QFQ Documentation

Release 20.9.0

Carsten Rose, Benjamin Baer, Rafael Ostertag, Marc Egger

Oct 30, 2020

Contents

Extension key qfq

Version 20.9.0

Language en

Authors Carsten Rose, Benjamin Baer, Marc Egger

Further Contributors Rafael Ostertag, Elias Villiger, Nicola Chiapolini

Email carsten.rose@math.uzh.ch, benjamin.baer@math.uzh.ch, marc.egger@uzh.ch

Status Productive, new features and bug fixes comes all the time.

Rendered Oct 30, 2020

TYPO3

The content of this document is related to TYPO3 CMS, a GNU/GPL CMS/Framework available from typo3.org .

About this manual:

This manual is a reference. Some basic examples are at the end.

Community documentation:

This document is *not* official TYPO3 documentation.

It is maintained as part of a third party extension.

If you find an error or something is missing, please report an issue

Extension Manual

This documentation is for the TYPO3 extension **qfq**.

Sitemap

CHAPTER 1

General

- Project homepage: https://qfq.io
- Latest releases: https://qfq.io/download
- Development: https://git.math.uzh.ch/typo3/qfq
- Chat: https://hello.math.uzh.ch > QFQ

CHAPTER 2

Installation

The following features are only tested / supported on linux hosts:

- General: QFQ is coded to run on Linux hosts, preferable on Debian derivates like Ubuntu.
- HTML to PDF conversion - command *wkhtmltopdf*.
- Concatenation of PDF files - command *pdfunite*.
- Convert of images to PDF files - command *img2pdf*.
- PDF decrypt (used for merge with pdfunite) - command *qpdf*.
- PDF decrypt (used for merge with pdfunite) - command *gs* - in case *qpdf* is not successful.
- Mime type detection for uploads - command *file*.

2.1 Preparation

2.1.1 Report & Form

To normalize UTF8 input, *php-intl* package is needed by

- normalizer::normalize()

For the *Download* function, the programs *img2pdf*, *pdfunite*, *qpdf*, *gs* and *file* are necessary to concatenate PDF files.

Preparation for Ubuntu:

```
sudo apt install php-intl
# for file upload, PDF and 'HTML to PDF' (wkhtmltopdf), PDF split
sudo apt install poppler-utils libxrender1 file pdf2svg qpdf ghostscript img2pdf
sudo apt install inkscape imagemagick            # to render thumbnails
```

2.1.2 wkhtmltopdf

wkhtmltopdf will be used by QFQ to offer 'website print' and 'HTML to PDF' conversion. The program is not included in QFQ and has to be manually installed.

- The Ubuntu package *wkhtmltopdf* needs a running Xserver - this does not work on a headless webserver.
  - Best is to install the QT version from the named website above.
  - In case of trouble with wkhtmltopdf, also install 'libxrender1'.
  - The current version 0.12.4 might have trouble with https connections. Version 0.12.5-dev (github master branch) seems more reliable. Please contact the QFQ authors if you need a compiled Ubuntu version of wkhtmltopdf.

In *Configuration* specify:

```
config.cmdWkhtmltopdf:  /opt/wkhtmltox/bin/wkhtmltopdf
config.baseUrl: http://www.example.com/
```

If wkhtml has been compiled with dedicated libraries (not part of LD_LIBRARY_PATH), specify the LD_LIBRARY_PATH together with the path-filename:

```
config.cmdWkhtmltopdf: LD_LIBRARY_PATH=/opt/wkhtmltox/lib /opt/wkhtmltox/bin/
↪wkhtmltopdf
```

Important: To access FE_GROUP protected pages or content, it's necessary to disable the *[FE][lockIP]* check! *wkhtml* will access the Typo3 page locally (localhost) and that IP address is different from the client (=user) IP.

Configure via Typo3 Installtool *All configuration* > *$TYPO3_CONF_VARS['FE']*:

```
[FE][lockIP] = 0
```

Warning: `[FE][lockIP] = 0` disables an important anti-'session hijacking' protection. The security level of the whole installation will be *lowered*! Again, this is only needed if *wkhtml* needs access to FE_GROUP protected pages & content. As an alternative to lower the security level, create a separated page subtree which is only accessible (configured via Typoscript) from specific IPs **or** if a FE-User is logged in.

If there are problems with converting/downloading FE_GROUP protected pages, check *Configuration showDebugInfo = download* to debug.

Note: Converting HTML to PDF gives no error message but RC=-1? Check carefully all includes of CSS, JS, images and so on! Typically some of them fails to load and wkhtml stops running! Verify the correct loading of all elements by calling the site via a regular browser and bypassing any browser cache (Ctrl F5).

Note: On Ubuntu, Apache is started by default with *LANG=C*. This is true even when the OS default locale is set to *en_US.UTF-8*. Furthermore, all child processes of Apache will inherit *LANG=C*. Some PHP functions (like 'escapeshellarg()') or *wkhtml* will strip all non-ASCII characters (e.g. commandline arguments).

Let Apache run with the system locale: */etc/apache/envvars*, activate the line . */etc/default/locale* and restart Apache.

2.1.2.1 Checklist wkhtml problems

- *config.baseUrl* is configured and correct. The baseUrl has to be the same protocol as the website (http or https).
- To track down problems:
  - In *Configuration* set *debug.showDebugInfo=auto,download*.
  - Do the download.
  - Check *QFQ Log* for any output.
  - Grab the URL given in the *QFQ Log*, open a browser in private mode (no existing browser session) and open the URL.
  - Check the *–cookie-jar '/tmp/qfq.cookie....'* file for the cookie.
  - Call wkhtml manually on the webserver, with the same options as given in the *QFQ Log*.

2.1.2.2 HTML to PDF conversion

wkhtmltopdf converts a website (local or remote) to a (multi)-page PDF file. It's mainly used in *Download*.

2.1.2.3 Print

Different browser prints the same page in different variations. To prevent this, QFQ implements a small PHP wrapper *print.php* with uses *wkhtmltopdf* to convert HTML to PDF.

Provide a *print this page*-link (replace 'current pageId'):

```
<a href="typo3conf/ext/qfq/Classes/Api/print.php?id={current pageId}">Print this page
↪</a>
```

Any parameter specified after *print.php* will be delivered to *wkhtmltopdf* as part of the URL.

Typoscript code to implement a print link on every page:

```
10 = TEXT
10 {
  wrap = <a href="typo3conf/ext/qfq/Classes/Api/print.php?id=|&type=99"><span class=
↪"glyphicon glyphicon-print" aria-hidden="true"></span> Printview</a>
  data = page:uid
}
```

2.1.3 Send Email

QFQ sends mail via *sendEmail* http://caspian.dotconf.net/menu/Software/SendEmail/ - a small perl script without a central configuration.

By default, *sendEmail* uses the local installed MTA, writes a logfile to *fileadmin/protected/log/mail.log* and handles attachments via commandline options. A basic HTML email support is implemented.

The latest version is v1.56, which has at least one bug. That one is patched in the QFQ internal version v1.56p1 (see QFQ GIT sources in directory 'patches/sendEmail.patch').

Nevertheless, on latest system the TLS support is broken - please check *sendEmail: Error => TLS setup failed*.

The Typo3 sendmail eco-system is not used at all by QFQ.

2.1.4 Thumbnail

Thumbnails will be rendered via ImageMagick (https://www.imagemagick.org/) 'convert' and 'inkscape' (https://inkscape.org). 'inkscape' is only used for '.svg' files.

The Typo3 graphic eco-system is not used at all by QFQ.

Usage: *Column: _thumbnail*.

2.2 Setup

- Install the extension via the Extension Manager.
  - If you install the extension by manual download/upload and get an error message "can't activate extension": rename the downloaded zip file to *qfq.zip* or *qfq_<version>.zip* (e.g. version: 18.12.0).
  - If the Extension Manager stops after importing: check your memory limit in php.ini.
- Copy/rename the file *<site path>/typo3conf/ext/qfq/config-example.qfq.php* to *<site path>/typo3conf/config.qfq.php*. Configure the necessary settings *Configuration* The configuration file is outside of the extension directory, to not loose it during de-install and install again.
- When the QFQ Extension is called the first time on the Typo3 frontend, the file *<ext_dir>/Classes/Sql/formEditor.sql* will played and fills the database with the *Form editor* records. This also happens automatically after each update of QFQ.
- Configure Typoscript to include Bootstrap, jQuery, QFQ javascript and CSS files.

2.2.1 Setup CSS & JS

```
page.meta {
    X-UA-Compatible = IE=edge
    X-UA-Compatible.attribute = http-equiv
    viewport=width=device-width, initial-scale=1
}

page.includeCSS {
    file01 = typo3conf/ext/qfq/Resources/Public/Css/bootstrap.min.css
    file02 = typo3conf/ext/qfq/Resources/Public/Css/bootstrap-theme.min.css
    file03 = typo3conf/ext/qfq/Resources/Public/Css/jqx.base.css
    file04 = typo3conf/ext/qfq/Resources/Public/Css/jqx.bootstrap.css
    file05 = typo3conf/ext/qfq/Resources/Public/Css/qfq-bs.css
    file06 = typo3conf/ext/qfq/Resources/Public/Css/tablesorter-bootstrap.css
    file07 = typo3conf/ext/qfq/Resources/Public/Css/font-awesome.min.css

    # Only needed in case FullCalendar is used
    file08 = typo3conf/ext/qfq/Resources/Public/Css/fullcalendar.min.css
}

page.includeJS {
    file01 = typo3conf/ext/qfq/Resources/Public/JavaScript/jquery.min.js
    file02 = typo3conf/ext/qfq/Resources/Public/JavaScript/bootstrap.min.js
    file03 = typo3conf/ext/qfq/Resources/Public/JavaScript/validator.min.js
    file04 = typo3conf/ext/qfq/Resources/Public/JavaScript/jqx-all.js
    file05 = typo3conf/ext/qfq/Resources/Public/JavaScript/globalize.js
    file06 = typo3conf/ext/qfq/Resources/Public/JavaScript/tinymce.min.js
```

(continues on next page)

(continued from previous page)

```
    file07 = typo3conf/ext/qfq/Resources/Public/JavaScript/EventEmitter.min.js
    file08 = typo3conf/ext/qfq/Resources/Public/JavaScript/typeahead.bundle.min.js
    file09 = typo3conf/ext/qfq/Resources/Public/JavaScript/qfq.min.js
    file10 = typo3conf/ext/qfq/Resources/Public/JavaScript/jquery.tablesorter.
↪combined.min.js
    file11 = typo3conf/ext/qfq/Resources/Public/JavaScript/jquery.tablesorter.pager.
↪min.js
    file12 = typo3conf/ext/qfq/Resources/Public/JavaScript/widget-columnSelector.min.
↪js

    # Only needed in case FormElement 'annotate' is used.
    file13 = typo3conf/ext/qfq/Resources/Public/JavaScript/fabric.min.js
    file14 = typo3conf/ext/qfq/Resources/Public/JavaScript/qfq.fabric.min.js

    # Only needed in case FullCalendar is used
    file15 = typo3conf/ext/qfq/Resources/Public/JavaScript/moment.min.js
    file16 = typo3conf/ext/qfq/Resources/Public/JavaScript/fullcalendar.min.js
}
```

2.3 FormEditor

Setup a *report* to manage all *forms*:

- Create a Typo3 page.
- Set the 'URL Alias' to *form* (recommended) or the individual defined value in parameter *editFormPage* (*configuration*).
- Insert a content record of type *qfq*.
- In the bodytext insert the following code:

```
# If there is a form given by SIP: show
form={{form:SE}}

# In case indexQfq != indexData, set dbIndex=indexQfq.
dbIndex = {{indexQfq:Y}}

10 {
    # Table header.
    sql = SELECT CONCAT('p:{{pageAlias:T}}&form=form|A:data-reference=newForm')␣
↪as _pagen, '#', 'Name', 'Title', 'Table', ''
    head = {{'b|p:id={{pageAlias:T}}&form=copyFormFromExt|t:Copy form from␣
↪ExtForm|A:data-reference=copyForm' AS _link}}
            <table class="table table-hover qfq-table-50 tablesorter tablesorter-
↪filter" id="{{pageAlias:T}}-form">
    tail = </table>
    rbeg = <thead class="qfq-sticky"><tr>
    rend = </tr></thead>
    fbeg = <th>
    fend = </th>

    10 {
        # All forms
        sql = SELECT CONCAT('p:{{pageAlias:T}}&form=form&r=', f.id, '|A:data-
↪reference=editForm', f.name) as _pagee
```

(continues on next page)

(continued from previous page)

```
                    , f.id, f.name, QMORE(strip_tags(f.title), 50), f.tableName
                    , CONCAT('U:form=form&r=', f.id, '|A:data-reference=deletForm
↪') as _paged
                FROM Form AS f
                ORDER BY f.name
        rbeg = <tr>
        rend = </tr>
        fbeg = <td>
        fend = </td>
    }
}
```

To keep the overview about all forms, it's useful to know which form has been used, how often, on which page and when. Find these information included in the *FormEditor with usage* report.

2.4 Installation: Check List

- Protect the directory *<T3 installation>/fileadmin/protected* in Apache against direct file access.
  - *<T3 installation>/fileadmin/protected/* should be used for confidential (uploaded / generated) data.
  - *<T3 installation>/fileadmin/protected/log/...* is the default place for QFQ log files.
- Protect the directory *<T3 installation>/fileadmin* in Apache to not execute PHP Scripts - malicious uploads won't be executed.
- Setup a log rotation rule for *sqlLog*.
- Check that *sqlLogMode* is set to *modify* on productive sites. With *none* you have no chance to find out who changed which data and *all* really logs a mass of data.

2.5 Configuration

2.5.1 config.qfq.php

Keyword	Example	Description
DB_<n>_USER	DB_1_USER=qfqUser	Credentials configured in MySQL
DB_<n>_PASSWORD	DB_1_PASSWORD=1234567890	Credentials configured in MySQL
DB_<n>_SERVER	DB_1_SERVER=localhost	Hostname of MySQL Server
DB_<n>_NAME	DB_1_NAME=qfq_db	Database name
LDAP_1_RDN LDAP_1_PASSWORD	LDAP_1_RDN='ou=Admin,ou=example,dc=com' LDAP_1_PASSWORD='mySecurePassword'	Credentials for non-anonymous LDAP access. Only one set supported.

Example: *typo3conf/config.qfq.php*:

```
<?php

// QFQ configuration
//
// Save this file as: <site path>/typo3conf/config.qfq.php
```

(continues on next page)

(continued from previous page)

```
return [
    'DB_1_USER' => '<DBUSER>',
    'DB_1_SERVER' => '<DBSERVER>',
    'DB_1_PASSWORD' => '<DBPW>',
    'DB_1_NAME' => '<DB>',

    //DB_2_USER => <DBUSER>
    //DB_2_SERVER => <DBSERVER>
    //DB_2_PASSWORD => <DBPW>
    //DB_2_NAME => <DB>

    // DB_n ...
    // ...

    // LDAP_1_RDN => 'ou=Admin,ou=example,dc=com'
    // LDAP_1_PASSWORD => 'mySecurePassword'
];
```

2.5.2 Extension Manager: QFQ Configuration

Keyword	Default / Example	Description
Config		
flagProduction	yes	yes|no: used to differentiate production and development site.
render	single	both|single: QFQ will show form and/or report. In most cases only one at a time is needed. Options: 'single' (default) or 'both' (legacy). In mode 'single' prefer 'form' over 'report'.
maxFileSize	10M	If empty, take minimum of 'post_max_size' and 'upload_max_filesize'.
baseUrl	http://example.com	URL where wkhtmltopdf will fetch the HTML (no parameter, those comes later)
dateFormat	yyyy-mm-dd	Possible options: yyyy-mm-dd, dd.mm.yyyy.
thumbnailDirSecure	fileadmin/protected/qfqThumbnail	Important: secure directory 'protected' (recursive) against direct access.
thumbnailDirPublic	typo3temp/qfqThumbnail	Both thumbnail directories will be created if not existing.
cmdInkscape	inkscape	If inkscape is not available, specify an empty string.
cmdConvert	convert	GraphicsMagics 'convert' is recommended.
cmdWkhtmltopdf	/usr/bin/wkhtmltopdf	PathFilename of wkhtmltopdf. Optional variables like LD_LIBRARY_PATH=...

Continued on next page

Table 1 – continued from previous page

Keyword	Default / Example	Description
cmdQpdf	qpdf	PathFilename of qpdf. Optional variables like LD_LIBRARY_PATH=...
cmdGs	gs	PathFilename of gs. Optional variables like LD_LIBRARY_PATH=...
cmdPdfunite	pdfunite	PathFilename of pdfunite. Optional variables like LD_LIBRARY_PATH=...
cmdImg2pdf	img2pdf	PathFilename of img2pdf. Optional variables like LD_LIBRARY_PATH=...
sendEMailOptions	-o tls=yes	General options. Check: http://caspian.dotconf.net/menu/Software/SendEmail
documentation	http://docs.typo3.org...	Link to the online documentation of QFQ. Every QFQ installation also contains a local copy: typo3conf/ext/qfq/Documentation/html/Manual.html
Dynamic		
fillStoreSystemBySql1/2/3	SELECT s.id AS ...	Specific values read from the database to fill the system store during QFQ load. See *Fill STORE_SYSTEM by SQL* for a usecase.
fillStoreSystemBySqlErrorMsg1/2/3	No current period found	Only define an error message, if QFQ should stop running in case of an SQL error or not exact one record.
Debug		
throwExceptionGeneralError	auto	*yes*: 'general errors' in QFQ (PHP) will throw an exception. *auto*: becomes 'yes', if 'flagProduction'!='yes', else 'no'. *no*: 'general errors' in QFQ (PHP) will be silently ignored.
formSubmitLogMode	all	*all*: every form submission will be logged. *none*: no logging. See *Form Submit Log page* for example QFQ code to display the log.
redirectAllMailTo	john@doe.com	If set, redirect all QFQ generated mails (Form, Report) to the specified.

Continued on next page

Table 1 – continued from previous page

Keyword	Default / Example	Description
sqlLogMode	modify	*all*: every statement will be logged - this might be a lot. *modifyAll*: log all statements which might change data, even if 0 rows affected. *modify*: log only statements which change data (affected rows > 0). *error*: log only DB errors. *none*: no SQL log at all.
sqlLogModeAutoCron	error	Applies only to AutoCron Jobs. For production 'error' should be fine.
sqlLog	fileadmin/protected/log/sql.log	Filename to log SQL commands: relative to <site path> or absolute. If the directory does not exist, create it.
qfqLog	fileadmin/protected/log/qfq.log	Filename to log general QFQ events:relative to <site path> or absolute. If the directory does not exist, create it.
mailLog	fileadmin/protected/log/mail.log	Filename to log *sendEmail* commands: relative to <site path> or absolute. If the directory does not exist, create it.
showDebugInfo	auto	FE - Possible values: yes|no|auto|download. For 'auto': If a BE User is logged in, a debug information will be shown on the FE.
Database		
init	init=SET names utf8; SET sql_mode = "NO_ENGINE_SUBSTITUTION"	Global init for using the database. For 'sql_mode="NO_ENGINE_SUBSTITUTION"' see #7407.
update	auto	*auto*: apply DB Updates only if there is a newer version. *always*: apply DB Updates always, especially play formEditor.sql every time QFQ is called - *not* recommended! *never*: never apply DB Updates.
indexData	1	Optional. Default: 1. Retrieve the current setting via {{dbNameData:Y}}.

Continued on next page

Table 1 – continued from previous page

Keyword	Default / Example	Description
indexQfq	1	Optional. Default: 1. Retrieve the current setting via {{dbNameQfq:Y}}.
Security		
escapeTypeDefault	m	All variables *{{...}}* get this escape class by default. See *Escape/Action class*.
securityVarsHoneypot	email,username,password	If empty: no check. All named variables will rendered as INPUT elements.
securityAttackDelay	5	If an attack is detected, sleep 'x' seconds and exit PHP process.
securityShowMessage	true	If an attack is detected, show a message.
securityGetMaxLength	50	GET vars longer than 'x' chars triggers an *attack-recognized*. *Exception for SECURITY_GET_MAX_LENGTH*.
securityFailedAuthDelay	3	If REST authorization fails, sleep 'x' seconds before answering.
Form-Config		
recordLockTimeoutSeconds	900	Timeout for record locking. After this time, a record will be replaced.
sessionTimeoutSeconds	1800	Timeout for FE User session. See *FE-User: Session timeout seconds*
enterAsSubmit	enterAsSubmit = 1	0: off, 1: Pressing *enter* in a form means *save* and *close*.
editFormPage	form	T3 Pagealias to edit a form.
formDataPatternError	please check pattern error	Customizable error message used in validator.js. 'pattern' violation.
formDataRequiredError	missing value	Customizable error message used in validator.js. 'required' fields.
formDataMatchError	type error	Customizable error message used in validator.js. 'match' retype mismatch.
formDataError	generic error	Customizable error message used in validator.js. 'no specific' given.
Form-Layout		
labelAlign	left	Label align (left/center/right)/ Default: left. Will be inherited to Form.
cssClassQfqContainer	container	QFQ with own Bootstrap: 'container'. QFQ already nested in Bootstrap of mainpage: <empty>.
cssClassQfqForm	qfq-color-base	Wrap around QFQ 'Form'.
cssClassQfqFormPill	qfq-color-grey-1	Wrap around title bar for pills: CSS Class, typically a background color.

Continued on next page

Table 1 – continued from previous page

Keyword	Default / Example	Description
cssClassQfqFormBody	qfq-color-grey-2	Wrap around FormElements: CSS Class, typically a background color.
formBsColumns	col-md-12 col-lg-10	The whole form will be wrapped. See *Custom field width*
formBsLabelColumns	col-md-3 col-lg-3	The column get the width. See *Custom field width*
formBsInputColumns	col-md-6 col-lg-6	
formBsNoteColumns	col-md-3 col-lg-3	
extraButtonInfoInline	<img src="info.png">	Image for *extraButtonInfo* (inline).
extraButtonInfoBelow	<img src="info.png">	Image for *extraButtonInfo* (below).
extraButtonInfoPosition	below	'auto' (default) or 'below'. See *extraButtonInfo*.
extraButtonInfoClass	pull-right	'' (default) or 'pull-right'. See *extraButtonInfo*.
Form-Language		
formLanguage[ABCD]Id	E.g.: 1	In Typo3 configured pageLanguage id. The number after the 'L' parameter.
formLanguage[ABCD]Label	E.G.: english	Label shown in *Form editor*, on the 'basic' tab.
saveButtonText		Text on the form save button. Typically none.
saveButtonTooltip	Save	Tooltip on the form save button.
saveButtonClass	btn btn-default navbar-btn	Bootstrap CSS class for save button on top of the form.
buttonOnChangeClass	alert-info btn-info	Bootstrap CSS class for save button showing 'data changed'.
saveButtonGlyphIcon	glyphicon-ok	Icon for the form save button.
closeButtonText		Text on the form close button. Typically none.
closeButtonTooltip	close	Tooltip on the form close button.
closeButtonClass	btn btn-default navbar-btn	Bootstrap CSS class for close button on top of the form.
closeButtonGlyphIcon	glyphicon-remove	Icon for the form close button.
deleteButtonText		Text on the form delete button. Typically none.
deleteButtonTooltip	delete	Tooltip on the form delete button.
deleteButtonClass	btn btn-default navbar-btn	Bootstrap CSS class for delete button on top of the form.
deleteButtonGlyphIcon	glyphicon-trash	Icon for the form delete button.
newButtonText		Text on the form new button. Typically none.
newButtonTooltip	new	Tooltip on the form new button.
newButtonClass	btn btn-default navbar-btn	Bootstrap CSS class for new button on top of the form.
newButtonGlyphIcon	glyphicon-plus	Icon for the form new button.
showIdInFormTitle	0 (off), 1 (on)	Append at the form title the current record id.
cssClassColumnId	text-muted	A column in a subrecord with the name id|ID|Id gets this class.

After parsing the configuration, the following variables will be set automatically in STORE_SYSTEM:

Keyword	Description
dbNameData	Name of the 'data'-database. '{{dbNameData:Y}}
dbNameQfq	Name of the 'QFQ'-database. '{{dbNameQfq:Y}}
dbNameT3	Name of the 'T3'-database. '{{dbNameT3:Y}}
sitePath	Absolute path of the current T3 instance. '{{sitePath:Y}}
extPath	Absolute path of the QFQ extension. '{{extPath:Y}}

2.5.3 Custom variables

Up to 30 custom variables can be defined in *Configuration*.

E.g. to setup a contact address and reuse the information inside your installation do:

```
custom1: ADMINISTRATIVE_CONTACT = john@doe.com
custom2: ADMINISTRATIVE_ADDRESS = John Doe, Hollywood Blvd. 1, L.A.
custom3: ADMINISTRATIVE_NAME = John Doe
```

- Somewhere in a *Form* or in *Report*:

```
{{ADMINISTRATIVE_CONTACT:Y}}, {{ADMINISTRATIVE_ADDRESS:Y}}, {{ADMINISTRATIVE_NAME}
↪}
```

It's also possible to configure such variables directly in *config.qfq.php*.

2.5.4 Fill STORE_SYSTEM by SQL

A specified SELECT statement in *Configuration* in variable *fillStoreSystemBySql1* (or *2*, or *3*) will be fired. The query should have 0 (nothing happens) or 1 row. All columns will be **added** to the existing STORE_SYSTEM. Existing variables will be overwritten. Be careful not to overwrite system values.

This option is useful to make generic custom values, saved in the database, accessible to all QFQ Report and Forms. Access such variables via *{{<varname>:Y}}*.

In case QFQ should stop working if a given query does not select exact one record (e.g. a missing period), define an error message:

```
fillStoreSystemBySql1: SELECT name FROM Person WHERE name='Doe'
fillStoreSystemBySqlErrorMsg1: Too many or to few "Doe's" in our database
```

2.5.4.1 periodId

This is

- a usecase, implemented via *Fill STORE_SYSTEM by SQL*,
- a way to access *Period.id* with respect to the current period (the period itself is custom defined).

After a full QFQ installation:

- a table *Period* (extend / change it to your needs, fill them with your periods),
- one sample record in table *Period*,

Websites, delivering semester data, school year schedules, or any other type or periods, often need an index to the *current* period.

In *Configuration*:

```
fillStoreSystemBySql1: SELECT id AS periodId FROM Period WHERE start<=NOW() ORDER BY␣
↪start DESC LIMIT 1
```

a variable 'periodId' will automatically computed and filled in STORE SYSTEM. Access it via *{{periodId:Y0}}*. To get the name and current period:

```
SELECT name, ' / ', start FROM Period WHERE id={{periodId:Y0}}
```

Typically, it's necessary to offer a 'previous' / 'next' link. In this example, the STORE SIP holds the new periodId:

```
SELECT CONCAT('p:{{pageAlias:T}}&periodId=', {{periodId:SY0}}-1, '|Next') AS _page, '
↪', name, ' ',
  CONCAT('p:{{pageAlias:T}}&periodId=', {{periodId:SY0}}+1, '|Next') AS _page FROM␣
↪Period AS s WHERE s.id={{periodId:SY0}}
```

Take care for minimum and maximum indexes (do not render the links if out of range).

2.5.5 DB USER privileges

The specified DB User needs privileges to the database of at least: SELECT / INSERT / UPDATE / DELETE / SHOW.

To apply automatically QFQ-'DB UPDATE' the following rights are mandatory too: CREATE / ALTER

To get access to the Typo3 installation, 'dbuser' should also have access to the Typo3 Database with at least SELECT / INSERT / UPDATE / DELETE.

2.5.6 Exception for SECURITY_GET_MAX_LENGTH

If it is necessary to use a GET variable which exceeds *securityGetMaxLength* limit, name the variable with '_<num>' at the end. E.g. *my_long_variable_130*. Such a variable has an allowed length of 130 chars. Access the variable as usual with the variable name: *{{my_long_variable_130:C:...}}*.

2.5.7 FE-User: Session timeout seconds

There is no timeout for website users who are not logged in (but typically those users don't have access to protected content).

For logged in users, the default timeout is the php.ini settings for *session.cookie_lifetime* and *session.gc_maxlifetime* (minimum of both). These timeout only affects QFQ related content and can be specified a) globally (QFQ configuration) and b) specific per Form.

The maximum timeout depends on the minimal value of php.ini *session.cookie_lifetime* and *session.gc_maxlifetime*. Specifying a higher value produces an error in the front end.

Every access to QFQ related content resets the timeout.

After FE login, the next access to QFQ related content starts the timeout counter.

CHAPTER 3

Concept

3.1 SIPs

The following is a technical background information. Not needed to just use QFQ.

The SIPs (=Server Id Pairs) are uniq timestamps, created/registered on the fly for a specific browser session (=user). Every SIP is registered on the server (= stored in a browser session) and contains one or more key/value pairs. The key/value pairs never leave the server. The SIPs will be used:

- to protect values not to be spoofed by anyone,
- to protect values not to be altered by anyone,
- to grant access, e.g.:
  - load and save forms,
  - upload files,
  - download files,
  - PHP AJAX pages.

SIPs becomes invalid, as soon as the current browser session is destroyed. The client (= user) can't manipulate the content of SIPs - it's only possible to reuse already registered SIPs by the user, who already owns the session.

3.2 Access privileges

The Typo3 FE Groups can be used to implement access privileges. Such groups are assigned to

- Typo3 FE users,
- Typo3 pages,
- and/or Typo3 content records (e.g. QFQ records).

This will be used for general page structure privileges.

A *record base* privileges controlling (e.g. which user can edit which person record) will be implicit configured, by the way that records are viewable / editable (or not) through SQL in the specific QFQ tt-content statements.

3.3 Typo3 QFQ content element

Insert one or more QFQ content elements on a Typo3 page. Specify column and language per content record as wished.

The title of the QFQ content element will not be rendered on the frontend. It's only visible to the webmaster in the backend for orientation.

3.3.1 QFQ Keywords (Bodytext)

All of these parameters are optional.

Name	Explanation
form	Formname. Static: **form = person** By SIP: **form = {{form:SE}}** By SQL: **form = {{SELECT c.form FROM Config AS c WHERE c.id={{a:C}} }}**
r	<record id>. The form will load the record with the specified id. Static: **r = 123** By SQL: **r = {{SELECT . . . }}** If not specified, the SIP parameter 'r' is used.
dbIndex	E.g. *dbIndex = {{indexQfq:Y}}* Select a DB index. Only necessary if a different than the standard DB should be used.
debugShowBodyText	If='1' and *Configuration*:*showDebugInfo: yes*, shows a tooltip with bodytext
sqlLog	Overwrites *Configuration*: *SQL Log* . Only affects *Report*, not *Form*.
sqlLogMode	Overwrites *Configuration*: *SQL_LOG_MODE* . Only affects *Report*, not *Form*.
render	See *Report: render*. Overwrites *Configuration*: render.
<level>.fbeg	Start token for every field (=column)
<level>.fend	End token for every field (=column)
<level>.fsep	Separator token between fields (=columns)
<level>.fskipwrap	Skip wrapping (via fbeg, fsep, fend) of named columns. Comma separated list of column id's (starting at 1). See also the special column name '_noWrap' to suppress wrapping.
<level>.shead	Static start token for whole <level>, independent if records are selected Shown before *head*.
<level>.stail	Static end token for whole <level>, independent if records are selected. Shown after *tail*.
<level>.head	Dynamic start token for whole <level>. Only if at least one record is select.
<level>.tail	Dynamic end token for whole <level>. Only if at least one record is select.
<level>.rbeg	Start token for row.
<level>.rbgd	Alternating (per row) token.
<level>.rend	End token for row. Will be rendered **before** subsequent levels are processed
<level>.renr	End token for row. Will be rendered **after** subsequent levels are processed
<level>.rsep	Seperator token between rows
<level>.sql	SQL Query
<level>.twig	Twig Template
<level>.althead	If <level>.sql has no rows selected (empty), these token will be rendered.
<level>.altsql	If <level>.sql has no rows selected (empty) or affected (delete, update, insert) the <altsql> will be fired. Note: Sub queries of <level> are not fired, even if <altsql> selects some rows.
<level>.content	

3.4 Report: render

QFQ will render Report or Form in three situations:

1. Browser: The QFQ content is described by QFQ-Report syntax.
2. Browser: The QFQ content is described by a QFQ-Form.
3. API: The QFQ content is handled without Typo3. Typically a single tt-content record is specified in the request - only that one is rendered by QFQ. This mode is typically used to export data like Excel Export.

Option 1 and 2 are distinguished by the parameter *form* (STORE_SIP or STORE_TYPO3). If 'form' is given, in most cases only a Form should be shown (not the report).

render	Use
single	Display Form or Report, but not both at the same time. If a SIP parameter 'form' is given, the form is rendered, else the report.
both	Display Form and Report, both at the same time.
api	Create output only when called via API (no Typo3).

Example:

```
render = both
form = {{form:SE}}

10 {
  sql = SELECT ...
}
```

3.5 QFQ Database

Recommended setup for Typo3 & QFQ Installation is with *two* databases. One for the Typo3 installation and one for QFQ. A good practice is to name both databases equal, appending the suffix '_t3' and '_db'.

When QFQ is called, it checks for QFQ system tables. If they do not exist or have a lower version than the installed QFQ version, the system tables will be automatically installed or updated.

3.5.1 System tables

Name	Use	Database
Clipboard	Temporary	QFQ
Cron	Persistent	QFQ
Dirty	Temporary	QFQ | Data
Form	Persistent	QFQ
FormElement	Persistent	QFQ
FormSubmitLog	Persistent	QFQ | Data
MailLog	Persistent	QFQ | Data
Period	Persistent	Data
Split	Persistent	Data

See *Mail Log page (Table MailLog)* and *Form Submit Log page* for some Frontend views for these tables.

- Check Bug #5459 / support of system tables in different DBs not supported.

3.5.2 Multi Database

3.5.2.1 Base: T3 & QFQ

QFQ typically interacts with one database, the QFQ database. The database used by Typo3 is typically a separate one. Theoretically it might be the same (never tested), but it's strongly recommended to use a separated QFQ database to have no problems on Typo3 updates and to have a clean separation between Typo3 and QFQ.

3.5.2.2 QFQ: System & Data

QFQ itself can be separated in 'QFQ system' (see *System tables*) and 'QFQ data' databases (even more than one are possible). The 'QFQ system' stores the forms, record locking, log tables and so on - *QFQ data* is for the rest.

A *Multi Database* setup is given, if 'QFQ system' is different from 'QFQ data'.

3.5.2.3 Data: Data1, Data2, . . . , Data n

Every database needs to be configured via *Configuration* with it's own *index*.

QFQ data might switch between different 'data'-databases. In *Configuration* one main *QFQ data* index will be specified in *indexQfq*. If specific forms or reports should use a different database than that, *dbIndex* might change *indexData* temporarily.

dbIndex: A *Report* (field *dbIndex*) as well as a *Form* (field *parameter*.'dbIndex') can operate on a specific database.

A *Form* will:

- load the form-definition from *indexQfq* (table *Form* and *FormElement*),
- loads and save data from/in *indexData* (config.qfq.php) / *dbIndex* (form.parameter.dbIndex),
- retrieve extra information via *dbIndexExtra* - this is useful to offer information from a database and save them in a different one.

The simplest setup, QFQ system & data in the same database, needs no *indexQfq / indexData* definition in *Configuration* or one or both of them set to '1'

To separate QFQ system and data, indexQfq and indexData will have different indexes.

A Multi Database setup might be useful for:

- several independent Typo3 QFQ installations (each have it's own form repository) and one central database, or
- one QFQ installation which should display / load /save records from different databases, or
- a combination of the above two.

Note:

- Option 'A' is the most simple and commonly used.
- Option 'B' separate the T3 and QFQ databases on two database hosts.
- Option 'C' is like 'B' but with a shared 'QFQ data'-database between three 'Typo3 / QFQ' instances.
- Further variants are possible.

	Domain	Website Host	T3	QFQ system	QFQ data
A	standalone.edu	'w'	<dbHost>, <dbname>_t3, <dbnameSingle>_db		
B	appB1.edu	'wApp'	<dbHostApp>, <dbnameB1>_t3	<dbHostB1>, <dbnameApp>_db	
B	appB2.edu	'wApp'	<dbHostApp>, <dbnameB2>_t3	<dbHostB2>, <dbnameApp>_db	
C	appC1.edu	'wAppC'	<dbHostAppC>, <dbnameC1>_t3	<dbHostC>, <dbnameSysC1>_db	<dbHostData>_db, <dbNameData>_db
C	appC2.edu	'wAppC'	<dbHostAppC>, <dbnameC2>_t3	<dbHostC>, <dbnameSysC2>_db	<dbHostData>_db, <dbNameData>_db
C	appC3.edu	'wAppC3'	<dbHostAppC3>, <dbnameC3>_t3	<dbHostC3>, <dbnameSysC3>_db	<dbHostData>_db, <dbNameData>_db

In *config.qfq.php* mutliple database credentials can be prepared. Mandatory is at least one credential setup like *DB_1_USER*, *DB_1_SERVER*, *DB_1_PASSWORD*, *DB_1_NAME*. The number '1' indicates the *dbIndex*. Increment the number to specify further database credential setups.

Typically the credentials for *DB_1* also have access to the T3 database.

3.5.3 Different QFQ versions, shared database

When using different QFQ versions and a shared 'QFQ data'-database, there is some risk of conflicting 'QFQ system' tables. Best is to always use the same QFQ version on all instances or use a Multi Database setup.

CHAPTER 4

Debug

4.1 QFQ Log

Setup in *Configuration*

- *qfqLog*
  - Filename where to log QFQ debug and error messages.
  - File is relative to the <*site path*> or absolute (starting with '/').
  - Content: error situations of QFQ and debug, if enabled.

All non SQL related information will be logged to QFQ log file.

4.2 SQL Log

Setup in *Configuration*

- *sqlLog*
  - Filename where to log SQL queries and statistical data.
  - File is relative to the <*site path*> or absolute (starting with '/').
  - Content: SQL queries and timestamp, formName/formId, fe_user, success, affected rows, newly created record id's and accessed from IP.
  - The global setting can be overwritten by defining *sqlLog* inside of a QFQ tt-content record.
- *sqlLogMode: all|modify|error|none*
  - *all*: logs every SQL statement.
  - *modify*: logs only statements who might potentially change data.
  - *error*: logs only queries which generate SQL errors.

  - *none*: no query logging at all.
  - The global setting can be overwritten by defining *sqlLogMode* inside of a QFQ tt-content record.

- *showDebugInfo = [yes|no|auto],[download]*

  If active, displays additional information in the Frontend (FE). This is typically helpful during development.

  - *yes*:
    * Form:
      · For every internal link/button, show tooltips with decoded SIP on mouseover.
      · Shows an 'Edit form'-button (wrench symbol) on a form. The link points to the T3 page with the *FormEditor*.
    * Report: Will be configured per tt-content record.

      debugShowBodyText = 1
  - *no*: No debug info.
  - *auto*: Depending if there is a Typo3 BE session, set internally:
    * *showDebugInfo = yes* (BE session exist)
    * *showDebugInfo = no* (no BE session)
  - *download*:
    * During a download (especially by using wkhtml), temporary files are not deleted automatically. Also the `wkhtmltopdf`, `pdfunite`, `pdf2img` command lines will be logged to *QFQ Log*. Use this only to debug problems on download.

4.3 MAIL Log

Setup in *Configuration*

- *mailLog*
  - File which *sendEmail* logs sending mail.
  - File is relative to the <*site path*> or absolute (starting with '/').

4.4 Mail Log page (Table MailLog)

For debugging purposes you may like to add a Mail Log page in the frontend. The following QFQ code could be used for that purpose (put it in a QFQ PageContent element)

Note: If you do not use/have the Ggroup table, then remove the "# Filters" block.

```
# Page parameters
1.sql = SELECT @grId := '{{grId:C0:digit}}' AS _grId
2.sql = SELECT @summary := IF('{{summary:CE:alnumx}}' = 'true', 'true', 'false') AS _s

# Filters
10 {
  sql = SELECT "'", gr.id, IF(gr.id = @grId, "' selected>", "'>"), gr.value, ' (Id: ',
↪ gr.id, ')'
```

(continues on next page)

(continued from previous page)

```
            FROM Ggroup AS gr
            INNER JOIN MailLog AS ml ON ml.grId = gr.id
            GROUP BY gr.id
  head = <form onchange='this.submit();' class='form-inline'><input type='hidden'␣
↪name='id' value='{{pageAlias:T0}}'>
            Filter By Group: <select name='grId' class='form-control'><option value=''>
↪</option>
  rbeg = <option value=
  rend = </option>
  tail = </select>
}

20 {
  sql = SELECT IF(@summary = 'true', ' checked', '')
  head = <div class='checkbox'><label><input type='checkbox' name='summary' value=
↪'true'
  tail = >Summary</label></div></form>
}

# Mail Log
50 {
  sql = SELECT id, '</td><td>', grId, '</td><td>', xId, '</td><td>'
               , REPLACE(receiver, ',', '<br>'), '</td><td>', REPLACE(sender, ',', '
↪<br>'), '</td><td>'
               , DATE_FORMAT(modified, '%d.%m.%Y<br>%H:%i:%s'), '</td><td style="word-
↪break:break-word;">'
               , CONCAT('<b>', subject, '</b><br>', IF(@summary = 'true',␣
↪CONCAT(SUBSTR(body, 1
                        , LEAST(IF(INSTR(body, '\n') = 0, 50, INSTR(body, '\n')),␣
↪IF(INSTR(body, '<br>') = 0, 50
                        , INSTR(body, '<br>')))-1), ' ...'), CONCAT('<br>',␣
↪REPLACE(body, '\n', '<br>'))) )
          FROM MailLog
          WHERE (grId = @grId OR @grId = 0)
          ORDER BY modified DESC
          LIMIT 100
  head = <table class="table table-condensed table-hover"><tr>
             <th>Id</th><th>grId</th><th>xId</th><th>To</th><th>From</th><th>Date</th>
↪<th>E-Mail</th></tr>
  tail = </table>
  rbeg = <tr><td>
  rend = </td></tr>
}
```

4.5 Redirect all mail to (catch all)

Setup in *Configuration*

- *redirectAllMailTo=john@doe.com*
  - During the development, it might be helpful to configure a 'catch all' email address, which QFQ uses as the final receiver instead of the original intended one.
  - The setting will:
    * Replace the 'To' with the configured one.

* Clear 'CC' and 'Bcc'
* Write a note and the original configured receiver at the top of the email body.

4.6 Show log files realtime

Display QFQ log files in realtime.

The following QFQ code could be used for that purpose (put it in a QFQ PageContent element):

```
#
# {{logfile:SU}}
#

# Show buttons to select log file.
10 {
  sql = SELECT '{{logfile:SU:::sql.log}}' AS '_=logfile'
  head = <p>
  tail = </p>

  20.sql = SELECT CONCAT('p:{{pageAlias:T}}&logfile=sql.log|t:sql.log|b:', IF('{
↪{logfile:R}}'='sql.log','primary','')) AS _page, ' '
                   , CONCAT('p:{{pageAlias:T}}&logfile=qfq.log|t:qfq.log|b:', IF('{
↪{logfile:R}}'='qfq.log','primary','')) AS _page, ' '
                   , CONCAT('p:{{pageAlias:T}}&logfile=mail.log|t:mail.log|b:', IF('{
↪{logfile:R}}'='mail.log','primary','')) AS _page
}

# Show selected log file.
100 {
  sql = SELECT 'file:fileadmin/protected/log/{{logfile:R}}' AS _monitor
  head = <pre id="monitor-1">Please wait</pre>
}
```

4.7 Form Submit Log page

For debugging purposes you may like to add a Form Submit Log page in the frontend. The following QFQ code could be used for that purpose (put it in a QFQ PageContent element):

```
# Filters
20.shead = <form onchange='this.submit()' class='form-inline'><input type='hidden'␣
↪name='id' value='{{pageAlias:T0}}'>
20 {
  sql = SELECT "'", id, IF(id = '{{formId:SC0}}', "' selected>", "'>"), name
          FROM Form
          ORDER BY name
  head = <label for='formId'>Form:</label> <select name='formId' id='formId' class=
↪'form-control'><option value=0></option>
  tail = </select>
  rbeg = <option value=
  rend = </option>
}
30 {
  sql = SELECT feUser, IF(feUser = '{{feUser:SCE:alnumx}}', "' selected>", "'>"),␣
↪feUser
```

(continues on next page)

(continued from previous page)

```
          FROM FormSubmitLog
          GROUP BY feUser ORDER BY feUser
  head = <label for='feUser'>FE User:</label> <select name='feUser' id='feUser' class=
↪'form-control'><option value=''></option>
  tail = </select>
  rbeg = <option value='
  rend = </option>
}
30.stail = </form>

# Show Log
50 {
  sql = SELECT l.id
                , CONCAT('<b>Form</b>: ', f.name
                , '<br><b>Record Id</b>: ', l.recordId
                , '<br><b>Fe User</b>: ', l.feUser
                , '<br><b>Date</b>: ', l.created
                , '<br><b>Page Id</b>: ', l.pageId
                , '<br><b>Session Id</b>: ', l.sessionId
                , '<br><b>IP Address</b>: ', l.clientIp
                , '<br><b>User Agent</b>: ', l.userAgent
                , '<br><b>SIP Data</b>: <div style="margin-left:20px;">'
                , "<script>var data = JSON.parse('", l.sipData, "'); for (var key in␣
↪data) {
                            document.write('<b>' + key + '</b>: ' + data[key] + '<br>
↪'); }</script>", '</div>')
                , CONCAT("<script>var data = JSON.parse('", l.formData, "'); for␣
↪(var key in data) {
                            document.write('<b>' + key + '</b>: ' + data[key] + '<br>
↪'); }</script>")
           FROM FormSubmitLog AS l
           LEFT JOIN Form AS f
             ON f.id = l.formId
           WHERE (l.formId = '{{formId:SC0}}' OR '{{formId:SC0}}' = 0)
             AND (l.feUser = '{{feUser:SCE:alnumx}}' OR '{{feUser:SCE:alnumx}}' = '')
           ORDER BY l.created DESC
           LIMIT 100
  head = <table class="table table-hover">
        <tr><th>Id</th><th style="min-width:250px;">Environment</th><th>Submitted␣
↪Data</th>
  tail = </table>
  rbeg = <tr>
  renr = </tr>
  fbeg = <td>
  fend = </td>
}
```

CHAPTER 5

Variable

Variables in QFQ are surrounded by double curly braces. Four different types of variable substitution functionality is provided. Access to:

- *Store variables*
- *SQL variables*
- *Row column variables*
- *Link column variables*

Some examples, including nesting:

```
# Store
#-----------------------------------------------
{{r}}
{{index:FS}}
{{name:FS:alnumx:s:my default}}

# SQL
#-----------------------------------------------
{{SELECT name FROM Person WHERE id=1234}}

# Row columns
#-----------------------------------------------
{{10.pId}}
{{10.20.pId}}

# Nesting
#-----------------------------------------------
{{SELECT name FROM Person WHERE id={{r}} }}
{{SELECT name FROM Person WHERE id={{key1:C:alnumx}} }} # explained below
{{SELECT name FROM Person WHERE id={{SELECT id FROM Persfunction LIMIT 1}} }} # it's␣
↪more efficient to use only one query
```

(continues on next page)

(continued from previous page)

```
# Link Columns
{{p:form=Person&r=1|t:Edit Person|E|s AS link}}
```

Leading and trailing spaces inside curly braces are removed.

- *{{ SELECT "Hello World" }}* becomes *{{SELECT "Hello World"}}*
- *{{ varname }}* becomes *{{varname}}*

5.1 Types

5.1.1 Store variables

Note: {{ *variable name* : *Store* : *Sanitize class* : *Escape/Action class* : *Default* : *Type message violate* }}

Example:

```
{{pId}}
{{pId:FSE}}
{{pId:FSE:digit}}
{{pId::digit}}
{{name:FSE:alnumx:m}}
{{name:::m}}
{{name:FSE:alnumx:m:John Doe}}
{{name::::John Doe}}
{{name:FSE:alnumx:m:John Doe:forbidden characters}}
{{name:::::forbidden characters}}
```

- Zero or more stores might be specified to be searched for the given VarName.
- If no store is specified, the default for the searched stores are: **FSRVD** (=FORM > SIP > RECORD > VARS > DEFAULT).
- If the VarName is not found in one store, the next store is searched, up to the last specified store.
- If the VarName is not found and a default value is given, the default is returned.
- If no value is found, nothing is replaced - the string `{{<VarName>}}` remains.
- If anywhere along the line an empty string is found, this **is** a value: therefore, the search will stop.

5.2 Sanitize class

Values in STORE_CLIENT *C* (Client=Browser) and STORE_FORM *F* (Form, HTTP 'post') are checked against a sanitize class. Values from other stores are *not* checked against any sanitize class, even if a sanitize class is specified.

- Variables get by default the sanitize class defined in the corresponding *FormElement*. If not defined, the default class is `digit`.
- A default sanitize class can be overwritten by individual definition: *{{a:C:alnumx}}*
- If a value violates the specific sanitize class, see *Type message violate* for default or customized message. By default the value becomes *!!<name of sanitize class>!!*. E.g. *!!digit!!*.

For QFQ variables and FormElements:

Only in FormElement:

auto	Form		Only supported for FormElements. Most suitable checktype is dynamically evaluated based on native column definition, the FormElement type, and other info. See below for details.
email	Form	Query	[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}
pattern	Form		Compares the value against a regexp.

Rules for CheckType Auto (by priority):

- TypeAheadSQL or TypeAheadLDAP defined: **alnumx**
- Table definition * integer type: **digit** * floating point number: **numerical**
- FE Type * `password`, `note`: **all** * `editor`, `text` and encode = `specialchar`: **all**
- None of the above: **alnumx**

5.3 Escape/Action class

The following *escape* & *action* types are available:

Token	Description
c	Config - the escape type configured in *Configuration*.
C	Colon `:` will be escaped by `\:`
d	Double ticks `"` will be escaped by `\"`.
l	LDAP search filter values: ldap-escape() (LDAP_ESCAPE_FILTER).
L	LDAP DN values. ldap-escape() (LDAP_ESCAPE_DN).
s	Single ticks `'` will be escaped by `\'`.
S	Stop replace. If the replaced value contains nested variables, they won't be replaced.
m	real_escape_string() (m = mysql)
p	Password hashing: depends on the hashing type in the Typo3 installation, includes salting if configured.
w	wipe out current key/value pair from SIP store *variable-escape-wipe-key*
X	Throw exception if variable is not found in the given store(s). Outputs *Type message violate*
''	Nothing defined - the escape/action class type configured in *Configuration*.
-	No escaping.

- The `escape/action` class is defined by the fourth parameter of the variable. E.g.: `{{name:FE:alnumx:m}}` (m = mysql).
- It's possible to combine multiple `escape/action` classes, they will be processed in the order given. E.g. `{{name:FE:alnumx:Ls}}` (L, s).
- Escaping is typically necessary for all user supplied content, especially if they are processed via SQL or LDAP queries.
- Be careful when escaping nested variables. Best is to escape **only** the most outer variable.
- In *Configuration* a global `escapeTypeDefault` can be defined. The configured `escape/action` class applies to all substituted variables, who *do not* contain a *specific* `escape/action` class.
- Additionally a `defaultEscapeType` can be defined per `Form` (separate field in the *Form editor*). This overwrites the global definition of `configuration`. By default, every `Form.defaultEscapeType` = 'c' (=config), which means the setting in *Configuration*.

- To suppress an escape type, define the `escape type = '-'` on the specific variable. E.g.: `{{name:FE:alnumx:-}}`.

5.3.1 Escape

To *escape* a character typically means: a character, which have a special meaning/function, should not treated as a special character. E.g. a string is surrounded by single ticks `'`. If such a string should contain the same single tick inside, the inside single tick has to be escaped - if not, the string end's at the second tick, not the third. This is typically done by a backlash: \

QFQ offers different ways of escaping. Which of them to use, depends on the situation.

Especially variables used in SQL statements might cause trouble when using: NUL (ASCII 0), \n, \r, \, ', ", or Control-Z.

5.3.2 Action

- *password* - `p`: transforms the value of the variable into a Typo3 salted password hash. The hash function is the one used by Typo3 to encrypt and salt a password. This is useful to manipulate FE user passwords via QFQ. See *Set FE-User password*
- *stop replace* - `S`: typically QFQ will replace nested variables as long as there are variables to replace. This options stops this
- *exception* - `X`: If a variable is not found in any given store, it's replace by a default value or an error message. In special situation it might be useful to do a full stop on all current actions (no further procession). A custom message can be defined via: *Type message violate*.
- *wipe* - `w`: In special cases it might be useful to get a value via SIP only one time and after retrieving the value it will be deleted in STORE SIP . Further access to the variable will return *variable undefined*. At time of writing only the STORE SIP supports the feature *wipe*. This is useful to suppress any repeating events by using the browser history. The following example will send a mail only the first when it is called with a given SIP:

```
10.sql = SELECT '...' AS _sendmail FROM Person AS p WHERE '{{action:S::w}}'='send
↪' AND p.id={{pId:S}}
```

5.4 Default

- Any string can be given to define a default value.
- If a default value is given, it makes no sense to define more than one store: with a default value given, only the first store is considered.
- If the default value contains a `:`, that one needs to be escaped by \
- For dedicated variables this value has a special meaning. E.g. `{{randomUniq:V}}` uses this as `expire` argument.

5.5 Type message violate

If a value violates the sanitize class, the following actions are possible:

- `c` - The violated class will be set as content, surrounded by *!!*. E.g. *!!digit!!*. This is the default.

- `e` - Instead of the value an empty string will be set as content.
- `0` - Instead of the value the string *0* will be set as content.
- *custom text ...* - Instead of the value, the custom text will be set as content. If the text contains a `:`, that one needs to be escaped by `\`. Check *Escape/Action class* qualifier `C` to let QFQ do the colon escaping.

5.5.1 SQL variables

- The detection of an SQL command is case *insensitive*.
- Leading whitespace will be skipped.
- The following commands are interpreted as SQL commands:
  - SELECT
  - INSERT, UPDATE, DELETE, REPLACE, TRUNCATE
  - SHOW, DESCRIBE, EXPLAIN, SET
- An SQL Statement might contain variables, including additional SQL statements. Inner SQL queries will be executed first.
- All variables will be substituted one by one from inner to outer.
- The number of variables inside an input field or an SQL statement is not limited.

5.5.1.1 Result: string

A result of an SQL statement will be imploded over all: concat all columns of a row, concat all rows - there is no glue string.

5.5.1.2 Result: row

A few functions needs more than a returned string, instead separate columns are necessary. To indicate an array result, specify those with an `!`:

```
{{!SELECT ...}}
```

This manual will specify the individual QFQ elements, who needs an array instead of a string. It's an error to return a string where an array is needed and vice versa.

5.5.1.3 Database index

To access different databases in a *Multi Database* setup, the database index can be specified after the opening curly braces.

```
{{[1]SELECT ... }}
```

For using the indexData and indexQfq (:ref:configuration), it's a good practice to specify the variable name instead of the numeric index.

```
{{[{{indexData:Y}}]SELECT ...}}
```

If no dbIndex is given, *{{indexData:Y}}* is used.

5.5.1.4 Example

```
{{SELECT id, name FROM Person}}
{{SELECT id, name, IF({{feUser:T0}}=0, 'Yes', 'No')  FROM Person WHERE id={{r:S}} }}
{{SELECT id, city FROM Address AS adr WHERE adr.accId={{SELECT id FROM Account AS acc␣
↪WHERE acc.name={{feUser:T0}} }} }}
{{!SELECT id, name FROM Person}}
{{[2]SELECT id, name FROM Form}}
{{[{{indexQfq:Y}}]SELECT id, name FROM Form}}
```

5.5.1.5 Row column variables

Syntax: *{{<level>.<column>}}*

Only used in report to access outer columns. See *Access column values* and *Syntax of report*.

There might be name conflicts between VarName / SQL keywords and <line identifier>. QFQ checks first for <*level*>, than for *SQL keywords* and than for *VarNames* in stores.

All types might be nested with each other. There is no limit of nesting variables.

Very specific: Also, it's possible that the content of a variable is again (including curly braces) a variable - this is sometimes used in text templates, where the template is retrieved from a record and specific locations in the text will be (automatically by QFQ) replaced by values from other sources.

General note: using this type of variables is only the second choice. First choice is *{{column:R}}* (see *Access column values*) - using the STORE_RECORD is more portable cause no renumbering is needed if the level keys change.

5.5.2 Link column variables

These variables return a link, completely rendered in HTML. The syntax and all features of *Column: _link* are available. The following code will render a *new person* button:

```
{{p:form&form=Person|s|N|t:new person AS link}}
```

For better reading, the format string might be wrapped in single or double quotes (this is optional):

```
{{"p:form&form=Person|s|N|t:new person" AS link}}
```

These variables are especially helpful in:

- *report*, to create create links or buttons outside of an SQL statement. E.g. in *head*, *rbeg*, ...
- *form*, to create links and buttons in labels or notes.

CHAPTER 6

Security

All values passed to QFQ will be:

- Checked against max. length and allowed content, on the client and on the server side. On the server side, the check happens before any further processing. The 'length' and 'allowed' content is specified per *FormElement*. 'digit' or 'alnumx' is the default. Violating the rules will stop the 'save record' process (Form) or result in an empty value (Report). If a variable is not replaced, check the default sanitize class.
- Only elements defined in the *Form* definition or requested by *Report* will be processed.
- UTF8 normalized (normalizer::normalize) to unify different ways of composing characters. It's more a database interest, to work with unified data.

SQL statements are typically fired as *prepared statements* with separated variables. Further *custom* SQL statements will be defined by the webmaster - those do not use *prepared statements* and might be affected by SQL injection. To prevent SQL injection, every variable is by default escaped with *mysqli::real_escape_string*.

QFQ notice:

- Variables passed by the client (=Browser) are untrusted and use the default sanitize class 'digit' (if nothing else is specified). If alpha characters are submitted, the content violates *digit* and becomes therefore *!!<name of sanitize class>!!* - there is no error message. Best is to always use SIP (value is trustful) or at least digits for GET (=client) parameter (user might change those and therefore those are *not* trustful).

6.1 Get Parameter

QFQ security restriction:

- GET parameter might contain urlencoded content (%xx). Therefore all GET parameter will be processed by 'urldecode()'. As a result a text like '%nn' in GET variables will always be decoded. It's not possible to transfer '%nn' itself.
- GET values are limited to securityGetMaxLength (*Extension Manager: QFQ Configuration*) chars - any violation will stop QFQ. Individual exceptions are defined via *Exception for SECURITY_GET_MAX_LENGTH*.

- GET parameter 'type' and 'L' might affected by (T3, configuration dependent) cache poisoning. If they contain non digit values, only the first character is used (if this is a digit) or completely cleaned (else).

6.2 Post Parameter

Per *FormElement* (HTML input) the default is to *htmlspecialchars()* the input. This means `&<>'"` will be encoded as htmlentity and saved as a htmlentity in the database. In case any of these characters (e.g. for HTML tags) are required, the encoding can be disabled per FormElement: *encode=none* (default is *specialchar*).

During Form load, htmlentities are decoded again.

6.3 $_SERVER

All $_SERVER vars are htmlentities encoded (all, not only specialchars!) .

6.4 Honeypot

Every QFQ Form contains 'honeypot'-HTML input elements (HTML: hidden & readonly). Which of them to use is configured in *Configuration* (default: 'username', 'password' and 'email'). On every start of QFQ (form, report, save, ...), these variables are tested if they are non-empty. In such a case a probably malicious bot has send the request and the request will not be processed.

If any of the default configured variable names are needed (which will never be the case for QFQ), an explicit variable name list have to be configured in *Configuration*.

QFQ security restriction:

- The honeypot variables can't be used in GET or POST as regular HTML input elements - any values of them will terminate QFQ.

6.5 Violation

On any violation, QFQ will sleep *securityAttackDelaySeconds* (*Configuration*) and than exit the running PHP process. A detected attack leads to a complete white (=empty) page.

If *securityShowMessage*: true (*Configuration*), at least a message is displayed after the delay.

6.6 Client Parameter via SIP

Links with URL parameters, targeting to the local website, are typically SIP encoded. Instead of transferring the parameter as part of the URL, only one unique GET parameter 's' is appended at the link. The parameter 's' is unique (equal to a timestamp) for the user. Assigned variables are stored as a part of the PHP user session on the server. Two users might have the same value of parameter 's', but the content is completely independent.

Variables needed by Typo3 remains on the link and are not 'sip-encoded'.

6.7 Secure direct file access

If the application uploads files, mostly it's not necessary and often a security issue, to offer a direct download of the uploaded files. Best is to create a directory, e.g. *<site path>/fileadmin/protected* and deny direct access via webbrowser to it. E.g. for Apache set a rule:

```
<Directory "/var/www/html/fileadmin/protected">
    Require all denied
</Directory>
```

If you only have access to *.htaccess*, create a file *<site path>/fileadmin/protected/.htaccess* with:

```
<IfModule mod_authz_core.c>
     Require all denied
</IfModule>
```

Important: All QFQ uploads should save files only in/below such a protected directory.

To offer download of those files, use the reserved column name '_download' (see *Download*) or variants.

Important: To protect the installation against executing of uploaded malicious script code, disable PHP for the final upload directory. E.g. *fileadmin* (Apache):

```
<Directory "/var/www/html/fileadmin">
  php_admin_flag engine Off
</Directory>
```

This is in general a good security improvement for directories with user supplied content.

6.8 File upload

By default the mime type of every uploaded file is checked against a white list of allowed mime types. The mime type of a file can be (easily) faked by an attacker. This check is good to handle regular user file upload for specific file types but won't help to prevent attacks against uploading and executing malicious code.

Instead prohibit the execution of user contributed files by the webserver config (*SecureDirectFileAccess*).

6.9 Typo3 Setup - best practice

- Activate notification emails for every BE login (if there are only few BE users). In case the backend has been hacked, unusual login's (time or username) will appear:

```
[BE][warning_email_addr] = <your email>
[BE][warning_mode] = 1
```

CHAPTER 7

Store

Only variables that are known in a specified store can be substituted.

Name	Description	Content
B	*Store: BEFORE - B*: Record - the current record loaded in the form before any update.	All columns of the current record from the current table. See *Store: BEFORE - B*.
C	*Store: CLIENT - C*: POST variable, if not found: GET variable.	Parameter sent from the Client (=Browser). See *Store: CLIENT - C*.
D	Default values column : The *table.column* specified *default value*.	
E	*Empty* - always an empty string, might be helpful if a variable is empty or undefined and will be used in an SQL statement.	Any key
F	*Store: FORM - F*: data not saved in database yet.	All native *FormElements*. Recent values from the Browser. See: *Store: FORM - F*
L	*Store: LDAP - L*: Will be filled on demand during processing of a *FormElement*.	Custom specified list of LDAP attributes. See *Store: LDAP - L*.
M	Column type: The *table.column* specified *type*.	
P	Parent record. E.g.: on multi & copy forms the current record of the outer query.	All columns of the MultiSQL Statement from the table for the current row
R	*Store: RECORD - R*: Record - the current record loaded in the form.	All columns of the current record from the current table. See *Store: RECORD - R*.
S	*Store: SIP - S*: Client parameter 's' will indicate the current SIP, which will be loaded from the SESSION repo to the SIP-Store.	sip, r (recordId), form. See *Store: SIP - S*.
T	*Store: TYPO3 (Bodytext) - T*: a) Bodytext (ttcontent record), b) Typo3 internal variables.	See Typo3 tt_content record configuration. See *STORE_TYPO3*.
U	*Store: USER - U*: per user variables, valid as long as the browser session lives.	Set via report: '...' AS '_=<var name>' See: *Store: USER - U*, *STORE_USER examples*
V	*Store: VARS - V*: Generic variables.	See *Store: VARS - V*.
Y	*Store: SYSTEM - Y*: a) Database, b) helper vars for logging/debugging: SYSTEM_SQL_RAW ... SYSTEM_FORM_ELEMENT_COLUMN, c) Any custom fields: CONTACT, HELP, ...	See *Store: SYSTEM - Y*.
0	*Zero* - always value: 0, might be helpful if a variable is empty or undefined and will be used in an SQL statement.	Any key

- Default <*prio*>: *FSRVD* - Form / SIP / Record / Vars / Table definition.
- Hint: Preferable, parameter should be submitted by SIP, not by Client (=URL).
  - Warning: Data submitted via 'Client' can be easily spoofed and altered.
  - Best: Data submitted via SIP never leaves the server, cannot be spoofed or altered by the user.
  - SIPs can _only_ be defined by using *Report*. Inside of *Report* use columns 'Link' (with attribute 's'), 'page?' or 'Page?'.

7.1 Store: *FORM* - F

- Sanitized: *yes*
- Represents the values in the form, typically before saving them.
- Used for:

- *FormElements* which will be rerendered, after a parent *FormElement* has been changed by the user.
- *FormElement* actions, before saving the form.
- Values will be sanitized by the class configured in corresponding the *FormElement*. By default, the sanitize class is *alnumx*.

Name	Explanation
<FormElement name>	Name of native *FormElement*. To get, exactly and only, the specified *FormElement* (for 'pId'): *{{pId:F}}*

7.2 Store: *SIP* - S

- Sanitized: *no*
- Filled automatically by creating links. E.g.:
  - in *Report* by using *_page?* or *_link* (with active 's')
  - in *Form* by using subrecords: 'new', 'edit', 'delete' links (system) or by column type *_page?*, *_link*.

Name	Explanation
sip	13 char uniqid
r	current record id
form	current form name
table	current table name
urlparam	all non Typo3 parameter in one string
<user defined>	additional user defined link parameter

7.3 Store: *RECORD* - R

- Sanitized: *no*
- *Form*: Current record.
- *Report*: See *Access column values*
- If r=0, all values are empty.

Name	Type | Explanation	
<column name>	Form	Name of a column of the primary table (as defined in the current form). Example: *{{pId:R}}*
<column name>	Report	Name of a column of a previous fired SQL query. Example: *{{pId:R}}*
&<column name>	Report (final)	Name of a column of a previous fired SQL query, typically used by columns with a *Special column names*. Final value. Example: '{{link:R}}' returns 'p:home&form=Person|s|b:success|t:Edit'. Whereas '{{&link:R}}' returns 'Edit

7.4 Store: *BEFORE* - B

- Sanitized: *no*
- Current record loaded in Form without any modification.
- If r=0, all values are empty.

This store is handy to compare new and old values of a form.

Name	Explanation
<column name>	Name of a column of the primary table (as defined in the current form). To get, exactly and only, the specified form *FormElement*: *{{pId:B}}*

7.5 Store: *CLIENT* - C

- Sanitized: *yes*

Name	Explanation
s	=SIP
r	record id. Only if specified as GET parameter - typically stored in SIP (=STORE_SIP)
form	Name of form to load. Only if specified as GET parameter - typically stored in SIP (=STORE_SIP)
HTTP_client_host	HTTP HOST
REMOTE_ADDR	Client IP address
All HTTP Header variables	SERVER_ADDR, SERVER_NAME, SERVER_SOFTWARE,SERVER_PROTOCOL, REQUEST_METHOD, REQUEST_TIME, REQUEST_TIME_FLOAT, QUERY_STRING, DOCUMENT_ROOT, HTTP_ACCEPT, HTTP_ACCEPT_CHARSET, HTTP_ACCEPT_ENCODING, HTTP_ACCEPT_LANGUAGE, HTTP_CONNECTION, HTTP_HOST, HTTP_REFERER, HTTP_USER_AGENT, HTTPS, REMOTE_ADDR, REMOTE_HOST, REMOTE_PORT, REMOTE_USER, REDIRECT_REMOTE_USER, SCRIPT_FILENAME, SERVER_ADMIN, SERVER_PORT, SERVER_SIGNATURE, PATH_TRANSLATED, SCRIPT_NAME, REQUEST_URI, PHP_AUTH_DIGEST, PHP_AUTH_USER, PHP_AUTH_PW, AUTH_TYPE, PATH_INFO, ORIG_PATH_INFO
Authorization	Value of the HTTP Header 'Authorization'. This is typically not set. Mostly used for authentication of REST requests

7.6 Store: *TYPO3* (Bodytext) - T

- Sanitized: *no*

Name	Explanation	Note
form	Formname defined in ttcontent record bodytext * Fix. E.g. *form = person* * via SIP. E.g. *form = {{form:SE}}*	see note
pageId	Record id of current Typo3 page	see note
pageAlias	Alias of current Typo3 page. If empty, take pageId.	see note
pageTitle	Title of current Typo3 page	see note
pageType	Current selected page type (typically URL parameter 'type')	see note
pageLanguage	Current selected page language (typically URL parameter 'L')	see note
ttcontentUid	Record id of current Typo3 content element	see note
feUser	Logged in Typo3 FE User	
feUserUid	Logged in Typo3 FE User uid	
feUserGroup	FE groups of logged in Typo3 FE User	
beUser	Logged in Typo3 BE User	
beUserLoggedIn	'yes' | 'no' - Status if a BE-User is logged in	

- **note**: not available:
  - in *Dynamic Update* or
  - by *FormElement* class 'action' with type 'beforeSave', 'afterSave', 'beforeDelete', 'afterDelete'.

7.7 Store: *VARS* - V

- Sanitized: *no*

Name	Explanation
random	Random string with length of 32 alphanum chars (lower & upper case). This variable is always filled. Each access gives a new value. Remember: QFQ variables will be replaced **before** a SQL statement is fired. Uniqueness is given via PHP function *uniqid()*.
slaveId	see *Parameter: slaveId*
allRequiredGiven	Form save - Set during check of all required FE. If every *required* FE is given: *1*. Else: *0*. If there is no required FE: *1*. Even with *formModeGlobal = requiredOff | requiredOffButMark* the variable will be set.

- FormElement 'upload':

Name	Explanation
filename	Original filename of an uploaded file via an 'upload'-FormElement. Valid only during processing of the current 'upload'-formElement.
filename-Only	Like filename, but without path.
filename-Base	Like *filename*, but without an optional extension. E.g. filename='image.png' comes to filename-Base='image'
filename-Ext	Like *filename*, but only the optional extension. E.g. filename='image.png' comes to filename-Ext='png'
fileDesti-nation	Destination (path & filename) for an uploaded file. Defined in an 'upload'-FormElement.parameter. Valid: same as 'filename'.
fileSize	Size of the uploaded file.
mimeType	Mime type of the uploaded file.

The directive *fillStoreVar* will fill the store VARS with custom values. Existing Store VARS values will be merged together with them. E.g.:

```
fillStoreVar = {{! SELECT p.name, p.email FROM Person AS p WHERE p.id={{pId:S}} }}
```

- After filling the store, the values can be retrieved via *{{name:V}}* and *{{email:V}}*.
- Be careful by specifying general purpose variables like *id*, *r*, *pageId* and so on. This might conflict with existing variables.
- *fillStoreVar* can be used in *Parameter* and *Attributes defined in the parameter field*

7.8 Store: *LDAP* - L

- Sanitized: *yes*
- See also *LDAP*:

Name	Explanation
<custom defined>	See *ldapAttributes*

7.9 Store: *SYSTEM* - Y

- Sanitized: *no*

See *Configuration* for a list of all settings.

7.10 Store: *USER* - U

- Sanitized: *no*

At start of a new browser session (=user calls the website the first time or was logged out before) the store is empty. As soon as a value is set in the store, it remains as long as the browser session is alive (=until user logs out).

Values can be set via report '... AS "_=<var name>"'

See also: *STORE_USER examples*

CHAPTER 8

LDAP

A form can retrieve data from LDAP server(s) to display or to save them. Configuration options for LDAP will be specified in the *parameter* field of the *Form* and/or the *FormElement*. Definitions of the *FormElement* will overwrite definitions of the *Form*. One LDAP Server can be configured per *FormElement*. Multiple *FormElements* might use individual LDAP Server configurations.

To decide which Parameter should be placed on *Form.parameter* and which on *FormElement.parameter*: If LDAP access is ...

- only necessary in one *FormElement*, most useful setup is to specify all values in that specific *FormElement*,
- needed on multiple *FormElement*s (of the same *Form*, e.g. one *input* with *typeAhead*, one *note* and one *action*), it's more efficient to specify the base parameter *ldapServer*, *ldapBaseDn* in *Form.parameter* and the rest on the current *FormElement*.

Parameter	Example	Description	Form	FormElement	Used for
ldapServer	ldaps://directory.example.com:636	Host name	x	x	TA, FSL
ldapBaseDn	ou=Addressbook,dc=example,dc=com	Base DN to start the search	x	x	TA, FSL
ldapAttributes	cn, email	List of attributes to save in STORE_LDAP	x	x	FSL
ldapSearch	(mail=john.doe@example.com)	Regular LDAP search expression	x	x	FSL
ldapTimeLimit	3 (default)	Maximum time to wait for an answer of the LDAP Server	x	x	TA, FSL
ldapUseBindCredentials	ldapUseBindCredentials=1	Use LDAP_1_* credentials from *config.qfq.php* for ldap_bind()	x	x	TA, FSL
typeAheadLdap		Enable LDAP as 'Typeahead' data source		x	TA
typeAheadLdapSearch	*(|(cn=*?*)(mail=*?*))*	Regular LDAP search expression, returns upto typeAheadLimit	x	x	TA
typeAheadLdapSearchOnlyExact	*(mail=?)*	Regular LDAP search expression, typically return one record	x	x	TA
typeAheadLdapSearchPerToken		**Split search value in token and OR-combine every search with** the individual tokens	x	x	TA
typeAheadLdapValueFormat	*'%s', cn, mail*	Custom format to display attributes, as *value*	x	x	TA
typeAheadLdapIdFormat	*'%s', mail*	Custom format to display attributes, as *id*	x	x	TA
typeAheadLimit	20 (default)	Result will be limited to this number of entries	x	x	TA
typeAheadPedantic	typeAheadPedantic=0	Turn off 'pedantic' mode - allow any values (see below)	x	x	TA
typeAheadMinLength	2 (default)	Minimum number of characters before starting the search	x	x	TA
fillStoreLdap		Activate *Fill STORE LDAP* with the first		x	FSL

- *typeAheadLimit*: there might be a hard limit on the server side (e.g. 100) - which can't be extended.
- *ldapUseBindCredentials* is only necessary if *anonymous* access is not possible. RDN and password has to be configured in *config.qfq.php*.

8.1 Typeahead (TA) - LDAP

See also *Type Ahead*

Typeahead offers continuous searching of a LDAP directoy by using a regular *FormElement* of type *text*. The *FormElement.parameter*=*typeAheadLdap* will trigger LDAP searches on every user **keystroke** (starting after *typeAheadMinLength* keystrokes) for the current *FormElement* - this is different from *dynamicUpdate* (triggered by leaving focus of an input element). Typeahead delivers a list of elements.

- *FormElement.parameter.typeAheadLdap* - activate the mode *Typeahead* - no value is needed, the existence is sufficient.
- *Form.parameter* or *FormElement.parameter*:
  - *ldapServer = ldaps://directory.example.com:636*
  - *ldapBaseDn = ou=Addressbook,dc=example,dc=com*
  - *typeAheadLdapSearch = (|(cn=*?*)(mail=*?*))*
  - *typeAheadLdapValuePrintf = '%s / %s', cn, email*
  - *typeAheadLdapIdPrintf = '%s', email*
  - Optional: *ldapUseBindCredentials* = 1

All fetched LDAP values will be formatted with:

- *typeAheadLdapValuePrintf*, shown to the user in a drop-down box and
- *typeAheadLdapIdPrintf*, which represents the final data to save.

The *id/value* translation is compareable to a regular select drop-down box with id/value pairs. Only attributes, defined in *typeAheadLdapValuePrintf* / *typeAheadLdapIdPrintf* will be fetched from the LDAP directory. To examine all possible values of an LDAP server, use the commandline tool *ldapsearch*. E.g.:

```
ldapsearch -x -h directory.example.com -L -b ou=Addressbook,dc=example,dc=com
↪"(mail=john.doe@example.com)"
```

All occurrences of a '?' in *ldapSearch* will be replaced by the user data typed in via the text-*FormElement*. The typed data will be escaped to fulfill LDAP search limitations. Regular *Form* variables might be used on all parameter and will be evaluated during form load (!) - *not* at the time when the user types something.

8.1.1 Pedantic

The *typeAheadPedantic* mode ensures that the typed value (technically this is the value of the *id*, latest in the moment when loosing the focus) is valid (= exists on the LDAP server or is defined in *typeAheadSql*). If the user typed something and that is not a valid *id*, the client (=browser) will delete the input when losing the focus. To identify the exact *id*, an additional search filter is necessary: *typeAheadLdapSearchPrefetch* - see next topic.

typeAheadPedantic is active by default when *typeAheadLdap* or *typeAheadSql* is defined, but can be turned off with *typeAheadPedantic=0*.

- *Form.parameter* or *FormElement.parameter*:

  - *typeAheadPedantic=0*

8.1.2 Prefetch

After 'form load' with an existing record, the user expects to see the previous saved data. In case there is an *id* to *value* translation, the *value* does not exist in the database, instead it has to be fetched again dynamically. A precise LDAP or SQL query has to be defined to force this:

- *Form.parameter* or *FormElement.parameter*:
  - *typeAheadLdapSearchPrefetch = (mail=?)*
  - *typeAheadSqlPrefetch = SELECT firstName, ' ', lastName FROM Person WHERE id = ?*

This situation also applies in *pedantic* mode to verify the user input after each change.

8.1.3 PerToken

Sometimes a LDAP server only provides attributes like 'sn' and 'givenName', but not 'displayName' or another practical combination of multiple attributes - than it is difficult to search for 'firstname' *and* (=human AND) 'lastname'. E.g. 'John Doe', results to search like *(|(sn=*John Doe*)(givenName=*John Doe*))* which will be probably always be empty. Instead, the user input has to be split in token and the search string has to repeated for every token.

- *Form.parameter* or *FormElement.parameter*:
  - *typeAheadLdapSearchPerToken* - no value needed.

This will repeat the search string per token.

E.g.:

```
User search string: X Y
Ldap search string: (|(a=*?*)(b=*?*))

Result: (& (|(a=*X*)(b=*X*)) (|(a=*Y*)(b=*Y*))
```

Attention: this option is only useful in specific environments. Only use it, if it is really needed. The query becomes much more cpu / IO intensive.

8.2 Fill STORE LDAP (FSL)

Before processing a *FormElement*, an optional configured FSL-action loads **one** record from a LDAP directory and stores the named attributes in STORE_LDAP. If the LDAP search query selects more than one record, only the first record is processed. The attributes names always becomes lowercase (PHP implentation detail on get_ldap_entries()) in the store. To make accessing STORE_LDAP easily, the keys are implemented case insensitive for this specific store. FLS is triggered during *Form-...*

- load,
- dynamic update,
- save.

The FLS happens *before* the *FormElement* processing starts. Therefore the fetched LDAP data (specified by *ldapAttributes*), are available via *{{<attributename>:L:allbut:s}}* during the regular *FormElement* processing. Take care to specify a sanitize class and optional escaping on further processing of those data.

Also, the STORE LDAP remains filled, during the whole form processing time. E.g. if the values are needed for a person name and email, it's sufficient to fire one FSL on the first FormElement Action element, and use the same values during further FormElement Action Elements.

Important: LDAP access might slow down the *Form* processing on load, update or save! The timeout (default: 3 seconds) have to be multiplied by the number of accesses. E.g. a broken LDAP connection and 3 *FormElements* with *FSL* results to 9 seconds delay on save. Also be prepared not to receive the expected data.

- *FormElement.parameter.fillStoreLdap* - activate the mode *Fill S* - no value is needed, the existence is sufficient.
- *Form.parameter* or *FormElement.parameter*:
  - *ldapServer = ldaps://directory.example.com:636*
  - *ldapBaseDn = ou=Addressbook,dc=example,dc=com*
  - *typeAheadLdapSearch = (|(cn=*?*)(mail=*?*))*
  - *ldapAttributes = givenName, sn, telephoneNumber, email*
  - *ldapSearch = (mail={{email:F0:alnumx:l}})*
  - Optional: *ldapUseBindCredentials* = 1

After filling the store, access the content via *{{<attributename>:L:allbut:s}}*.

CHAPTER 9

Form

9.1 General

- Forms will be created by using the *Form Editor* on the Typo3 frontend (HTML form).
- The *Form editor* itself consist of two predefined QFQ forms: *form* and *formElement* - these forms are often updated during the installation of new QFQ versions.
- Every form consist of a) a *Form* record and b) multiple *FormElement* records.
- A form is assigned to a *table*. Such a table is called the *primary table* for this form.
- Forms can roughly categorized into:
  - *Simple* form: the form acts on one record, stored in one table.
    - The form will create necessary SQL commands for insert, update and delete (only primary record) automatically.
  - *Advanced* form: the form acts on multiple records, stored in more than one table.
    - Fields of the primary table acts like a *simple* form, all other fields have to be specified with *action/afterSave* records.
  - *Multi* form: (*multi-form*) the form acts simultaneously on more than one record. All records use the same *FormElements*.
    - The *FormElements* are defined as a regular *simple* / or *advanced* form, plus an SQL Query, which selects and iterates over all records. Those records will be loaded at the same time.
  - *Delete* form: any form can be used as a form to *Delete Record*.
- Form mode: The parameter 'r' (given via URL or via SIP) decide if the form is in mode:
  - *New*:
    - Missing parameter 'r' or 'r=0'
    - On form load, no record is displayed.

        * Saving the form creates a new primary record.
        * E.g.: *http://example.com/index.php?id=home&form=Person* or *http://example.com/index.php?id=home&form=Person*
    - *Edit*:
        * Parameter 'r>0'
        * On form load, the specified record (<tablename>.id= <r>) will be loaded and displayed.
        * Saving the form will update the existing record.
        * E.g.: *http://example.com/index.php?id=home&form=Person&r=123*
    - Providing additional parameter:

      Often, it is necessary to store additional, for the user not visible, parameter in a record. See *Form Magic*.
- Display a form:
    - Create a QFQ tt_content record on a Typo 3 page.
    - Inside the QFQ record: *form* = <*formname*>. E.g.:
        * Static: *form* = *Person*
        * Dynamic 1: *form* = *{{form:SE}}* (the left *form* is a keyword for QFQ, the right *form* is a variable name)
        * Dynamic 2: *form* = *{{SELECT f.name FROM Form AS f WHERE f.id=... }}* (the left *form* is a keyword for QFQ, the right *form* is a variable name)
    - With the *Dynamic* option, it's easily possible to use one Typo3 page and display different forms on that specific page.

9.2 Form process order

Depending on *form load / save* or *record delete*, different tasks are performed. Processing is divided into:

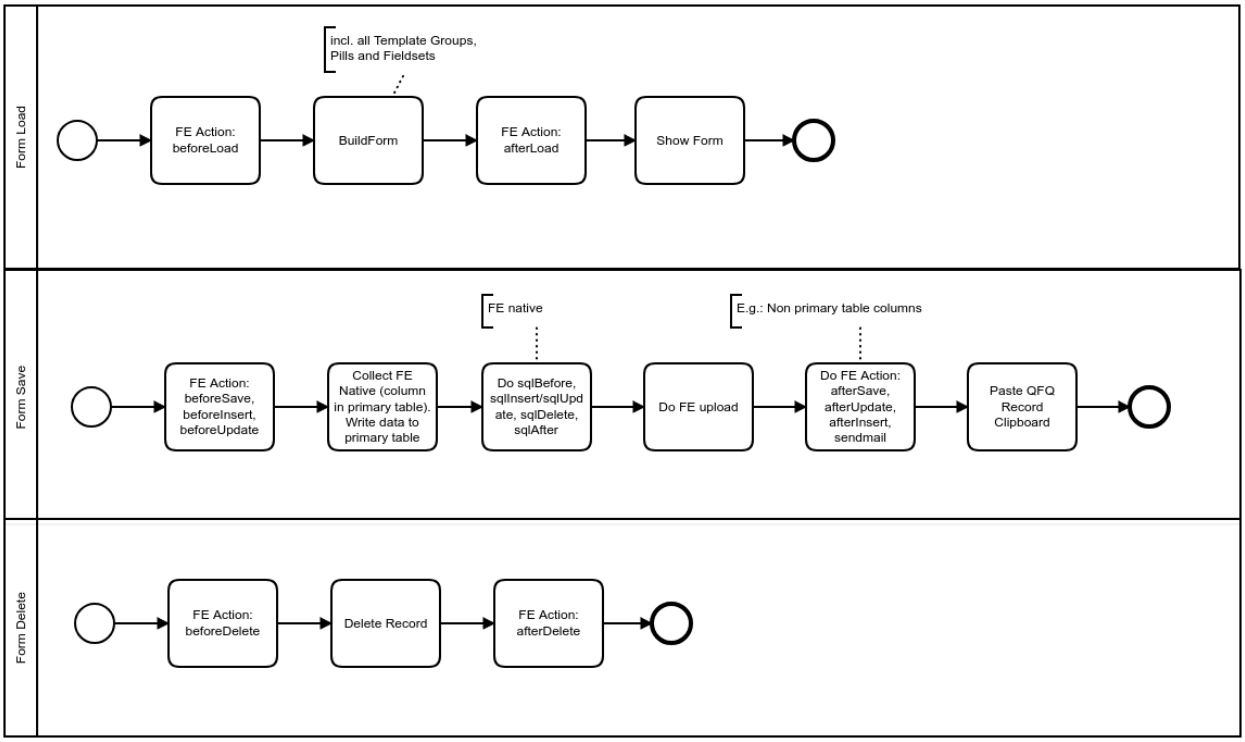
- Native FormElements like: *input*, *select list*, *checkbox*, *radio*, ...
    - *upload* elements are categorized as native FormElement, but will be processed later.
    - *pill*, *fieldset* and *subrecord* elements are only processed during form load, they do not impact *save* or *delete*.
- Action FormElement like *before...*, *after...*, *sendmail* ...

Each FormElement has an order.

Native FormElements which 'name':

- corresponds to a column in the form primary table: are grouped together in one insert or update query.
- do not correspond to a column in the primary table: needs an explicit defined Action FormElement to be handled.

FormElement processing order:



9.3 Record locking

Forms and 'record delete'-function support basic record locking. A user opens a form: starting with the first change of content, a record lock will be acquired in the background. If the lock is denied (e.g. another user already owns a lock on the record) an alert is shown. By default the record lock mode is 'exclusive' and the default timeout is 15 minutes. Both can be configured per form. The general timeout can also be configured in *Configuration* (will be copied to the form during creating the form).

The lock timeout counts from the first change, not from the last modification on the form.

If a timeout expires, the lock becomes invalid. During the next change in a form, the lock is acquired again.

A lock is assigned to a record of a table. Multiple forms, with the same primary table, uses the same lock for a given record.

If a *Form* acts on further records (e.g. via FE action), those further records are not protected by this basic record locking.

If a user tries to delete a record and another user already owns a lock on that record, the delete action is denied.

If there are different locking modes in multiple forms, the most restricting mode applies for the current lock.

9.3.1 Exclusive

An existing lock on a record forbids any write action on that record.

9.3.2 Advisory

An existing lock on a record informs the user that another user is currently working on that record. Nevertheless, writing is allowed.

9.3.3 None

No locking at all.

9.4 Comment- and space-character

The following applies to the fields *Form.parameter* and *FormElement.parameter*:

- Lines will be trimmed - leading and trailing spaces will be removed.
- If a leading and/or trailing space is needed, escape it: \ hello world \ > ‘ hello world ‘.
- Lines starting with a ‘#’ are treated as a comment and will not be parsed. Such lines are treated as ‘empty lines’.
- The comment sign can be escaped with \ .

9.5 Form Settings

Name	Description
Name	Unique and speaking name of the *Form*. Form will be identified by this name. Use only '[a-zA-Z0-9._+-]'. form-name
Title	Title, shown on/above the form. form-title
Note	Personal editor notes. form-note
Table	Primary table of the form. form-tablename
Primary Key	Primary key of the indicated table. Only needed if != 'id'. form-primary-key
Required Parameter NEW	Name of required SIP parameter to create a new record (r=0), separated by comma. '#' as comment delimiter. See *Required Parameter New\Edit*
Required Parameter EDIT	Name of required SIP parameter to edit an existing record (r>0), separated by comma. '#' as comment delimiter. See *Required Parameter New\Edit*
Permit New	'sip, logged_in, logged_out, always, never' (Default: sip): See *permitNew & permitEdit*
Permit Edit	'sip, logged_in, logged_out, always, never' (Default: sip): See *permitNew & permitEdit*
Permit REST	See *REST*
Escape Type Default	See *Escape/Action class*.
Show button	'new, delete, close, save' (Default: 'new,delete,close,save'): Shown named buttons in the upper right corner of the form. See *showButton*
Forward Mode	'auto | close | no | url | url-skip-history | url-sip | url-sip-skip-history' (Default: auto): See *Forward: Save / Close*.
Forward (Mode) Page	a) URL / Typo3 page id/alias or b) Forward Mode (via '{{...}}') or combination of a) & b). See *Forward: Save / Close*.
labelAlign	Label align (default/left/center/right)/ Default: 'default' (defined by Config). form-label-align
Parameter	Misc additional parameters. See *Parameter*.
BS Label Columns	The bootstrap grid system is based on 12 columns. The sum of *bsLabelColumns*, *bsInputColumns* and *bsNoteColumns* should be 12. These values here are the base values for all *FormElements*. Exceptions per *FormElement* can be specified per *FormElement*. Default: label=col-md-3, input=col-md-6, note=col-md-3. See *Form Layout*.
BS Input Columns	
BS Note Columns	
multiMode	NOT IMPLEMENTED - 'none, horizontal, vertical' (Default 'none')
multiSql	NOT IMPLEMENTED - Optional. SQL Query which selects all records to edit.
multiDetailForm	NOT IMPLEMENTED - Optional. Form to open, if a record is selected to edit (double click on record line)
multiDetailFormParameter	NOT IMPLEMENTED - Optional. Translated Parameter submitted to detailform (like subrecord parameter)

9.5.1 permitNew & permitEdit

Depending on *r*, the following access permission will be taken:

- *r=0* - permitNew
- *r>0* - permitEdit

Option	Description
sip	The parameter 'form' and 'r' must be supplied via SIP or hard coded in the QFQ tt_content record.
logged_in	The form will only be shown if the current User is logged in as a FE User
logged_out	The form will only be shown if the current User is *not* logged in as a FE User
always	No access restriction, the form will always be shown
never	The form will never be shown

- *sip*
  - is *always* the preferred way. With 'sip' it's not necessary to differ between logged in or not, cause the SIP only exist and is only valid, if it's created via QFQ/report earlier. This means 'creating' the SIP implies 'access granted'. The grant will be revoked when the QFQ session is destroyed - this happens when a user loggs out or the web browser is closed.
- *logged_in* / *logged_out*: for forms which might be displayed without a SIP, but maybe on a protected or even unprotected page. *The option is probably not often used.*
- *always*: such a form is always allowed to be loaded.
  - *permitNew=always*: Public accessible forms (e.g. for registration) will allow users to fill and save the form.
  - *permitEdit=always*: Public accessible forms will allow users to update existing data. This is dangerous, cause the URL parameter (recordId) 'r' might be changed by the user (URL manipulating) and therefore the user might see and/or change data from other users. *The option is probably not often used.*
- *never*: such a form is not allowed to be loaded.
  - *permitNew=never*: Public accessible forms. It's not possible to create new records.
  - *permitEdit=none*: Public accessible forms. It's not possible to update records.

9.5.2 Required Parameter New|Edit

Comma separated list of variable names. On form load, an error message will be shown in case of missing parameters. The parameters must be given by SIP.

The list of required parameter has to be defined for *New* (r=0, create a new record) and for *Edit* (r>0, edit existing record).

Optional a comment might be attached after the parameter definition.

E.g.:

```
New: grId, pId # Always specify a person, grId2
Edit: pId
```

9.5.3 showButton

Display or hide the button *new*, *delete*, *close*, *save*.

- *new*: Creates a new record. If the form needs any special parameter via SIP or Client (=browser), hide this 'new' button - the necessary parameter are not provided.
- *delete*: This either deletes the current record only, or (if defined via action *FormElement* 'beforeDelete') any specified subrecords.
- *close*: Close the current form. If there are changes, a popup opens and ask to save / close / cancel. The last page from the history will be shown.
- *save*: Save the form.
- Default: show all buttons.

9.5.4 Forward: Save / Close

9.5.4.1 Forward (=forwardMode)

After the user presses *Save*, *Close*, *Delete* or *New*, different actions are possible where the browser redirects to.

- *auto* (default) - the QFQ browser Javascript logic, decides to stay on the page or to force a redirection to a previous page. The decision depends on:
  - *Close* goes back (feels like close) to the previous page. Note: if there is no history, QFQ won't close the tab, instead a message is shown.
  - *Save* stays on the current page.
- *close* - goes back (feels like close) to the previous page. Note: if there is no history, QFQ won't close the tab, instead a message is shown.
- *no* - no change, the browser remains on the current side. Close does not close the page. It just triggers a save if there are modified data.
- *url* - the browser redirects to the URL or T3 page named in *Forward URL / Page*. Independent if the user presses *save* or *close*.
- *url-skip-history* - same as *url*, but the current location won't saved in the browser history.
- *url-sip* - like *url*, but any given parameter will be SIP encoded. Only useful if *url* points to current web instance.
- *url-sip-skip-history* - like *url-sip*, but skips the Browser history.

Only with *Forward* == *url* | *url-skip-history*, the definition of *Forward URL / Page* becomes active.

9.5.4.2 Forward URL / Page (=forwardPage)

Format: [<url>] or [<mode>|<url>]

- *<url>*:
  - *http://www.example.com/index.html?a=123#bottom*
  - *website.html?a=123#bottom*
  - *?id=<T3 Alias pageid>&a=123#bottom, ?id=<T3 page id>&a=123#bottom*
  - *{{SELECT . . . }}*
  - *<mode>|<url>*
- *<mode>* - Valid keywords are as above: *auto|close|no|url|url-skip-history|url-sip|url-sip-skip-history*

Specifying the mode in *forwardPage* overwrites *formMode* (but only if *formMode* is *url...*).

Also regular QFQ statements like {{var}} or {{SELECT ...}} are possible in *forwardPage*. This is useful to dynamically redirect to different targets, depending on user input or any other dependencies.

If a forwardMode 'url...' is specified and there is no *forwardPage*, QFQ falls down to *auto* mode.

On a form, the user might click 'save' or 'save,close' or 'close' (with modified data this leads to 'save,close'). The CLIENT *submit_reason* shows the user action:

- *{{submit_reason:CE:alnumx}} = save* or *save,close*

9.5.5 Example forwardPage

- *{{SELECT IF('{{formModeGlobal:S}}'='requiredOff', 'no', 'auto') }}*
- *{{SELECT IF('{{submit_reason:CE:alnumx}}'='save', 'no', 'url'), '\http://example.com' }}*

9.5.6 Type: combined dynamic mode & URL/page

Syntax: *forwardPage=<mode>|<page>*

- *forwardPage={{SELECT IF(a.url='','no','url'), 'l', a.url FROM Address AS a }}*

9.5.7 Parameter

- The following parameter are optional and can be configured in the *Form.parameter* field.

Name	Type	Description
dbIndex	int	Database credential index, given via *config.qfq.php* to let the current *Form* operate on the
bsColumns	string	Wrap the whole form in ' . See *Custom field width*.
maxVisiblePill	int	Show pills upto <maxVisiblePill> as button, all further in a drop-down menu. Eg.: maxVi
class	string	HTML div with given class, surrounding the whole form. Eg.: class=container-fluid.
classTitle	string	CSS class inside of the *title* div. Default 'qfq-form-title'.
classPill	string	HTML div with given class, surrounding the *pill* title line.
classBody	string	HTML div with given class, surrounding all *FormElement*.
extraDeleteForm	string	Name of a form which specifies how to delete the primary record and optional slave recor
data-pattern-error	string	Pattern violation: Text for error message used for all FormElements of current form.
data-required-error	string	Required violation: Text for error message used for all FormElements of current form.
data-match-error	string	Match violation: Text for error message used for all FormElements of current form.
data-error	string	If none specific is defined: Text for error message used for all FormElements of current fo
buttonOnChangeClass	string	Color for save button after user modified some content or current form. E.g.: 'btn-info ale
ldapServer	string	FQDN Ldap Server. E.g.: directory.example.com.
ldapBaseDn	string	E.g.: *ou=Addressbook,dc=example,dc=com*.
ldapAttributes	string	List of attributes to fill STORE_LDAP with. E.g.: cn, email.
ldapSearch	string	E.g.: *(mail={{email::alnumx:l}})*
ldapTimeLimit	int	Maximum time to wait for an answer of the LDAP Server.
typeAheadLdap		Enable LDAP as 'Typeahead' data source.
typeAheadLdapSearch	string	Regular LDAP search expression. E.g.: *(|(cn=*?*)(mail=*?*))*
typeAheadLdapValuePrintf	string	Value formatting of LDAP result, per entry. E.g.: *'%s / %s / %s', mail, roomnumber, telep*
typeAheadLdapIdPrintf	string	Key formatting of LDAP result, per entry. E.g.: *'%s', mail*
typeAheadLdapSearchPerToken		Split search value in token and OR-combine every search with the individual tokens.

Table 1 – continued from previous page

Name	Type	Description
typeAheadLimit	int	Maximum number of entries. The limit is applied to the server (LDAP or SQL) and the Cl
typeAheadMinLength	int	Minimum number of characters which have to typed to start the search.
mode	string	The value *readonly* will activate a global readonly mode of the form - the user can't chang
activateFirstRequiredTab	digit	0: off, 1: (default) - with formModeGlobal=requiredOffButMark bring pill to front on sav
enterAsSubmit	digit	0: off, 1: Pressing *enter* in a form means *save* and *close*. Takes default from *Configuratio*
submitButtonText	string	Show a save button at the bottom of the form, with <submitButtonText> . See *submitButt*
saveButtonActive		0: off, 1: Make the 'save'-button active on *Form* load (instead of waiting for the first user
saveButtonText	string	Overwrite default from *Configuration*
saveButtonTooltip	string	Overwrite default from *Configuration*
saveButtonClass	string	Overwrite default from *Configuration*
saveButtonGlyphIcon	string	Overwrite default from *Configuration*
closeButtonText	string	Overwrite default from *Configuration*
closeButtonTooltip	string	Overwrite default from *Configuration*
closeButtonClass	string	Overwrite default from *Configuration*
closeButtonGlyphIcon	string	Overwrite default from *Configuration*
deleteButtonText	string	Overwrite default from *Configuration*
deleteButtonTooltip	string	Overwrite default from *Configuration*
deleteButtonClass	string	Overwrite default from *Configuration*
deleteButtonGlyphIcon	string	Overwrite default from *Configuration*
newButtonText	string	Overwrite default from *Configuration*
newButtonTooltip	string	Overwrite default from *Configuration*
newButtonClass	string	Overwrite default from *Configuration*
newButtonGlyphIcon	string	Overwrite default from *Configuration*
extraButtonInfoClass	string	Overwrite default from *Configuration*
extraButtonInfoMinWidth	string	See *extraButtonInfo*
fillStoreVar	string	Fill the STORE_VAR with custom values. See *Store: VARS - V*.
showIdInFormTitle	string	Overwrite default from *Configuration*
formSubmitLogMode	string	Overwrite default from *Configuration*
sessionTimeoutSeconds	int	Overwrite default from *Configuration* . See *FE-User: Session timeout seconds*.
maxFileSize	int	Overwrite default from *Configuration* .
requiredPosition	int	See *Required Position* .

- Example:
  - maxVisiblePill = 5
  - class = container-fluid
  - classBody = qfq-form-right

9.5.7.1 submitButtonText

If specified and non empty, display a regular submit button at the bottom of the page with the given text. This gives the form a ordinary HTML-form look'n' feel. With this option, the standard buttons on the top right border should be hided to not confuse the user.

- Optional.
- Default: Empty

9.5.7.2 class

- Optional.

- Default: *container*
- Any CSS class name(s) can be specified.
- Check *typo3conf/ext/qfq/Resources/Public/Css/qfq-bs.css* for predefined classes.
- Typical use: adjust the floating rules of the form.
  - See: http://getbootstrap.com/docs/3.4/css/#overview-container
  - Expand the form over the whole area: *container-fluid*

9.5.7.3 classPill

- Optional.
- Default: *qfq-color-grey-1*
- Any CSS class name(s) can be specified.
- Check *typo3conf/ext/qfq/Resources/Public/Css/qfq-bs.css* for predefined classes.
- Typical use: adjust the background color of the *pill title* area.
- Predefined background colors: *qfq-color-white*, *qfq-color-grey-1* (dark), *qfq-color-grey-2* (light), *qfq-color-blue-1* (dark), *qfq-color-blue-2*. (light)
- *classPill* is only visible on forms with container elements of type 'Pill'.

9.5.7.4 classBody

- Optional.
- Default: *qfq-color-grey-2*
- Any CSS class name(s) can be specified.
- Check *typo3conf/ext/qfq/Resources/Public/Css/qfq-bs.css* for predefined classes.
- Typical use:
  - adjust the background color of the *FormElement* area.
  - make all form labels right align: *qfq-form-right*.
- Predefined background colors: *qfq-color-white*, *qfq-color-grey-1* (dark), *qfq-color-grey-2* (light), *qfq-color-blue-1* (dark), *qfq-color-blue-2*. (light)

9.5.7.5 extraDeleteForm

Depending on the database definition, it might be necessary to delete the primary record *and* corresponding slave records. To not repeat such 'slave record delete definition', an 'extraDeleteForm' can be specified. If the user opens a record in a form and clicks on the 'delete' button, a defined 'extraDeleteForm'-form will be used to delete primary and slave records instead of using the current form. E.g. if there are multiple different forms to work on the same table, all of theses forms might reference to the same 'extraDeleteForm'-form. This simplifies the maintenance.

The 'extraDeleteForm' parameter might be specified for a 'form' and/or for 'subrecords'

See also: *Delete Record*.

9.5.7.6 Form mode global

A form can be operated in modes: *standard* | **readonly** | **requiredOff** | **requiredOffButMark**.

Mode *standard* is given if none of the others are defined.

The *mode* is given via (in this priority):

- Via STORE_USER: {{formModeGlobal:U}}
- Via STORE_SIP: {{formModeGlobal:S}}
- Via Form: *form.parameter.formModeGlobal=...*

Mode

- **standard**:
  - The form will behave like defined in the form editor.
  - Missing required values will a) be indicated and b) block saving the record.
- **readonly**: all *FormElement* of the whole form are temporarily in *readonly* mode.
  - Fast way to display the form data, without a possibility for the user to change any data of the form.
  - The mode will be be inherited to all subforms.
- **requiredOff**:
  - All *FormElement* with *mode=required*, will be handled as *mode=show*.
  - The user can save the form without filling all required inputs!
  - Required inputs are indicated by a red star - missing values **won't** be complained.
- **requiredOffButMark**:
  - All *FormElement* with *mode=required*, will be handled as *mode=show*.
  - The user can save the form without filling all required inputs!
  - Missing required inputs will be marked:
    - On lost focus.
    - When the user saves the record.
      - After saving the record, by default the first pill with a missing input comes to front.
      - This behaviour can be switch on/off with *form.parameter.activateFirstRequiredTab=0*

Extra

Via *Store: VARS - V* the variable *{{allRequiredGiven:V}}* shows, if the form has been filled completely - independent of the mode. The variable is only accessible during form save.

Usage example

Readonly

Code: `SELECT 'p:{{pageAlias}}&form=person&r=1&formModeGlobal=readonly|s|t:View|s|b' AS _link`

- The form is called with SIP parameter `formModeGlobal=readonly` or `form.parameter.mode=readonly`.
- The user can't change any data.

Readonly systemwide

Code (somewhere): `SELECT 'requiredoff' AS '_=formModeGlobal'`

Code: `SELECT 'p:{{pageAlias}}&form=person&r=1|s|t:View|s|b' AS _link`

- The STORE_USER variable is set *formModeGlobal=readonly*.
- All forms will be shown in readonly mode - fast option to give a user access to the website, without the possibility to change any data.

Draft Mode 1

Code: `SELECT 'p:{{pageAlias}}&form=person&r=1&formModeGlobal=readquiredOff|s|t:View|s|b' AS _link`

- A form has one or more FormElement with 'fe.type=required'.
- Opening the form with *formModeGlobal=requiredOff* will allow the user to save the form, even if not all FE.type=required elements are given. This can be called 'draft' mode.
- Opening the form without *formModeGlobal* (that's the default), forces the user to fill out all FE.type=required fields. This can be called 'final submit' mode.

Draft Mode 2

Code: `SELECT 'p:{{pageAlias}}&form=person&r=1&formModeGlobal=readquiredOff|s|t:View|s|b' AS _link`

- A form has one or more FormElement with 'fe.type=required'.
- Calling the form with *formModeGlobal=requiredOff* will allow the user to save the form, even if not all FE.type=required elements are given.
- Define an FE-Action 'afterSave', and do some action on *{{allRequiredGiven:V0}}* like:

```
{{UPDATE <table> SET dataValid={{allRequiredGiven:V0}} WHERE id={{id:R}} }}
```

- In the application logic, open the next process step if all data is given by evaluating *dataValid*.

9.6 FormElements

- Each *form* contains one or more *FormElement*.
- The *FormElements* are divided in three categories:
  - *Class: Container*
  - *Class: Native*
  - *Class: Action*

- Ordering and grouping: Native *FormElements* and Container-Elements (both with feIdContainer=0) will be ordered by 'ord'.
- Inside of a container, all nested elements will be displayed.
- Technical, it's *not* necessary to configure a FormElement for the primary index column *id*.
- Additional options to a *FormElement* will be configured via the *FormElement.parameter* field (analog to *Form.parameter* for *Forms*).
  - See also: *Comment- and space-character*

9.7 Class: Container

- Pills are containers for 'fieldset' *and* / *or* 'native' *FormElements*.
- Fieldsets are containers for 'native' *FormElements*.
- TemplateGroups are containers for 'fieldset' *and* / *or* 'native' *FormElements*.

9.7.1 Type: fieldset

- Native *FormElements* can be assigned to a fieldset.
- FormElement settings:
  - *name*: technical name, used as HTML identifier.
  - *label*: Shown title of the fieldset.

9.7.2 Type: pill (tab)

- Pill is synonymous for a tab and looks like a tab.
- If there is at least one pill defined:
  - every native *FormElement* needs to be assigned to a pill or to a fieldset.
  - every *fieldset* needs to be assigned to a pill.
- Mode:
  - *show*: all child elements will be shown.
  - *required*: same as 'show'. This mode has no other meaning than 'show'.
  - *readonly*: technical it's like HTML/CSS *disabled*.
    - The pill title is shown, but not clickable.
    - The *FormElements* on the pill still exist, but are not reachable for the user via UI.
  - *hidden*:
    - The pill is invisible.
    - The *FormElements* on the pill still exist, but are not reachable for the user via UI.
  - Note: Independent of the *mode*, all child elements are always rendered and processed by the client/server.
- Pills are 'dynamicUpdate' aware. *title* and *mode* are optional recalculated during 'dynamicUpdate'.
- FormElement settings:

  - *name*: technical name, used as HTML identifier.
  - *label*: Label shown on the corresponding pill button or inside the drop-down menu.
  - *mode*:
    * *show*, *required*: regular mode. The pill will be shown.
    * *readonly*: the pill and it's name is visible, but not clickable.
    * *hidden*: the pill is not shown at all.
  - *modeSql*:
  - *type*: *pill*
  - *feIdContainer*: *0* - Pill's can't be nested.
  - *tooltip*: Optional tooltip on hover. Especially helpful if the pill is set to *readonly*.
  - *parameter*:
    * *maxVisiblePill*: <*nr*> - Number of Pill-Buttons shown. Undefined means unlimited. Excess Pill buttons will be displayed as a drop-down menu.

9.7.3 Type: templateGroup

TemplateGroups will be used to create a series of grouped (by the given *templateGroup*) *FormElements*.

FormElements can be assigned to a *templateGroup*. These *templateGroup* will be rendered upto *n*-times. On 'form load' only a single (=first) copy of the *templateGroup* will be shown. Below the last copy of the *templateGroup* an 'add'-button is shown. If the user click on it, an additional copy of the *templateGroup* is displayed. This can be repeated up to *templateGroup.maxLength* times. Also, the user can 'remove' previously created copies by clicking on a remove button near beside every *templateGroup*. The first copy of a templateGroup can't be removed.

- FormElement settings:
  - *label*: Shown in the FormElement-editor *container* field.
  - *maxLength*: Maximum number of copies of the current *templateGroup*. Default: 5.
  - *bsLabelColumn*, *bsInputColumn*, *bsNoteColumn*: column widths for row with the 'Add' button.
  - *parameter*:
    * *tgAddClass*: Class of the 'add' button. Default: *btn btn-default*.
    * *tgAddText*: Text shown on the button. Default: *Add*.
    * *tgRemoveClass*: Class of the 'remove' button. Default: *btn btn-default*.
    * *tgRemoveText*: Text shown on the button. Default: *Remove*.
    * *tgClass*: Class wrapped around every copy of the *templateGroup*. E.g. the class *qfq-child-margin-top* adds a margin between two copies of the *templateGroup*. Default: empty

Multiple *templateGroups* per form are allowed.

The name of the native FormElements, inside the templateGroup, which represents the effective table columns, uses the placeholder *%d*. E.g. the columns *grade1*, *grade2*, *grade3* needs a *FormElement.name* = *grade%d*. The counting will always start with 1. The placeholder *%d* can also be used in the *FormElement.label*

Example of styling the Add/ Delete Button: *Icons Template Group*

9.7.3.1 Column: primary record

If the columns *<name>%d* are real columns on the primary table, saving and delete (=empty string) are done automatically. E.g. if there are up to five elements *grade1, . . . , grade5* and the user inputs only the first three, the remaining will be set to an empty string.

9.7.3.2 Column: non primary record

Additional logic is required to load, update, insert and/or delete slave records.

Load

On each native *FormElement* of the *templateGroup* define an SQL query in the *FormElement.value* field. The query has to select **all** values of all possible existing copies of the *FormElement* - therefore the exclamation mark is necessary. Also define the order. E.g. *FormElement.value*:

```
{{!SELECT street FROM Address WHERE personId={{id}} ORDER BY id}}
```

Insert / Update / Delete

Add an *action* record, type='afterSave', and assign the record to the given *templateGroup*.

In the parameter field define:

```
slaveId = {{SELECT id FROM Address WHERE personId={{id}} ORDER BY id LIMIT %D,1}}
sqlHonorFormElements = city%d, street%d
sqlUpdate = {{UPDATE Address SET city='{{city%d:FE:alnumx:s}}', street='{{street
↪%d:FE:alnumx:s}}'  WHERE id={{slaveId}} LIMIT 1}}
sqlInsert = {{INSERT INTO Address (`personId`, `city`, `street`) VALUES ({{id}}, '{
↪{city%d:FE:alnumx:s}}' , '{{street%d:FE:alnumx:s}}') }}
sqlDelete = {{DELETE FROM Address WHERE id={{slaveId}} LIMIT 1}}
```

The *slaveId* needs attention: the placeholder *%d* starts always at 1. The *LIMIT* directive starts at 0 - therefore use *%D* instead of *%d*, cause *%D* is always one below *%d* - but can **only** be used on the action element.

9.8 Class: Native

Fields:

Name	Type	Description
Container	int	0 or *FormElement*.id of container element (pill, fieldSet, templateGroup) part the current *Form*
Enabled	enum('yes'|'no')	Process the current FormElement
Dynamic Update	enum('yes'|'no')	In the browser, *FormElements* with "dynamicUpdate='yes'" will be updated depending on user input. *Dynamic Update*
Name	string	
Label	string	Label of *FormElement*. Depending on layout model, left or on top of the *FormElement* Additional label description can be added by wrapping in HTML tag '<small>'
Mode	enum('show', 'readonly', 'required', 'hidden')	*Show*: regular user input field. This is the default. *Required*: User has to specify a value. Typically, an <empty string> represents 'no value'. *Readonly*: User can't change. Data is not saved, except for FormElement with 'processReadOnly' *Hidden*: *FormElement* is not visible.
Mode sql	*SELECT* statement with a value like in *mode*	A value given here overwrites the setting from *mode*. Most useful with *Dynamic Update*. E.g.: {{SELECT IF('{{otherFunding:FR:alnumx}}'='yes' ,'show', 'hidden' }}
Class	enum('native', 'action', 'container')	Details below.
Type	enum('checkbox', 'date', 'time', 'datetime', 'dateJQW', 'datetimeJQW', 'extra', 'gridJQW', 'text', 'editor', 'annotate', 'imageCut', 'note', 'password', 'radio', 'select', 'subrecord', 'upload', 'fieldset', 'pill', 'beforeLoad', 'beforeSave', 'beforeInsert', 'beforeUpdate', 'beforeDelete', 'afterLoad', 'afterSave', 'afterInsert', 'afterUpdate', 'afterDelete', 'sendMail')	
Encode	'none', 'specialchar'	With 'specialchar' (default) the chars <>"'& will be encoded to their htmlentity. field-encode
Check Type	enum('auto', 'alnumx', 'digit', 'numerical', 'email', 'pattern', 'allbut', 'all')	See: *Sanitize class*
Check Pattern	'regexp'	field-checktype: If $checkType=='pattern': pattern to match
Order	string	Display order of *FormElements* ('order' is a reserved keyword) field-ord
labelAlign	left	Label align (default/left/center/right) default: 'default' (defined by Form).
Size	string	Depends on the FormElement type. E.g. visible length (and

9.8.1 FE: Value

By default this field is empty: QFQ will fill it with the corresponding existing column value on form load. For a customized default value define:

```
{{SELECT IF('{{column:RE}}'='','custom default', '{{column:R}}') }}
```

For non primary records, this is the place to load an existing value. E.g. we're on a 'Person' detail form and would like to edit, on the same form, a corresponding person email address (which is in a separate table):

```
{{SELECT a.email FROM Address AS a WHERE a.pId={{id:R0}} ORDER BY a.id LIMIT 1}}
```

Report syntax can also be used, see *FE: 'Report' notation*.

9.8.2 FE: 'Report' notation

The FE fields 'value' and 'note' understand the *Report* syntax. Nested SQL queries as well as links with SIP encoding are possible. To distinguish between 'Form' and 'Report' syntax, the first line has to be *#!report*:

```
#!report

10.sql = SELECT ...

20 {
  sql = SELECT ...
  5.sql = SELECT ...
}
```

9.8.3 Attributes defined in the parameter field

See also at specific *FormElement* definitions.

Name	Note
acceptZeroAsRequired	0|1 - Accept a '0' as a valid input. Default '0' (=0 is not a valid input)
autofocus	See *autofocus*
capture, accept, maxFileSize, fileDestination, fileTrash, fileTrashText, fileReplace, autoOrient, autoOrientCmd, autoOrientMimeType, chmodFile, chmodDir, slaveId, sqlBefore, sqlInsert, sqlUpdate, sqlDelete, sqlAfter, importToTable, importToColumns, importRegion, importMode, importType, importNamedSheetsOnly, importSetReadDataOnly, importListSheetNames,	See *Type: upload*
checkBoxMode checked unchecked label2 itemList emptyHide emptyItemAtStart emptyItemAtEnd buttonClass	See *Type: checkbox*, *Type: radio*, *Type: select*
dateFormat	yyyy-mm-dd / dd.mm.yyyy
data-pattern-error	Pattern violation: Text for error message
data-required-error	Required violation: Text for error message
data-match-error	Match violation: Text for error message
data-error	Violation of 'check-type': Text for error message

Continued on next page

Table 2 – continued from previous page

Name	Note
decimalFormat	[precision,scale] Limits and formats input to a decimal number with the specified precision and scale. If no precision and scale are specified, the decimal format is pulled from the table definition.
htmlAfter	HTML Code wrapped after the complete *FormElement*
htmlBefore	HTML Code wrapped before the complete *FormElement*
extraButtonLock	[0|1] Show a 'lock' on the right side of the input element. See *extraButtonLock*
extraButtonPassword	[0|1] Show an 'eye' on the right side of the input element. See *extraButtonPassword*
extraButtonInfo	Text. Show an 'i' on the right side of the input element. See *extraButtonInfo*
extraButtonInfoClass	By default empty. Specify any class to be assigned to wrap extraButtonInfo
extraButtonInfoMinWidth	See *extraButtonInfo*
editor-plugins, editor-toolbar, editor-statusbar,	See *Type: editor*
fileButtonText	Overwrite default 'Choose File'
fillStoreVar	Fill the STORE_VAR with custom values. See *Store: VARS - V*.
form, page, title, extraDeleteForm, detail, subrecordTableClass,	See *Type: subrecord*
min s/d/n	Minimum and/or maximum allowed values for input field. Can be used for numbers, dates, or strings. *Always use the international format 'yyyy-mm-dd[hh:mm[:ss]]*
max s/d/n	
processReadOnly	[0|1] By default FE's with type='readonly' are not processed during 'save'. This option forces to process them during 'save' as well.
retype, retypeLabel, retypeNote, characterCountWrap, hideZero, emptyMeansNull,	See *Type: text*
showSeconds	0|1 - Shows the seconds on form load. Default: 0
showZero	0|1 - Empty timestamp: '0'(default) - nothing shown, '1' - the string '0000-00-00 00:00:00' is displayed
timeIsOptional	0|1 - Used for datetime input. 0 (default): Time is required - 1: Entering a time is optional (defaults to 00:00:00 if none entered).
typeAheadLimit, typeAheadInitialSuggestion, typeAheadMinLength, typeAheadSql, typeAheadSqlPrefetch, typeAheadPedantic	See *Type Ahead*
typeAheadTag, typeAheadGlueInsert, typeAheadGlueDelete, typeAheadTagInsert	See *Type Ahead Tag*
wrapRow	If specified, skip default wrapping (<*div class='col-md-?'*>). Instead the given string is used.
wrapLabel	
wrapInput	
wrapNote	
trim	By default, whitespace is trimmed. To disable, use 'trim=none'. You can also specify custom trim characters: 'trim=\ ' only trims spaces.
sqlValidate	See *Parameter: sqlValidate*

Continued on next page

Table 2 – continued from previous page

Name	Note
expectRecords	
messageFail	
dataReference	Optional. See *Application Test*
requiredPosition	See *Required Position*.
minWidth	See *Checkbox / Radio: minWidth*.

- *s/d/n*: string or date or number.

9.8.3.1 slaveId, sqlBefore, sqlAfter, . . .

See *Parameter: slaveId*

9.8.3.2 Native *FormElements*

- Like 'input', 'checkbox', . . .

autofocus

The first *FormElement* with this attribute will get the focus after form load. If there is no such attribute given to any *FormElement*, the attribute will be automatically assigned to the first editable *FormElement*.

To disable 'autofocus' on a form, set 'autofocus=0' on the first editable *FormElement*.

Note: If there are multiple pills defined on a form, only the first pill will be set with 'autofocus'.

extraButtonLock

- The user has to click on the lock, before it's possible to change the value. This will protect data against unwanted modification.
- After Form load, the value is shown, but not editable.
- Shows a 'lock' on the right side of an input element of type *text*, *date*, *time* or *datetime*.
- This option is not available for FormElements with *mode=readonly*.
- There is no value needed for this parameter.

extraButtonPassword

- The user has to click on the eye (unhide) to see the value.
- After Form load, the data is hided by asteriks.
- Shows an 'eye' on the right side of an input element of type *text*, *date*, *time* or *datetime*.
- There is no value needed for this parameter.

extraButtonInfo

- After Form load, the *info* button/icon is shown but the information message is hidden.
- The user has to click on the *info* button/icon to see an additional message.
- The value of this parameter is the text shown to the user.
- Shows an *info* button/icon, depending of *extraButtonInfoPosition* in *Configuration*
  - *auto*, depending on *FormElement* type:
    - on the right side of an input element for type *text*, *date*, *time* or *datetime*,
    - below the FormElement for all other types.
  - *below*: below the FormElement for all types.
- *extraButtonInfoMinWidth*: default is 250 and defines a minimal width.
- For *FormElement* with mode *below*, a *span* element with the given class in *extraButtonInfoClass* (FE, F, *Configuration*) will be applied. E.g. this might be *pull-right* to align the *info* button/icon on the right side below the input element.

9.8.4 Checkbox / Radio: minWidth

Checkbox and Radio Elements, shown in plain horizontal mode, receives a minWidth to align them. The default is 80px and might be defined per Form or per FormElement.

9.8.5 Required Position

By default, input elements with *Mode=required* will be displayed with a 'red asterix' right beside the label. The position of the 'red asterix' can be choosen via the *parameter* field:

```
requiredPosition = label-left|label-right|input-left|input-right|note-left|note-right
```

The default is 'label-right'.

The definition can be set per Form (=affects all FormElements) or per FormElement.

9.8.6 Type: checkbox

Checkboxes can be rendered in mode:

- *single*:
  - One column in a table corresponds to one checkbox.
  - The value for statuses *checked* and *unchecked* are free to choose.
  - This mode is selected, if a) *checkBoxMode* = single, or b) *checkBoxMode* is missing **and** the number of fields of the column definition is <3.
  - *FormElement.parameter*:
    - *checkBoxMode* = single (optional)
    - *checked* = <value> (optional, the value which represents 'checked')

        - If *checked* is empty or missing: If *type* = 'enum' or 'set', get first item of the definition. If *type* = string, get default.
    - *unchecked* = <value> (optional, the value which represents 'unchecked')
        - If *unchecked* is empty or missing: If *type* = 'enum' or 'set', get second item of checked. If *type* = 'string', get ''.
    - *label2* = <value> (Text right beside checkbox) (optional)

- *multi*:
  - One column in a table represents multiple checkboxes. This is typically useful for the column type *set*.
  - The value for status *checked* are free to choose, the value for status *unchecked* is always the empty string.
  - Each field key (or the corresponding value from the key/value pair) will be rendered right beside the checkbox.
  - *FormElement.parameter*
    - *checkBoxMode* = multi
    - *itemList* - E.g.:
      - `itemList=red,blue,orange`
      - `itemList=1:red,2:blue,3:orange`
      - If ':' or ',' are part of key or value, it needs to escaped by \ . E.g.: *itemList=1:red\: (with colon),2:blue\, (with comma),3:orange'*
  - *FormElement.sql1* = `{{!SELECT id, value FROM SomeTable}}`
  - *FormElement.maxlength* - vertical or horizontal alignment:
    - Value: '', 0, 1 - The check boxes will be aligned vertical.
    - Value: >1 - The check boxes will be aligned horizontal, with a linebreak every 'value' elements.

- *FormElement.parameter*:
  - *emptyHide*: Existence of this item hides an entry with an empty string. This is useful for e.g. Enums, which have an empty entry, but the empty value should not be selectable.
  - *emptyItemAtStart*: Existence of this item inserts an empty entry at the beginning of the selectlist.
  - *emptyItemAtEnd*: Existence of this item inserts an empty entry at the end of the selectlist.
  - *buttonClass*: Instead of the plain HTML checkbox fields, Bootstrap buttons. are rendered as *checkbox* elements. Use one of the following classes:
    - *btn-default* (default, grey),
    - *btn-primary* (blue),
    - *btn-success* (green),
    - *btn-info* (light blue),
    - *btn-warning* (orange),
    - *btn-danger* (red).

    With a given *buttonClass*, all buttons (=radios) are rendered horizontal. A value in *FormElement.maxlength* has no effect.

- *No preselection*:

- If a form is in 'new' mode and if there is a default value configured on a table column, such a value is shown by default. There might be situations, where the user should be forced to select a value (e.g. specifying the gender). An unwanted default value can be suppressed by specifying an explicit definition on the FormElement field *value*:

```
{{<columnName>:RZ}}
```

For existing records the shown value is as expected the value of the record. For new records, it's the value *0*, which is typically not one of the ENUM / SET values and therefore nothing is selected.

9.8.7 Type: date

- Range datetime: '1000-01-01' to '9999-12-31' or '0000-00-00'. (http://dev.mysql.com/doc/refman/5.5/en/datetime.html)
- Optional:
  - *FormElement.parameter.dateFormat*: yyyy-mm-dd | dd.mm.yyyy

9.8.8 Type: datetime

- Range datetime: '1000-01-01 00:00:00' to '9999-12-31 23:59:59' or '0000-00-00 00:00:00'. (http://dev.mysql.com/doc/refman/5.5/en/datetime.html)
- Optional:
  - *FormElement.parameter*:
    - *dateFormat* = yyyy-mm-dd | dd.mm.yyyy
    - *showSeconds* = 0|1 - shows the seconds. Independent if the user specifies seconds, they are displayed '1' or not '0'.
    - *showZero* = 0|1 - For an empty timestamp, With '0' nothing is displayed. With '1' the string '0000-00-00 00:00:00' is displayed.

9.8.9 Type: extra

- The element behaves like, and can be used as, a HTML hidden input element - with the difference & advantage, that the element never leaves the server and therefore can't be manipulated by a user.
- The following names are reserved and can't be used to name 'extra' FormElements: 'id', 'type', 'L'.
- The element is not transferred to the the browser.
- The element can be used to define / pre calculate values for a column, which do not already exist as a native *FormElement*.
- The element is build / computed on form load and saved alongside with the SIP parameter of the current form.
- The element is not recalculated during save - it's stored during 'Form Load' inside the current form SIP handle.
- Access the value without specifying any store (default store priority is sufficient).

9.8.10 Type: text

General input for any text.

- By default, the maximum length of input data is automatically restricted by the underlying database column.
- HTML decides between one line input (=Input text) and multiline input (=Textarea).
- *FormElement.size* = [<width>[,<min height (lines)>[,<max height (pixel)>]]]
  - The parameter is optional and controls the behaviour of the input / textarea element.
  - The <*width*> is counted in 'characters'.
    - But: the *visible* width of an input element is defined by the Bootstrap column width (and *not* the width given here). Finally: the value here is meaningless. Nevertheless it has to be given for future compatibility.
  - <*min-height*>:
    - Counted as 'lines'.
    - If not set the height is treated as 1.
    - A <*min-height*> of 1 forces an one line input. Exception: <*max-height*> > 0 enables *auto-grow*.
  - <*max-height*>:
    - Controls the *auto-grow*-Mode.
    - Counted in 'pixel'.
    - If not set it becomes the default of 350 pixels.
    - If > 0, the *auto-grow* mode is activated and the height of the textarea will be dynamically updated up to <*max-height*>.
    - If = 0, the *auto-grow* mode is disabled.
- *FormElement.parameter*:
  - *retype* = 1 (optional): Current input element will be rendered twice. The form can only submitted if both elements are equal.
    - *retypeLabel* = <text> (optional): The label of the second element.
    - *retypeNote* = <text> (optional): The note of the second element.
  - *characterCountWrap* = Count: | (optional). Displays a character counter below the input/textarea element.
  - Also check the *Attributes defined in the parameter field data-...-error* to customize error messages shown by the validator.
  - *hideZero* = 0|1 (optional): *with hideZero=1* a '0' in the value will be replaced by an empty string.
  - *emptyMeansNull* = [0|1] (optional): with *emptyMeansNull* or *emptyMeansNull=1* a NULL value will be written if the value is an empty string
  - *inputType* = number (optional). Typically the HTML tag 'type' will be 'text', 'textarea' or 'number' (detected automatically). If necessary, the HTML tag 'type' might be forced to a specific given value.
  - *step* = Step size of the up/down buttons which increase/decrease the number of in the input field. Optional. Default 1. Only useful with *inputType=number* (defined explicit via *inputType* or detected automatically).
  - *textareaResize* = 0|1 (optional). Be default = 1 (=on). A textarea element is resizable by the user.

9.8.10.1 Type Ahead

Activating *typeahead* functionality offers an instant lookup of data and displaying them to the user, while the user is typing, a drop-down box offers the results. As datasource the regular SQL connection or a LDAP query can be used. With every keystroke (starting from the *typeAheadMinLength* characters), the already typed value will be transmitted to the server, the lookup will be performed and the result, upto *typeAheadLimit* entries, are displayed as a drop-down box.

- *FormElement.parameter*:
  - *typeAheadLimit* = <number>. Max numbers of result records to be shown. Default is 20.
  - *typeAheadMinLength* = <number>. Minimum length to type before the first lookup starts. Default is 2.

Depending of the *typeahead* setup, the given FormElement will contain the displayed *value* or *id* (if an id/value dict is configured).

Configuration via Form / FormElement

All of the *typeAhead** (except *typeAheadLdap*, *typeAheadInitialSuggestion*) and *ldap** parameter can be specified either in *Form.parameter* or in *FormElement.parameter*.

SQL

- *FormElement.parameter*:
  - *typeAheadSql* = `SELECT ... AS 'id', ... AS 'value' FROM ... WHERE name LIKE ? OR firstName LIKE ? LIMIT 100`
    - If there is only one column in the SELECT statement, that one will be used and there is no dict (key/value pair).
    - If there is no column *id* or no column *value*, then the first column becomes *id* and the second column becomes *value*.
    - The query will be fired as a 'prepared statement'.
    - The value, typed by the user, will be replaced on all places where a *?* appears.
    - All *?* will be automatically surrounded by '*%*'. Therefore wildcard search is implemented: ... *LIKE '%<?>%'* ...
  - *typeAheadSqlPrefetch* = `SELECT firstName, ' ', lastName FROM Person WHERE id = ?`
    - If the query returns several results, only the first one is returned and displayed.
    - If the query selects multiple columns, the columns are concatenated.
  - *typeAheadInitialSuggestion* = `{{!SELECT fr.id AS id, fr.name AS value FROM Fruit AS fr}}`
    - Shows suggestions when the input element gets the focus, before the user starts to type anything.
    - If given, *typeAheadMinLength* will be set to 0.
    - Limit the number of rows via SQL `... LIMIT ...` clause.

LDAP

See *Typeahead (TA) - LDAP*

9.8.10.2 Type Ahead Tag

Extend a TypeAhead input element to take more than one token (=tag) in the same input element.

This mode supports only *typeAheadSql* (no LDAP).

Usage: A user might choose one or more tags from a typeahead list (to minimize typos and to reuse already given tags).

The user starts typing and for each keypress *typeAheadSql* is searched for all matches. The user selects an element by clicking on it or by using one of the *typeAheadTagDelimiter* key presses (by default tab or comma). If a tag is selected, it will be visual separated from the input cursor. Already selected tags can not be edited but removed (clicking on the x). Further tags can be added.

typeAheadTag support two different modes: a) *Tag* , b) *Glue*.

Mode: Tag

Tags will be loaded and saved as a comma separated list. Maximum length of saved tags is limit by the size of the column (incl. separator).

Additional arguments needed for *typeAheadTag*:

- *FormElement.parameter*:
  - *typeAheadTag* = [0|1] - Default 0 (=off), existence or =1 switches the mode *typeAheadTag* on.
  - *typeAheadTagDelimiter* = List of ASCII codes to separate tags during input. Default '9,44' (tab and comma).

Mode: Glue

For each selected tag a glue record, pointing to the tag, is created.

The *Glue* mode will be activated by setting *FormElement.parameter.typeAheadGlueInsert* with a corresponding SQL statement.

Glue records will be created or deleted, as the user select or deselect tags. Processing of those Glue records will be done after the primary form record has been written and before any after*-action FormElements will be processed.

FormElement.name should **not** point to a column in the form primary table. Instead a free name should be used for the *typeAhead* FormElement.

The maximum number of tags is not limited - but take care to size the FormElement big enough (*FormElement.maxLength*) to show all tags.

On *Form load* (to show already assigned tags) a comma separated list has to be given in *FormElement.value*, based on the previously saved Glue records. The string format is identically to the one used in mode *Tag*.

Extra parameter for mode = *Tag* :

- *FormElement.parameter*:
  - *typeAheadTagInsert* = {{INSERT INTO Tag (....) VALUES (...)}} - Only needed with *typeAheadPedantic=0*.

  - *typeAheadGlueInsert* = {{INSERT INTO glueTag (. . .) VALUES (. . .)}}
  - *typeAheadGlueDelete* = {{DELETE FROM glueTag WHERE . . . }}

Example:

Table *Person* with some records. Table *Fruit* with a list of fruits. Table *FruitPerson* with glue records.

Usage: assign favourite fruits to a person. The fruits are the tags, the glue records will assign the fruits to a person.

The form will be open with a person record and has only one FormElement.

- Form.name=personFavouriteFruits
- Form.title=Person Favourite Fruits
- Form.primaryTable = Person
- FormElement[1].name = myFavoriteFruits
- FormElement[1].type = Text
- FormElement[1].value = {{SELECT GROUP_CONCAT(CONCAT(f.id, ':', f.name)) FROM FruitPerson AS fp, Fruit AS f WHERE fp.pId={{id:R}} AND fp.fruitId=f.id ORDER BY f.name}}
- FormElement[1].parameter:
  - typeAheadTag = 1
  - typeAheadSql = SELECT f.id AS 'id', f.name AS 'value' FROM Fruit AS f WHERE f.name LIKE ?
  - typeAheadMinLength = 1
  - typeAheadGlueInsert = {{INSERT INTO FruitPerson (pId, fruitId) VALUES ({{id:R}}, {{tagId:V}}) }}
  - typeAheadGlueDelete = {{DELETE FROM FruitPerson WHERE pId={{id:R}} AND fruitId={{tagId:V}} }}

Explanation:

- On form load, without any assigned tags (=fruits), *FormElement.value* will be empty.
- The User will assign three fruits: Apple, Banana, Lemon.
- On form save, QFQ does:
  - compares the old tag assigment (empty) with the new tag assigment (3 elements).
  - for each new assigned tag:
    - the *tagId* and *tagValue* will be stored in STORE_VAR (that's the one selected by the user and defined via *typeAheadSql*)
    - *typeAheadGlueInsert* will be fired (with the replaced variable *{{tagId:V}}*).
- The user loads the person favourite fruit form again (same user).
- *FormElement.value* will now be: `1:Apple,3:Banana,10:Lemon`.
- The user removes 'Banana' and adds 'Orange'.
- On form save, QFQ does:
  - compares the old tag assigment (3 elements) with the new tag assigment (also 3 elements, but different).
  - for each new assigned tag:
    - the *tagId* and *tagValue* will be stored in STORE_VAR.
    - *typeAheadGlueInsert* will be fired (with the replaced variable *{{tagId:V}}*).

  - for each removed assigned tag:
    * the *tagId* and *tagValue* will be stored in STORE_VAR.
    * *typeAheadGlueDelete* will be fired (with the replaced variable *{{tagId:V}}*).

9.8.11 Type: editor

- TinyMCE (https://www.tinymce.com, community edition) is used as the QFQ Rich Text Editor.
- The content will be saved as HTML inside the database.
- All configuration and plugins will be configured via the 'parameter' field. Just prepend the word 'editor-' in front of each TinyMCE keyword. Check possible options under:
  - https://www.tinymce.com/docs/configure/,
  - https://www.tinymce.com/docs/plugins/,
  - https://www.tinymce.com/docs/advanced/editor-control-identifiers/#toolbarcontrols
- Bars:
  - Top: *menubar* - by default hidden.
  - Top: *toolbar* - by default visible.
  - Bottom: *statusbar* - by default hidden, exception: *min_height* and *max_height* are given via size parameter.
- The default setting in *FormElement.parameter* is:

```
editor-plugins=code link lists searchreplace table textcolor textpattern␣
↪visualchars
editor-toolbar=code searchreplace undo redo | styleselect link table | fontselect␣
↪fontsizeselect | bullist numlist outdent indent | forecolor backcolor bold␣
↪italic editor-menubar=false
editor-statusbar=false
```

- To deactivate the surrouding <*p*> tag, configure in *FormElement.parameter*:

```
editor-forced_root_block = false
```

  This might have impacts on the editor. See https://www.tinymce.com/docs/configure/content-filtering/#forced_root_block

- Set 'extended_valid_elements' to enable HTML tags and their attributes. Example:

```
editor-extended_valid_elements = span[class|style]
```

- Set 'editor-content_css' to use a custom CSS to style elements inside the editor. Example:

```
editor-content_css = fileadmin/custom.css
```

- *FormElement.size* = <min_height>,<max_height>: in pixels, including top and bottom bars. E.g.: 300,600

9.8.12 Type: annotate

Annotate image or text. Typically the image or text has been uploaded during a previous step. The annotation will be saved in *FormElement.name* column of the current record. The uploaded file itself will not be modified. The annotations can be shown in edit (and might be modified) or in readonly mode.

Two modes are available:

grafic A simple grafic editor to paint on top of the image (best by a tablet with pen or grafic tablet). The uploaded image is shown in the background. All drawings are saved as a JSON fabric.js data string. Supported file types: **png, svg**. PDF files can be easily divided into per page SVG files during upload - see *Split PDF Upload*

text Per code line, annotation(s) can be added. Different users can add individual annotations. A user can only edit the own annotations. The annotations are saved as QFQ internal JSON string.

Note: Drawing with fabric.js might produce a lot data. Take care the column type/size is big enough (>=64kB).

9.8.12.1 Grafic

An image, specified by `FormElement.parameter.imageSource={{pathFileName}}`, will be displayed in the background. On form load, both, the image and an optional already given grafical annotations, will be displayed. The image is SIP protected and will be loaded on demand.

Form.parameter

Attribute	Value	Description
annotateType	grafic	*grafic|text*. Default is *grafic*. Select mode.
imageSource	<path filename>	Background image. E.g. *fileadmin/images/scan.png*
defaultPenColor	<rgb hex value> | Pen default color, after loading the fabric element. Default is '0000FF' (blue).	

Note: By using the the *FormElement annotate*, the JS code *fabric.min.js* and *qfq.fabric.min.js* has to be included. See *Setup CSS & JS*.

9.8.12.2 Code

Form.parameter

Attribute	Value	Description
annotateType	text	*grafic|text*. Default is *grafic*. Select mode.
textSource	<path filename>	Text file to annotate.
annotateUserName	<john doe>	Will be shown at annotation line.
annotateUserUid	<123>	Will be shown at annotation line.
annotateUserAvatar	<https://gravatar...>	Will be shown at annotation line.
highlight	auto	off,auto,javascript,qfq,python,matlab

9.8.13 Type: imageCut

Uploaded images can be cut or rotate via QFQ (via fabric.js). The modified image is saved under the given pathFileName.

- The 'value' of the *FormElement* has to be a valid PathFileName to an image.
- Valid image file formats are SVG, PNG, JPG, GIF.

- Invalid or missing filenames results to an empty 'imageCut' element.
- *FormElement.parameter*:
  - *resizeWidth* = <empty>|[width in pixel] - the final width of the modified image. If empty (or not given), no change.
  - *keepOriginal* = <empty>|[string] - By default: '.save'. If empty (no string given), don't keep the original. If an extension is given and if there is not already a <pathFileName><.extension>, than the original file is to copied to it.

9.8.14 Type: note

An FormElement without any 'input' functionality -just to show some text. Use the typical fields 'label', 'value' and 'note' to be displayed in the corresponding three standard columns.

9.8.15 Type: password

- Like a *text* element, but every character is shown as an asterisk.

9.8.16 Type: radio

- Radio Buttons will be built from one of three sources:
  1. 'sql1': E.g. *{{!SELECT type AS label FROM Car }}* or *{{!SELECT type AS label, typeNr AS id FROM Car}}* or *{{!SHOW tables}}*.
     - Resultset format 'named': column 'label' and optional a column 'id'.
     - Resultset format 'index':
       - One column in resultset >> first column represents *label*
       - Two or more columns in resultset >> first column represents *id* and second column represents *label*.
  2. *FormElement.parameter*:
     - *itemList* = <*attribute*> E.g.: *itemList=red,blue,orange* or *itemList=1:red,2:blue,3:orange*
     - If ':' or ',' are part of key or value, it needs to escaped by \ . E.g.: *itemList=1:red\: (with colon),2:blue\, (with comma),3:orange*
  3. Definition of the *enum* or *set* field (only labels, ids are not possible).
- *FormElement.maxlength* = <*value*>
  - Applies only to 'plain' radio elements (not the Bootstrap 'buttonClass' from below)
  - *vertical* or *horizontal* alignment:
    - <*value*>: '', 0, 1 - The radios will be aligned *vertical*.
    - <*value*>: >1 - The readios will be aligned *horizontal*, with a linebreak every 'value' elements.
- *FormElement.parameter*:
  - *emptyHide*: Existence of this item hides an entry with an empty string. This is useful for e.g. Enums, which have an empty entry, but the empty value should not be selectable.
  - *emptyItemAtStart*: Existence of this item inserts an empty entry at the beginning of the selectlist.

  - *emptyItemAtEnd*: Existence of this item inserts an empty entry at the end of the selectlist.
  - *buttonClass* = <class> - Instead of the plain radio fields, Bootstrap buttons. are rendered as *radio* elements. Use one of the following classes:
    * *btn-default* (default, grey),
    * *btn-primary* (blue),
    * *btn-success* (green),
    * *btn-info* (light blue),
    * *btn-warning* (orange),
    * *btn-danger* (red).

    With a given *buttonClass*, all buttons (=radios) are rendered horizontal. A value in *FormElement.maxlength* has no effect.

- *No preselection*:
  - If there is a default configured on a table column, such a value is selected by default. If the user should actively choose an option, the 'preselection' can be omitted by specifying an explicit definition on the FormElement field *value*:

```
{{<columnName>:RZ}}
```

    For existing records the shown value is as expected the value of the record. For new records, it's the value *0*, which is typically not one of the ENUM values and therefore nothing is selected.

9.8.17 Type: select

- Select lists will be built from one of three sources:
  - *FormElement.sql1 = {{!<SQL Query}}*
    * E.g. *{{!SELECT type AS label FROM Car }}* or *{{!SELECT type AS label, typeNr AS id FROM Car}}* or *{{!SHOW tables}}*.
    * Resultset format 'named': column 'label' and optional a column 'id'.
    * Resultset format 'index':
      · One column in resultset >> first column represents *label*
      · Two or more columns in resultset >> first column represents *id* and second column represents *label*.
  - *FormElement.parameter*:
    * *itemList* = <*attribute*> - E.g.: *itemList=red,blue,orange* or *itemList=1:red,2:blue:3:orange*
    * If ':' or ',' are part of key or value, it needs to escaped by \ . E.g.: *itemList=1:red\: (with colon),2:blue\, (with comma),3:orange*
  - Definition of the *enum* or *set* field (only labels, ids are not possible).
- *FormElement.size* = <*value*>
  - <*value*>: <empty>|0|1: drop-down list.
  - <*value*>: >1: Select field with *size* rows height. Multiple selection of items is possible.
- *FormElement.parameter*:

- *emptyItemAtStart*: Existence of this item inserts an empty entry at the beginning of the selectlist.
- *emptyItemAtEnd*: Existence of this item inserts an empty entry at the end of the selectlist.
- *emptyHide*: Existence of this item hides the empty entry. This is useful for e.g. Enums, which have an empty entry and the empty value should not be an option to be selected.
- *datalist*: Similar to 'typeAhead'. Enables the user to select a predefined option (sql1, itemList) or supply any free text. Attention: Safari (and some other) browsers do not support this fully - https://caniuse.com/#search=datalist.

9.8.18 Type: subrecord

The *FormElement* type 'subrecord' renders a list of records (so called secondary records), typically to show, edit, delete or add new records. The list is defined as an SQL query. The number of records shown is not limited. These *FormElement* will be rendered inside the form as a HTML table.

- *mode / modeSql = <type/value>*
  - *show / required*: the regular mode to show the subrecords
  - *readonly*: New / Edit / Delete Buttons are disabled
  - *hidden*: The FormElement is rendered, but hidden with *display='none'*.
- *dynamicUpdate* - not supported at the moment.
- *sql1 = {{!SQL Query}}*
  - SQL query to select records. E.g.:

```
{{!SELECT addr.id AS id, CONCAT(addr.street, addr.streetnumber) AS a, addr.
↪city AS b, addr.zip AS c FROM Address AS addr}}
```

  - Notice the **exclamation mark** after '{{' - this is necessary to return an array of elements, instead of a single string.
  - Exactly one column **'id'** has to exist; it specifies the primary record for the target form. In case the id should not be visible to the user, it has to be named **'_id'**.
  - Column name: *[title=]<title>[|[maxLength=]<number>][|nostrip][|icon][|link][|url][|mailto][|_rowClass][|_rowTooltip]*
    - If the keyword is used, all parameter are position independent.
    - Parameter are separated by '|'.
    - *[title=]<text>*: Title of the column. The keyword 'title=' is optional. Columns with a title starting with '_' won't be rendered.
    - *[maxLength=]<number>*: Max. number of characters displayed per cell. The keyword 'maxLength=' is optional. Default maxLength '20'. A value of '0' means no limit. This setting also affects the title of the column.
    - *nostrip*: by default, html tags will be stripped off the cell content before rendering. This protects the table layout. 'nostrip' deactivates the cleaning to make pure html possible.
    - *icon*: the cell value contains the name of an icon in *typo3conf/ext/qfq/Resources/Public/icons*. Empty cell values will omit an html image tag (=nothing rendered in the cell).
    - *link*: value will be rendered as described under *Column: _link*
    - *url*: value will be rendered as a href url.
    - *mailto*: value will be rendered as a href mailto.

    * *_rowClass*
        - The value is a CSS class name(s) which will be rendered in the <*tr class="*<*_rowClass*>*"*> of the subrecord table.
        - The column itself is not rendered.
        - By using Bootstrap, the following predefined classes are available:
        - Text color: *text-muted|text-primary|text-success|text-info|text-warning|text-danger* (http://getbootstrap.com/docs/3.4/css/#helper-classes)
        - Row background: *active|success|info|warning|danger* (http://getbootstrap.com/docs/3.4/css/#tables-contextual-classes)
    * *_rowTooltip*
        - Defines the title attribute (=tooltip) of a subrecord table row.
    * Examples:

```
{{!SELECT id, note1 AS 'Comment', note2 AS 'Comment|50' , note3 AS
↪'title=Comment|maxLength=100|nostrip', note4 AS '50|Comment',
'checked.png' AS 'Status|icon', email AS 'mailto', CONCAT(homepage,
↪'|Homepage') AS 'url',
CONCAT('d|s|F:', pathFileName) AS 'Download|link',
ELT(status,'info','warning','danger') AS '_rowClass', help AS '_rowTooltip
↪' ...}}
```

- *FormElement.parameter*
  - *form* = <*form name*> - Target form, e.g. *form=person*
  - *page* = <*T3 page alias or id*> - Target page with detail form. If none specified, use the current page.
  - *extraDeleteForm*: Optional. The per row delete Button will reference the form specified here (for deleting) instead of the default (*form*).
  - *detail* = <*string*> - Mapping of values from
    * a) the primary form,
    * b) the current row,
    * c) any constant or '{{...}}' -

    to the target form (defined via *form=...*).
    * Syntax:

```
<source table column name 1|&constant 1>:<target column name 1>[,<source␣
↪table column name 2|&constant 2>:<target column name 2>][...]
```

    * Example: *detail=id:personId,rowId:secId,&12:xId,&{{a}}:personId* (rowId is a column of the current selected row defined by sql1)
    * By default, the given value will overwrite values on the target record. In most situations, this is the wished behaviour.
    * Exceptions of the default behaviour have to be defined on the target form in the corresponding *FormElement* in the field *value* by changing the default Store priority definition. E.g. *{{<columnName>:RS0}}* - For existing records, the store *R* will provide a value. For new records, store *R* is empty and store S will be searched for a value: the value defined in *detail* will be choosen. At last the store '0' is defined as a fallback.

    - *source table column name*: E.g. A person form is opened with person.id=5 (r=5). The definition *detail=id:personId* and *form=address* maps person.id to address.personId. On the target record, the column personId becomes '5'.
    - *Constant '&'*: Indicate a 'constant' value. E.g. *&12:xId* or *{{...}}* (all possibilities, incl. further SELECT statements) might be used.
  - *subrecordTableClass*: Optional. Default: 'table table-hover qfq-subrecord-table qfq-color-grey-2'. If given, the default will be overwritten. Example:

```
subrecordTableClass = table table-hover qfq-subrecord-table qfq-table-50
```

  - Tablesorter in Subrecord:

```
subrecordTableClass = table table-hover qfq-subrecord-table tablesorter␣
↪tablesorter-pager tablesorter-filter
```

  - *subrecordColumnTitleEdit*: Optional. Will be rendered as the column title for the new/edit column.
  - *subrecordColumnTitleDelete*: Optional. Will be rendered as the column title for the delete column.

Subrecord DragAndDrop

Subrecords inherently support drag-and-drop, see also *Drag and drop*. The following parameters can be used in the *parameter* field to customize/activate drag-and-drop:

- *orderInterval*: The order interval to be used, default is 10.
- *dndTable*: The table that contains the records to be ordered. If not given, the table name of the form specified via *form=...* is used.
- *orderColumn*: The dedicated order column in the specified dndTable (needs to match a column in the table definition). Default is *ord*.

If *dndTable* is a table with a column *orderColumn*, QFQ automatically applies drag-and-drop logic to the rendered subrecord. It does so by using the subrecord field *sql1*. The *sql1* query should include a column *id* (or *_id*) and a column *ord* (or *_ord*). E.g.:

```
FE.sql1 = {{!SELECT p.id AS _id, p.ord AS _ord, p.name FROM Person WHERE p.email!=''␣
↪ORDER BY p.ord}}
```

Tips:

- If you want to deactivate a drag-and-drop that QFQ automatically renders, set the *orderColumn* to a non-existing column. E.g., *orderColumn = nonExistingColumn*. This will deactivate drag-and-drop.
- In order to evaluate the *sql1* query dynamically during a drag-and-drop event, the STORE_RECORD (with the current subrecord) is loaded.
- The stores STORE_RECORD, STORE_SIP and STORE_SYSTEM are supported during a drag-and-drop event and can be used in FE.sql1 query.
  - STORE_SIP: SIP values on form load
  - STORE_RECORD: values of the current record loaded in the form.
- If the subrecord is rendered with drag-and-drop active, but the order is not affected upon reload, there is most likely a problem with evaluating the *sql1* query at runtime.

9.8.19 Type: time

- Range time: '00:00:00' to '23:59:59' or '00:00:00'. (http://dev.mysql.com/doc/refman/5.5/en/datetime.html)

- Optional:
- *FormElement.parameter*
  - *showSeconds = 0|1* - shows the seconds. Independent if the user specifies seconds, they are displayed '1' or not '0'.
  - *showZero = 0|1* - For an empty timestamp, With '0' nothing is displayed. With '1' the string '00:00[:00]' is displayed.

9.8.20 Type: upload

An upload element is based on a 'file browse'-button and a 'trash'-button (=delete). Only one of them is shown at a time. The 'file browse'-button is displayed, if there is no file uploaded already. The 'trash'-button is displayed, if there is a file uploaded already.

After clicking on the browse button, the user select a file from the local filesystem. After choosing the file, the upload starts immediately, shown by a turning wheel. When the server received the whole file and accepts (see below) the file, the 'file browse'-button disappears and the filename is shown, followed by a 'trash'-button. Either the user is satisfied now or the user can delete the uploaded file (and maybe upload another one).

Until this point, the file is cached on the server but not copied to the *fileDestination*. The user have to save the current record, either to finalize the upload and/or to delete a previously uploaded file.

The FormElement behaves like a

- 'native FormElement' (showing controls/text on the form) as well as an
- 'action FormElement' by firing queries and doing some additional actions during form save.

Inside the *Form editor* it's shown as a 'native FormElement'. During saving the current record, it behaves like an action FormElement and will be processed after saving the primary record and before any action FormElements are processed.

- *FormElement.value = <string>* - By default, the full path of any already uploaded file is shown. To show something different, e.g. only the filename, define:

```
a) {{filenameBase:V}}
b) {{SELECT SUBSTRING_INDEX( '{{pathFileName:R}}', '/', -1)  }}
```

See also *download Button* to offer a download of an uploaded file.

9.8.20.1 FormElement.parameter

- *fileButtonText*: Overwrite default 'Choose File'
- *capture = camera* - On a smartphone, after pressing the 'open file' button, the camera will be opened and a choosen picture will be uploaded. Automatically set/overwrite *accept=image/**.
- *accept = <mime type>,image/*,video/*,audio/*,.doc,.docx,.pdf*
  - List of mime types (also known as 'media types'): http://www.iana.org/assignments/media-types/media-types.xhtml
  - If none mime type is specified, 'application/pdf' is set. This forces that always (!) one type is specified.
  - To allow any type, specify `*` or `*/*` or `*.*`.
  - One or more media types might be specified, separated by ','.
  - Different browser respect the given definitions in different ways. Typically the 'file choose' dialog offer:

        - the specified mime type (some browers only show 'custom', if more than one mime type is given),
        - the option 'All files' (the user is always free to **try** to upload other file types) - but the server won't accept them,
        - the 'file choose' dialog only offers files of the selected (in the dialog) type.
    - If for a specific file type is no mime type available, the definition of file extension(s) is possible. This is **less secure**, cause there is no *content* check on the server after the upload.
- *maxFileSize* = <*size*> - max filesize in bytes (no unit), kilobytes (k/K) or megabytes (m/M) for an uploaded file. If empty or not given, take value from Form, System or System default.
- *fileTrash* = [0|1] - Default: '1'. This option en-/disables the trash button right beside the file chooser. By default the trash is visible. The trash is only visible if a) there is already a file uploaded or b) a new file has been chosen.
- *fileTrashText* = <*string*> - Default: ''. Will be shown right beside the trash glyph-icon.
- *fileDestination* = <*pathFileName*> - Destination where to copy the file. A good practice is to specify a relative *fileDestination* - such an installation (filesystem and database) are moveable.
    - If the original filename should be part of *fileDestination*, the variable *{{filename}}* (STORE_VARS) can be used. Example

```
fileDestination={{SELECT 'fileadmin/user/pictures/', p.name, '-{{filename}}'␣
↪FROM Person AS p WHERE p.id={{id:R0}} }}
```

        - Several more variants of the filename and also mimetype and filesize are available. See *store variables form element upload*.
        - The original filename will be sanitized: only '<alnum>', '.' and '_' characters are allowed. German 'umlaut' will be replaced by 'ae', 'ue', 'oe'. All non valid characters will be replaced by '_'.
    - If a file already exist under *fileDestination*, an error message is shown and 'save' is aborted. The user has no possibility to overwrite the already existing file. If the whole workflow is correct, this situation should no arise. Check also *fileReplace* below.
    - All necessary subdirectories in *fileDestination* are automatically created.
    - Using the current record id in the *fileDestination*: Using {{r}} is problematic for a 'new' primary record: that one is still '0' at the time of saving. Use *{{id:R0}}* instead.
    - Uploading of malicious code (e.g. PHP files) is hard to detect. The default mime type check can be easily faked by an attacker. Therefore it's recommended to use a *fileDestination*-directory, which is secured against script execution (even if the file has been uploaded, the webserver won't execute it) - see *Secure direct file access*.
- *sqlBefore*, *sqlAfter*: available in *Upload simple mode* and *Upload advanced mode*.
- *slaveId*, *sqlInsert*, *sqlUpdate*, *sqlDelete*, *sqlUpdate*: available only in *Upload advanced mode*.
- *fileSize* / *mimeType*
    - In *Upload simple mode* the information of *fileSize* and *mimeType* will be automatically updated on the current record, if table columns *fileSize* and/or *mimeType* exist.
        - If there are more than one Upload FormElement in a form, the automatically update for *fileSize* and/or *mimeType* are not done automatically.
    - In *Upload advanced mode* the *fileSize* and / or *mimeType* have to be updated with an explicit SQL statement:

```
sqlAfter = {{UPDATE Data SET mimeType='{{mimeType:V}}', fileSize={
↪{fileSize:V}} WHERE id={{id:R}} }}
```

- *fileReplace* = *always* - If *fileDestination* exist - replace it by the new one.
- *chmodFile* = <unix file permission mode> - e.g. *660* for owner and group read and writeable. Only the numeric mode is allowed.
- *chmodDir* = <unix file permission mode> - e.g. *770* for owner and group read, writeable and executable. Only the numeric mode is allowed. Will be applied to all new created directories.
- *autoOrient:* images might contain EXIF data (e.g. captured via mobile phones) incl. an orientation tag like TopLeft, BottomRight and so on. Web-Browser and other grafic programs often understand and respect those information and rotate such images automatically. If not, the image might be displayed in an unwanted oritentation. With active option 'autoOrient', QFQ tries to normalize such images via 'convert' (part of ImageMagick). Especially if images are processed by the QFQ internal 'Fabric'-JS it's recommended to normalize images first. The normalization process does not solve all orientation problems.
  - *autoOrient* = [0|1]
  - *autoOrientCmd* = 'convert -auto-orient {{fileDestination:V}} {{fileDestination:V}}.new; mv {{fileDestination:V}}.new {{fileDestination:V}}'
  - *autoOrientMimeType* = image/jpeg,image/png,image/tiff

  If the defaults for *autoOrientCmd* and *autoOrientMimeType* are sufficient, it's not necessary to specify them.
- *downloadButton* = *t:<string>* - If given, shows a button to download the previous uploaded file - instead of the string given in *fe.value*. The button is only shown if *fe.value* points to a readable file on the server.
  - If *downloadButton* is empty, just shows the regular download glyph.
  - To just show the filename: *downloadButton* = *t:{{filenameOnly:V}}*
  - Additional attributes might be given like *downloadButton* = *t:Download|o:check file*. Please check *Download*.
    - The following attributes are hard coded (can't be changed): *s|M:file|d|F*
- *fileUnzip* - If the file is a ZIP file (only then) it will be unzipped. If no directory is given via `fileUnzip`, the basedir of `fileDestination` is taken, appended by `unpack`.

  If an unzip will be done, for each file of the archive STORE_VAR will be filled (name, path of the extracted file, mime type, size) and the following will be triggered: *sqlValidate, slaveId, sqlBefore, sqlAfter, sqlInsert, sqlUpdate*.

  Example:

```
fileDestination = fileadmin/file_{{id:R}}.zip
fileUnzip
sqlValidate ={{! SELECT '' FROM (SELECT '') AS fake WHERE '{{mimeType:V}}' LIKE
↪'application/pdf%' }}
expectRecords=1
messageFail=Unexpected filetype

# Set new
sqlAfter={{INSERT INTO Upload (pathFileName) VALUES '{{filename:V}}' }}
```

- *fileSplit*, *fileDestinationSplit*, *tableNameSplit*: see *Split PDF Upload*
- Excel Import: QFQ offers functionality to directly import excel data into the database. This functionality can optionally be combined with saving the file by using the above parameters like *fileDestination*. The data is imported without formatting. Please note that this means Excel dates will be imported as a number (e.g. 43214), which is the serial value date in Excel. To convert such a number to a MariaDb date, use: *DATE_ADD('1899-12-30', INTERVAL serialValue DAY)*.

- *importToTable* = <[db.]tablename> - **Required**. Providing this parameter activates the import. If the table doesn't exist, it will be created.
- *importToColumns* = <col1>,<col2>,... - If none provided, the Excel column names A, B, ... are used. Note: These have to match the table's column names if the table already exists.
- *importRegion* = [tab],[startColumn],[startRow],[endColumn],[endRow]|... - All parts are optional (default: entire 1st sheet). Tab can either be given as an index (1-based) or a name. start/endColumn can be given either numerically (1, 2, ...) or by column name (A, B, ...). Note that you can specify several regions to import.
- *importMode* = *append* (default) | *replace* - The data is either appended or replace in the specified table.
- *importType* = *auto* (default) | *xls* | *xlsx* | *ods* | *csv* - Define what kind of data should be expected by the Spreadsheet Reader.
- *importNamedSheetsOnly* = <comma separated list of sheet names>. Use this option if specific sheets cause problems during import and should be skipped, by naming only those sheets, who will be read. This will also reduce the memory usage.
- *importSetReadDataOnly* = 0|1. Read only cell data, not the cell formatting. Warning: cell types other than numerical will be misinterpreted.
- *importListSheetNames* = 0|1. For debug use only. Will open a dialog and report all found worksheet names.

Immediately after the upload finished (before the user press save), the file will be checked on the server for it's content or file extension (see 'accept').

The maximum size is defined by the minimum of *upload_max_filesize*, *post_max_size* and *memory_limit* (PHP script) in the php.ini.

In case of broken uploads, please also check *max_input_time* in php.ini.

9.8.20.2 Deleting a record and the referenced file

If the user deletes a record (e.g. pressing the delete button on a form) which contains reference(s) to files, such files are deleted too. Slave records, which might be also deleted through a 'delete'-form, are *not* checked for file references and therefore such files are not deleted on the filesystem.

Only column(name)s which contains *pathFileName* as part of their name, are checked for file references.

If there are other records, which references the same file, such files are not deleted. It's a very basic check: just the current column of the current table is compared. In general it's not a good idea to have multiple references to a single file. Therefore this check is just a fallback.

9.8.20.3 Upload simple mode

Requires: *'upload'-FormElement.name* = *'column name'* of an column in the primary table.

After moving the file to *fileDestination*, the current record/column will be updated to *fileDestination*. The database definition of the named column has to be a string variant (varchar, text but not numeric or else). On form load, the column value will be displayed as the whole value (pathFileName)

Deleting an uploaded file in the form (by clicking on the trash near beside) will delete the file on the filesystem as well. The column will be updated to an empty string.

This happens automatically without any further definiton in the 'upload'-FormElement.

Multiple 'upload'-FormElements per form are possible. Each of it needs an own table column.

9.8.20.4 Upload advanced mode

Requires: *'upload'-FormElement.name* is unknown as a column in the primary table.

This mode will serve further database structure scenarios.

A typical name for such an 'upload'-FormElement, to show that the name does not exist in the primary table, might start with 'my', e.g. 'myUpload1'.

- *FormElement.value* = <*string*> - The path/filename, shown during 'form load' to indicate a previous uploaded file, has to be queried with this field. E.g.:

```
{{SELECT pathFileNamePicture FROM Note WHERE id={{slaveId}} }}
```

- *FormElement.parameter*:
  - *fileDestination* = <*pathFileName*> - determine the path/filename. E.g.:

```
fileDestination=fileadmin/person/{{name:R0}}_{{id:R}}/uploads/picture_{
↪{filename}}
```

  - *slaveId* = <*id*> - Defines the target record where to retrieve and store the path/filename of the uploaded file. Check also *Parameter: slaveId*. E.g.:

```
slaveId={{SELECT id FROM Note WHERE pId={{id:R0}} AND type='picture' LIMIT 1}
↪}
```

  - *sqlBefore* = *{{<query>}}* - fired during a form save, before the following queries are fired.
  - *sqlInsert* = *{{<query>}}* - fired if *slaveId=0* and an upload exist (user has choosen a file):

```
sqlInsert={{INSERT INTO Note (pId, type, pathFileName) VALUE ({{id:R0}},
↪'image', '{{fileDestination}}') }}
```

  - *sqlUpdate* = *{{<query>}}* - fired if *slaveId>0* and an upload exist (user has choosen a file). E.g.:

```
sqlUpdate={{UPDATE Note SET pathFileName = '{{fileDestination}}' WHERE id={
↪{slaveId}} LIMIT 1}}
```

  - *sqlDelete* = *{{<query>}}* - fired if *slaveId>0* and no upload exist (user has not choosen a file). E.g.:

```
sqlDelete={{DELETE FROM Note WHERE id={{slaveId:V}}  LIMIT 1}}
```

  - *sqlAfter* = *{{<query>}}* - fired after all previous queries have been fired. Might update the new created id to a primary record. E.g.:

```
sqlAfter={{UPDATE Person SET noteIdPicture = {{slaveId}} WHERE id={{id:R0}}␣
↪LIMIT 1 }}
```

9.8.20.5 Split PDF Upload

Additional to the upload, it's possible to split the uploaded file (only PDF files) into several SVG or JPEG files, one file per PDF page. The split is done via a) http://www.cityinthesky.co.uk/opensource/pdf2svg/ or b) Image Magick *convert*.

Currently, QFQ can only split PDF files.

If the source file is not of type PDF, activating `fileSplit` has no impact: no split and NO complain about invalid file type.

- *FormElement.parameter*:
  - *fileSplit* = <*type*> - Activate the splitting process. Possible values: *svg* or *jpeg*. No default.
  - *fileSplitOptions* = <*command line options*>.
    - [svg] - no default
    - [jpeg] - default: *-density 150 -quality 90*
  - *fileDestinationSplit* = <*pathFileName (pattern)*> - Target directory and filename pattern for the created & split'ed files. Default <fileDestination>.split/split.<nr>.<fileSplit>. If explicit given, respect that SVG needs a printf style for <nr>, whereas JPEG is numbered automatically. E.g.

```
[svg] fileDestinationSplit = fileadmin/protected/{{id:R}}.{{filenameBase:V}}.
↪%02d.svg
[jpeg] fileDestinationSplit = fileadmin/protected/{{id:R}}.{{filenameBase:V}}
↪.jpg
```

  - *tableNameSplit* = <*tablename*> - Default: name of table of current form. This name will be saved in table *Split*

The splitting happens immediately after the user pressed *save*.

To easily access the split files via QFQ, per file one record is created in table 'Split'.

Table 'Split':

Column	Description
id	Uniq auto increment index
tableName	Name of the table, where the reference to the original file (multipage PDF file) is saved.
xId	Primary id of the reference record.
pathFileName	Path/filename reference to one of the created files

One usecase why to split an upload: annotate individual pages by using the *FormElement*.type='annotate'.

9.9 Class: Action

9.9.1 Type: before... | after...

These type of 'action' *FormElements* will be used to implement data validation or creating/updating additional records.

Types:

- beforeLoad (e.g. good to check access permission)
- afterLoad
- beforeSave (e.g. to prohibit creating of duplicate records)
- afterSave (e.g. to to create & update additional records)
- beforeInsert
- afterInsert
- beforeUpdate
- afterUpdate
- beforeDelete (e.g. to delete slave records)

- afterDelete
- paste (configure copy/paste forms)

9.9.1.1 Parameter: sqlValidate

Perform checks by firing an SQL query and expecting a predefined number of selected records.

- OK: the *expectRecords* number of records has been selected. Continue processing the next *FormElement*.
- Fail: the *expectRecords* number of records has not been selected (less or more): Display the error message *messageFail* and abort the whole (!) current form load or save.

FormElement.parameter:

- *requiredList* = <*fe.name[s]*> - List of *native-FormElement* names: only if all of those elements are filled (!=0 and !=''), the *current action-FormElement* will be processed. This will enable or disable the check, based on the user input! If no *native-FormElement* names are given, the specified check will always be performed.
- *sqlValidate* = *{{<query>}}* - validation query. E.g.: *sqlValidate={{SELECT id FROM Person AS p WHERE p.name LIKE {{name:F:all}} AND p.firstname LIKE {{firstname:F:all}} }}*
- *expectRecords* = <*value*>- number of expected records.
  - *expectRecords = 0* or *expectRecords = 0,1* or *expectRecords = {{SELECT COUNT(id) FROM Person}}*
  - Separate multiple valid record numbers by ','. If at least one of those matches, the check will pass successfully.
- *messageFail* = <*string*> - Message to show. E.g.: *messageFail = There is already a person called {{firstname:F:all}} {{name:F:all}}*

9.9.1.2 Parameter: slaveId

FormElement.parameter

- *slaveId* = <*id*>:
  - Auto fill: name the action *action-FormElement* equal to an existing column (table from the current form definition). *slaveId* will be automatically filled with the value of the named column.
    - If there is no such named column name, set *slaveId = 0*.
  - Explicit definition: *slaveId = 123* or *slaveId = {{SELECT id . . . }}*

Note:

- *{{slaveId:V}}* can be used in any query of the current *FormElement*.
- If the *action-FormElement* name exist as a column in the master record: Update that column *automatically* with the recent slaveId
- After an INSERT the *last_insert_id()* becomes the *{{slaveId:V}}*.
- *fillStoreVar* is fired first, than *slaveId*.
- If *slaveId* is known in *fillStoreVar*, set: *slaveId={{someId:V}}*.

9.9.1.3 Parameter: sqlBefore / sqlInsert / sqlUpdate / sqlDelete / sqlAfter

- Save values of a form to different record(s), optionally on different table(s).
- Typically useful on 'afterSave' - be careful when using it earlier, e.g. beforeLoad.

FormElement.parameter

- *requiredList* = *<fe.name[s]>* - List of *native-FormElement*: only if all of those elements are filled, the current *action-FormElement* will be processed.
- *sqlBefore* = *{{<query>}}* - always fired (before any *sqlInsert*, *sqlUpdate*, ..)
- *sqlInsert* = *{{<query>}}* - fired if *slaveId* == *0* or *slaveId* == *''*.
- *sqlUpdate* = *{{<query>}}* - fired if *slaveId* > *0*.
- *sqlDelete* = *{{<query>}}* - fired if *slaveId* > *0*, after *sqlInsert* or *sqlUpdate*. Be careful not to delete filled records! Always add a check, if values given, not to delete the record! *sqlHonorFormElements* helps to skip such checks.
- *sqlAfter* = *{{<query>}}* - always fired (after *sqlInsert*, *sqlUpdate* or *sqlDelete*).
- *sqlHonorFormElements* = *<fe.name[s]>* list of *FormElement* names (this parameter is optional).
  - If one of the named *FormElements* is not empty:
    - fire *sqlInsert* if *slaveId* == *0*,
    - fire *sqlUpdate* if *slaveId* > *0*
  - If all of the named *FormElements* are empty:
    - fire *sqlDelete* if *slaveId* > *0*

9.9.1.4 Example

Situation 1: master.xId=slave.id (1:1)

- Name the action element 'xId': than {{slaveId}} will be automatically set to the value of 'master.xId'
  - {{slaveId}} == 0 ? 'sqlInsert' will be fired.
  - {{slaveId}} != 0 ? 'sqlUpdate' will be fired.
- In case of firing 'sqlInsert', the 'slave.id' of the new created record are copied to master.xId (the database will be updated automatically).
- If the automatic update of the master record is not suitable, the action element should have no name or a name which does not exist as a column of the master record. Define *slaveId={{SELECT id ... }}*
- Two *FormElements myStreet* and *myCity*:
  - Without *sqlHonorFormElements*. Parameter:

```
sqlInsert = {{INSERT INTO address (`street`, `city`) VALUES ('{
↪{myStreet:FE:alnumx:s}}', '{{myCity:FE:alnumx:s}}') }}
sqlUpdate = {{UPDATE address SET `street` = '{{myStreet:FE:alnumx:s}}',␣
↪`city` = '{{myCity:FE:alnumx:s}}'  WHERE id={{slaveId}} LIMIT 1 }}
sqlDelete = {{DELETE FROM Address WHERE id={{slaveId}} AND '{
↪{myStreet:FE:alnumx:s}}''' AND '{{myCity:FE:alnumx:s}}''' LIMIT 1 }}
```

  - With *sqlHonorFormElements*. Parameter:

```
sqlHonorFormElements = myStreet, myCity     # Non Templategroup
sqlInsert = {{INSERT INTO address (`street`, `city`) VALUES ('{
↪{myStreet:FE:alnumx:s}}', '{{myCity:FE:alnumx:s}}') }}
sqlUpdate = {{UPDATE address SET `street` = '{{myStreet:FE:alnumx:s}}',␣
↪`city` = '{{myCity:FE:alnumx:s}}'  WHERE id={{slaveId}} LIMIT 1 }}
sqlDelete = {{DELETE FROM Address WHERE id={{slaveId}} LIMIT 1 }}

# For Templategroups: sqlHonorFormElements = myStreet%d, myCity%d
```

Situation 2: master.id=slave.xId (1:n)

- Name the action element *different* to any column name of the master record (or no name).
- Determine the slaveId: *slaveId={{SELECT id FROM Slave WHERE slave.xxx={{...}} LIMIT 1}}*
  - {{slaveId}} == 0 ? 'sqlInsert' will be fired.
  - {{slaveId}} != 0 ? 'sqlUpdate' will be fired.
- Two *FormElements myStreet* and *myCity*. The *person* is the master record, *address* is the slave:
  - Without *sqlHonorFormElements*. Parameter:

```
slaveId = {{SELECT id FROM Address WHERE personId={{id}} ORDER BY id LIMIT 1␣
↪}}
sqlInsert = {{INSERT INTO address (`personId`, `street`, `city`) VALUES ({
↪{id}}, '{{myStreet:FE:alnumx:s}}', '{{myCity:FE:alnumx:s}}') }}
sqlUpdate = {{UPDATE address SET `street` = '{{myStreet:FE:alnumx:s}}',␣
↪`city` = '{{myCity:FE:alnumx:s}}'  WHERE id={{slaveId}} LIMIT 1 }}
sqlDelete = {{DELETE FROM Address WHERE id={{slaveId}} AND '{
↪{myStreet:FE:alnumx:s}}'='' AND '{{myCity:FE:alnumx:s}}'='' LIMIT 1 }}
```

  - With *sqlHonorFormElements*. Parameter:

```
slaveId = {{SELECT id FROM Address WHERE personId={{id}} ORDER BY id LIMIT 1␣
↪}}
sqlHonorFormElements = myStreet, myCity       # Non Templategroup
sqlInsert = {{INSERT INTO address (`personId`, `street`, `city`) VALUES ({
↪{id}}, '{{myStreet:FE:alnumx:s}}', '{{myCity:FE:alnumx:s}}') }}
sqlUpdate = {{UPDATE address SET `street` = '{{myStreet:FE:alnumx:s}}',␣
↪`city` = '{{myCity:FE:alnumx:s}}'  WHERE id={{slaveId}} LIMIT 1 }}
sqlDelete = {{DELETE FROM Address WHERE id={{slaveId}} LIMIT 1 }}

# For Templategroups: sqlHonorFormElements = myStreet%d, myCity%d
```

9.9.2 Type: sendmail

- Send mail(s) will be processed after:
  - saving the record ,
  - processing all uploads,
  - together with *after...* action *FormElements* in the given order.
- *FormElement.value* = <*string*> - Body of the email. See also: *html-formatting*
- *FormElement.parameter*:

- *sendMailTo* = <*string*> - Comma-separated list of receiver email addresses. Optional: 'realname <john@doe.com>. If there is no recipient email address, **no** mail will be sent.
- *sendMailCc* = <*string*> - Comma-separated list of receiver email addresses. Optional: 'realname <john@doe.com>.
- *sendMailBcc* = <*string*> - Comma-separated list of receiver email addresses. Optional: 'realname <john@doe.com>.
- *sendMailFrom* = <*string*> - Sender of the email. Optional: 'realname <john@doe.com>'. **Mandatory**.
- *sendMailSubject* = <*string*> - Subject of the email.
- *sendMailReplyTo* = <*string*> - Reply this email address. Optional: 'realname <john@doe.com>'.
- *sendMailAttachment* = <*string*> - List of 'sources' to attach to the mail as files. Check *Attachment* for options.
- *sendMailHeader* = <*string*> - Specify custom header.
- *sendMailFlagAutoSubmit* = <*string*> - **on|off** - If 'on' (default), the mail contains the header 'Auto-Submitted: auto-send' - this suppress a) OoO replies, b) forwarding of emails.
- *sendMailGrId* = <*string*> - Will be copied to the mailLog record. Helps to setup specific logfile queries.
- *sendMailXId* = <*string*> - Will be copied to the mailLog record. Helps to setup specific logfile queries.
- *sendMailXId2* = <*string*> - Will be copied to the mailLog record. Helps to setup specific logfile queries.
- *sendMailXId3* = <*string*> - Will be copied to the mailLog record. Helps to setup specific logfile queries.
- *sendMailMode* = <*string*> - **html** - if set, the e-mail body will be rendered as html.
- *sendMailSubjectHtmlEntity* = <*string*> - **encode|decode|none** - the mail subject will be htmlspecialchar() encoded / decoded (default) or none (untouched).
- *sendMailBodyHtmlEntity*= '<*string*>' - ***encode|decode|none** - the mail body will be htmlspecialchar() encoded, decoded (default) or none (untouched).
- *sqlBefore* / *sqlAfter* = <*string*> - can be used like with other action elements (will be fired before/after sending the e-mail).

- An **empty** *sendMailTo* will **cancel** any sendmail action, even if *sendMailCc|Bcc* is set. This can be used to determine during runtime if sending is wished.
- To use values of the submitted form, use the STORE_FORM. E.g. *{{name:F:allbut}}*
- To use the *id* of a new created or already existing primary record, use the STORE_RECORD. E.g. *{{id:R}}*.
- By default, QFQ stores values 'htmlspecialchars()' encoded. If such values have to send by email, the html entities are unwanted. Therefore the default setting for 'subject' und 'body' is to decode the values via 'html-specialchars_decode()'. If this is not wished, it can be turned off by *sendMailSubjectHtmlEntity=none* and/or *sendMailBodyHtmlEntity=none*.
- For debugging, please check *Redirect all mail to (catch all)*.

Example to attach one file1.pdf (with the attachment filename 'readme.pdf') and concatenate two PDF, created on the fly from the www.example.com and ?export (with the attachment filename 'personal.pdf'):

```
sendMailAttachmemt = F:fileadmin/file1.pdf|d:readme.pdf|C|u:http://www.example.com|p:?
↪id=export&r=123&_sip=1|d:personal.pdf
```

9.9.3 Type: paste

See also *Copy Form*.

- *sql1 = {{<query>}}* - e.g. *{{!SELECT {{id:P}} AS id, '{{myNewName:FE:allbut}}' AS name}}* (only one record) or *{{!SELECT i.id AS id, {{basketId:P}} AS basketId FROM Item AS i WHERE i.basketId={{id:P}} }}* (multiple records)
  - Pay attention to '!'.
  - For every row, a new record is created in *recordDestinationTable*.
  - Column 'id' is not copied.
  - The *recordSourceTable* together with column *id* will identify the source record.
  - Columns not specified, will be copied 1:1 from source to destination.
  - Columns specified, will overwrite the source value.
- *FormElement.parameter*:
  - *recordSourceTable* = <*tableName*> - Optional: table from where the records will be copied. Default: <recordDestinationTable>
  - *recordDestinationTable* = <*tableName*> - table where the new records will be copied to.
  - *translateIdColumn* = <*column name*> - column name to update references of newly created id's.

9.10 Form Magic

9.10.1 Parameter

- Table column *id*: QFQ expect that each table, which will be loaded in a form, contains an autoincrement column of name *id*. It's not necessary to create a FormElement *id* in a form - but it won't disturb.
- Parameter (one or more) in the SIP url, which exist as a column in the form table (SIP parameter name is equal to a table column name), will be automatically saved in the record. This acts as 'hidden magic'.

  Example: A slave record (e.g. an address of a person) has to be assigned to a master record (a person). Just give the *pId* in the link who calls the address form. The following creates a 'new' button for an address for all persons, and the pId will be automatically saved in the address table:

```
SELECT CONCAT('p:{{pageAlias:T}}&form=address&r=0&pId=', p.id) AS _pagen FROM␣
↪Person AS p
```

  Such parameter, which the form expects to be in the SIP url, should be specified in Form.permitNew and/or Form.permitEdit. It's only a check for the webmaster, not to forgot a parameter in a SIP url.
- FormElement.type = subrecord

  Subrecord's will automatically create *new*, *edit* and *delete* links. To inject parameter in those automatically created links, use *FormElement.parameter.detail* . See *Type: subrecord*.
- FormElement.type = extra

  If a table column should be saved with a specific value, and the value should not be shown to the user, the FE.type='extra' will do the job. The value could be static or calculated on the fly. Often it's easier to specify such a parameter/value in the SIP url, but if the form is called from multiple places, an *extra* element is more suitable.

9.10.2 Variables

- Form.parameter.fillStoreVar / FormElement.parameter.fillStoreVar

  An SQL statement will fill STORE_VARS. Such values can be used during form load and/or save.

9.10.3 Action

- Action FE

  Via *FormElement.parameter.requiredList* an element can be enabled / disabled, depending of a user provided input in one of the specified required FEs.

9.11 Multi Form

Multi Forms are like a regular form with the difference that the shown FormElements are repeated for *each* selected record (defined by *multiSql*).

Name		
multiSql	{{!SELECT id, name FROM Person}}	Query to select MulitForm records
multiMgsNoRecord	Default: No data	Message shown if *multiSql* selects no records

- Multi Form do not use 'record-locking' at all.

The Form is shown as a HTML table.

- *multiSql*: Selects the records where the defined FormElements will work on each.
  - A uniq column 'id' or '_id' (not shown) is mandatory and has to reflect an existing record id in table *primary table*.
  - Additional columns, defined in *multiSql*, will be shown on the form in the same line, before the FormElements.

9.11.1 Simple

General:

- It's not possible to create new records in simple mode, only existing records can be modified.

Form:

- Per row, the STORE_RECORD is filled with the whole record of the primary table, referenced by *multiSql.id*.

FormElement:

- The FormElement.name represents a column of the defined primary table.
- The existing values of such FormElements are automatically loaded.
- No further definition is required.

9.11.2 Advanced

To handle foreign records (insert/update/delete), use the *Parameter: slaveId* concept.

Typically the *FormElement.name* is not a column of the primary table.

9.12 Multiple languages

QFQ Forms might be configured for up to 5 different languages. Per language there is one extra field in the *Form editor*. Which field represents which language is configured in *Configuration*.

- The Typo3 installation needs to be configured to handle different languages - this is independent of QFQ and not covered here. QFQ will use the Typo3 internal variable 'pageLanguage', which typically correlates to the URL parameter 'L' in the URL.
- In *Configuration* the Typo3 language index (value of 'L') and a language label have to be configured for each language. Only than, the additional language fields in the *Form editor* will be shown.

9.12.1 Example

Assuming the Typo3 page has the

- default language, L=0
- English, L=1
- Spanish, L=2

Configuration in *Configuration*:

```
formLanguageAId = 1
formLanguageALabel = English

formLanguageBId = 2
formLanguageBLabel = Spanish
```

The default language is not covered in *Configuration*.

The *Form editor* now shows on the pill 'Basic' (Form and FormEditor) for both languages each an additional parameter input field. Any input field in the *Form editor* can be redeclared in the corresponding language parameter field. Any missing definition means 'take the default'. E.g.:

- Form: 'person'

Column	Value
title	Eingabe Person
languageParameterA	title=Input Person
languageParameterB	title=Persona de entrada

- FormElement 'firstname' in Form 'person':

Column	Value
title	Vorname
note	Bitte alle Vornamen erfassen
languageParameterA	title=Firstname note=Please give all firstnames
languageParameterB	title=Persona de entrada note=Por favor, introduzca todos los nombres

The following fields are possible:

- Form: *title, showButton, forwardMode, forwardPage, bsLabelColumns, bsInputColumns, bsNoteColumns, recordLockTimeoutSeconds*
- FormElement: *label, mode, modeSql, class, type, subrecordOption, encode, checkType, ord, size, maxLength, bsLabelColumns, bsInputColumns, bsNoteColumns,rowLabelInputNote, note, tooltip, placeholder, value, sql1, feGroup*

9.13 Dynamic Update

The 'Dynamic Update' feature makes a form more interactive. If a user changes a *FormElement* who is tagged with 'dynamicUpdate', *all* elements who are tagged with 'dynamicUpdate', will be recalculated and rerendered.

The following fields will be recalculated during 'Dynamic Update'

- 'modeSql' - Possible values: 'show', 'required', 'readonly', 'hidden'
- 'label'
- 'value'
- 'note'
- 'parameter.*' - especially 'itemList'

To make a form dynamic:

- Mark all *FormElements* with *dynamic update*'='*enabled*, which should **initiate** or **receive** updates.

See #3426 / Dynamic Update: Inputs loose the new content and shows the old value:

- On **all** *dynamic update FormElements* an explicit definition of *value*, including a sanitize class, is necessary (except the field is numeric). **A missing definition let's the content overwrite all the time with the old value**. A typical definition for *value* looks like (default store priority is: FSRVD):

```
{{<FormElement name>::alnumx}}
```

- Define the receiving *FormElements* in a way, that they will interpret the recent user change! The form variable of the specific sender *FormElement {{<sender element>:F:<sanitize>}}* should be part of one of the above fields to get an impact. E.g.:

```
[receiving *FormElement*].parameter: itemList={{ SELECT IF({
↪{carPriceRange:FE:alnumx}}='expensive','Ferrari,Tesla,Jaguar','General Motors,
↪Honda,Seat,Fiat') }}
```

(continues on next page)

(continued from previous page)

Remember to specify a 'sanitize' class - a missing sanitize class means 'digit', every content, which is not numeric, violates the sanitize class and becomes therefore an empty string!

- If the dynamic update should work on existing and *new* records, it's important to guarantee that the query result is not empty! even if the primary record does not exist! E.g. use a *LEFT JOIN*. The following query is ok for *new* and *edit*.

```
{{SELECT IF( IFNULL(adr.type,'') LIKE '%token%','show','hidden') FROM (SELECT 1)␣
↪AS fake LEFT JOIN Address AS adr ON adr.type='{{type:FR0}}' LIMIT 1}}
```

9.13.1 Examples

- Master FormElement 'music' is a radio/enum of 'classic', 'jazz', 'pop'.

9.13.1.1 Content of a select list

- Slave FormElement 'interpret' is 'select'-list, depending of 'music'

```
sql={{!SELECT name FROM Interpret WHERE music={{music:FE:alnumx}} ORDER BY name}}
```

9.13.1.2 Show / Hide a *FormElement*

- Slave 'interpret' is displayed only for 'pop'. Field 'modeSql':

```
{{SELECT IF( '{{music:FR:alnumx}}'='pop' ,'show', 'hidden' ) }}
```

9.14 Form Layout

The forms will be rendered with Bootstrap CSS classes, based on the 12 column grid model (Bootstrap 3.x). Generally a 3 column layout for *label* columns on the left side, an *input* field column in the middle and a *note* column on the right side will be rendered.

The used default column (=bootstrap grid) width is *3,6,3* (col-md , col-lg) for *label, input, note*.

- The system wide defaults can be changed via *Configuration*.
- Per *Form* settings can be done in the *Form* parameter field. They overwrite the system wide default.
- Per *FormElement* settings can be done in the *FormElement* parameter field. They overwrite the *Form* setting.

A column will be switched off (no wrapping via <*div class='col-md-?*>) by setting a *0* on the respective column.

9.14.1 Custom field width

Per *FormElement* set *BS Label Columns*, *BS Input Columns* or *BS Note Columns* to customize an individual width. If only a number is specified, it's used as *col-md-<number>*. Else the whole text string is used as CSS class, e.g. *col-md-3 col-lg-2*.

9.14.2 Multiple Elements per row

Every row is by default wrapped in a <*div class='form-group'*> and every column is wrapped in a <*div class='col-md-?'*>. To display multiple input elements in one row, the wrapping of the *FormElement* row and of the three columns can be customized via the checkboxes of *Label / Input / Note*. Every open and every close tag can be individually switched on or off.

E.g. to display 2 *FormElements* in a row with one label (first *FormElement*) and one note (last *FormElement*) we need the following (switch off all non named):

- First *FormElement*
  - open row tag: *row* ,
  - open and close label tag: *label*, */label*,
  - open and close field tag: *input*, */input*,
- Second *FormElement*
  - open and close field tag: *input*, */input*,
  - open and close note tag: *note*, */note*,
  - close row tag: */row* ,

9.15 Copy Form

Records (=master) and child records can be duplicated (=copied) by a regular *Form*, extended by *FormElements* of type 'paste'. A 'copy form' works either in:

- 'copy and paste now' mode: the 'select' and 'paste' *Form* is merged in one form, only one master record is possible,
- 'copy now, paste later' mode: the 'select' *Form* selects master record(s), the 'paste' Form paste's them later.

9.15.1 Concept

A 'select action' (e.g. a *Form* or a button click) creates record(s) in the table *Clipboard*. Each clipboard record contains:

- the 'id(s)' of the record(s) to duplicate,
- the 'paste' form id (that *Form* defines, to which table the master records belongs to, as well as rules of how to duplicate any slave records) and where to copy the new records
- user identifier (QFQ cookie) to separate clipboard records of different users inside the Clipboard table.

The 'select action' is also responsible to delete old clipboard records of the current user, before new clipboard records are created.

The 'paste form' iterates over all master record id(s) in the *Clipboard* table. For each master record id, all FormElements of type *paste* are fired (incl. the creating of slave records).

E.g. if there is a basket with different items and you want to duplicate the whole basket including new items, create a form with the following parameter

- Form
  - Name: *copyBasket*

  - Table: *Clipboard*
  - Show Button: only *close* and *save*
- FormElement 1: Record id of the source record.
  - Name: *idSrc*
  - Lable: *Source Form*
  - Class: *native*
  - Type: *select*
  - sql1: *{{! SELECT id, title FROM Basket }}*
- FormElement 2: New name of the copied record.
  - Name: *myNewName*
  - Class: *native*
  - Type: *text*
- FormElement 3: a) Check that there is no name conflict. b)Purge any old clipboard content of the current user.
  - Name: *clearClipboard*
  - Class: *action*
  - Type: *beforeSave*
  - Parameter:
    - *sqlValidate={{SELECT f.id FROM Form AS f WHERE f.name LIKE '{{myName:FE:alnumx}}' LIMIT 1}}*
    - *expectRecords = 0*
    - *messageFail = There is already a form with this name*
    - *sqlAfter={{DELETE FROM Clipboard WHERE cookie='{{cookieQfq:C0:alnumx}}' }}*
- FormElement 4: Update the clipboard source reference, with current {{cookieQfq:C}} identifier.
  - Name: *updateClipboardRecord*
  - Class: *action*
  - Type: *afterSave*
  - Parameter: *sqlAfter={{UPDATE Clipboard SET cookie='{{cookieQfq:C0:alnumx}}', formId-Paste={{formId:S0}} /* PasteForm */ WHERE id={{id:R}} LIMIT 1 }}*
- FormElement 5: Copy basket identifier.
  - Name: *basketId*
  - Class: *action*
  - Type: *paste*
  - sql1: *{{!SELECT {{id:P}} AS id, '{{myNewName:FE:allbut}}' AS name}}*
  - Parameter: *recordDestinationTable=Basket*
- FormElement 6: Copy items of basket.
  - Name: *itemId*
  - Class: *action*

  - Type: *paste*
  - sql1: *{{!SELECT i.id AS id, {{basketId:P}} AS basketId FROM Item AS i WHERE i.basketId={{id:P}} }}*
  - Parameter: *recordDestinationTable=Item*

9.15.2 Table self referencing records

Records might contain references to other records in the same table. E.g. native FormElements might assigned to a fieldSet, templateGroup or pill, a fieldSet might assigned to other fieldsets or pills and so on. When duplicating a *Form* and the corresponding *FormElements* all internal references needs to be updated as well.

On each FormElement.type='paste' record, the column to be updated is defined via:

- parameter: translateIdColumn = <column name>

For the 'copyForm' this would be 'feIdContainer'.

The update of the records is started after all records have been copied (of the specific FormElement.type='paste' record).

9.16 Delete Record

Deleting record(s) via QFQ might be solved by either:

- using the *delete* button on a form on the top right corner.
- by letting *Report* creating a special link (see below). The link contains the record id and:
  - a form name, or
  - a table name.

Deleting a record just by specifying a table name, will only delete the defined record (no slave records).

- By using a delete button via *report* or in a *subrecord* row, a ajax request is send.
- By using a delete button on the top right corner of the form, the form will be closed after deleting the record.

Example for report:

```
SELECT p.name, CONCAT('U:form=person&r=', p.id) AS _paged FROM Person AS p
SELECT p.name, CONCAT('U:table=Person&r=', p.id) AS _paged FROM Person AS p
```

To automatically delete slave records, use a form and create *beforeDelete* FormElement(s) on the form:

- class: action
- type: beforeDelete
- parameter: sqlAfter={{DELETE FROM <slaveTable> WHERE <slaveTable>.<masteId>={{id:R}} }}

You might also check the form 'form' how the slave records 'FormElement' will be deleted.

9.17 Locking Record / Form

Support for record locking is given with mode:

- *exclusive*: user can't force a write.

  - Including a timeout (default 15 mins recordLockTimeoutSeconds in *Configuration*) for maximum lock time.

- *advisory*: user is only warned, but allowed to overwrite.
- *none*: no bookkeeping about locks.

For 'new' records (r=0) there is no locking at all.

The record locking protection is based on the *tablename* and the *record id*. Different *Forms*, with the same primary table, will be protected by record locking. On the other side, action-*FormElements* updating non primary table records are not protected by 'record locking': the QFQ record locking is *NOT 100%*.

The 'record locking' mode will be specified per *Form*. If there are multiple Forms with different modes, and there is already a lock for a *tablename* / *record id* pair, the most restrictive will be applied.

9.18 Best practice

9.18.1 View: List vs. Detail

As 'list' a number of data/rows shown on the page is meant.

As 'detail' a form is meant, which shows one single data record and let the user edit it.

To provide an easy understandable navigation structure, it's nice for the user to stay on the same page, even the user is in 'detail' or 'list' mode. Create a single QFQ tt-content record on a fresh page:

```
form = {{form:SE}}

10.sql = SELECT p.name, CONCAT('p:{{pageAlias:T}}&form=Person&r=', p.id) AS _pagee␣
↪FROM Person AS p
10.rend = <br>
```

- If the page is called without any parameter, a list of persons is shown.
- Behind each name, a button is shown. A click on it opens the form 'Person' (with the selected person record) on the same page.

Mode 'list' or 'detail' is detected automatically: if a form is given via STORE_SIP or STORE_TYPO3, the form (=detail) is shown else the report (=list).

9.18.2 Custom default value only for 'new records'

9.18.2.1 Method 1

On *Form.parameter* define a *fillStoreVar* query with a column name equal to a form field. That's all.

Example:

```
FormElement.name = technicalContact
Form.parameter.fillStoreVar = {{! SELECT CONCAT(p.firstName, ' ', p.name) AS␣
↪technicalContact FROM Person AS p WHERE p.account='{{feUser:T}}' }}
```

What we use here is the default STORE prio FSRVD. If the form loads with r=0, 'F', 'S' and 'R' are empty. 'V' is filled. If r>0, than 'F' and 'S' are empty and 'R' is filled.

9.18.2.2 Method 2

In the specific *FormElement* set *value={{columnName:RSE}}*. The link to the form should be rendered with '"...&columnName=<data>&..." AS _page'. The trick is that the STORE_RECORD is empty for new records, and therefore the corresponding value from STORE_SIP will be returned. Existing records will use the already saved value.

9.18.3 Central configured values

Any variable in *Configuration* can be used by *{{<varname>:Y}}* in form or report statements.

E.g.

TECHNICAL_CONTACT = jane.doe@example.net

Could be used in an *FormElement.type* = sendmail with *parameter* setting *sendMailFrom={{TECHNICAL_CONTACT:Y}}*.

9.18.4 Debug Report

Writing "report's" in the nested notation or long queries broken over several lines, might not interpreted as wished. Best for debugging is to specify in the tt-content record:

```
debugShowBodyText = 1
```

Note: Debug information is only display if it's enabled in *Configuration* by

- *showDebugInfo: yes* or
- *showDebugInfo: auto* and logged in in the same Browser as a Typo3 backend user.

9.18.5 More detailed error messages

If *showDebugInfo* is enabled, a full stacktrace and variable contents are displayed in case of an error.

9.18.6 Form search

QFQ content record:

```
# Creates a small form that redirects back to this page
10 {
  sql = SELECT '_'
  head = <form action='#' method='get'><input type='hidden' name='id' value='{
↪{pageAlias:T}}'>Search: <input type='text' name='search' value='{{search:CE:all}}'>
↪<input type='submit' value='Submit'></form>
}

# SQL statement will find and list all the relevant forms - be careful not to open a␣
↪cross site scripting door: the parameter 'search' needs to be sanitized.
20 {
  sql = SELECT CONCAT('p:{{pageAlias:T}}&form=form&r=', f.id) AS _pagee, f.id, f.name,
↪ f.title
            FROM Form AS f
            WHERE f.name LIKE  '%{{search:CE:alnumx}}%'
```

(continues on next page)

(continued from previous page)

```
  head = <table class='table'>
  tail = </table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
}
```

9.18.7 Form: compute next free 'ord' automatically

Requirement: new records should automatically get the highest number plus 10 for their 'ord' value. Existing records should not be altered.

9.18.7.1 Version 1

Compute the next 'ord' in advance in the subrecord field of the primary form. Submit that value to the new record via SIP parameter to the secondary form.

On the secondary form: for 'new' records choose the computed value, for existing records leave the value unchanged.

- Master form, *subrecord FormElement*, field *parameter*: set

```
detail=id:formId,{{SELECT '&', IFNULL(fe.ord,0)+10 FROM Form AS f LEFT JOIN␣
↪*FormElement* AS fe ON fe.formId=f.id WHERE
f.id={{r:S0}} ORDER BY fe.ord DESC LIMIT 1}}:ord
```

- Slave form, *ord FormElement*, field *value*: set

```
`{{ord:RS0}}`.
```

9.18.7.2 Version 2

Compute the next 'ord' as default value direct inside the secondary form. No change is needed for the primary form.

- Secondary form, *ord FormElement*, field *value*: set *{{SELECT IF({{ord:R0}}=0, MAX(IFNULL(fe.ord,0))+10,{{ord:R0}}) FROM (SELECT 1) AS a LEFT JOIN FormElement AS fe ON fe.formId={{formId:S0}} GROUP BY fe.formId}}*.

9.18.8 Form: Person Wizard - firstname, city

Requirement: A form that displays the column 'firstname' from table 'Person' and 'city' from table 'Address'. If the records not exist, the form should create it.

Form primary table: Person

Form slave table: Address

Relation: *Person.id = Address.personId*

- Form: wizard
  - Name: wizard
  - Title: Person Wizard

  - Table: Person
  - Render: bootstrap
- *FormElement*: firstname
  - Class: **native**
  - Type: **text**
  - Name: firstname
  - Label: Firstname
- *FormElement*: email, text, 20
  - Class: **native**
  - Type: **text**
  - Name: city
  - Label: City
  - Value: *{{SELECT city FROM Address WHERE personId={{r}} ORDER BY id LIMIT 1}}*
- *FormElement*: insert/update address record
  - Class: **action**
  - Type: **afterSave**
  - Label: Manage Address
  - Parameter:
    - *slaveId={{SELECT id FROM Address WHERE personId={{r}} ORDER BY id LIMIT 1}}*
    - *sqlInsert={{INSERT INTO Address (personId, city) VALUES ({{r}}, '{{city:F:allbut:s}}') }}*
    - *sqlUpdate={{UPDATE Address SET city='{{city:F:allbut:s}}' WHERE id={{slaveId:V}} }}*
    - *sqlDelete={{DELETE FROM Address WHERE id={{slaveId:V}} AND ''='{{city:F:allbut:s}}' LIMIT 1}}*

9.18.9 Form: Person Wizard - firstname, single note

Requirement: A form that displays the column 'firstname' from table 'Person' and 'note' from table 'Note'. If the records don't exist, the form should create it. Column Person.noteId points to Note.id

Form primary table: Person

Form slave table: Address

Relation: *Person.id = Address.personId*

- Form: wizard
  - Name: wizard
  - Title: Person Wizard
  - Table: Person
  - Render: bootstrap
- *FormElement*: firstname
  - Class: **native**

  - Type: **text**
  - Name: firstname
  - Label: Firstname
- *FormElement*: email, text, 20
  - Class: **native**
  - Type: **text**
  - Name: note
  - Label: Note
  - Value: *{{SELECT Note FROM Note AS n, Person AS p WHERE p.id={{r}} AND p.noteId=n.id ORDER BY id }}*
- *FormElement*: insert/update address record
  - Class: **action**
  - Type: **afterSave**
  - Name: noteId
  - Label: Manage Note
  - Parameter:
    * *sqlInsert={{INSERT INTO Note (note) VALUES ('{{note:F:allbut:s}}') }}*
    * *sqlUpdate={{UPDATE Note SET note='{{note:F:allbut:s}}' WHERE id={{slaveId:V}} }}*

9.18.10 Icons Template Group

This example will display graphics instead of text 'add' and 'remove'. Also there is a distance between the template-Groups.

- FormElement.parameter:

```
tgClass = qfq-child-margin-top
tgAddClass = btn alert-success
tgAddText = <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>
tgRemoveClass = btn btn-danger alert-danger
tgRemoveText = <span class="glyphicon glyphicon-remove" aria-hidden="true"></span>
```

9.18.11 Chart

- QFQ delivers a chart JavaScript lib: https://github.com/nnnick/Chart.js.git. Docs: http://www.chartjs.org/docs/
- The library is not sourced in the HTML page automatically. To do it, either include the lib *typo3conf/ext/qfq/Resources/Public/JavaScript/Chart.min.js*:
  - in the specific tt_content record (shown below in the example) or
  - system wide via Typo3 Template record.
- By splitting HTML and JavaScript code over several lines, take care not accidentally to create a 'nesting'-end token. Check the line after *10.tail =*. It's '}' alone on one line. This is a valid 'nesting'-end token!. There are two options to circumvent this:

- Don't nest the HTML & JavaScript code - bad workaround, this is not human readable.
- Select different nesting token, e.g. '<' (check the first line on the following example).

```
# <

10.sql = SELECT '_'
10.head =
  <div style="height: 1024px; width: 640px;">
    <h3>Distribution of FormElement types over all forms</h3>
    <canvas id="barchart" width="1240" height="640"></canvas>
  </div>
  <script src="typo3conf/ext/qfq/Resources/Public/JavaScript/Chart.min.js"></
↪script>
  <script>
    $(function () {
      var ctx = document.getElementById("barchart");
      var barChart = new Chart(ctx, {
        type: 'bar',
          data: {

10.tail =
          }
      });
    });
  </script>

# Labels
10.10 <
  sql = SELECT "'", fe.type, "'" FROM FormElement AS fe GROUP BY fe.type␣
↪ORDER BY fe.type
  head = labels: [
  tail = ],
  rsep = ,
>

# Data
10.20 <
  sql = SELECT COUNT(fe.id) FROM FormElement AS fe GROUP BY fe.type ORDER BY␣
↪fe.type
  head = datasets: [ {   data: [
  tail = ],  backgroundColor: "steelblue", label: "FormElements" } ]
  rsep = ,
>
```

9.18.12 Upload Form Simple

Table Person

Name	Type
id	int
name	varchar(255)
pathFileNamePicture	varchar(255)
pathFileNameAvatar	varchar(255)

- Form:

  - Name: UploadSimple
  - Table: Person
- FormElements:
  - Name: name
    - Type: text
    - Label: Name
  - Name: pathFileNamePicture
    - Type: upload
    - Label: Picture
    - Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-picture-{{filename}}
```

  - Name: pathFileNameAvatar
    - Type: upload
    - Label: Avatar
    - Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-avatar-{{filename}}
```

9.18.13 Upload Form Advanced 1

Table: Person

Name	Type
id	int
name	varchar(255)

Table: Note

Name	Type
id	int
pId	int
type	varchar(255)
pathFileName	varchar(255)

- Form:
  - Name: UploadAdvanced1
  - Table: Person
- FormElements
  - Name: name
    - Type: text
    - Label: Name

  - Name: mypathFileNamePicture
    * Type: upload
    * Label: Picture
    * Value: {{SELECT pathFileName FROM Note WHERE id={{slaveId}} }}
    * Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-picture-{{filename}}
slaveId={{SELECT id FROM Note WHERE pId={{id:R0}} AND type='picture'␣
↪LIMIT 1}}
sqlInsert={{INSERT INTO Note (pathFileName, type, pId) VALUE ('{
↪{fileDestination}}', 'picture', {{id:R0}}) }}
sqlUpdate={{UPDATE Note SET pathFileName = '{{fileDestination}}' WHERE␣
↪id={{slaveId}} LIMIT 1}}
sqlDelete={{DELETE FROM Note WHERE id={{slaveId}}  LIMIT 1}}
```

  - Name: mypathFileNameAvatar
    * Type: upload
    * Label: Avatar
    * Value: {{SELECT pathFileName FROM Note WHERE id={{slaveId}} }}
    * Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-avatar-{{filename}}
slaveId={{SELECT id FROM Note WHERE pId={{id:R0}} AND type='avatar'␣
↪LIMIT 1}}
sqlInsert={{INSERT INTO Note (pathFileName, type, pId) VALUE ('{
↪{fileDestination}}', 'avatar', {{id:R0}}) }}
sqlUpdate={{UPDATE Note SET pathFileName = '{{fileDestination}}' WHERE␣
↪id={{slaveId}} LIMIT 1}}
sqlDelete={{DELETE FROM Note WHERE id={{slaveId}}  LIMIT 1}}
```

9.18.14 Upload Form Advanced 2

Table: Person

Name	Type
id	int
name	varchar(255)
noteIdPicture	int
noteIdAvatar	int

Table: Note

Name	Type
id	int
pathFileName	varchar(255)

- Form:
  - Name: UploadAdvanced2
  - Table: Person

- FormElements
  - Name: name
    - Type: text
    - Label: Name
  - Name: mypathFileNamePicture
    - Type: upload
    - Label: Picture
    - Value: {{SELECT pathFileName FROM Note WHERE id={{slaveId}} }}
    - Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-picture-{{filename}}
slaveId={{SELECT id FROM Note WHERE id={{noteIdPicture}} LIMIT 1}}
sqlInsert={{INSERT INTO Note (pathFileName) VALUE ('{{fileDestination}}
↪') }}
sqlUpdate={{UPDATE Note SET pathFileName = '{{fileDestination}}' WHERE␣
↪id={{slaveId}} LIMIT 1}}
sqlDelete={{DELETE FROM Note WHERE id={{slaveId}}  LIMIT 1}}
sqlAfter={{UPDATE Person SET noteIdPicture={{slaveId}} WHERE id={{id:R0}}
↪ LIMIT 1
```

  - Name: mypathFileNameAvatar
    - Type: upload
    - Label: Avatar
    - Value: {{SELECT pathFileName FROM Note WHERE id={{slaveId}} }}
    - Parameter:

```
fileDestination=fileadmin/user/{{id:R0}}-avatar-{{filename}}
slaveId={{SELECT id FROM Note WHERE id={{noteIdAvatar}} LIMIT 1}}
sqlInsert={{INSERT INTO Note (pathFileName) VALUE ('{{fileDestination}}
↪') }}
sqlUpdate={{UPDATE Note SET pathFileName = '{{fileDestination}}' WHERE␣
↪id={{slaveId}} LIMIT 1}}
sqlDelete={{DELETE FROM Note WHERE id={{slaveId}}  LIMIT 1}}
sqlAfter={{UPDATE Person SET noteIdAvatar={{slaveId}} WHERE id={{id:R0}}␣
↪LIMIT 1
```

9.18.15 Typeahead: SQL

Table: Person

Name	Type
id	int
name	varchar(255)

- Form:
  - Name: PersonNameTypeahead
  - Table: Person

- FormElements
  - Name: name
    - Type: text
    - Label: Name
    - Parameter: `typeAheadSql = SELECT name FROM Person WHERE name LIKE ? OR firstName LIKE ? LIMIT 100`

9.18.16 Typeahead: LDAP with additional values

Table: Person

Name	Type
id	int
name	varchar(255)
firstname	varchar(255)
email	varchar(255)

- Form:
  - Name: PersonNameTypeaheadSetNames
  - Table: Person
  - Parameter:

```
ldapServer = directory.example.com
ldapBaseDn = ou=Addressbook,dc=example,dc=com
```

- FormElements
  - Name: email
    - Class: native
    - Type: text
    - Label: Email
    - Note: Name: {{cn:LE}}
Email: {{mail:LE}}
    - dynamicUpdate: checked
    - Parameter:

```
# Typeahead
typeAheadLdapSearch = (|(cn=*?*)(mail=*?*))
typeAheadLdapValuePrintf '%s / %s', cn, email
typeAheadLdapIdPrintf  '%s', email

# dynamicUpdate: show note
fillStoreLdap
ldapSearch = (mail={{email::alnumx}})
ldapAttributes = cn, email
```

  - Name: fillLdapValues
    - Class: action

* Type: afterSave
* Parameter:

```
fillStoreLdap
ldapSearch = (mail={{email::alnumx}})
ldapAttributes = cn, email

slaveId={{id:R0}}
sqlUpdate={{ UPDATE Person AS p SET p.name='{{cn:L:alnumx:s}}' WHERE p.
↪id={{slaveId}} LIMIT 1 }}
```

9.19 Import/merge form

The form *copyFormFromExt* copies a form from table *ExtForm / ExtFormElement* to *Form / FormElement*. The import/merge form:

- offers a drop down list with all forms of *ExtForm*,
- an input element for the new form name,
- create new Form.id
- copied FormElements get the new Form.id.
- the copied form will be opened in the FormEditor.

Installation:

- Play (do all sql statements on your QFQ database, e.g. via *mysql <dbname> < copyFormFromExt.sql* or *phpMyAdmin*) the file *<ext_dir>/Classes/Sql/copyFormFromExt.sql*.
- Insert a link/button 'Copy form from ExtForm' to open the import/merge form. A good place is the list of all forms (see *FormEditor*). E.g.:

```
10.head = {{'b|p:id={{pageAlias:T}}&form=copyFormFromExt|t:Copy form from ExtForm
↪' AS _link }} ...
```

If there are several T3/QFQ instances and if forms should be imported frequently/easily, set up a one shot 'import Forms from db xyz' like:

```
10.sql = CREATE OR REPLACE table ExtForm SELECT * FROM <db xyz>.Form
20.sql = CREATE OR REPLACE table ExtFormElement SELECT * FROM <db xyz>.FormElement
```

CHAPTER 10

Report

10.1 QFQ content element

The QFQ extension is activated through tt-content records. One or more tt-content records per page are necessary to render *forms* and *reports*. Specify column and language per content record as wished.

The title of the QFQ content element will not be rendered. It's only visible in the backend for orientation of the webmaster.

To display a report on any given TYPO3 page, create a content element of type 'QFQ Element' (plugin) on that page.

10.1.1 A simple example

Assume that the database has a table person with columns firstName and lastName. To create a simple list of all persons, we can do the following:

```
10.sql = SELECT firstName, lastName FROM Person
```

The '10' indicates a *root level* of the report (see section *Structure*). The expression '10.sql' defines an SQL query for the specific level. When the query is executed, it will return a result having one single column name containing first and last name separated by a space character.

The HTML output, displayed on the page, resulting from only this definition, could look as follows:

```
JohnDoeJaneMillerFrankStar
```

I.e., QFQ will simply output the content of the SQL result row after row for each single level.

10.1.1.1 Format output: mix style and content

Variant 1: pure SQL

To format the upper example, just create additional columns:

```
10.sql = SELECT firstName, ", ", lastName, "<br>" FROM Person
```

HTML output:

```
Doe, John<br>
Miller, Jane<br>
Star, Frank<br>
```

Variant 2: SQL plus QFQ helper

QFQ provides several helper functions to wrap rows and columns, or to separate them. In this example 'fsep'='field separate and 'rend' = row end:

```
10.sql = SELECT firstName, lastName FROM Person
10.fsep = ', '
10.rend = <br>
```

HTML output:

```
Doe, John<br>
Miller, Jane<br>
Star, Frank<br>
```

Check out all QFQ helpers under *QFQ Keywords (Bodytext)*.

Due to mixing style and content, this becomes harder to maintain with more complex layout.

10.1.1.2 Format output: separate style and content

The result of the query can be passed to the Twig template engine in order to fill a template with the data.:

```
10.sql = SELECT firstName, lastName FROM Person
10.twig = {% for row in result %}
    {{ row.lastName }}, {{ row.firstName }<br />
{% endfor %}
```

HTML output:

```
Doe, John<br>
Miller, Jane<br>
Star, Frank<br>
```

Check out *Using Twig*.

10.2 Syntax of *report*

All **root level queries** will be fired in the order specified by 'level' (Integer value).

For **each** row of a query (this means *all* queries), all subqueries will be fired once.

- E.g. if the outer query selects 5 rows, and a nested query select 3 rows, than the total number of rows are 5 x 3 = 15 rows.

There is a set of **variables** that will get replaced before ('count' also after) the SQL-Query gets executed:

{{<name>[:<store/s>[:...]]}} Variables from specific stores.

{{<name>:R}} - use case of the above generic definition. See also *Access column values*.

{{<level>.<columnName>}} Similar to {{<name>:R}} but more specific. There is no sanitize class, escape mode or default value.

{{<level>.line.count}} - Current row index This variable is specific, as it will be replaced before the query is fired in case of <level> is an outer/previous level or it will be replaced after a query is fired in case <level> is the current level.

{{<level>.line.total}} Total rows (MySQL num_rows for *SELECT* and *SHOW*, MySQL affected_rows for *UPDATE* and *INSERT*.

{{<level>.line.insertId}} Last insert id for *INSERT*.

{{<level>.line.content}} If the content of <*level*> have been stored, e.g. <*level*>.*content=hide*.

{{<level>.line.altCount}} - Like 'line.count' but for 'alt' query.

{{<level>.line.altTotal}} - Like 'line.total' but for 'alt' query.

{{<level>.line.altInsertId}} - Like 'line.insertId' but for 'alt' query.

See *Variable* for a full list of all available variables.

Different types of SQL queries are possible: SELECT, INSERT, UPDATE, DELETE, SHOW, REPLACE

Only SELECT and SHOW queries will fire subqueries.

Processing of the resulting rows and columns:

- In general, all columns of all rows will be printed out sequentially.
- On a per column base, printing of columns can be suppressed by starting the column name with an underscore '_'. E.g. *SELECT id AS _id*.

  This might be useful to store values, which will be used later on in another query via the *{{id:R}}* or *{{<level>.columnName}}* variable. To suppress printing of a column, use a underscore as column name prefix. E.g. *SELECT id AS _id*

Reserved column names have a special meaning and will be processed in a special way. See *Processing of columns in the SQL result* for details.

There are extensive ways to wrap columns and rows. See *Wrapping rows and columns: Level*

10.3 Using Twig

How to write Twig templates is documented by the Twig Project.

QFQ passes the result of a given query to the corresponding Twig template using the Twig variable *result*. So templates need to use this variable to access the result of a query.

10.3.1 Specifying a Template

By default the string passed to the **twig**-key is used as template directly, as shown in the simple example above:

```
10.twig = {% for row in result %}
    {{ row.lastName }}, {{ row.firstName }}<br />
{% endfor %}
```

However, if the string starts with **file:**, the template is read from the file specified.:

```
10.twig = file:table_sortable.html.twig
```

The file is searched relative to *<site path>* and if the file is not found there, it's searched relative to QFQ's *twig_template* folder where the included base templates are stored.

The following templates are available:

tables/default.html.twig A basic table with column headers, sorting and column filters using tablesorter and bootstrap.

tables/single_vertical.html.twig A template to display the values of a single record in a vertical table.

10.3.2 Links

The link syntax described in *Column: _link* is available inside Twig templates using the *qfqlink* filter:

```
{{ "u:http://www.example.com" | qfqlink }}
```

will render a link to *http://www.example.com*.

10.3.3 Json Decode

A String can be JSON decoded in Twig the following way:

```
{% set decoded = '["this is one", "this is two"]' | json_decode%}
```

This can then be used as a normal object in Twig:

```
{{ decoded[0] }}
```

will render *this is one*.

10.3.4 Available Store Variables

QFQ also provides access to the following stores via the template context.

- record
- sip
- typo3
- user
- system
- var

All stores are accessed using their lower case name as attribute of the context variable *store*. The active Typo3 front-end user is therefore available as:

```
{{ store.typo3.feUser }}
```

10.3.5 Example

The following block shows an example of a QFQ report.

10.sql selects all users who have been assigned files in our file tracker.

10.10 then selects all files belonging to this user, prints the username as header and then displays the files in a nice table with links to the files.

```
10.sql = SELECT assigned_to AS _user FROM FileTracker
            WHERE assigned_to IS NOT NULL
            GROUP BY _user
            ORDER BY _user

10.10.sql = SELECT id, path_scan FROM FileTracker
WHERE assigned_to = '{{user:R}}'
10.10.twig = <h2>{{ store.record.user }}</h2>
<table class="table table-hover tablesorter" id="{{pageAlias:T}}-twig">
  <thead><tr><th>ID</th><th>File</th></tr></thead>
  <tbody>
  {% for row in result %}
    <tr>
      <td>{{ row.id }}</td>
      <td>{{ ("d:|M:pdf|s|t:"~ row.path_scan ~"|F:" ~ row.path_scan ) | qfqlink }}</
↪td>
    </tr>
  {% endfor %}
  </tbody>
</table>
```

10.4 Debug the bodytext

The parsed bodytext could be displayed by activating 'showDebugInfo' (*Debug*) and specifying

```
debugShowBodyText = 1
```

A small symbol with a tooltip will be shown, where the content record will be displayed on the webpage. Note: *Debug* information will only be shown with *showDebugInfo: yes* in *Configuration*.

10.5 Inline Report editing

For quick changes it can be bothersome to go to the TYPO3 backend to update the page content and reload the page. For this reason, QFQ offers an inline report editing feature whenever there is a TYPO3 BE user logged in. A small link symbol will appear on the right-hand side of each report record. Please note that the TYPO3 Frontend cache is also deleted upon each inline report save.

In order for the inline report editing to work, QFQ needs to be able to access the T3 database. This database is assumed to be accessible with the same credentials as specified with indexQfq.

10.6 Structure

A report can be divided into several levels. This can make report definitions more readable because it allows for splitting of otherwise excessively long SQL queries. For example, if your SQL query on the root level selects a number of person records from your person table, you can use the SQL query on the second level to look up the city where each person lives.

See the example below:

```
10.sql = SELECT id AS _pId, CONCAT(firstName, " ", lastName, " ") AS name FROM Person
10.rsep = <br>

10.10.sql = SELECT CONCAT(postal_code, " ", city) FROM Address WHERE pId = {{10.pId}}
10.10.rbeg = (
10.10.rend = )
```

This would result in:

```
John Doe (3004 Bern)
Jane Miller (8008 Zürich)
Frank Star (3012 Bern)
```

10.6.1 Text across several lines

To get better human readable SQL queries, it's possible to split a line across several lines. Lines with keywords are on their own (*QFQ Keywords (Bodytext)* start a new line). If a line is not a 'keyword' line, it will be appended to the last keyword line. 'Keyword' lines are detected on:

- <level>.<keyword> =
- {
- <level>[.<sub level] {

Example:

```
10.sql = SELECT 'hello world'
           FROM Mastertable
10.tail = End

20.sql = SELECT 'a warm welcome'
           'some additional', 'columns'
           FROM AnotherTable
           WHERE id>100

20.head = <h3>
20.tail = </h3>
```

10.6.1.1 Join mode: SQL

This is the default. All lines are joined with a *space* in between. E.g.:

```
10.sql = SELECT 'hello world'
           FROM Mastertable
```

Results to: `10.sql = SELECT 'hello world' FROM Mastertable`

Notice the space between "...world'" and "FROM ...".

10.6.1.2 Join mode: strip whitespace

Ending a line with a \ strips all leading and trailing whitespaces of that line joins the line directly (no extra space in between). E.g.:

```
10.sql = SELECT 'hello world', 'd:final.pdf \
                               |p:id=export  \
                               |t:Download' AS _pdf \
```

Results to: `10.sql = SELECT 'hello world', 'd:final.pdf|p:id=export|t:Download' AS _pdf`

Note: the \ does not force the joining, it only removes the whitespaces.

To get the same result, the following is also possible:

```
10.sql = SELECT 'hello world', CONCAT('d:final.pdf'
                               '|p:id=export',
                               '|t:Download') AS _pdf
```

10.6.2 Nesting of levels

Levels can be nested. E.g.:

```
10 {
  sql = SELECT ...
  5 {
      sql = SELECT ...
      head = ...
  }
}
```

This is equal to:

```
10.sql = SELECT ...
10.5.sql = SELECT ...
10.5.head = ...
```

10.6.3 Leading / trailing spaces

By default, leading or trailing whitespaces are removed from strings behind '='. E.g. 'rend = test ' becomes 'test' for rend. To prevent any leading or trailing spaces, surround them by using single or double ticks. Example:

```
10.sql = SELECT name FROM Person
10.rsep = ' '
10.head = "Names: "
```

10.6.4 Braces character for nesting

By default, curly braces '{}' are used for nesting. Alternatively angle braces '<>', round braces '()' or square braces '[]' are also possible. To define the braces to use, the **first line** of the bodytext has to be a comment line and the last character of that line must be one of '{[(<'. The corresponding braces are used for that QFQ record. E.g.:

```
# Specific code. >
10 <
  sql = SELECT
  head = <script>
         data = [
           {
             10, 20
           }
         ]
         </script>
>
```

Per QFQ tt-content record, only one type of nesting braces can be used.

Be careful to:

- write nothing else than whitespaces/newline behind an **open brace**
- the **closing brace** has to be alone on a line:

```
10.sql = SELECT 'Yearly Report'

20 {
      sql = SELECT companyName FROM Company LIMIT 1
      head = <h1>
      tail = </h1>
}

30 {
      sql = SELECT depName FROM Department
      head = <p>
      tail = </p>
      5 {
            sql = SELECT 'detailed information for department'
            1.sql = SELECT name FROM Person LIMIT 7
            1.head = Employees:
      }
}

30.5.tail = More will follow

50

{
       sql = SELECT 'A query with braces on their own'
}
```

10.6.5 Access column values

Columns of the upper / outer level result can be accessed via variables in two ways

- STORE_RECORD: *{{pId:R}}*

- Level Key: *{{10.pId}}*

The STORE_RECORD will always be merged with previous content. The Level Keys are unique.

Important: Multiple columns, with the same column name, can't be accessed individually. Only the last column is available.

Important: Retrieving the *final* value of *Special column names* is possible via '{{&<column>:R}} (there is an '&' direct behind '{{')

Example:

```
10.sql = SELECT 'p:home&form=Person|s|b:success|t:Edit' AS _link
10.20.sql = SELECT '{{link:R}}', '{{&link:R}}'
```

The first column of row *10.20* returns 'p:home&form=Person|s|b:success|t:Edit',the second column returns 'Edit'.

Example STORE_RECORD:

```
10.sql= SELECT p.id AS _pId, p.name FROM Person AS p
10.5.sql = SELECT adr.city, 'dummy' AS _pId FROM Address AS adr WHERE adr.pId={{pId:R}
↪}
10.5.20.sql = SELECT '{{pId:R}}'
10.10.sql = SELECT '{{pId:R}}'
```

The line '10.10' will output 'dummy' in cases where there is at least one corresponding address. If there are no addresses (all persons) it reports the person id. If there is at least one address, it reports 'dummy', cause that's the last stored content.

Example 'level':

```
10.sql= SELECT p.id AS _pId, p.name FROM Person AS p
10.5.sql = SELECT adr.city, 'dummy' AS _pId FROM Address AS adr WHERE adr.pId={{10.
↪pId}}
10.5.20.sql = SELECT '{{10.pId}}'
10.10.sql = SELECT '{{10.pId}}'
```

Notes to the level:

Levels	A report is divided into levels. The Example has levels *10*, *20*, *30*, *30.5*, *30.5.1*, *50*
Qualifier	A level is divided into qualifiers *30.5.1* has 3 qualifiers *30*, *5*, *1*
Root levels	Is a level with one qualifier. E.g.: 10
Sub levels	Is a level with more than one qualifier. E.g. levels *30.5* or *30.5.1*
Child	The level *30* has one child and child child: *30.5* and *30.5.1*
Example	*10*, *20*, *30*, *50** are root level and will be completely processed one after each other. *30.5* will be executed as many times as *30* has row numbers. *30.5.1* will be executed as many times as *30.5* has row numbers.

Report Example 1:

```
# Displays current date
10.sql = SELECT CURDATE()

# Show all students from the person table
20.sql = SELECT p.id AS pId, p.firstName, " - ", p.lastName FROM Person AS p WHERE p.
↪typ LIKE "student"

# Show all the marks from the current student ordered chronological
20.25.sql = SELECT e.mark FROM Exam AS e WHERE e.pId={{20.pId}} ORDER BY e.date

# This query will never be fired, cause there is no direct parent called 20.30.
20.30.10.sql = SELECT 'never fired'
```

10.7 Wrapping rows and columns: Level

Order and nesting of queries, will be defined with a TypoScript-alike syntax:

```
level.sublevel1.subsublevel2. ...
```

- Each 'level' directive needs a final key, e.g: 20.30.10. **sql**.
- A key **sql** is necessary in order to process a level.

See all *QFQ Keywords (Bodytext)*.

10.8 Processing of columns in the SQL result

- The content of all columns of all rows will be printed sequentially, without separator (except one is defined).
- Rows with *Special column names* will be rendered internally by QFQ and the QFQ output of such processing (if there is any) is taken as the content.

10.9 Special column names

Note: Twig: respect that the 'special column name'-columns are rendered before Twig becomes active. The recommended way by using Twig is *not to use* special column names at all. Use the Twig version *qfqLink*.

QFQ don't care about the content of any SQL-Query - it just copy the content to the output (=Browser).

One exception are columns, whose name starts with '_'. E.g.:

```
10.sql = SELECT 'All', 'cats' AS red, 'are' AS _green, 'grey in the night' AS _link
```

- The first and second column are regular columns. No QFQ processing.
- The third column (alias name 'green') is no QFQ special column name, but has an '_' at the beginning: this column content will be hidden.
- The fourth column (alias name 'link') uses a QFQ special column name. Here, only in this example, it has no further meaning.

- All columns in a row, with the same special column name (e.g. `... AS _page`) will have the same column name: 'page'. To access individual columns a uniq column title is necessary and can be added `|_column1`:

```
10.sql = SELECT '..1..' AS '_page|column1', '..2..' AS '_page|column2'
```

  Those columns can be accessed via `{{10.column1}}` , `{{10.column2}}` or (recommended) `{{column1:R}}`,`{{column2:R}}`

- To skip wrapping via `fbeg`, `fsep`, `fend` for dedicated columns, add the keyword `|_noWrap` to the column alias. Example:

```
10.sql = SELECT 'world' AS 'title|_noWrap'
```

Summary:

- Special column names always start with '_'.
- Columns starting with a '_' but not defined as as QFQ special column name are hidden(!) - in other words: they are not **printed** as output.
- SQL hint: If there is no column alias defined, the column name becomes the value of the first row. E.g. if the first selected row contains a '_' as the first character, the column name becomes '_...' - which will be hidden! To be safe: always define an alias!
- Access to 'hidden' columns via `{{<level>.<column>}}` or `{{<column>:R}}` are possible and often used.
- Important: to reference a column, the name has to be given without '_'. E.g. `SELECT 'cat' AS _pet` will be used with `{{pet:R}}` (notice the missing '_').
- For 'special column names': the column name together with the value controls how QFQ processes the column.

Reserved column name	Purpose
_link	*Column: _link* - Build links with different features.
_pageX or _PageX	*Columns: _page[X]* - Shortcut version of the link interface for fast creation of internal links. The
_download	*Download* - single file (any type) or concatenate multiple files (PDF, ZIP)
_pdf, _file, _zip _Pdf, _File, _Zip	*Column: _pdf | _file | _zip* - Shortcut version of the link interface for fast creation of *Download* li
_savePdf	*Column: _savePdf* - pre render PDF files
_excel	*Excel export* - creates Excel exports based on QFQ Report queries, optional with pre uploaded Ex
_yank	*Copy to clipboard*. Shortcut version of the link interface
_mailto	*Column: _mailto* - Build email links. A click on the link will open the default mailer. The addres
_sendmail	*Column: _sendmail* - Send emails.
_exec	*Column: _exec* - Run batch files or executables on the webserver.
_script	*Column: _script* - Run php function defined in an external script
_vertical	*Column: _vertical* - Render Text vertically. This is useful for tables with limited column width.
_img	*Column: _img* - Display images.
_bullet	Display a blue/gray/green/pink/red/yellow bullet. If none color specified, show nothing.
_check	Display a blue/gray/green/pink/red/yellow checked sign. If none color specified, show nothing.
_nl2br	All newline characters will be converted to <*br*>.
_striptags	HTML Tags will be stripped.
_htmlentities	Characters will be encoded to their HTML entity representation.
_fileSize	Show file size of a given file
_mimeType	Show mime type of a given file
_thumbnail	*thumbnail* - Create thumbnails on the fly. See *Column: _thumbnail*.
_monitor	*Column: _monitor* - Constantly display a file. See *Column: _monitor*.
_XLS	Used for Excel export. Append a *newline* character at the end of the string. Token must be part of

Table 1 – continued from prev

Reserved column name	Purpose
_XLSs	Used for Excel export. Prepend the token 's=' and append a *newline* character at the string. See E
_XLSb	Used for Excel export. Like '_XLSs' but encode the string base64. See *Excel export*.
_XLSn	Used for Excel export. Prepend 'n=' and append a *newline* character around the string. See *Excel*
_noWrap	Skip wrapping via *fbeg*, `fsep`, `fend`.
_hide	Column is hidden.
_+<html-tag attributes>	The content will be wrapped with '<html-tag attributes>'. Example: SELECT 'example' AS '_+a
_=<varname>	The content will be saved in store 'user' under 'varname'. Retrieve it later via {{varname:U}}. E
_<nonReservedName>	Suppress output. Column names with leading underscore are used to select data from the databas

10.9.1 Column: _link

- Most URLs will be rendered via class link.
- Column names like *_pagee*, *_mailto*, ... are wrapper to class link.
- The parameters for link contains a prefix to make them position-independent.
- Parameter have to be separated with 'l'. If 'l' is part of the value, it needs to be escaped '\l'. This can be done via QFQ SQL function *QBAR()* - see *QBAR: Escape QFQ Delimiter*.

URL	IMG	Meaning	Qualifier	Example	Description
x		URL	u:<url>	u:http://www.example.com	If an image is specified, it will be rendered inside the link, default link class: external
x		Mail	m:<email>	m:info@example.com	Default link class: email
x		Page	p:<pageId>	p:impressum	Prepend '?' or '?id=', no hostname qualifier (automatically set by browser)
x		Download	d:[<exportFilename>:]<sip>	d:complete.pdf	Link points to .../typo3conf/ext/qfq/Api/download. Additional parameter SIP encoded. 'Download' needs SIP. See *Download*.

Continued on next page

Table 2 – continued from previous page

URL	IMG	Meaning	Qualifier	Example	Description
x		Copy to clipboard	y:[some content]	y:this will be copied	Click on it copies the value of 'y:' to the clipboard. Optional a file ('F:...') might be specified as source. See *Copy to clipboard*.
		Dropdown menu	z	z|p:home|t:Home	Creates a dropdown menu. See *Dropdown Menu*.
		websocket	w:ws://<host>:<port>/<path>	w:ws://localhost:1234	Send mo message given in 't:...' to websocket. See *WebSocket*.
		Text	t:<text>	t:Firstname Lastname	
		Render	r:<mode>	r:3	See: *Render mode*, Default: 0
		Button	b[:0|1|<btn class>]	b:0, b:1, b:success	'b', 'b:1': a bootstrap button is created. 'b:0' disable the button. <btn class>: default, primary, success, info, warning,danger
	x	Picture	P:<filename>	P:bullet-red.gif	Picture '<img src="bullet-red.gif"alt="....">'.
	x	Edit	E	E	Show 'edit' icon as image
	x	New	N	N	Show 'new' icon as image
	x	Delete	D	D	Show 'delete' icon as image (only the icon, no database record 'delete' functionality)
	x	Help	H	H	Show 'help' icon as image
	x	Info	I	I	Show 'information' icon as image

Continued on next page

Table 2 – continued from previous page

URL	IMG	Meaning	Qualifier	Example	Description
	x	Show	S	S	Show 'show' icon as image
	x	Glyph	G:<glyphname>	G:glyphicon-envelope	Show <glyphname>. Check: https://getbootstrap.com/docs/3.4/components/
	x	Bullet	B:[<color>]	B:green	Show bullet with '<color>'. Colors: blue, gray, green, pink, red, yellow. Default Color: green.
	x	Check	C:[<color>]	C:green	Show checked with '<color>'. Colors: blue, gray, green, pink, red, yellow. Default Color: green.
	x	Thumbnail	T:<pathFileName>	T:fileadmin/file.pdf	Creates a thumbnail on the fly. Size is specified via 'W'. See *Column: _thumbnail*
		Dimension	W:[width]x[height]	W:50x , W:x60 , W:50x60	Defines the pixel size of thumbnail. See *Column: _thumbnail*
		URL Params	U:<key1>=<value1>[&<key2>=<value2>]...	U:a=b&c=d	Add a number of additional Params. Links to forms: U:form=Person&r=1234. Used to create 'record delete'-links.
		Tooltip	o:<text>	o:More information here	Tooltip text

Continued on next page

Table 2 – continued from previous page

URL	IMG	Meaning	Qualifier	Example	Description
		Alttext	a:<text>	a:Name of person	a) Alttext for images, b) Message text for *Download* popup window.
		Class	c:[n|<text>]	c:text-muted	CSS class for link. n:no class attribute, <text>: explicit named
		Attribute	A:<key>=”<value”>	A:data-reference=”person”	Custom attributes and a corresponding value. Might be used by application tests.
		Target	g:<text>	g:_blank	target=_blank,_self,_parent,<custom Default: no target
		Question	q:<text>	q:please confirm	See: *Alert: Question*. Link will be executed only if user clicks ok/cancel, default: ‘Please confirm’
		Encryption	e:0|1|...	e:1	Encryption of the e-mail: 0: no encryption, 1:via Javascript (default)
		Right	R	R	Defines picture position: Default is ‘left’ (no definition) of the ‘text’. ‘R’ means ‘right’ of the ‘text’

Continued on next page

Table 2 – continued from previous page

URL	IMG	Meaning	Qualifier	Example	Description
		SIP	s[:0|1]	s, s:0, s:1	If 's' or 's:1' a SIP entry is generated with all non Typo 3 Parameters. The URL contains only parameter 's' and Typo 3 parameter
		Mode	M:file|pdf|zip	M:file, M:pdf, M:zip	Mode. Used to specify type of download. One or more element sources needs to be configured. See *Download*.
		File	F:<filename>	F:fileadmin/file.pdf	Element source for download mode file|pdf|zip. See *Download*.
		Delete record	x[:a|r|c]	x, x:r, x:c	a: ajax (only QFQ internal used), r: report (default), c: close (current page, open last page)

10.9.2 Render mode

The following table might be hard to read - but it's really useful to understand. It solves a lot of different situations. If there are no special condition (like missing value, or suppressed links), render mode 0 is sufficient. But if the URL text is missing, or the URL is missing, OR the link should be rendered in sql row 1-10, but not 5, then render mode might dynamically control the rendered link.

- Column *Mode* is the render mode and controls how the link is rendered.

Mode	Given: url & text	Given: only url	Given: only text	Description
0 (default)	text	url		text or image will be shown, only if there is a url, page or mailto
1	text	url	text	text or image will be shown, independently of whether there is a url or not
2	text			no link if text is empty
3	text	url	text	**no link, only text or image, incl. Optio** r:3 will set the button to disable and no link/sip is rendered.
4	url	url	text	no link, show text, if text is empty, show url, incl. optional tooltip
5				nothing at all
6	pure text		pure text	no link, pure text
7	pure url	pure url		no link, pure url
8	pure sip	pure sip		no link, no html, only the 13 digit sip code.

Example:

```
10.sql = SELECT CONCAT('u:', p.homepage, IF(p.showHomepage='yes', '|r:0', '|r:5') )␣
↪AS _link FROM Person AS p
```

Tip:

An easy way to switch between different options of rendering a link, incl. Bootstrap buttons, is to use the render mode.

- no render mode or 'r:0' - the full functional link/button.
- 'r:3' - the link/button is rendered with text/image/glyph/tooltip ... but without a HTML a-tag! For Bootstrap button, the button get the 'disabled' class.
- 'r:5' - no link/button at all.

10.9.3 Link Examples

SQL-Query	Result
SELECT “m:info@example.com” AS _link	info@example.com as linked text, encrypted with javascript, class=external
SELECT “m:info@example.com|c:0” AS _link	info@example.com as linked text, not encrypted, class=external
SELECT “m:info@example.com|P:mail.gif” AS _link	info@example.com as linked image mail.gif, encrypted with javascript, class=external
SELECT “m:info@example.com|P:mail.gif|o:Email” AS _link	*info@example.com* as linked image mail.gif, encrypted with javascript, class=external, tooltip: “sendmail”
SELECT “m:info@example.com|t:mailto:info@example.com|o:Email” AS link	‘*info@example.com*’ as linked text, encrypted with javascript, class=external
SELECT “u:www.example.com” AS _link	www.example as link, class=external
SELECT “u:http://www.example.com” AS _link	*http://www.example* as link, class=external
SELECT “u:www.example.com|q:Please confirm” AS _link	www.example as link, class=external, See: *Alert: Question*
SELECT “u:www.example.com|c:nicelink” AS _link	*http://www.example* as link, class=nicelink
SELECT “p:form_person&note=Text|t:Person” AS _link	Person
SELECT “p:form_person|E” AS _link	<img alttext=”Edit” src=”typo3conf/ext/qfq/Resources/Public/icons/edit.gif”>
SELECT “p:form_person|E|g:_blank” AS _link	<img alttext=”Edit” src=”typo3conf/ext/qfq/Resources/Public/icons/edit.gif”>
SELECT “p:form_person|C” AS _link	<img alttext=”Check” src=”typo3conf/ext/qfq/Resources/Public/icons/checked-green.gif”>
SELECT “p:form_person|C:green” AS _link	<img alttext=”Check” src=”typo3conf/ext/qfq/Resources/Public/icons/checked-green.gif”>
SELECT “U:form=Person&r=123|x|D” as _link	<sp class=”glyphicon glyphicon-trash” >”>
SELECT “U:form=Person&r=123|x|t:Delete” as _link	<a href=”typo3conf/ext/qfq/Classes/Api/delete.php?s=badcaffee1234”>Del
SELECT “s:1|d:full.pdf|M:pdf|p:id=det1&r=12|p:id=det2|F:cv.pdf| t:Download|a:Create complete PDF - please wait” as _link	<a href=”typo3conf/ext/qfq/Classes/Api/download.php?s=badcaffee1234”>
SELECT “y:iatae3Ieem0jeet|t:Password|o:Clipboard|b” AS _link	<button class=”btn btn-info” onClick=”new QfqNS.Clipboard({text: ‘iatae3Ieem0jeet’});” title=’Copy to clipboard’>Password</button>
SELECT “y|s:1|F:dir/data.R|t:Data|o:Clipboard|b” AS _link	<button class=”btn btn-info” onClick=”new QfqNS.Clipboard({uri: ‘typo3conf/. . . /download.php?s=badcaffee1234’});” title=’Copy to clipboard’>Data</button>

10.9.4 Alert: Question

Syntax

```
q[:<alert text>[:<level>[:<positive button text>[:<negative button text>[:<timeout>[:
↪<flag modal>]]]]]]
```

- If a user clicks on a link, an alert is shown. If the user answers the alert by clicking on the 'positive button', the browser opens the specified link. If the user click on the negative answer (or waits for timout), the alert is closed and the browser does nothing.
- All parameter are optional.
- Parameter are separated by ':'
- To use ':' inside the text, the colon has to be escaped by . E.g. 'ok\ : I understand'.

Parameter	Description
Text	The text shown by the alert. HTML is allowed to format the text. Any ':' needs to be escaped. Default: 'Please confirm'.
Level	success, info, warning, danger
Positive button text	Default: 'Ok'
Negative button text	Default: 'Cancel'. To hide the second button: '-'
Timeout in seconds	0: no timeout, >0: after the specified time in seconds, the alert will dissapear and behaves like 'negative answer'
Flag modal	0: Alert behaves not modal. 1: (default) Alert behaves modal.

Examples:

SQL-Query	Result
SELECT "p:form_person|q:Edit Person:warn" AS _link	Shows alert with level 'warn'
SELECT "p:form_person|q:Edit Person::I do:No way" AS _link	Instead of 'Ok' and 'Cancel', the button text will be 'I do' and 'No way'
SELECT "p:form_person|q:Edit Person:::10" AS _link	The Alert will be shown 10 seconds
SELECT "p:form_person|q:Edit Person:::10:0" AS _link	The Alert will be shown 10 seconds and is not modal.

10.9.5 Columns: _page[X]

The colum name is composed of the string *page* and a trailing character to specify the type of the link.

Syntax:

```
10.sql = SELECT "[options]" AS _page[<link type>]

with: [options] = [p:<page & param>][|t:<text>][|o:<tooltip>][|q:<question parameter>
↪][|c:<class>][|g:<target>][|r:<render mode>]

<link type> = c,d,e,h,i,n,s
```

column name	Purpose	default value of question parameter	Mandatory parameters
_page	Internal link without a grafic	empty	p:<pageId/pageAlias>[&param]
_pagec	Internal link without a grafic, with question	*Please confirm!*	p:<pageId/pageAlias>[&param]
_paged	Internal link with delete icon (trash)	*Delete record ?*	U:form=<formname>&r=<record id> *or* U:table=<tablename>&r=<record id>
_pagee	Internal link with edit icon (pencil)	empty	p:<pageId/pageAlias>[&param]
_pageh	Internal link with help icon (question mark)	empty	p:<pageId/pageAlias>[&param]
_pagei	Internal link with information icon (i)	empty	p:<pageId/pageAlias>[&param]
_pagen	Internal link with new icon (sheet)	empty	p:<pageId/pageAlias>[&param]
_pages	Internal link with show icon (magnifier)	empty	p:<pageId/pageAlias>[&param]

- All parameter are optional.
- Optional set of predefined icons.
- Optional set of dialog boxes.

Parameter	Description	Default value	Example
<page>	TYPO3 page id or page alias.	The current page: *{{pageId}}*	45 application application&N_param1=1045
<text>	Text, wrapped by the link. If there is an icon, text will be displayed to the right of it.	empty string	
<tooltip>	Text to appear as a ToolTip	empty string	
<question>	If there is a question text given, an alert will be opened. Only if the user clicks on 'ok', the link will be called	**Expected "=" to follow "see"**	
<class>	CSS Class for the <a> tag		
<target>	Parameter for HTML 'target='. F.e.: Opens a new window	empty	P
<render mode>	Show/render a link at all or not. See *Render mode*		
<create sip>	s		's': create a SIP

10.9.6 Column: _paged

These column offers a link, with a confirmation question, to delete one record (mode 'table') or a bunch of records (mode 'form'). After deleting the record(s), the current page will be reloaded in the browser.

Syntax

```
10.sql = SELECT "U:table=<tablename>&r=<record id>|q:<question>|..." AS _paged
10.sql = SELECT "U:form=<formname>&r=<record id>|q:<question>|..." AS _paged
```

If the record to delete contains column(s), whose column name match on *%pathFileName%* and such a column points to a real existing file, such a file will be deleted too. If the table contains records where the specific file is multiple times referenced, than the file is not deleted (it would break the still existing references). Multiple references are not found, if they use different columnnames or tablenames.

10.9.6.1 Mode: table

- *table=<table name>*
- *r=<record id>*

Deletes the record with id '<record id>' from table '<table name>'.

10.9.6.2 Mode: form

- *form=<form name>*
- *r=<record id>*

Deletes the record with id '<record id>' from the table specified in form '<form name>' as primary table. Additional action *FormElement* of type *beforeDelete* or *afterDelete* will be fired too.

10.9.6.3 Examples

```
10.sql = SELECT 'U:table=Person&r=123|q:Do you want delete John Doe?' AS _paged
10.sql = SELECT 'U:form=person-main&r=123|q:Do you want delete John Doe?' AS _paged
```

10.9.7 Columns: _Page[X]

- Similar to *_page[X]*
- Parameter are position dependent and therefore without a qualifier!

```
"[<page id|alias>[&param=value&...]] | [text] | [tooltip] | [question parameter] |␣
↪[class] | [target] | [render mode]" as _Pagee.
```

10.9.8 Column: _Paged

- Similar to *_paged*
- Parameter are position dependent and therefore without a qualifier!

```
"[table=<table name>&r=<record id>[&param=value&...] | [text] | [tooltip] | [question␣
↪parameter] | [class] | [render mode]" as _Paged.
"[form=<form name>&r=<record id>[&param=value&...] | [text] | [tooltip] | [question␣
↪parameter] | [class] | [render mode]" as _Paged.
```

10.9.9 Column: _vertical

Use instead *Table: vertical column title*

Warning: The '... AS _vertical' is deprecated - do not use it anymore.

Render text vertically. This is useful for tables with limited column width. The vertical rendering is achieved via CSS tranformations (rotation) defined in the style attribute of the wrapping tag. You can optionally specify the rotation angle.

Syntax

```
10.sql = SELECT "<text>|[<angle>]" AS _vertical
```

Parameter	Description	Default value
<text>	The string that should be rendered vertically.	none
<angle>	How many degrees should the text be rotated? The angle is measured clockwise from baseline of the text.	*270*

The text is surrounded by some HTML tags in an effort to make other elements position appropriately around it. This works best for angles close to 270 or 90.

Minimal Example

```
10.sql = SELECT "Hello" AS _vertical
20.sql = SELECT "Hello|90" AS _vertical
20.sql = SELECT "Hello|-75" AS _vertical
```

10.9.10 Column: _mailto

Easily create Email links.

Syntax

```
10.sql = SELECT "<email address>|[<link text>]" AS _mailto
```

Parameter	Description	Default value
<emailaddress>	The email address where the link should point to.	none
<linktext>	The text that should be displayed on the website and be linked to the email address. This will typically be the name of the recipient. If this parameter is omitted, the email address will be displayed as link text.	none

Minimal Example

```
10.sql = SELECT "john.doe@example.com" AS _mailto
```

Advanced Example

```
10.sql = SELECT "john.doe@example.com|John Doe" AS _mailto
```

10.9.11 Column: _sendmail

Format:

```
t:<TO:email[,email]>|f:<FROM:email>|s:<subject>|b:<body>
    [|c:<CC:email[,email]]>[|B:<BCC:email[,email]]>[|r:<REPLY-TO:email>]
    [|A:<flag autosubmit: on/off>][|g:<grId>][|x:<xId>][|y:<xId2>][|z:<xId3>][|h:
↪<mail header>]
    [|e:<subject encode: encode/decode/none>][E:<body encode: encode/decode/none>
↪][|mode:html]
    [|C][d:<filename of the attachment>][|F:<file to attach>][|u:<url>][|p:<T3 uri>]
```

The following parameters can also be written as complete words for ease of use:

```
to:<email[,email]>|from:<email>|subject:<subject>|body:<body>
    [|cc:<email[,email]]>[|bcc:<email[,email]]>[|reply-to:<email>]
    [|autosubmit:<on/off>][|grid:<grid>][|xid:<xId>][|xid2:<xId2>][|xid3:<xId3>
↪][|header:<mail header>]
    [|mode:html]
```

Send emails. Every mail will be logged in the table *mailLog*. Attachments are supported.

Syntax

```
10.sql = SELECT "t:john@doe.com|f:jane@doe.com|s:Reminder tomorrow|b:Please dont miss␣
↪the meeting tomorrow" AS _sendmail
10.sql = SELECT "t:john@doe.com|f:jane@doe.com|s:Reminder tomorrow|b:Please dont miss␣
↪the meeting tomorrow|A:off|g:1|x:2|y:3|z:4" AS _sendmail
```

Token short / long	**Parameter**	**Description**	**Required**
f from	email	**FROM**: Sender of the email. Optional: 'real-name <john@doe.com>'	yes
t to	email[,email]	**TO**: Comma separated list of receiver email addresses. Optional: *real-name <john@doe.com>*	yes
c cc	email[,email]	**CC**: Comma separated list of receiver email addresses. Optional: 'real-name <john@doe.com>'	yes
B bcc	email[,email]	**BCC**: Comma separated list of receiver email addresses. Optional: 'real-name <john@doe.com>'	yes
r reply-to	REPLY-TO:email	**Reply-to**: Email address to reply to (if different from sender)	yes
s subject	Subject	**Subject**: Subject of the email	yes
b body	Body	**Body**: Message - see also: *html-formatting*	yes
h header	Mail header	**Custom mail header**: Separate multiple header with \r\n	yes
F	Attach file	**Attachment**: File to attach to the mail. Repeatable.	
u	Attach created PDF of a given URL	**Attachment**: Convert the given URL to a PDF and attach it the mail. Repeatable.	
p	Attach created PDF of a given T3 URL	**Attachment**: Convert the given URL to a PDF and attach it the mail. Repeatable.	
d	Filename of the attachment	**Attachment**: Useful for URL to PDF converted attachments. Repeatable.	
C	Concat multiple F|p|u| together	**Attachment: All following (until the next 'C') 'F|p|u' concatenated to o** Repeatable.	
	flagAutoSubmit 'on' / 'off'	If 'on' (default), add mail header 'Auto-Submitted:	yes

- **e|E**: By default, QFQ stores values 'htmlspecialchars()' encoded. If such values have to send by email, the html entities are unwanted. Therefore the default setting for 'subject' und 'body' is to decode the values via 'htmlspecialchars_decode()'. If this is not wished, it can be turned off by *e=none* and/or *E=none*.

Minimal Example

```
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available." AS _sendmail
```

This will send an email with subject *Latest News* from company@example.com to john.doe@example.com.

Advanced Examples

```
10.sql = SELECT "t:customer1@example.com,Firstname Lastname <customer2@example.com>,␣
↪Firstname Lastname <customer3@example.com>| \\
                f:company@example.com|s:Latest News|b:The new version is now␣
↪available.|r:sales@example.com|A:on|g:101|x:222|c:ceo@example.com|B:backup@example.
↪com" AS _sendmail
```

This will send an email with subject *Latest News* from company@example.com to customer1, customer2 and customer3 by using a realname for customer2 and customer3 and suppress generating of OoO answer if any receiver is on vacation. Additional the CEO as well as backup will receive the mail via CC and BCC.

For debugging, please check *Redirect all mail to (catch all)*.

Mail Body HTML Formatting

In order to send an email with HTML formatting, such as bold text or bullet lists, specify 'mode=html'. The subsequent contents will be interpreted as HTML and is rendered correctly by most email programs.

10.9.11.1 Attachment

The following options are provided to attach files to an email:

Token	Example	Comment
F	F:fileadmin/file3.pdf	Single file to attach
u	u:www.example.com/index.html?key=val&k2=v2	A URL will be converted to a PDF and than attached.
p	p:?id=export&r=123&_sip=1	A SIP protected local T3 page. Will be converted to a PDF and than attached.
d	d:myfile.pdf	Name of the attachment in the email.
C	C|u:http://www.example.com|F:file1.pdf|C|F:file2.pdf	Concatenate all named sources to one PDF file. The souces has to be PDF files or a web page, which will be converted to a PDF first.

Any combination (incl. repeating them) are possible. Any source will be added as a single attachment.

Optional any number of sources can be concatenated to a single PDF file: 'C|F:<file1>|F:<file2>|p:export&a=123'.

Examples in Report:

```
# One file attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|F:fileadmin/summary.pdf" AS _sendmail

# Two files attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|F:fileadmin/summary.pdf|F:fileadmin/detail.pdf" AS _
↪sendmail
```

(continues on next page)

(continued from previous page)

```
# Two files and a webpage (converted to PDF) are attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|F:fileadmin/summary.pdf|F:fileadmin/detail.pdf|p:?
↪id=export&r=123|d:person.pdf" AS _sendmail

# Two webpages (converted to PDF) are attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|p:?id=export&r=123|d:person123.pdf|p:?id=export&
↪r=234|d:person234.pdf" AS _sendmail

# One file and two webpages (converted to PDF) are *concatenated* to one PDF and␣
↪attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|C|F:fileadmin/summary.pdf|p:?id=export&r=123|p:?id=export&
↪r=234|d:complete.pdf" AS _sendmail

# One T3 webpage, protected by a SIP, are attached.
10.sql = SELECT "t:john.doe@example.com|f:company@example.com|s:Latest News|b:The new␣
↪version is now available.|p:?id=export&r=123&_sip=1|d:person123.pdf" AS _sendmail
```

10.9.12 Column: _img

Renders images. Allows to define an alternative text and a title attribute for the image. Alternative text and title text are optional.

- If no alternative text is defined, an empty alt attribute is rendered in the img tag (since this attribute is mandatory in HTML).
- If no title text is defined, the title attribute will not be rendered at all.

Syntax

```
10.sql = SELECT "<path to image>|[<alt text>]|[<title text>]" AS _img
```

Parameter	Description	Default value/behaviour
<pathtoimage>	The path to the image file.	none
<alttext>	Alternative text. Will be displayed if image can't be loaded (alt attribute of img tag).	empty string
<titletext>	Text that will be set as image title in the title attribute of the img tag.	no title attribute rendered

Minimal Example

```
10.sql = SELECT "fileadmin/img/img.jpg" AS _img
```

Advanced Examples

```
10.sql = SELECT "fileadmin/img/img.jpg|Aternative Text" AS _img            # alt=
↪"Alternative Text, no title
20.sql = SELECT "fileadmin/img/img.jpg|Aternative Text|" AS _img           # alt=
↪"Alternative Text, no title
```

(continues on next page)

(continued from previous page)

```
30.sql = SELECT "fileadmin/img/img.jpg|Aternative Text|Title Text" AS _img # alt=
↪"Alternative Text, title="Title Text"
40.sql = SELECT "fileadmin/img/img.jpg|Alternative Text" AS _img           # alt=
↪"Alternative Text", no title
50.sql = SELECT "fileadmin/img/img.jpg" AS _img                            # empty␣
↪alt, no title
60.sql = SELECT "fileadmin/img/img.jpg|" AS _img                           # empty␣
↪alt, no title
70.sql = SELECT "fileadmin/img/img.jpg||Title Text" AS _img                # empty␣
↪alt, title="Title Text"
80.sql = SELECT "fileadmin/img/img.jpg||" AS _img                          # empty␣
↪alt, no title
```

10.9.13 Column: _exec

Run any command on the web server.

- The command is run via web server, so with the uid of the web server.
- The current working directory is the current web instance (e.g. `/var/www/html`) .
- All text send to 'stdout' will be returned.
- Text send to 'stderr' is not returned at all.
- If 'stderr' should be shown, redirect the output:

```
SELECT 'touch /root 2>&1' AS _exec
```

- If 'stdout' / 'stderr' should not be displayed, redirect the output:

```
SELECT 'touch /tmp >/dev/null' AS _exec
SELECT 'touch /root 2>&1 >/dev/null' AS _exec
```

- Multiple commands can be concatenated by ;:

```
SELECT 'date; date' AS _exec
```

- If the return code is not 0, the string '[<rc>] ', will be prepended.
- If it is not wished to see the return code, just add `true` to fake rc of 0 (only the last rc will be reported):

```
SELECT 'touch /root; true' AS _exec
```

Syntax

```
<command>
```

Parameter	Description	Default value
<command>	The command that should be executed on the server.	none

Minimal Examples

```
10.sql = SELECT "ls -s" AS _exec
20.sql = SELECT "./batchfile.sh" AS _exec
```

10.9.14 Column: _script

Run a php function defined in an external script.

- All **column parameters are passed** as an associative array to the function as the first argument.
- The second argument (here called $qfq) is an object which acts as an **interface to QFQ functionality** (see below).
- **The current working directory inside the function is the current web instance (e.g. location of index.php).**
  - Hint: Inside the script `dirname(__FILE__)` gives the path of the script.
- All **output (e.g. using echo) will be rendered** by the special column as is.
- If the function returns an associative array, then the **key-value pairs will be accessible via the VARS store 'V'**.
- If the function throws an **exception** then a standard QFQ error message is shown.
- Text sent to 'stderr' by the php function is not returned at all.
- **The script has access to the following qfq functions using the interface (see examples below):**
  - **$qfq::apiCall($method, $url, $data = '', $header = [], $timeout = 5)**
    - **arguments:**
      - string $method: can be PUT/POST/GET/DELETE
      - string $url
      - string $data: a json string which will be added as GET parameters or as POST fields respectively.
      - array $header: is of the form ['Content-type: text/plain', 'Content-length: 100']
      - int $timeout: is the number of seconds to wait until call is aborted.
    - **return array:**
      - [0]: Http status code
      - [1]: API answer as string.
  - **$qfq::getVar($key, $useStores = 'FSRVD', $sanitizeClass = '', &$foundInStore = '', $typeMessageViolate = 'c')**
    - **arguments:**
      - string $key: is the name of qfq variable
      - string $useStores: are the stores in which variable is searched (in order from left to right). see *Store*.
      - string $sanitizeClass: (see *Sanitize class*)
      - string $foundInStore: is filled with the name of the store in which the variable was found.
      - **string $typeMessageViolate: defines what to return if the sanitize class was violated:**

        'c' : returns '!!<sanitize class>!!'

        '0' : returns '0'

        'e' : returns ''

* **return string|false:**
  - The value of the variable if found.
  - A placeholder if the variable violates the sanitize class. (see argument *$typeMessageViolate*)
  - *false* if the variable was not found.

Column Parameters

Token	Example	Comment
F	F:fileadmin/scripts/my_script.php	Path to the custom script relative to the current web instance
call	call:my_function	PHP function to call
arg	arg:a1=Hello&a2=World"	Arguments are parsed and passed to the function together with the other parameters

Example

- QFQ report

```
5.sql = SELECT "IAmInRecordStore" AS _savedInRecordStore
10.sql = SELECT "F:fileadmin/scripts/my_script.php|call:my_function|arg:a1=Hello&
↪a2=World" AS _script
20.sql = SELECT "<br><br>Returened value: {{IAmInVarStore:V:alnumx}}"
```

- PHP script (*fileadmin/scripts/my_script.php*)

```
<?php
 function my_function($param, $qfq) {

     echo 'The first argument contains all attributes including "F" and "c":<br>';
     print_r($param);

     echo '<br><br>get variable from record store:<br>';
     print_r($qfq::getVar('savedInRecordStore', 'RE'));

     echo '<br><br>Make API call:<br>';
     list($http_code, $answer) = $qfq::apiCall('GET', 'google.com');
     echo 'Http code: ' . $http_code;

     // Returned array fills VARS store
     return ["IAmInVarStore" => "FooBar"];
 }
```

- Output

```
The first argument contains all parameters including "F", "call" and "arg":
Array ( [a1] => Hello [a2] => World [F] => fileadmin/scripts/my_script.php [call]␣
↪=> my_function [arg] => a1=Hello&a2=World )

get variable from record store:
IAmInRecordStore

Make API call:
Http code: 301

Returened value: FooBar
```

10.9.15 Column: _pdf | _file | _zip

Detailed explanation: *Download*

Most of the other Link-Class attributes can be used to customize the link.

```
10.sql = SELECT "[options]" AS _pdf, "[options]" AS _file, "[options]" AS _zip

with: [options] = [d:<exportFilename>][|p:<params>][|U:<params>][|u:<url>][|F:file][|t:
↪<text>][|a:<message>][|o:<tooltip>][|c:<class>][|r:<render mode>]
```

- Parameter are position independent.
- *<params>*: see *Parameter and (element) sources*
- For column *_pdf* and *_zip*, the element sources *p:...*, *U:...*, *u:...*, *F:...* might repeated multiple times.
- To only render the page content without menus add the parameter *type=2*. For example: *U:id=pageToPrint&type=2&_sip=1&r=', r.id*
- Example:

```
10.sql = SELECT "F:fileadmin/test.pdf" as _pdf,  "F:fileadmin/test.pdf" as _file,␣
↪ "F:fileadmin/test.pdf" as _zip
10.sql = SELECT "p:id=export&r=1" as _pdf,  "p:id=export&r=1" as _file,
↪"p:id=export&r=1" as _zip

10.sql = SELECT "t:Download PDF|F:fileadmin/test.pdf" as _pdf,  "t:Download␣
↪PDF|F:fileadmin/test.pdf" as _file,  "t:Download ZIP|F:fileadmin/test.pdf" as _
↪zip
10.sql = SELECT "t:Download PDF|p:id=export&r=1" as _pdf,  "t:Download␣
↪PDF|p:id=export&r=1" as _file,  "t:Download ZIP|p:id=export&r=1" as _zip

10.sql = SELECT "d:complete.pdf|t:Download PDF|F:fileadmin/test1.pdf|F:fileadmin/
↪test2.pdf" as _pdf, "d:complete.zip|t:Download ZIP|F:fileadmin/test1.
↪pdf|F:fileadmin/test2.pdf" as _zip

10.sql = SELECT "d:complete.pdf|t:Download PDF|F:fileadmin/test.pdf|p:id=export&
↪r=1|u:www.example.com" AS _pdf
```

10.9.16 Column: _savePdf

Generated PDFs can be stored directly on the server with this functionality. The link query consists of the following parameters:

- One or more element sources (such as *F:*, *U:*, *p:*, see *Parameter and (element) sources*), including possible wkhtmltopdf parameters
- The export filename and path as *d:* - for security reasons, this path has to start with *fileadmin/* and end with *.pdf*.

Tips:

- Please note that this option does not render anything in the front end, but is executed each time it is parsed. You may want to add a check to prevent multiple execution.
- It is not advised to generate the filename with user input for security reasons.
- If the target file already exists it will be overwriten. To save individual files, choose a new filename, for example by adding a timestamp.

Example:

```
SELECT "d:fileadmin/result.pdf|F:fileadmin/_temp_/test.pdf" AS _savePdf
SELECT "d:fileadmin/result.pdf|F:fileadmin/_temp_/test.pdf|U:id=test&--
↪orientation=landscape" AS _savePdf
```

10.9.17 Column: _thumbnail

For file *T:<pathFileName>* a thumbnail will be rendered, saved (to be reused) and a HTML *<img>* tag is returned, With the SIP encoded thumbnail.

The thumbnail:

- Size is specified via *W:<dimension>*. The file is only rendered once and subsequent access is delivered via a local QFQ cache.
- Will be rendered, if the source file is newer than the thumbnail or if the thumbnail dimension changes.
- The caching is done by building the MD5 of pathFileName and thumbnail dimension.
- Of multi page files like PDFs, the first page is used as the thumbnail.

All file formats, which 'convert' ImageMagick (https://www.imagemagick.org/) supports, can be used. Office file formats are not supported. Due to speed and quality reasons, SVG files will be converted by inkscape. If a file format is not known, QFQ tries to show a corresponding file type image provided by Typo3 - such an image is not scaled.

In *Configuration* the exact location of *convert* and *inkscape* can be configured (optional) as well as the directory names for the cached thumbnails.

Token	Example	Comment
T	T:fileadmin/file3.pdf	File render a thumbnail
W	W:200x, W:x100, W:200x100	Dimension of the thumbnail: '<width>x<height>. Both parameter are otional. If non is given the default is W:150x
s	s:1, s:0	Optional. Default: *s:1*. If SIP is enabled, the rendered URL is a link via *api/download.php?...* Else a direct pathFileName.
r	r:7	Render Mode. Default 'r:0'. With 'r:7' only the url will be delivered.

The render mode '7' is useful, if the URL of the thumbnail have to be used in another way than the provided html-'<img>' tag. Something like *<body style="background-image:url(bgimage.jpg)">* could be solved with *SELECT "<body style="background-image:url(", 'T:fileadmin/file3.pdf' AS _thumbnail, ')">'*

Example:

```
# SIP protected, IMG tag, thumbnail width 150px
10.sql = SELECT 'T:fileadmin/file3.pdf' AS _thumbnail

# SIP protected, IMG tag, thumbnail width 50px
20.sql = SELECT 'T:fileadmin/file3.pdf|W:50' AS _thumbnail

# No SIP protection, IMG tag, thumbnail width 150px
30.sql = SELECT 'T:fileadmin/file3.pdf|s:0' AS _thumbnail

# SIP protected, only the URL to the image, thumbnail width 150px
40.sql = SELECT 'T:fileadmin/file3.pdf|s:1|r:7' AS _thumbnail
```

10.9.17.1 Dimension

ImageMagick support various settings to force the thumbnail size. See https://www.imagemagick.org/script/command-line-processing.php#geometry or http://www.graphicsmagick.org/GraphicsMagick.html#details-geometry.

10.9.17.2 Cleaning

By default, the thumbnail directories are never cleaned. It's a good idea to install a cronjob which purges all files older than 1 year:

```
find /path/to/files -type f -mtime +365 -delete
```

10.9.17.3 Render

Public thumbnails are rendered at the time when the T3 QFQ record is executed. *Secure* thumbnails are rendered when the 'download.php?s=...' is called. The difference is, that the 'public' thumbnails blocks the page load until all thumbnails are rendered, instead the *secure* thumbnails are loaded asynchonous via the browser - the main page is already delivered to browser, all thumbnails appearing after a time.

A way to *pre render* thumbnails, is a periodically called (hidden) T3 page, which iterates over all new uploaded files and triggers the rendering via column *_thumbnail*.

10.9.17.4 Thumbnail: secure

Mode 'secure' is activated via enabling SIP (*s:1*, default). The thumbnail is saved under the path *thumbnailDirSecure* as configured in *Configuration*.

The secure path needs to be protected against direct file access by the webmaster / webserver configuration too.

QFQ returns a HTML 'img'-tag:

```
<img src="api/download.php?s=badcaffee1234">
```

10.9.17.5 Thumbnail: public

Mode 'public' has to be explicit activated by specifying *s:0*. The thumbnail is saved under the path *thumbnailDirPublic* as configured in *Configuration*.

QFQ returns a HTML 'img'-tag:

```
<img src="{{thumbnailDirPublic:Y}}/<md5 hash>.png">
```

10.9.18 Column: _monitor

Detailed explanation: *Monitor*

Syntax

```
10.sql = SELECT 'file:<filename>|tail:<number of last lines>|append:<0 or 1>|interval:
↪<time in ms>|htmlId:<id>' AS _monitor
```

Parameter	Description	Default value/behaviour
<file-name>	The path to the file. Relative to T3 installation directory or absolute.	none
<tail>	Number of last lines to show	30
<append>	0: Retrieved content replaces current. 1: Retrieved content will be added to current.	0
<htmlId>	Reference to HTML element to whose content replaced by the retrieve one.	monitor-1

10.9.19 Copy to clipboard

Token	Example	Comment
y[:<content>]	y, y:some content	Initiates 'copy to clipboard' mode. Source might given text or page or url
F:<pathFileName>	F:fileadmin/protected/data.R	pathFileName in DocumentRoot

Example:

```
10.sql = SELECT 'y:hello world (yank)|t:content direct (yank)' AS _yank
                , 'y:hello world (link)|t:content direct (link)' AS _link
                , CONCAT('F:', p.pathFileName,'|t:File (yank)|o:', p.pathFileName) AS␣
↪_yank
                , CONCAT('y|F:', p.pathFileName,'|t:File (link)|o:', p.pathFileName)␣
↪AS _link
            FROM Person AS p
```

10.9.20 API Call QFQ Report (e.g. AJAX)

Note: QFQ Report functionality protected by SIP offered to simple API calls: `typo3conf/ext/qfq/Classes/Api/dataReport.php?s=....`

- General use API call to fire a specific QFQ tt-content record. Useful for e.g. AJAX calls. No Typo3 is involved. *No FE-Group access control.*
- This defines just a simple API endpoint. For defining a rest API see: *REST*.
- **Custom response headers can be defined by setting the variable *apiResponseHeader* in the record store.**
  - Multiple headers should be separated by *n* or *rn*. e.g.: *Content-Type: application/jsonncustom-header: fooBar*
- **If the api call succeeds the rendered content of the report is returned as is. (no additional formatting, no JSON encoding)**
  - You can use MYSQL to create Json. See: MYSQL create Json and MariaDB Json functions
- If a QFQ error occurs then a http-status of 400 is returned together with a JSON encoded response of the form: *{"status":"error", "message":"..."}*

Example QFQ record JS (with tt_content.uid=12345):

```
5.sql = SELECT "See console log for output"

# Register SIP with given arguments.
10.sql = SELECT 'U:uid=12345&arg1=Hello&arg2=World|s|r:8' AS '_link|col1|_hide'

# Build JS
10.tail = <script>
    console.log('start api request');
    $.ajax({
    url: 'typo3conf/ext/qfq/Classes/Api/dataReport.php?s={{&col1:RE}}',
    data: {arg3:456, arg4:567},
    method: 'POST',
    dataType: 'TEXT',
    success: function(response, status, jqxhr) {console.log(response); console.
↪log(jqxhr.getAllResponseHeaders());},
    error: function(jqXHR, textStatus, errorThrown) {console.log(jqXHR.responseText,␣
↪textStatus, errorThrown);}
    });
  </script>
```

Example QFQ record called by above AJAX:

```
# Create a dedicated tt-content record (on any T3 page, might be on the same page as␣
↪the JS code).
# The example above assumes that this record has the tt_content.uid=12345.
render = api
10.sql = SELECT '{{arg1:S}} {{arg2:S}} {{arg3:C}} {{arg4:C}}', NOW()
, 'Content-Type: application/json\ncustom-header: fooBar' AS _apiResponseHeader
```

Example text returned by the above AJAX call:

```
Hello World 456 5672020-09-22 18:09:47
```

10.9.21 REST Client

Note: POST and GET data to external REST interfaces or other API services.

Access to external services via HTTP / HTTPS is triggered via special column name *restClient*. The received data might be processed in subsequent calls.

Example:

```
# Retrieve information. Received data is delivered in JSON and decoded / copied on␣
↪the fly to CLIENT store (CLIENT store is emptied beforehand)
10.sql = SELECT 'n:https://www.dummy.ord/rest/person/id/123' AS _restClient
20.sql = SELECT 'Status: {{http-status:C}}<br>Name: {{name:C:alnumx}}<br>Surname: {
↪{surname:C:alnumx}}'

# Simple POST request via https. Result is printed on the page.
10.sql = SELECT 'n:https://www.dummy.ord/rest/person/id/123|method:POST|content:{"name
↪":"John";"surname":"Doe"}' AS _restClient
```

Token	Example | Comment	
n	n:https://www.dummy.ord/rest/person |	
method	method:POST	GET, POST, PUT or DELETE
content	content:{"name":"John";"surname":"Doe"}	Depending on the REST server JSON might be expected
header	*see below*	
timeout	timeout:5	Default: 5 seconds.

Header

- Each header must be separated by `\r\n` or *n*.
- An explicit given header will overwrite the named default header.
- Default header:
  - *content-type: application/json* - if *content* starts with a `{`.
  - *content-type: text/plain* - if *content* does not start with a `{`.
  - *connection: close* - Necessary for HTTP 1.1.

Result received

- After a *REST client* call is fired, QFQ will wait up to *timeout* seconds for the answer.
- By default, the whole received answer will be shown. To suppress the output: `... AS '_restClient|_hide'`
- The variable `{{http-status:C}}` shows the **'HTTP status code<https://en.wikipedia.org/wiki/List_of_HTTP_status_cod**
  A value starting with '2..' shows success.
- In case of an error, `{{error-message:C:allbut}}` shows some details.
- **In case the returned answer is a valid JSON string, it is flattened and automatically copied to STORE_CLIENT with corr**
  - NOTE: The CLIENT store is emptied beforehand!

JSON answer example:

```
Answer from Server:  { 'name' : 'John', 'address' : {'city' : 'Bern'} }
Retrieve the values via:  {{name:C:alnumx}}, {{city:C:alnumx}}
```

10.10 Special SQL Functions (prepared statements)

10.10.1 QBAR: Escape QFQ Delimiter

The SQL function QBAR(text) replaces "|" with "\|" in *text* to prevent conflicts with the QFQ special column notation. In general this function should be used when there is a chance that unplanned '|'-characters occur.

Example:

```
10.sql = SELECT CONCAT('p:notes|t:Information: ', QBAR(Note.title), '|b') AS _link␣
↪FROM Note
```

In case 'Note.title' contains a '|' (like 'fruit | numbers'), it will confuse the '... AS _link' class. Therefore it's necessary to 'escape' (adding a '' in front of the problematic character) the bar which is done by using *QBAR()*.

10.10.2 QCC: Escape colon / coma

The SQL function QCC(text) replaces ":" with "\:" and "," with "\," in *text* to prevent conflicts with the QFQ notation.

10.10.3 QNL2BR: Convert newline to HTML '
'

The SQL function QNL2BR(text) replaces *LF* or *CR/LF* by <*br*>. This can be used for data (containing LF) to output on a HTML page with correctly displayed linefeed.

Example:

```
10.sql = SELECT QNL2BR(Note.title) FROM Note
```

One possibility how *LF* comes into the database is with form elements of type *textarea* if the user presses *enter* inside.

10.10.4 QMORE: Truncate Long Text - more/less

The SQL function QMORE(text, n) truncates *text* if it is longer than *n* characters and adds a "more.." button. If the "more..." button is clicked, the whole text is displayed. The stored procedure QMORE() will inject some HTML/CSS code.

Example:

```
10.sql = SELECT QMORE("This is a text which is longer than 10 characters", 10)
```

Output:

```
This is a `more..`
```

10.10.5 QIFEMPTY: if empty show token

The SQL function QIFEMPTY(input, token) returns 'token' if 'input' is 'empty string' / '0' / '0000-00-00' / '0000-00-00 00:00:00'.

Example:

```
10.sql = SELECT QIFEMPTY('hello world','+'), QIFEMPTY('','-')
```

Output:

```
hello world-
```

10.10.6 QDATE_FORMAT: format a timestamp, show '-' if empty

The SQL function QDATE_FORMAT(timestamp) returns 'dd.mm.YYYY hh:mm', if 'timestamp' is 0 returns '-'

Example:

```
10.sql = SELECT QDATE_FORMAT( '2019-12-31 23:55:41' ), ' / ', QDATE_FORMAT( 0 ), ' /
↪', QDATE_FORMAT( '' )
```

Output:

```
31.12.2019 23:55 / - / -
```

10.10.7 QSLUGIFY: clean a string

Convert a string to only use alphanumerical characters and '-'. Characters with accent will be replaced without the accent. Non alphanumerical characters are stripped off. Spaces are replaced by '-'. All characters are lowercase.

Example:

```
10.sql = SELECT QSLUGIFY('abcd ABCD ae.ä.oe.ö.ue.ü z[]{}()<>.,?Z')
```

Output:

```
abcd-abcd-ae-a-oe-o-ue-u-z-z
```

10.10.8 strip_tags: strip html tags

The SQL function strip_tags(input) returns 'input' without any HTML tags.

Example:

```
10.sql = SELECT strip_tags('<a href="https://example.com"><b>my name</b> <i>is john</
↪i></a> - end of sentence')
```

Output:

```
my name is john - end of sentence
```

10.11 Download

Download offers:

- Single file - download a single file (any type),
- PDF create - one or concatenate several files (uploaded) and/or web pages (=HTML to PDF) into one PDF output file,
- ZIP archive - filled with several files ('uploaded' or 'HTML to PDF'-converted).
- Excel - created from scratch or fill a template xlsx with database values.

The downloads are SIP protected. Only the current user can use the link to download files.

By using the *_link* column name:

- the option *d:...* initiate creating the download link and optional specifies an export filename,
- the optional *M:...* (Mode) specifies the export type (file, pdf, zip, export),
- setting *s:1* is recommended for the download function (file / path name is hidden to the user),
- the alttext *a:...* specifies a message in the download popup.

By using *_pdf*, *_Pdf*, *_file*, *_File*, *_zip*, *_Zip*, *_excel* as column name, the options *d*, *M* and *s* will be set.

All files will be read by PHP - therefore the directory might be protected against direct web access. This is the preferred option to offer secure downloads via QFQ.

In case the download needs a persistant URL (no SIP, no user session), a regular link, pointing directly to a file, have to be used - the download functionality described here is not appropriate for such a scenario. If necessary, *Column: _savePdf* can be used to generate such a file.

10.11.1 Parameter and (element) sources

- *download*: *d[:<exportFilename>]*
  - *exportFilename* = <filename for save as> - Name, offered in the 'File save as' browser dialog. Default: 'output.<ext>'.

    If there is no *exportFilename* defined, then the original filename is taken (if there is one, else: output...).

    The user typically expects meaningful and distinct file names for different download links.
- *popupMessage*: *a:<text>* - will be displayed in the popup window during download. If the creating/download is fast, the window might disappear quickly.
- *mode*: *M:<mode>*
  - *mode* = <file | pdf | zip | excel>
    - If *M:file*, the mime type is derived dynamically from the specified file. In this mode, only one element source is allowed per download link (no concatenation).
    - In case of multiple element sources, only *pdf*, *zip* and *excel* (template mode) is supported.
    - If *M:zip* is used together with *p:...*, *U:...* or *u:..*, those HTML pages will be converted to PDF. Those files get generic filenames inside the archive.
    - If not specified, the **default** 'Mode' depends on the number of specified element sources (=file or web page):
      - If only one *file* is specified, the default is *file*.
      - If there is a) a page defined or b) multiple elements, the default is *pdf*.
- *element sources* - for *M:pdf* or *M:zip*, all of the following element sources may be specified multiple times. Any combination and order of these options are allowed.
  - *file*: *F:<pathFileName>* - relative or absolute pathFileName offered for a) download (single), or to be concatenated in a PDF or ZIP.
  - *page*: *p:id=<t3 page>&<key 1>=<value 1>&<key 2>=<value 2>&...&<key n>=<value n>*.
    - By default, the options given to wkhtml will *not* be encoded by a SIP!
    - To encode the parameter via SIP: Add '_sip=1' to the URL GET parameter.

      E.g. *p:id=form&_sip=1&form=Person&r=1*.

      In that way, specific sources for the *download* might be SIP encrypted.
    - Any current HTML cookies will be forwarded to/via *wkhtml*. This includes the current FE Login as well as any QFQ session. Also the current User-Agent are faked via the *wkhtml* page request.
    - If there are trouble with accessing FE_GROUP protected content, please check *wkhtmltopdf*.
  - *url*: *u:<url>* - any URL, pointing to an internal or external destination.
  - *uid*: *uid:<tt-content record id>* - the tt_content.uid of a QFQ PageContent record (shown on hover in the backend). This will render only the specified QFQ content record, without any Typo3 layout elements (Menu, Body,...) QFQ will retrieve the tt-content's bodytext from the Typo3 database, parse it, and render it as a PDF. Parameters can be passed: *uid:<tt-content record id>[&arg1=value1][&arg2=value2][...]*

and will be available in the SIP store for the QFQ PageContent, or passed as wkhtmltopdf arguments, if applicable.

- *WKHTML Options* for *page*, *urlParam* or *url*:
  - The 'HTML to PDF' will be done via *wkhtmltopdf*.
  - All possible options, suitable for *wkhtmltopdf*, can be submitted in the *p:...*, *u:...* or *U:...* element source. Check wkhtmltopdf.txt for possible options. Be aware that key/value tuple in the documentation is separated by a space, but to respect the QFQ key/value notation of URLs, the key/value tuple in *p:...*, *u:...* or *U:...* has to be separated by '='. Please see last example below.
  - If an option contains an '&' it must be escaped with double \ . See example.

Most of the other Link-Class attributes can be used to customize the link as well.

Example *_link*:

```
# single `file`. Specifying a popup message window text is not necessary, cause a␣
↪file directly accessed is fast.
SELECT "d:file.pdf|s|t:Download|F:fileadmin/pdf/test.pdf" AS _link

# single `file`, with mode
SELECT "d:file.pdf|M:pdf|s|t:Download|F:fileadmin/pdf/test.pdf" AS _link

# three sources: two pages and one file
SELECT "d:complete.pdf|s|t:Complete PDF|p:id=detail&r=1|p:id=detail2&r=1|F:fileadmin/
↪pdf/test.pdf" AS _link

# three sources: two pages and one file
SELECT "d:complete.pdf|s|t:Complete PDF|p:id=detail&r=1|p:id=detail2&r=1|F:fileadmin/
↪pdf/test.pdf" AS _link

# three sources: two pages and one file, parameter to wkhtml will be SIP encoded
SELECT "d:complete.pdf|s|t:Complete PDF|p:id=detail&r=1&_sip=1|p:id=detail2&r=1&_
↪sip=1|F:fileadmin/pdf/test.pdf" AS _link

# three sources: two pages and one file, the second page will be in landscape and␣
↪pagesize A3
SELECT "d:complete.pdf|s|t:Complete PDF|p:id=detail&r=1|p:id=detail2&r=1&--
↪orientation=Landscape&--page-size=A3|F:fileadmin/pdf/test.pdf" AS _link

# One source and a header file. Note: the parameter to the header URL is escaped with␣
↪double backslash.
SELECT "d:complete.pdf|s|t:Complete PDF|p:id=detail2&r=1&--orientation=Landscape&--
↪header={{URL:R}}?indexp.php?id=head\\&L=1|F:fileadmin/pdf/test.pdf" AS _link
```

Example *_pdf*, *_zip*:

```
# File 1: p:id=1&--orientation=Landscape&--page-size=A3
# File 2: p:id=form
# File 3: F:fileadmin/file.pdf
SELECT 't:PDF|a:Creating a new PDF|p:id=1&--orientation=Landscape&--page-
↪size=A3|p:id=form|F:fileadmin/file.pdf' AS _pdf

# File 1: p:id=1
# File 2: u:http://www.example.com
# File 3: F:fileadmin/file.pdf
SELECT 't:PDF - 3 Files|a:Please be patient|p:id=1|u:http://www.example.
↪com|F:fileadmin/file.pdf' AS _pdf
```

(continues on next page)

(continued from previous page)

```
# File 1: p:id=1
# File 2: p:id=form
# File 3: F:fileadmin/file.pdf
SELECT CONCAT('t:ZIP - 3 Pages|a:Please be patient|p:id=1|p:id=form|F:', p.
↪pathFileName) AS _zip
```

Use the *–print-media-type* as wkhtml option to access the page with media type 'printer'. Depending on the website configuration this switches off navigation and background images.

10.11.2 Rendering PDF letters

wkhtmltopdf, with the header and footer options, can be used to render multi page PDF letters (repeating header, pagination) in combination with dynamic content. Such PDFs might look-alike official letters, together with logo and signature.

Best practice:

1. Create a clean (=no menu, no website layout) letter layout in a separated T3 branch:

```
page = PAGE
page.typeNum = 0
page.includeCSS {
  10 = typo3conf/ext/qfq/Resources/Public/Css/qfq-letter.css
}

// Grant access to any logged in user or specific development IPs
[usergroup = *] || [IP = 127.0.0.1,192.168.1.* ]
  page.10 < styles.content.get
[else]
  page.10 = TEXT
  page.10.value = access forbidden
[global]
```

2. Create a T3 *body* page (e.g. page alias: 'letterbody') with some content. Example static HTML content:

```
<div class="letter-receiver">
  <p>Address</p>
</div>
<div class="letter-sender">
 <p><b>firstName name</b><br>
  Phone +00 00 000 00 00<br>
  Fax +00 00 000 00 00<br>
 </p>
</div>

<div class="letter-date">
  Zurich, 01.12.2017
</div>

<div class="letter-body">
 <h1>Subject</h1>

 <p>Dear Mrs...</p>
 <p>Lucas ipsum dolor sit amet organa solo skywalker darth c-3p0 anakin jabba␣
↪mara greedo skywalker.</p>
```

(continues on next page)

(continued from previous page)

```
 <div class="letter-no-break">
 <p>Regards</p>
 <p>Company</p>
 <img class="letter-signature" src="">
 <p>Firstname Name<br>Function</p>
 </div>
</div>
```

3. Create a T3 letter-*header* page (e.g. page alias: 'letterheader') , with only the header information:

```
<header>
<img src="fileadmin/logo.png" class="letter-logo">

<div class="letter-unit">
  <p class="letter-title">Department</p>
  <p>
   Company name<br>
   Company department<br>
   Street<br>
   City
  </p>
</div>
</header>
```

4. Create a) a link (Report) to the PDF letter or b) attach the PDF (on the fly rendered) to a mail. Both will call the *wkhtml* via the *download* mode and forwards the necessary parameter.

Use in *report*:

```
sql = SELECT CONCAT('d:Letter.pdf|t:',p.firstName, ' ', p.name
                    , '|p:id=letterbody&pId=', p.id, '&_sip=1'
                    , '&--margin-top=50mm'
                    , '&--header-html={{BASE_URL_PRINT:Y}}?id=letterheader'

                    # IMPORTANT: set margin-bottom to make the footer visible!
                    , '&--margin-bottom=20mm'
                    , '&--footer-right="Seite: [page]/[toPage]"'
                    , '&--footer-font-size=8&--footer-spacing=10') AS _pdf

             FROM Person AS p ORDER BY p.id
```

Sendmail. Parameter:

```
sendMailAttachment={{SELECT 'd:Letter.pdf|t:', p.firstName, ' ', p.name,
↪'|p:id=letterbody&pId=', p.id, '&_sip=1&--margin-top=50mm&--margin-bottom=20mm&--
↪header-html={{BASE_URL_PRINT:Y}}?id=letterheader&--footer-right="Seite: [page]/
↪[toPage]"&--footer-font-size=8&--footer-spacing=10' FROM Person AS p WHERE p.id={
↪{id:S}} }}
```

Replace the static content elements from 2. and 3. by QFQ Content elements as needed:

```
10.sql = SELECT '<div class="letter-receiver"><p>', p.name AS '_+br', p.street AS '_
↪+br', p.city AS '_+br', '</p>'
          FROM Person AS p
          WHERE p.id={{pId:S}}
```

10.11.3 Export area

This description might be interesting if a page can't be protected by SIP.

To offer protected pages, e.g. directly referenced files (remember: this is not recommended) in download links, the regular FE_GROUPs can't be used, cause the download does not have the current user privileges (it's a separate process, started as the webserver user).

Create a separated export tree in Typo3 Backend, which is IP access restricted. Only localhost or the FE_GROUP 'admin' is allowed to access:

```
tmp.restrictedIPRange = 127.0.0.1,::1
[IP = {$tmp.restrictedIPRange} ][usergroup = admin]
   page.10 < styles.content.get
[else]
   page.10 = TEXT
   page.10.value = Please access from localhost or log in as 'admin' user.
[global]
```

10.11.4 Excel export

This chapter explains how to create Excel files on the fly.

Hint: For just up/downloading of excel files (without modification), check the generic Form *Type: upload* element and the report 'download' (*column_pdf*) function.

The Excel file is build in the moment when the user request it, by clicking on a download link.

Mode building:

- *New*: The export file will be completely build from scratch.
- *Template*: The export file is based on an earlier uploaded xlsx file (template). The template itself is unchanged.

Injecting data into the Excel file is done in the same way in both modes: a Typo3 page (rendered without any HTML header or tags) contains one or more Typo3 QFQ records. Those QFQ records will create plain ASCII output.

If the export file has to be customized (colors, pictures, headlines, . . .), the *Template* mode is the preferred option. It's much easier to do all customizations via Excel and creating a template than by coding in QFQ / Excel export notation.

10.11.4.1 Setup

- Create a special column name *_excel* (or *_link*) in QFQ/Report. As a source, define a T3 PageContent, which has to deliver the dynamic content (also *excel-export-sample*).

```
SELECT CONCAT('d:final.xlsx|M:excel|s:1|t:Excel (new)|uid:<tt-content record id>
↪') AS _link
```

- Create a T3 PageContent which delivers the content.
  - It is recommended to use the *uid:<tt-content record id>* syntax for excel imports, because there should be no html code on the resulting content. QFQ will retrieve the PageContent's bodytext from the Typo3 database, parse it, and pass the result as the instructions for filling the excel file.
  - Parameters can be passed: *uid:<tt-content record id>?param=<value1>&param2=<value2>* and will be accessible in the SIP Store (S) in the QFQ PageContent.
  - Use the regular QFQ Report syntax to create output.

- The newline at the end of every line needs to be CHAR(10). To make it simpler, the special column name ... *AS _XLS* (see _XLS, _XLSs, _XLSb, _XLSn) can be used.
- One option per line.
- Empty lines will be skipped.
- Lines starting with '#' will be skipped (comments). Inline comment signs are NOT recognized as comment sign.
- Separate <keyword> and <value> by '='.

Keyword	Example	Description
'worksheet'	worksheet=main	Select a worksheet in case the excel file has multiple of them.
'mode'	mode=insert	Values: insert,overwrite.
'position'	position=A1	Default is 'A1'. Use the excel notation.
'newline'	newline	Start a new row. The column will be the one of the last 'position' statement.
'str', 's'	s=hello world	Set the given string on the given position. The current position will be shift one to the right. If the string contains newlines, option'b' (base64) should be used.
'b'	b=aGVsbG8gd29ybGQK	Same as 's', but the given string has to Base64 encoded and will be decoded before export.
'n'	n=123	Set number on the given position. The current position will be shift one to the right.
'f'	f==SUM(A5:C6)	Set a formula on the given position. The current position will be shift one to the right.

Create a output like this:

```
position=D11
s=Hello
s=World
s=First Line
newline
s=Second line
n=123
```

This fills D11, E11, F11, D12

In Report Syntax:

```
# With ... AS _XLS (token explicit given)
10.sql = SELECT 'position=D10' AS _XLS
                , 's=Hello' AS _XLS
                , 's=World' AS _XLS
                , 's=First Line' AS _XLS
                , 'newline' AS _XLS
                , 's=Second line' AS _XLS
                , 'n=123' AS _XLS

# With ... AS _XLSs (token generated internally)
20.sql = SELECT 'position=D20' AS _XLS
                , 'Hello' AS _XLSs
                , 'World' AS _XLSs
                , 'First Line' AS _XLSs
                , 'newline' AS _XLS
```

(continues on next page)

(continued from previous page)

```
                , 'Second line' AS _XLSs
                , 'n=123' AS _XLS

# With ... AS _XLSb (token generated internally and content is base64 encoded)
30.sql = SELECT 'position=D30' AS _XLS
                , '<some content with special characters like newline/carriage return>
↪' AS _XLSb
```

Excel export samples (54 is a example <tt-content record id>):

```
# From scratch (both are the same, one with '_excel' the other with '_link')
SELECT CONCAT('d:new.xlsx|t:Excel (new)|uid:54') AS _excel
SELECT CONCAT('d:new.xlsx|t:Excel (new)|uid:54|M:excel|s:1') AS _link

# Template
SELECT CONCAT('d:final.xlsx|t:Excel (template)|F:fileadmin/template.xlsx|uid:54') AS _
↪excel

# With parameter (via SIP) - get the Parameter on page 'exceldata' with '{{arg1:S}}'␣
↪and '{{arg2:S}}'
SELECT CONCAT('d:final.xlsx|t:Excel (parameter)|uid:54&arg1=hello&arg2=world') AS _
↪excel
```

10.11.4.2 Best practice

To keep the link of the Excel export close to the Excel export definition, the option *Report: render* can be used.

On a **single** T3 page create **two** QFQ tt-content records:

tt-content record 1:

- Type: QFQ
- Content:

```
render = single
10.sql = SELECT CONCAT('d:new.xlsx|t:Excel (new)|uid:54|M:excel|s:1') AS _link
```

tt-content record 2 (uid=54):

- Type: QFQ
- Content:

```
render = api
10.sql = SELECT 'position=D10' AS _XLS
                , 's=Hello' AS _XLS
                , 's=World' AS _XLS
                , 's=First Line' AS _XLS
                , 'newline' AS _XLS
                , 's=Second line' AS _XLS
                , 'n=123' AS _XLS
```

10.12 Dropdown Menu

Creates a menu with custom links. The same notation and options are used as with regular QFQ links.

Format String:

```
<dropdown menu symbol options>||<menu entry 1>||<menu entry 2>||...
```

Each menu entry is separated by two bars! A menu entry itself might contain multiple single bars.

Example 1:

```
SELECT 'z||p:home|t:Home|o:Jump to home||p:person&form=person&r=123|t:Edit: John Doe|s
↪' AS _link
```

This defines a menu (three vertical buttons) - a click on it shows two menu entries: 'Home' and 'Edit: John Doe'

Format the dropdown menu symbol:

- *Glyph*: Via *G:<glyphicon name>* any glyphicon can be defined. To hide the default glyph, specify: *G:0*.
- *Text*: Via *t:Menu* an additional text will be displayed for the menu symbol.
- *Tooltip*: Via *o:Detail menu* a tooltip is defined.
- *Render mode*: Via *r:3* the menu is disabled. No menu entries / links / sip are rendered.
- *Button*: Via *b* the dropdown meny symbol will be rendered with a button. Also *b:<style>* might set the BS color.

Format a menu entry:

- *qfq link*: All options as with a regular QFQ link.
- *header*: If a text starts with '===', it becomes a header in the dropdown menu. Multiple headers are possible. Headers can't be a link. An additional *r:1* is necessary.
- *separator*: If a text is exactly '—', it becomes a separator line between two menu entries. An additional *r:1* is necessary.
- *disabled menu entry*: If a text starts with '—' (like separator), the following text becomes a disable menu entry. An additional *r:1* is necessary.

Example 2:

```
SELECT CONCAT('z|t:Menu|G:0|o:Please select an option',
              '||p:home|t:Home|o:Jump to home|G:glyphicon-home|b:0',
              '||r:1|t:---',
              '||p:person&form=person&r=123|t:Edit: John Doe|s|q:Really edit?
↪|G:glyphicon-user|b:0',
              '||t:===Header|r:1',
              '||d|p:form&form=person&r=',p.id,'|s|t:Download Person|b:0',
              '||r:1|t:---Disabled entry') AS _link
```

Line 1: The dropdown menu symbol definition, with a text 'Menu' *t:Menu*, but without the three vertical bullets *G:0* and a tooltip for the menu *o:Please select an option*.

Line 2: First menu entry. Directs to T3 page 'home' *p:home*. *t:Home* sets the menu entry text. *G:glyphicon-home* set's glyphicon symbol in the menu entry. *b:0* switches off the button, which has been implicit activated by the use of *G:...*.

Line 3: A separator line.

Line 4: A SIP encoded edit record link, with a question dialog.

Line 5: A header line.

Line 6: A PDF download.

Line 7: A disabled menu entry.

10.13 WebSocket

Sending messages via WebSocket and receiving the answer is done via:

```
SELECT 'w:ws://<host>:<port>/<path>|t:<message>' AS _websocket
```

Instead of `... AS _websocket` it's also possible to use `... AS _link` (same syntax).

The answer from the socket (if there is something) is written to output and stored in STORE_RECORD by given column (in this case 'websocket' or 'link').

Tip: To suppress the direct output, add `|_hide` to the column name.

Example:

```
SELECT 'w:ws://<host>:<port>/<path>|t:<message>' AS '_websocket|_hide'
```

Tip: To define a uniq column name (easy access by column name via STORE_RECORD) add `|myName` (replace *myName*).

Example:

```
SELECT 'w:ws://<host>:<port>/<path>|t:<message>' AS '_websocket|myName'
```

Tip: Get the answer from STORE_RECORD by using `{{&...`. Check *access-column-values*.

Example:

```
SELECT 'w:ws://<host>:<port>/<path>|t:<message>' AS '_websocket|myName'

Results:

  '{{myName:R}}'  >> 'w:ws://<host>:<port>/<path>|t:<message>'
  '{{&myName:R}}'  >> '<received socket answer>'
```

10.14 Drag and drop

10.14.1 Order elements

Ordering of elements via *HTML5 drag and drop* is supported via QFQ. Any element to order should be represented by a database record with an order column. If the elements are unordered, they will be ordered after the first 'drag and drop' move of an element.

Functionality divides into:

- Display: the records will be displayed via QFQ/report.

- Order records: updates of the order column are managed by a specific definition form. The form is not a regular form (e.g. there are no FormElements), instead it's only a container to held the SQL update query as well as providing access control via SIP. The form is automatically called via AJAX.

10.14.1.1 Part 1: Display list

Display the list of elements via a regular QFQ content record. All 'drag and drop' elements together have to be nested by a HTML element. Such HTML element:

- With *class="qfq-dnd-sort"*.
- With a form name: *{{'form=<form name>' AS _data-dnd-api}}* (will be replaced by QFQ)
- Only *direct* children of such element can be dragged.
- Every children needs a unique identifier *data-dnd-id="<unique>"*. Typically this is the corresponding record id.
- The record needs a dedicated order column, which will be updated through API calls in time.

A * * example HTML output (HTML send to the browser):

```
<div class="qfq-dnd-sort" data-dnd-api="typo3conf/ext/qfq/Classes/Api/dragAndDrop.php?
↪s=badcaffee1234">
    <div class="anyClass" id="<uniq1>" data-dnd-id="55">
        Numbero Uno
    </div>
    <div class="anyClass" id="<uniq2>" data-dnd-id="18">
        Numbero Deux
    </div>
    <div class="anyClass" id="<uniq3>" data-dnd-id="27">
        Numbero Tre
    </div>
</div>
```

A typical QFQ report which generates those * * HTML:

```
10 {
  sql = SELECT '<div id="anytag-', n.id,'" data-dnd-id="', n.id,'">' , n.note, '</div>
↪'
                FROM Note AS n
                WHERE grId=28
                ORDER BY n.ord

  head = <div class="qfq-dnd-sort" {{'form=dndSortNote&grId=28' AS _data-dnd-api}}">
  tail = </div>
}
```

A *<table>* based setup is also possible. Note the attribute *data-columns="3"* - this generates a dropzone which is the same column width as the outer table.

```
<table>
    <tbody class="qfq-dnd-sort" data-dnd-api="typo3conf/ext/qfq/Classes/Api/
↪dragAndDrop.php?s=badcaffee1234" data-columns="3">
        <tr> class="anyClass" id="<uniq1>" data-dnd-id="55">
            <td>Numbero Uno</td><td>Numbero Uno.2</td><td>Numbero Uno.3</td>
        </tr>
        <tr class="anyClass" id="<uniq2>" data-dnd-id="18">
```

(continues on next page)

(continued from previous page)

```
            <td>Numbero Deux</td><td>Numbero Deux.2</td><td>Numbero Deux.3</td>
        </tr>
        <tr class="anyClass" id="<uniq3>" data-dnd-id="27">
            <td>Numbero Tre</td><td>Numbero Tre.2</td><td>Numbero Tre.3</td>
        </tr>
    </tbody>
</table>
```

A typical QFQ report which generates this HTML:

```
10 {
  sql = SELECT '<tr id="anytag-', n.id,'" data-dnd-id="', n.id,'" data-columns="3">' ,
↪ n.id AS '_+td', n.note AS '_+td', n.ord AS '_+td', '</tr>'
                FROM Note AS n
                WHERE grId=28
                ORDER BY n.ord

  head = <table><tbody class="qfq-dnd-sort" {{'form=dndSortNote&grId=28' AS _data-dnd-
↪api}} data-columns="3">
  tail = </tbody><table>
}
```

Show / update order value in the browser

The 'drag and drop' action does not trigger a reload of the page. In case the order number is shown and the user does a 'drag and drop', the order number shows the old. To update the dragable elements with the latest order number, a predefined html id has to be assigned them. After an update, all changed order number (referenced by the html id) will be updated via AJAX.

The html id per element is defined by *qfq-dnd-ord-id-<id>* where *<id>* is the record id. Same example as above, but with an updated *n.ord* column:

```
10 {
  sql = SELECT '<tr id="anytag-', n.id,'" data-dnd-id="', n.id,'" data-columns="3">' ,
↪ n.id AS '_+td', n.note AS '_+td',
                '<td id="qfq-dnd-ord-id-', n.id, '">', n.ord, '</td></tr>'
                FROM Note AS n
                WHERE grId=28
                ORDER BY n.ord

  head = <table><tbody class="qfq-dnd-sort" {{'form=dndSortNote&grId=28' AS _data-dnd-
↪api}} data-columns="3">
  tail = </tbody><table>
}
```

10.14.1.2 Part 2: Order records

A dedicated *Form*, without any *FormElements*, is used to define the reorder logic (database update definition).

Fields:

- Name: <custom form name> - used in Part 1 in the *_data-dnd-api* variable.
- Table: <table with the element records> - used to update the records specified by *dragAndDropOrderSql*.

- Parameter:

Attribute	Description
orderInterval = <number>	Optional. By default '10'. Might be any number > 0.
orderColumn = <column name>	Optional. By default 'ord'.
dragAndDropOrderSql = {{!SELECT n.id AS id, n.ord AS ord FROM Note AS n ORDER BY n.ord}}	Query to selects the *same* records as the report in the same *order!* Inconsistencies results in order differences. The columns *id* and *ord* are *mandatory.*

The form related to the example of part 1 ('div' or 'table'):

```
Form.name: dndSortNote
Form.table: Note
Form.parameter: orderInterval = 1
Form.parameter: orderColumn = ord
Form.parameter: dragAndDropOrderSql = {{!SELECT n.id AS id, n.ord AS ord FROM Note AS␣
↪n WHERE n.grId={{grId:S0}} ORDER BY n.ord}}
```

Re-Order:

QFQ iterates over the result set of *dragAndDropOrderSql*. The value of column *id* have to correspond to the dragged HTML element (given by *data-dnd-id*). Reordering always start with *orderInterval* and is incremented by *orderInterval* with each record of the result set. The client reports a) the id of the dragged HTML element, b) the id of the hovered element and c) the dropped position of above or below the hovered element. This information is compared to the result set and changes are applied where appropriate.

Take care that the query of part 1 (display list) does a) select the same records and b) in the same order as the query defined in part 2 (order records) via *dragAndDropOrderSql*.

If you find that the reorder does not work at expected, those two sql queries are not identical.

10.15 QFQ Icons

Located under ' typo3conf/ext/qfq/Resources/Public/icons':

```
black_dot.png     bullet-red.gif      checked-red.gif     emoji.svg    mail.gif    ␣
↪pointer.svg    up.gif
blue_dot.png      bullet-yellow.gif   checked-yellow.gif  gear.svg     marker.svg  ␣
↪rectangle.svg  upload.gif
bulb.png          checkboxinvert.gif  construction.gif    help.gif     new.gif     ␣
↪resize.svg     wavy-underline.gif
bullet-blue.gif   checked-blue.gif    copy.gif            home.gif     note.gif    ␣
↪show.gif       zoom.svg
bullet-gray.gif   checked-gray.gif    delete.gif          icons.svg    pan.svg     ␣
↪trash.svg
bullet-green.gif  checked-green.gif   down.gif            info.gif     paste.gif   ␣
↪turnLeft.svg
bullet-pink.gif   checked-pink.gif    edit.gif            loading.gif  pencil.svg  ␣
↪turnRight.svg
```

10.16 QFQ CSS Classes

- *qfq-table-50*, *qfq-table-80*, *qfq-table-100* - assigned to <*table*>, set min-width and column width to 'auto'.

- Background Color: *qfq-color-grey-1*, *qfq-color-grey-2* - assigned to different tags (table, row, cell).
- *qfq-100* - assigned to different tags, makes an element 'width: 100%'.
- *qfq-left*- assigned to different tags, Text align left.
- *qfq-sticky* - assigned to <*thead*>, makes the header sticky.
- *qfq-badge*, *qfq-badge-error*, *qfq-badge-warning*, *qfq-badge-success*, *qfq-badge-info*, *qfq-badge-invers* - colorized BS3 badges.
- *letter-no-break* - assigned to a *div* will protect a paragraph (CSS: page-break-before: avoid;) not to break around a page border (converted to PDF via wkhtml). Take care that *qfq-letter.css* is included in TypoScript setup.

10.17 Bootstrap

- Table: *table*
- Table > hover: *table-hover*
- Table > condensed: *table-condensed*

Example:

```
10.sql = SELECT id, name, firstName, ...
10.head = <table class='table table-condensed qfq-table-50'>
```

- *qfq-100*, *qfq-left* - makes e.g. a button full width and aligns the text left.

Example:

```
10.sql = SELECT "p:home&r=0|t:Home|c:qfq-100 qfq-left" AS _pagev
```

10.18 Tablesorter

QFQ includes a third-party client-side table sorter: https://mottie.github.io/tablesorter/docs/index.html

To turn any table into a sortable table:

- Ensure that your QFQ installation imports the appropriate js/css files, see *Setup CSS & JS*.
- Add the *class="tablesorter"* to your <*table*> element.
- Take care the <*table*> has a <*thead*> and <*tbody*> tag.
- Evey table with active tablesorter should have a uniq HTML id.

Important: Custom settings will be saved per table automatically in the browser local storage. To distinguish different table settings, define an uniq HTML id per table. Example: <*table class="tablesorter" id="{{pageAlias:T}}-person"*> - the *{{pageAlias:T}}* makes it easy to keep the overview over given name on the site.

The *tablesorter* options:

- Class *tablesorter-filter* enables row filtering.
- Class *tablesorter-pager* adds table paging functionality. A page navigation is shown.
- Class *tablesorter-column-selector* adds a column selector widget.

- Activate/Save/Delete *views*: Insert inside of a table html-tag the command:

```
{{ '<uniqueName>' AS _tablesorter-view-saver }}
```

  This adds a menu to save the current view (column filters, selected columns, sort order).

  - *<uniqueName>* should be a name which is at least unique inside the typo3 content element. Example:

```
<table {{ 'allperson' AS _tablesorter-view-saver }} class="tablesorter␣
↪tablesorter-filter tablesorter-column-selector" id="{{pageAlias:T}}-demo">␣
↪... </table>
```

  - 'Views' can be saved as:
    - public: every user will see the *view* and can modify it.
    - private: only the user who created the *view* will see/modify it.
    - **readonly: manually mark a *view* as readonly (no FE User can change it) by setting column *readonly='true'* in t** *Setting* of the corresponding view (identified by *name*).
  - Views will be saved in the table 'Setting'.
  - Include 'font-awesome' CSS in your T3 page setup: *typo3conf/ext/qfq/Resources/Public/Css/font-awesome.min.css* to get the icons.
  - The view 'Clear' is always available and can't be modified.
  - To preselect a view, append a HTML anker to the current URL. Get the anker by selecting the view and copy it from the browser address bar. Example:

```
https://localhost/index.php?id=person#allperson=public:email
```

    - 'allperson' is the '<uniqueName>' of the *tablesorter-view-saver* command.
    - 'public' means the view is tagged as 'public' visible.
    - 'email' is the name of the view, as it is shown in the dropdown list.
  - If there is a public view with the name 'Default' and a user has no choosen a view earlier, that one will be selected.

Customization of tablesorter:

- Add the desired classes or data attributes to your table html, e.g.:
  - Disable sorting *class="sorter-false"* on a '<th>' to disable sorting on that column (or: *data-sorter="false"*).
  - Disable filter *class="filter-false"* on a '<th>' to hide the filter field for that column
  - see docs for more options: https://mottie.github.io/tablesorter/docs/index.html
- You can pass in a default configuration object for the main *tablesorter()* function by using the attribute *data-tablesorter-config* on the table. Use JSON syntax when passing in your own configuration, such as:

```
data-tablesorter-config='{"theme":"bootstrap","widthFixed":true,"headerTemplate":"
↪{content} {icon}","dateFormat":"ddmmyyyy","widgets":["uitheme","filter",
↪"saveSort","columnSelector"],"widgetOptions":{"filter_columnFilters":true,
↪"filter_reset":".reset","filter_cssFilter":"form-control","columnSelector_
↪mediaquery":false} }'
```

- If the above customization options are not enough, you can output your own HTML for the pager and/or column selector, as well as your own *$(document).ready()* function with the desired config. In this case, it is

recommended not to use the above *tablesorter* classes since the QFQ javascript code could interfere with your javascript code.

Example:

```
10 {
  sql = SELECT id, CONCAT('form&form=person&r=', id) AS _Pagee, lastName, title FROM␣
↪Person
  head = <table class="table tablesorter tablesorter-filter tablesorter-pager␣
↪tablesorter-column-selector" id="{{pageAlias:T}}-ts1">
      <thead><tr><th>Id</th><th class="filter-false sorter-false">Edit</th>
      <th>Name</th><th class="filter-select" data-placeholder="Select a title">Title</
↪th>
      </tr></thead><tbody>
  tail = </tbody></table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
}
```

10.19 Monitor

Display a (log)file from the server, inside the browser, which updates automatically by a user defined interval. Access to the file is SIP protected. Any file on the server is possible.

- On a Typo3 page, define a HTML element with a unique html-id. E.g.:

```
10.head = <pre id="monitor-1">Please wait</pre>
```

- On the same Typo3 page, define an SQL column '_monitor' with the necessary parameter:

```
10.sql = SELECT 'file:fileadmin/protected/log/sql.
↪log|tail:50|append:1|refresh:1000|htmlId:monitor-1' AS _monitor
```

- Short version with all defaults used to display system configured sql.log:

```
10.sql = SELECT 'file:{{sqlLog:Y}}' AS _monitor, '<pre id="monitor-1" style=
↪"white-space: pre-wrap;">Please wait</pre>'
```

10.20 Calendar View

QFQ is shipped with the JavaScript library https://fullcalendar.io/ (respect that QFQ uses V3) to provides various calendar views.

Docs: https://fullcalendar.io/docs/v3

Include the JS & CSS files via Typoscript

- typo3conf/ext/qfq/Resources/Public/Css/fullcalendar.min.css
- typo3conf/ext/qfq/Resources/Public/JavaScript/moment.min.js
- typo3conf/ext/qfq/Resources/Public/JavaScript/fullcalendar.min.js

Integration: Create a * * with

- CSS class “qfq-calendar”
- Tag *data-config*. The content is a Javascript object.

Example:

```
10.sql = SELECT 'Calendar, Standard'
10.tail = <div class="qfq-calendar"
                data-config='{
                     "themeSystem": "bootstrap3",
                     "height": "auto",
                     "defaultDate": "2020-01-13",
                     "weekends": false,
                     "defaultView": "agendaWeek",
                     "minTime": "05:00:00",
                     "maxTime": "20:00:00",
                     "businessHours": { "dow": [ 1, 2, 3, 4 ], "startTime": "10:00",
↪"endTime": "18:00" },
                     "events": [
                       { "id": "a", "title": "my event",
                       "start": "2020-01-21"},
                       { "id": "b", "title": "my other event", "start": "2020-01-
↪16T09:00:00", "end": "2020-01-16T11:30:00"}
                      ]}'>
            </div>

# "now" is in the past to switchoff 'highlight of today'
20.sql = SELECT 'Calendar, 3 day, custom color, agend&list' AS '_+h2'
20.tail = <div class="qfq-calendar"
                data-config='{
                     "themeSystem": "bootstrap3",
                     "height": "auto",
                     "header": {
                       "left": "title",
                       "center": "",
                       "right": "agenda,listWeek"
                      },
                     "defaultDate": "2020-01-14",
                     "now": "1999-12-31",
                     "allDaySlot": false,
                     "weekends": false,
                     "defaultView": "agenda",
                     "dayCount": 3,
                     "minTime": "08:00:00",
                     "maxTime": "18:00:00",
                     "businessHours": { "dow": [ 1, 2, 3, 4 ], "startTime": "10:00",
↪"endTime": "18:00" },
                     "events": [
                       { "id": "a", "title": "my event",       "start": "2020-01-
↪15T10:15:00", "end": "2020-01-15T11:50:00", "color": "#25adf1", "textColor": "#000"}
↪,
                       { "id": "b", "title": "my other event", "start": "2020-01-
↪16T09:00:00", "end": "2020-01-16T11:30:00", "color": "#5cb85c", "textColor": "#000"}
↪,
                       { "id": "c", "title": "Eventli",        "start": "2020-01-
↪15T13:10:00", "end": "2020-01-15T16:30:00", "color": "#fbb64f", "textColor": "#000"}
↪,
                       { "id": "d", "title": "Evento",         "start": "2020-01-
↪15T13:50:00", "end": "2020-01-15T15:00:00", "color": "#fb4f4f", "textColor": "#000"}
↪,
```

(continues on next page)

(continued from previous page)

```
                    { "id": "d", "title": "Busy",           "start": "2020-01-
↪14T09:00:00", "end": "2020-01-14T12:00:00", "color": "#ccc",    "textColor": "#000"}
↪,
                    { "id": "e", "title": "Banana",         "start": "2020-01-
↪16T13:30:00", "end": "2020-01-16T16:00:00", "color": "#fff45b", "textColor": "#000"}
                ]}'>
            </div>
```

10.21 Report Examples

The following section gives some examples of typical reports.

10.21.1 Basic Queries

One simple query:

```
10.sql = SELECT "Hello World"
```

Result:

```
Hello World
```

Two simple queries:

```
10.sql = SELECT "Hello World"
20.sql = SELECT "Say hello"
```

Result:

```
Hello WorldSay hello
```

Two simple queries, with break:

```
10.sql = SELECT "Hello World<br>"
20.sql = SELECT "Say hello"
```

Result:

```
Hello World
Say hello
```

10.21.2 Accessing the database

Real data, one single column:

```
10.sql = SELECT p.firstName FROM ExpPerson AS p
```

Result:

```
BillieElvisLouisDiana
```

Real data, two columns:

```
10.sql = SELECT p.firstName, p.lastName FROM ExpPerson AS p
```

Result:

```
BillieHolidayElvisPresleyLouisArmstrongDianaRoss
```

The result of the SQL query is an output, row by row and column by column, without adding any formatting information. See *Formatting Examples* for examples of how the output can be formatted.

10.21.3 Formatting Examples

Formatting (i.e. wrapping of data with HTML tags etc.) can be achieved in two different ways:

One can add formatting output directly into the SQL by either putting it in a separate column of the output or by using concat to concatenate data and formatting output in a single column.

One can use 'level' keys to define formatting information that will be put before/after/between all rows/columns of the actual levels result.

Two columns:

```
# Add the formatting information as a column
10.sql = SELECT p.firstName, " " , p.lastName, "<br>" FROM ExpPerson AS p
```

Result:

```
Billie Holiday
Elvis Presley
Louis Armstrong
Diana Ross
```

One column 'rend' as linebreak - no extra column '
' needed:

```
10.sql = SELECT p.firstName, " " , p.lastName, " ", p.country FROM ExpPerson AS p
10.rend = <br>
```

Result:

```
Billie Holiday USA
Elvis Presley USA
Louis Armstrong USA
Diana Ross USA
```

Same with 'fsep' (column " " removed):

> 10.sql = SELECT p.firstName, p.lastName, p.country FROM ExpPerson AS p 10.rend =
 10.fsep = " "

Result:

```
Billie Holiday USA
Elvis Presley USA
Louis Armstrong USA
Diana Ross USA
```

More HTML:

```
10.sql = SELECT p.name FROM ExpPerson AS p
10.head = <ul>
10.tail = </ul>
10.rbeg = <li>
10.rend = </li>
```

Result:

```
o Billie Holiday
o Elvis Presley
o Louis Armstrong
o Diana Ross
```

The same as above, but with braces:

```
10 {
  sql = SELECT p.name FROM ExpPerson AS p
  head = <ul>
  tail = </ul>
  rbeg = <li>
  rend = </li>
}
```

Two queries:

```
10.sql = SELECT p.name FROM ExpPerson AS p
10.rend = <br>
20.sql = SELECT a.street FROM ExpAddress AS a
20.rend = <br>
```

Two queries: nested:

```
# outer query
10.sql = SELECT p.name FROM ExpPerson AS p
10.rend = <br>

# inner query
10.10.sql = SELECT a.street FROM ExpAddress AS a
10.10.rend = <br>
```

- For every record of '10', all records of 10.10 will be printed.

Two queries: nested with variables:

```
# outer query
10.sql = SELECT p.id, p.name FROM ExpPerson AS p
10.rend = <br>

# inner query
10.10.sql = SELECT a.street FROM ExpAddress AS a WHERE a.pId='{{10.id}}'
10.10.rend = <br>
```

- For every record of '10', all assigned records of 10.10 will be printed.

Two queries: nested with hidden variables in a table:

```
10.sql = SELECT p.id AS _pId, p.name FROM ExpPerson AS p
10.rend = <br>
```

(continues on next page)

(continued from previous page)

```
# inner query
10.10.sql = SELECT a.street FROM ExpAddress AS a WHERE a.pId='{{10.pId}}'
10.10.rend = <br>
```

Same as above, but written in the nested notation:

```
10 {
  sql = SELECT p.id AS _pId, p.name FROM ExpPerson AS p
  rend = <br>

  10 {
  # inner query
    sql = SELECT a.street FROM ExpAddress AS a WHERE a.pId='{{10.pId}}'
    rend = <br>
  }
}
```

Best practice *recommendation* for using parameter - see *Access column values*:

```
10 {
  sql = SELECT p.id AS _pId, p.name FROM ExpPerson AS p
  rend = <br>

  10 {
  # inner query
    sql = SELECT a.street FROM ExpAddress AS a WHERE a.pId='{{pId:R}}'
    rend = <br>
  }
}
```

Create HTML tables. Each column is wrapped in `<td>`, each row is wrapped in `<tr>`:

```
10 {
  sql = SELECT p.firstName, p.lastName, p.country FROM Person AS p
  head = <table class="table">
  tail = </table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
}
```

Maybe a few columns belongs together and should be in one table column.

Joining columns, variant A: firstName and lastName in one table column:

```
10 {
  sql = SELECT CONCAT(p.firstName, ' ', p.lastName), p.country FROM Person AS p
  head = <table class="table">
  tail = </table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
}
```

Joining columns, variant B: firstName and lastName in one table column:

```
10 {
  sql = SELECT '<td>', p.firstName, ' ', p.lastName, '</td><td>', p.country, '</td>'␣
↪FROM Person AS p
  head = <table class="table">
  tail = </table>
  rbeg = <tr>
  rend = </tr>
}
```

Joining columns, variant C: firstName and lastName in one table column. Notice `fbeg`, `fend` and `fskipwrap`:

```
10 {
  sql = SELECT '<td>', p.firstName, ' ', p.lastName, '</td>', p.country FROM Person␣
↪AS p
  head = <table class="table">
  tail = </table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
  fskipwrap = 1,2,3,4,5
}
```

Joining columns, variant D: firstName and lastName in one table column. Notice `fbeg`, `fend` and `fskipwrap`:

```
10 {
  sql = SELECT CONCAT('<td>', p.firstName, ' ', p.lastName, '</td>') AS '_noWrap', p.
↪country FROM Person AS p
  head = <table class="table">
  tail = </table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
}
```

10.21.4 Recent List

A nice feature is to show a list with last changed records. The following will show the 10 last modified (Form or FormElement) forms:

```
10 {
  sql = SELECT CONCAT('p:{{pageAlias:T}}&form=form&r=', f.id, '|t:', f.name,'|o:',␣
↪GREATEST(MAX(fe.modified), f.modified)) AS _page
          FROM Form AS f
          LEFT JOIN FormElement AS fe
            ON fe.formId = f.id
          GROUP BY f.id
          ORDER BY GREATEST(MAX(fe.modified), f.modified) DESC
          LIMIT 10
  head = <h3>Recent Forms</h3>
  rsep = , 
}
```

10.21.5 Table: vertical column title

To orientate a column title vertical, use the QFQ CSS classe *qfq-vertical* in td|th and *qfq-vertical-text* around the text.

HTML example (second column title is vertical):

```
<table><thead>
  <tr>
    <th>horizontal</th>
    <th class="qfq-vertical"><span class="qfq-vertical-text">text vertical</span></th>
  </tr>
</thead></table>
```

QFQ example:

```
10 {
  sql = SELECT title FROM Settings ORDER BY title
  fbeg = <th class="qfq-vertical"><span class="qfq-vertical-text">
  fend = </span></th>
  head = <table><thead><tr>
  rend = </tr></thead>
  tail = </table>

  20.sql = SELECT ...
}
```

10.21.6 STORE_USER examples

Keep variables per user session.

10.21.6.1 Two pages (pass variable)

Sometimes it's useful to have variables per user (=browser session). Set a variable on page 'A' and retrieve the value on page 'B'.

Page 'A' - set the variable:

```
10.sql = SELECT 'hello' AS '_=greeting'
```

Page 'B' - get the value:

```
10.sql = SELECT '{{greeting:UE}}'
```

If page 'A' has never been opened with the current browser session, nothing is printed (STORE_EMPTY gives an empty string). If page 'A' is called, page 'B' will print 'hello'.

10.21.6.2 One page (collect variables)

A page will be called with several SIP variables, but not at all at the same time. To still get all variables at any time:

```
# Normalize
10.sql = SELECT '{{order:USE:::sum}}' AS '_=order', '{{step:USE:::5}}' AS _step, '{
↪{direction:USE:::ASC}}' AS _direction
```

(continues on next page)

(continued from previous page)

```
# Different links
20.sql = SELECT 'p:{{pageAlias:T}}&order=count|t:Order by count|b|s' AS _link,
                'p:{{pageAlias:T}}&order=sum|t:Order by sum|b|s' AS _link,
                'p:{{pageAlias:T}}&step=10|t:Step=10|b|s' AS _link,
                'p:{{pageAlias:T}}&step=50|t:Step=50|b|s' AS _link,
                'p:{{pageAlias:T}}&direction=ASC|t:Order by up|b|s' AS _link,
                'p:{{pageAlias:T}}&direction=DESC|t:Order by down|b|s' AS _link

30.sql = SELECT * FROM Items ORDER BY {{order:U}} {{direction:U}} LIMIT {{step:U}}
```

10.21.6.3 Simulate/switch user: feUser

Just set the STORE_USER variable 'feUser'.

All places with *{{feUser:T}}* has to be replaced by *{{feUser:UT}}*:

```
# Normalize
10.sql = SELECT '{{feUser:UT}}' AS '_=feUser'

# Offer switching feUser
20.sql = SELECT 'p:{{pageAlias:T}}&feUser=account1|t:Become "account1"|b|s' AS _link,
                'p:{{pageAlias:T}}&feUser={{feUser:T}}|t:Back to own identity|b|s' AS␣
↪_link,
```

10.21.6.4 Semester switch (remember last choice)

A current semester is defined via configuration in STORE_SYSTEM '{{semId:Y}}'. The first column in 10.sql *'{{semId:SUY}}' AS '_=semId'* saves the semester to STORE_USER via '_=semId'. The priority 'SUY' takes either the latest choose (STORE_SIP) or reuse the last used (STORE_USER) or (first time call during browser session) takes the default from config (STORE_SYSTEM):

```
# Semester switch
10 {
  sql = SELECT '{{semId:SUY}}' AS '_=semId'
                , CONCAT('p:{{pageAlias:T}}&semId=', sp.id, '|t:', QBAR(sp.name),
↪'|s|b|G:glyphicon-chevron-left') AS _link
                , ' <button class="btn disabled ',   IF({{semId:Y0}}=sc.id, 'btn-
↪success', 'btn-default'), '">',sc.name, '</button> '
                , CONCAT('p:{{pageAlias:T}}&semId=', sn.id, '|t:', QBAR(sn.name),
↪'|s|b|G:glyphicon-chevron-right|R') AS _link
          FROM Semester AS sc

          LEFT JOIN semester AS sp
            ON sp.id=sc.id-1

          LEFT JOIN semester AS sn
            ON sc.id+1=sn.id AND sn.show_semester_from<=CURDATE()

          WHERE sc.id={{semId:SUY}}
          ORDER BY sc.semester_von
  head = <div class="btn-group" style="position: absolute; top: 15px; right: 25px;">
  tail = </div><p></p>
}
```

CHAPTER 11

REST

Via REST it's possible to access the QFQ based application. Each REST API endpoint has to be defined as a QFQ Form.

This describes the server side (=QFQ is server). For client access check *REST Client*.

The QFQ REST api implements the four most used REST HTTP methods:

GET - Read Shows a list of database records or a single record. The QFQ form holds the definition which and what to show.

List: `curl -X GET "http://localhost/qfq/typo3conf/ext/qfq/Classes/Api/rest.php/person/`

Data (id=123): `curl -X GET "http://localhost/qfq/typo3conf/ext/qfq/Classes/Api/rest.php/person/123`

POST - Create new record The QFQ form defines wich columns will be written in which table. Most of QFQ Form functionality can be used. Example:

```
curl -X POST "http://localhost/qfq/typo3conf/ext/qfq/Classes/Api/rest.php/person/" -d '{"name":"Miller","firstname":"Joe"}'
```

PUT - Update a record Similar to POST, but a given record will be updated.

```
curl -X PUT "http://localhost/qfq/typo3conf/ext/qfq/Classes/Api/rest.php/person/123" -d '{"name":"Miller","firstname":"Joe"}'
```

DELETE - Delete a record Similar to a QFQ Delete form.

```
curl -X DELETE "http://localhost/qfq/typo3conf/ext/qfq/Classes/Api/rest.php/person/123"
```

All data will imported / exported in JSON notation.

Any QFQ form becomes a REST form via: `Form > Access > Permit REST: get / insert / update / delete`

If the REST endpoint specifies an unknown form or access is forbidden, an HTTP error is reported.

11.1 Endpoint

Tip: The basic REST API endpoint: `<domain>/typo3conf/ext/qfq/Classes/Api/rest.php`

`<domain>/typo3conf/ext/qfq/Classes/Api/rest.php/<level1>/<id1>/<level2>/<id2>/.../?<var1>=<value1>&...`

Append level names and ids after `.../rest.php/`, each separated by '/' .

E.g.:

1. List of all persons: `<domain>/typo3conf/ext/qfq/Classes/Api/rest.php/person`
2. Data of person 123: `<domain>/typo3conf/ext/qfq/Classes/Api/rest.php/person/123`
3. Addresses of person 123: `<domain>/typo3conf/ext/qfq/Classes/Api/rest.php/person/123/address`
4. Address details of address 45 from person 123: `<domain>/typo3conf/ext/qfq/Classes/Api/rest.php/person/123/address/45`

QFQ 'Forms' are used as a 'container' (to define all details).

Tip: The QFQ `form name` represents the level name.

Only the last <level> of an URI will be processed. The former ones are just to fulfil a good looking REST API.

Note: Each level name (=form name) is available via STORE_CLIENT and name `_formX`. E.g. in example (1) `{{_form1:C:alnumx}}=person` and `{{_form2:C:alnumx}}=address`.

Each level id is available via STORE_CLIENT and name *_idX*. E.g. in example (2) `{{_id1:C}}=123` and `{{_id2:C}}=45`.

Also the `id` after the last `level` in the URI path, 123 in example (2) and 45 in example (4), is copied to variable `r` in STORE_TYPO3, access it via `{{r:T}}`.

11.2 GET - Read

A REST (GET) form has two modes:

data Specific content to a given id. Defined via `form.parameter.restSqlData`. This mode is selected if there is an id>0 given.

list A list of records will be exported. Defined via `form.parameter.restSqlList`. This mode is selected if there is no id or id=0.

Note: There are *no* native-FormElements necessary or loaded. Action FormElements will be processed.

To simplify access to id parameter of the URI, a mapping is possible via 'form.parameter.restParam'. E.g. `restParam=pId,adrId` with example d) makes `{{pId:C}}=123` and `{{adrId:C}}=45`. The order of variable names corresponds to the position in the URI. `_id1` is always mapped to the first parameter name, `_id2` to the second one and so on.

GET Variables provided via URL are available via STORE_CLIENT as usual.

Form

Attribute	Description
name	<*level*> Mandatory. Level name (Endpoint) in URI.
table	Mandatory. Name of the primary table
Permit REST	*get* Mandatory. The form can be loaded in REST mode.

Form.parameter

Attribute	Description
restSqlData	Mandatory. SQL query selects content shown in data mode. | `restSqlData={{!SELECT id, name, gender FROM Person WHERE id='{{r:T0}}'' }}`
restSqlList	Mandatory. SQL query selects content shown in data mode. | `restSqlData={{!SELECT id, name FROM Person }}`
restParam	Optional. CSV list of variable names. E.g.: `restParam=pId,adrId`
restToken	Optional. User defined string or dynamic token (see :ref:`restAuthorization`).

Note: There are no *Special column names* available in `restSqlData` or `restSqlList`. Also there are no SIPs possible, cause REST typically does not offer sessions/cookies (which are necessary for SIPs).

Important: If there is an `id` given, a record in the named primary with the specified table has to exist. If not, an error is thrown.

11.3 POST - Insert

Form

Attribute	Description
name	<*level*> Mandatory. Level name (Endpoint) in URI.
table	Mandatory. Name of the primary table
Permit REST	*insert* Mandatory. The form can be loaded in REST mode.
id	Missing or '0'.

Form.parameter

Attribute	Description
restParam	Optional. CSV list of variable names. E.g.: `restParam=pId,adrId`
restToken	Optional. User defined string or dynamic token (see *Authorization*).
restSqlPostPut	Optional. Instead of returning the last_insert_id, a customized result might be specified. E.g. `{{! SELECT id, token FROM Token WHERE id={{id:R0}} }}`

FormElement:

- For each column to save one FormElement with `FE.name=<column>` is necessary.
- A regular QFQ form can be used as REST Post endpoint.

11.4 PUT - Update

Form

Attribute	Description
name	*<level>* Mandatory. Level name (Endpoint) in URI.
table	Mandatory. Name of the primary table
Permit REST	*update* Mandatory. The form can be loaded in REST mode.
id	>0

Form.parameter

Attribute	Description
restParam	Optional. CSV list of variable names. E.g.: `restParam=pId,adrId`
restToken	Optional. User defined string or dynamic token (see *Authorization*).

FormElement:

- For each column to save one FormElement with `FE.name=<column>` is necessary.
- A regular QFQ form can be used as REST Post endpoint

11.5 DELETE - Delete

Form

Attribute	Description
name	*<level>* Mandatory. Level name (Endpoint) in URI.
table	Mandatory. Name of the primary table
Permit REST	*delete* Mandatory. The form can be loaded in REST mode.
id	>0

Form.parameter

Attribute	Description
restParam	Optional. CSV list of variable names. E.g.: `restParam=pId,adrId`
restToken	Optional. User defined string or dynamic token (see *Authorization*).

Note: There are *no* native-FormElements necessary - but might exist for dependent records to delete. Action FormElements will be processed.

11.6 Authorization

A QFQ form is only acessible via REST API, if `Form.permitRest` enables one of the HTTP Methods: **get, post, put, delete**

`Permit New` or `Permit Edit` don't apply to QFQ forms called via REST.

Important: By default, the REST API is public accessible.

If this is not wished:

- HTTP AUTH might be used (configured via webserver)
- Any other webserver based access restriction method
- or the QFQ internal 'HTTP header token based authorization'.

11.6.1 Token based authorization

A form will require a 'token based authorization', as soon as there is a `form.parameter.restToken` defined. Therefore the HTTP Header 'Authorization' has to be set with `token=<secret token>`. The 'secret token' will be checked against the server.

Example:

```
form.parameter.restToken=myCrypticString0123456789

Test via commandline: curl -X GET -H 'Authorization: Token␣
↪token=myCrypticString0123456789' "http://localhost/qfq/typo3conf/ext/qfq/Classes/
↪Api/rest.php/person/123/address/"
```

The static setup with `form.parameter.restToken=myCrypticString0123456789` is fine, as long as only one token exist. In case of multiple tokens, replace the static string against a SQL query.

Tip: The HTML Header Authorization token is available in STORE_CLIENT via '`{{Authorization:C:alnumx}}`.

Best Practice: For example all created tokens are saved in a table 'Auth' with a column 'token'. Define:

```
form.parameter.restToken={{SELECT a.token FROM Auth AS a WHERE a.token='{
↪{Authorization:C:alnumx}}' }}
```

To restrict access to a subset of data, just save the limitations inside the Auth record and update the query to check it:

```
form.parameter.restToken={{SELECT a.token FROM Auth AS a WHERE a.token='{
↪{Authorization:C:alnumx}}'}}
form.parameter.restSqlList={{!SELECT p.id, p.name, p.email FROM Person AS p, Auth AS␣
↪a WHERE a.token='{{Authorization:C:alnumx}}' AND a.attribute=p.attribute}}
form.parameter.restSqlData={{!SELECT p.* FROM Person AS p, Auth AS a WHERE a.token='{
↪{Authorization:C:alnumx}}' AND a.attribute=p.attribute AND p.id='{{r:T0}}' }}
```

If authorization is denied, the request will be answered with a delay of 3 seconds (configured via securityFailedAuthDelay).

CHAPTER 12

System

12.1 FormEditor with usage

With a large number of forms, it's important to know how often a form has been used, on which pages it's used and when has the form be used. The following report includes the regular *FormEditor* as well:

```
#
# Form
#
# a) List of forms: {{form:S}}='', {{formIdHistory:S}}=''
# b) Edit Form: {{form:S}} - Open form {{form:S}} with record {{r:S}} - typically the␣
↪FormEditor
# c) Use history of a given form: {{formIdHistory:S}}
#
# {{form:S}}
# {{formIdHistory:S0}} - usage history of form '{{formIdHistory:S}}'

form={{form:SE}}

10 {
  # List of Forms: Do not show this list of forms if there is a form given by SIP.
  # Table header.
  sql = SELECT '<th data-sorter="false" class="filter-false">'
              , CONCAT('p:{{pageAlias:T}}&form=Form&') as _pagen
              , '</th><th>Name'
              , '</th><th>Title'
              , '</th><th>Table'
              , '</th><th>#'
              , '</th><th><em>First</em>'
              , '</th><th><em>Last</em>'
              , '</th><th><em>PageId</em></th>'
          FROM (SELECT '') AS fake
          WHERE {{formIdHistory:S0}}=0
  head = <table class="table table-hover qfq-table-50 tablesorter tablesorter-filter"␣
↪id="{{pageAlias:T}}-form">
```

(continues on next page)

(continued from previous page)

```
  rbeg = <thead class="qfq-sticky"><tr>
  rend = </tr></thead><tbody>
  tail = </tbody></table>

  20 {
    # All forms
    sql = SELECT CONCAT('p:{{pageAlias:T}}&form=Form&r=', f.id) as _pagee
                , CONCAT(f.name, ' <span class="text-muted">(', f.id, ')</span>')
                , QMORE(strip_tags(f.title),50)
                , f.tableName
                , CONCAT('p:{{pageAlias:T}}&formIdHistory=', f.id, '|s|b|t:<span␣
↪class="badge">', COUNT(fsl.id), '</span>'
                        , IF(COUNT(fsl.id)=0, '|r:3','')  ) as _link
                , CONCAT( '<em><span title="',MIN(fsl.created), '">', DATE_FORMAT(␣
↪MIN( fsl.created), '%d.%m.%Y'), '</span></em>')
                , CONCAT( '<em><span title="',MAX(fsl.created), '">', DATE_FORMAT(␣
↪MAX( fsl.created), '%d.%m.%Y'), '</span></em>')
                , CONCAT('<em>', GROUP_CONCAT(DISTINCT fsl.pageId ORDER BY fsl.
↪pageId), '<em>')
              FROM Form AS f
              LEFT JOIN FormSubmitLog AS fsl
                ON fsl.formId=f.id
              WHERE {{formIdHistory:S0}}=0
              GROUP BY f.id
              ORDER BY f.name
    rbeg = <tr>
    rend = </tr>
    fbeg = <td>
    fend = </td>
  }
}

30 {
  # History of a Form {{formIdHistory:S0}}
  sql = SELECT f.name
               , fsl.feUser
               , fsl.recordId
               , fsl.pageId
               , fsl.created

           FROM FormSubmitLog AS fsl, Form AS f
           WHERE fsl.formId={{formIdHistory:S0}}
             AND fsl.formId=f.id
           ORDER BY fsl.created DESC

  head = <table class="table table-hover qfq-table-50 tablesorter tablesorter-filter"␣
↪id="{{pageAlias:T}}-formHistory">
         <thead class="qfq-sticky"><tr><th>Form</th><th>feUser</th><th>recordId</th>
↪<th>pageId</th><th>Submit</th></tr></thead><tbody>
  tail = </tbody></table>
  rbeg = <tr>
  rend = </tr>
  fbeg = <td>
  fend = </td>
  altsql = SELECT IF('{{formIdHistory:S0}}'='0', '', '<div class="alert alert-warning
↪">History is empty</div>')
}
```

12.2 AutoCron

The *AutoCron* service fires periodically jobs like *open a webpage* (typically a QFQ page which does some database actions) or *send mail*.

- AutoCron will be triggered via system cron. Minimal time distance therefore is 1 minute. If this is not sufficient, any process who starts *. . . /typo3conf/ext/qfq/Classes/External/autocron.php* via */usr/bin/php* frequently might be used.
- Custom start time and frequency.
- Per job:
  - If a job still runs and receives the next trigger, the running job will be completed first.
  - If more than one trigger arrives during a run, only one trigger will be processed.
  - If the system misses a run, it will be played as soon as the system is online again.
  - If multiple runs are missed, only one run is fired as soon as the system is online again.
- Running and processed jobs can easily be monitored via *lastRun, lastStatus, nextRun, inProgress*.

12.2.1 Setup

- Setup a system cron entry, typically as the webserver user ('www-data' on debian).
- Necessary privileges:
  - Read for *. . . /typo3conf/ext/qfq/**
  - Write, if a logfile should be written (specified per cron job) in the custom specified directory.

Cron task (user cron tab):

```
* * * * * /usr/bin/php /var/www/html/typo3conf/ext/qfq/Classes/External/autocron.php
```

AutoCron Jobs of type 'website' needs the php.ini setting:

```
allow_url_fopen = On
```

Remember: if a cron job fails for whatever reason, the cron daemon will send a mail to the userid who started the cron job. E.g. www-data. Setup a email forward of such account to a frequently read email account.

12.2.2 Create / edit *AutoCron* jobs

Create a T3 page with a QFQ record (similar to the formeditor). Such page should be access restricted and is only needed to edit *AutoCron* jobs:

```
dbIndex={{indexQfq:Y}}
form={{form:S}}

10 {
    # Table header.
    sql = SELECT CONCAT('p:{{pageAlias:T}}&form=cron') AS _pagen, 'id', 'Next run',
↪'Frequency','Comment'
                , 'Last run','In progress', 'Status', 'Auto generated'
    head = <table class='table table-hover qfq-table-50'>
```

(continues on next page)

(continued from previous page)

```
    tail = </table>
    rbeg = <thead><tr>
    rend = </tr></thead>
    fbeg = <th>
    fend = </th>

    10 {
    # All Cron Jobs
    sql = SELECT CONCAT('<tr class="'
                        , IF(c.lastStatus LIKE 'Error%','danger','')
                        , IF(c.inProgress!=0 AND DATE_ADD(c.inProgress, INTERVAL 10␣
↪MINUTE)<NOW(),' warning','')
                        , IF(c.status='enable','',' text-muted'),'" '

                        , IF(c.inProgress!=0 AND DATE_ADD(c.inProgress, INTERVAL 10␣
↪MINUTE)<NOW(),'title="inProgress > 10mins"'
                        , IF(c.lastStatus LIKE 'Error%','title="Status: Error"',''))
                        , '>')
                      , '<td>', CONCAT('p:{{pageAlias:T}}&form=cron&r=', c.id) AS _
↪pagee, '</td><td>'
                      , c.id, '</td><td>'
                      , IF( c.nextrun=0,"", DATE_FORMAT(c.nextrun, "%d.%m.%y %H:%i:%s
↪")), '</td><td>'
                      , c.frequency, '</td><td>'
                      , c.comment, '</td><td>'
                      , IF(c.lastrun=0,"", DATE_FORMAT(c.lastrun,"%d.%m.%y %H:%i:%s")),
↪'</td><td>'
                      , IF(c.inProgress=0,"", DATE_FORMAT(c.inProgress,"%d.%m.%y %H:%i:
↪%s")), '</td><td>'
                      , LEFT(c.laststatus,40) AS '_+pre', '</td><td>'
                      , c.autoGenerated
                      , CONCAT('U:form=cron&r=', c.id) AS _paged, '</td></tr>'
             FROM Cron AS c
    ORDER BY c.id
    }
}
```

12.2.3 Usage

The system *cron* service will call the *QFQ AutoCron* every minute. *QFQ AutoCron* checks if there is a pending job, by looking for jobs with *nextRun <= NOW()*. All found jobs will be fired - depending on their type, such jobs will send mail(s) or open a *webpage*. A *webpage* will mostly be a local T3 page with at least one QFQ record on it. Such a QFQ record might do some manipulation on the database or any other task.

A job with *nextRun'=0 or 'inProgress*!=0 won't never be started.

Due to checking *inProgress*, jobs will never run in parallel, even if a job needs more than 1 minute (interval system cron).

12.2.3.1 Job: repeating

- frequency: '1 MINUTE', '2 DAY', '3 MONTH', ...

After finishing a job, *nextRun* will be increased by *frequency*. If *nextRun* still points in the past, it will be increased by *frequency* again, until it points to the future.

12.2.3.2 Job: asynchronous

- frequency: <empty>

An 'AutoCron' job becomes 'asynchronous' if *frequency* is empty. Then, auto repeating is switched off.

If *nextRun* is > 0 and in the past, the job will be fired. After the job has been done, *nextRun* will be set to 0.

This is useful for jobs which have to be fired from time to time.

To fire such an asynchronous job, just set *nextRun=NOW()* and wait for the next system cron run.

If such a job is running and a new *nextRun=NOW()* is applied, the 'AutoCron' job will be fired again during the next system cron run.

12.2.3.3 Type: Mail

Currently QFQ uses a special sendmail notation - this will change in the future.

- *Mail*:

```
{{!SELECT 'john@doe.com' AS sendMailTo, 'Custom subject' AS sendMailSubject,
↪'jane@doe.com' AS sendMailFrom, 123 AS sendMailGrId, 456 AS sendMailXId}}
```

AutoCron will send as many mails as records are selected by the SQL query in field *Mail*. Field *Mail body* provides the mail text.

All columns of the SQL are available in STORE_PARENT.

12.2.3.4 Type: Website

The page specified in *URL* will be opened.

Optional the output of that page can be logged to a file (take care to have write permissions on that file).

- *Log output to file*'='*output.log* - creates a file in the Typo3 host directory.
- *Log output to file*'=/var/log/output.log' - creates a file in */var/log/* directory.

Also *overwrite* or *append* can be selected for the output file. In case of *append* a file rotation should be setup on OS level.

To check for a successful DB connection, it's a good practice to report a custom token on the T3 page / QFQ record like 'DB Connect: ok'. Such a string can be checked via *Pattern to look for on output=/DB Connect: ok/*. The pattern needs to be written in PHP PCRE syntax. For a simple search string, just surround them with '/'. If the pattern is found on the page, the job get's 'Ok' - else 'Error - . . . '.

Access restriction

To protect AutoCron pages not to be triggered accidental or by unprivileged access, access to those page tree might be limited to localhost. Some example Typoscript:

```
# Access allowed for any logged in user or via 'localhost'
[usergroup = *] || [IP = 127.0.0.1]
  page.10 < styles.content.get
[else]
  # Error Message
  page.10 = TEXT
```

(continues on next page)

(continued from previous page)

```
  page.10.value = <h2>Access denied</h2>Please log in or access this page from an␣
↪authorized host. Your current IP address: 
  page.20 = TEXT
  page.20.data = getenv : REMOTE_ADDR
[global]
```

AutoCron / website: HTTPS protocol

- For *https* the PHP extension *php_openssl* has to be installed.
- All certificates are accepted, even self signed without a correct chain or hostnames, not listed in the certificate. This is useful if there is a general 'HTTP >> HTTPS' redirection configured and the website is accessed via *https://localhost/...*

CHAPTER 13

Application Test

With a framework like https://www.seleniumhq.org/ it's possible to play and verify unattended test cases.

To assist such frameworks and to make the tests reliable, an individual tag might be assigned to HTML elements which have to interact with the test framework.

13.1 Form

By default every FormElement contains an attribute 'data-reference=<value>', whereas the '<value>' is either the name of the FormElement or a custom value, defined via 'FormElement.parameter.dataReference=<value>'.

13.2 Report

Any HTML output can be extended by a tag - that's done by the webmaster. For QFQ generated links, an attribute like 'data-reference' might be injected via token 'A' (attribute).

```
SELECT 'p:personedit&form=person&r=1|b|s|A:data-reference="person-edit"|t:Edit person
↪' AS _link
```

CHAPTER 14

General Tips

14.1 Errors

- Does the error happens on every *page* or only on a specific one?
- Does the error happens on every *form* or only on a specific one?

Tips:

- On general errors:
  - Always check the Javascript console of your browser, see *Javascript problem*.
  - Always check the Webserver log files.

14.2 Procedure to Find an Irreproducible Error

- Find out the date and time on which the error occurred as precisely as possible
- Find out the user-name of the person who experienced the error
- Search the logs at the time of the Error:
  - qfq.log
    - location: fileadmin/protected/log/qfq.log
    - Look for error messages which were sent to the user at that time. (search for user-name in file)
    - Look at actions performed during the time of Error
  - FormSubmitLog
    - location: table named 'FormSubmitLog' in the qfq database
    - In the SQL Table FormSubmitLog search the column FEUser for the user-name

    - does the data (in the column formData) submitted at the time of the error coincide with the data saved in the database?

- sql.log
    - location: fileadmin/protected/log/sql.og
    - search for the form that was active when the error occurred:
        - e.g.: If the form is named 'requestGreview' search for 'form:requestGreview'

14.3 Caching

Content, generated by QFQ, is generally not cached. But the QFQ content records are cached by Typo3. This means if there is a content element 'Insert Records' on a page 'A' which includes a QFQ record from page 'B' and such QFQ record is modified (the SQL definition, not the delivered output), the new definition becomes by default only visible if the cached is cleared.

To simplify the situation, set on the page of the QFQ record (B) in Page TS Config:

```
TCEMAIN.clearCacheCmd = pages
```

14.4 QFQ specific

14.4.1 A variable {{<var>}} is empty

The sanitize rule is violated and therefore the value has been removed. Set {{<var>:<store>:all}} as a test.

14.4.2 Page is white: no HTML code at all

This should not happen.

The PHP process stopped at all. Check the Apache error logfile, look for a stacktrace to find the latest function. Send a bug report.

14.4.3 Problem with query or variables

Specify the required sanitize class. Remember: for STORE_FORM and STORE_CLIENT the default is sanitize class is *digit*. This means if the variable content is a string, this violates the sanitize class and the variable will not be replaced.

Tip on Form: put the problematic variable or SQL statement in the 'title' or note 'field' of a *FormElement*. This should show the content. For SQL statements, remove the outer token (e.g. only one curly brace) to avoid triggering SQL:

```
FE.title: Person { SELECT ... WHERE id={{buggyVar:alnumx}} }
```

Tip on Report: In case the query did not contain any double ticks, just wrap all but 'SELECT' in double ticks:

```
Buggy query:  10.sql = SELECT id, ... FROM MyTable WHERE status={{myVar}} ORDER BY␣
↪status
Debug query:  10.sql = SELECT "id, ... FROM MyTable WHERE status={{myVar}} ORDER BY␣
↪status"
```

14.4.4 Error read file config.qfq.php: syntax error on line xx

Check the given line number. If it's an SQL statement, enclose it in single or double ticks.

14.4.5 Output a text, substitute embedded QFQ variables

The content will be copied to '_text'. In *10.tail* than the '{{text:R}}' will be substituted with all known variables. Note the '-' in '{{text:RE::-}}', this will prevent that QFQ escapes any character from the content.

```
10 {
  sql = SELECT no.text AS _text
           FROM Note AS no
           WHERE id=...
  tail = {{text:RE::-}}
}
```

14.4.6 TypeAhead list with T3 page alias names - use of the T3 DB

To define a typeahead list of T3 page alias names:

```
FE.type = text
FE.parameter.typeAheadSql = SELECT p.alias FROM {{dbNameT3:Y}}.Pages AS p WHERE p.
↪deleted=0 AND p.alias!='' AND p.alias LIKE ? ORDER BY p.alias LIMIT 20
FE.parameter.typeAheadMinLength = 1
```

14.4.7 Set FE-User password

To offer an FE User the possibility to change the own T3 FE password, create a form:

```
f.name = changeFePassword
f.title = Change Password
f.table = Person   (QFQ Table, not T3)
f.permitNew = never
f.permitEdit = sip

fe[1].name = myPassword
fe[1].title = Password
fe[1].class = native
fe[1].type = password
fe[1].mode = required
fe[1].parameter = retype

fe[2].class = action
fe[2].type = afterSave
fe[2].parameter = sqlAfter={{UPDATE {{dbNameT3:Y}}.fe_users SET password='{
↪{myPassword:FE:all:p}}' WHERE username='{{feUser:T}}' AND deleted=0
```

Call the form via SIP on an existing record. Often QFQ has an own table for persons and also the current user exist in T3 fe_users table.

14.5 Logging

14.5.1 General webserver error log

For apache: */var/log/apache2/error_log*

Especially if you got a blank page (no rendering at all), this is typically an uncaught PHP error. Check the error message and report the bug (https://qfq.io > Contact).

14.5.1.1 Call to undefined function qfq\mb_internal_encoding()

Check that all required php modules are installed. See *Preparation*.

14.6 Error Messages

14.6.1 Internal Server Error

The browser shows a red popup with 'Internal Server Error'. The message is generated in the browser. Happens e.g. an AJAX request response of QFQ (=Server) is broken. This might happen e.g. if PHP can't start successfully or PHP fails to run due to a missing php module or broken configuration.

14.6.2 Oops, an error occurred! Code: 20180612205917761fc593

You see this message on all places where a QFQ content record should produce some output. Typically the extension fails to load. If the error message disappears when the QFQ extension is disabled (instead a message *qfq_qfq can't be rendered* is shown), than QFQ is the problem.

Search the given code in *typo3temp/logs/**, in this example 20180612205917761fc593. You'll should find a stacktrace with a more detailed message.

The error might occur if there are problematic characters in config.qfq.php, like single or double ticks inside strings, wich are not enclosed (correctly).

14.6.3 sendEmail: Error => TLS setup failed

Switch off the TLS encryption. In *Configuration* specify for *config.sendEMailOptions*:

```
-o tls=no
```

14.7 Javascript problem

Open the 'Webdeveloper Tools' (FF: F12, Chrome/Opera: Right mouse click > Inspect Element) in your browser, switch to 'console' and reload the page. Inspect the messages.

14.8 TinyMCE

14.8.1 Glyph Icons in ''

TinyMCE forbids by default HTML tag 'span' with 'class' attribute. E.g.:

```
<span class="glyphicon glyphicon-user"></span>
```

To allow it, add 'span' to the valid elements in the FormElement.parameter field:

```
editor-extended_valid_elements = span[class|style]
```

The HTML span tag has to be added via 'source' view. At least in TinyMCE 4.7.13, the glyph is still not shown in the editor.

14.9 FE User

The FE User record (table: fe_users)

- Has to exist.
- Has to be assigned to at least one FE Group. Check `fe_users.usergroup`.
- Has to be assigned to a T3 page `fe_users.pid`.
- The T3 page has to be configured as `record store` on the T3 Plugin login box.
- Access time has to be zero or a currently valid period.

CHAPTER 15

Coding Guideline

The following is not mandatory but shows some best practices:

15.1 Constants

- Define constants in `Extensions > QFQ > Custom > ...`
- Dynamic values under `Extensions > QFQ > Dynamic > ...`

15.2 QFQ content record

- Name the record in the header field with:
  - Regular content: `[QFQ] ...`
  - Content in the left column: `[QFQ,L] ...`
  - Content in englisch: `[QFQ,E] ...`
- The first lines should be comments, explaining what the record does and list all passed variables:

```
#
# Shows list of Persons living in {{country:SE}}
#
# {{country:SE}}
#
```

- A good practice is to define all possible STORE_SIP Parameter in a SQL at the beginning and copy them to STORE_RECORD:

  10 { # Normalize variables sql = SELECT '{{country:SE}}' AS _country

  # List selected persons 20.sql = SELECT p.name FROM Person AS p WHERE p.country LIKE '{{country:R}}'

}

- Always comment the queries like shown above.

CHAPTER 16

Release

16.1 Version 20.x.x

Date: <date>

16.1.1 Notes

16.1.2 Features

16.1.3 Bug Fixes

16.2 Version 20.9.0

Date: 06.09.2020

16.2.1 Notes

New table 'uniq' to guarantee uniq random strings. New SIP protected AJAX calls. Report now fires calls to websocket (remote hosts). Report now fires REST calls to remote hosts.

16.2.2 Features

- #10979 / Report: do SIP protected AJAX calls to typo3conf/ext/qfq/Classes/Api/dataReport.php.
- #11076 / Report: trigger call to websockets.
- #11119 / Report: REST Client calls - incl. processing of answer.
- Add CodingGuideline.rst.

16.2.3 Bug Fixes

- #10919 / AutoCron: Fix missing FillStoreSystemBySql.
- #11075 / Form: Missing SQL error message in FE Action Elements.
- #11039 / Fix CSS for Checkbox.

16.3 Version 20.6.2

Date: 25.06.2020

16.3.1 Bug Fixes

- #10641 / TypeAheadTag: Fehler beim gleichzeitigen anlegen mehrerer neuer Tags
- #10794 / Documentation: Crontab entry more clearly

16.4 Version 20.6.1

Date: 24.06.2020

16.4.1 Features

- #10778 / Upload ZIP and unpack

16.5 Version 20.6.0

Date: 14.06.2020

16.5.1 Notes

- Add note in Installation.rst to start Apache with Locale en_US.UTF-8. This helps to support Umlaut and other characters in filenames and wkhtml commandline options (like header/footer).
- Migrate documentation from T3 to ReadTheDocs.io - looks older but 'search' is much more better. New: chapters separated in individual files.
- For the image to PDF feature, installation of *img2pdf* is required (please check **'preparation'_**).

16.5.2 Features

- #10751 / Allow images to be concatenated for PDF download.
- Fontawesome updated 5.13.
- Extend FE.label size to 1023.
- Local documentation rendering directly via Sphinx.

- Manual: Search is working, table width not truncated anymore, PDF & epub export, redirect qfq.io/doc to docs.qfq.io.
- Update copyright notice.

16.5.3 Bug Fixes

- #10507 / FormElement.type: 'annotate' is defined two times in Enum
- #10705 / New function 'HelperFile::joinPathFilename($pre, $post)'. Joins only if $post is without leading slash.

16.6 Version 20.4.1

Date: 30.04.2020

16.6.1 Notes

- Developer: update local npm/font awesome packages via *make bootstrap*.

16.6.2 Features

- #10433 / Update to Font Awesome 5.0
- #10379 / Stored Procedure: SLUGIFY()
- Manual.rst: Example Report 'render'. Update several places to fit latest ReST layout rules.

16.7 Version 20.4.0

Date: 05.04.2020

16.7.1 Notes

- New Feature: typeAheadTag - extend regular input with multiple values via typeAhead. _
- MySQL StoredProcedure:
  - strip_tags() - Simple strip html tags.
  - QCC() - Escape colon / coma. Useful for QFQ link arguments like 'text' or 'tooltip'.

16.7.2 Features

- #9686 / Download: sanitize output filename.
- #10358 / Configure path/environment via QFQ config: qpdf, gs, pdfunite.
- #9517, #10145, #10177, #10117 / typeAheadTag.
- #10152 / QCC() - Stored Procedure to escape colon / coma.
- FabricJS: replaced glyphicons with font awesome.

- Rename config.qfq.example.php config-example.qfq.php.

16.7.3 Bug Fixes

- #6798 / Close didn't worked with r=0.
- #10199 / Form.forwardMode: missing mode 'Close' / 'Auto'.
- #10173 / Dynamic Update: Readonly element can't be activated via dynamic update.
- Fix broken default value for Form.forwardMode.
- Fix problem with reporting broken TG-FormElements.
- Add error message if primary table does not exist.

16.8 Version 20.2.0

Date: 02.02.2020

16.8.1 Notes

- Add new keyword 'render' in tt-content Report and QFQ config. 'render' will control if a) only Form OR Report will be rendered (render=single) or b) as previous Form AND Report together (render=both).

  Advantage: with 'render=single' no more SELECT ... FROM (SELECT '') AS fake WHERE '{{form:SE}}'=''.

  Attention: NEW default behaviour in new QFQ installations - render=single. Behaviour in old installations is unchanged.
- tt-content records with 'render = api' can stay on the same page as the link to get the content (e.g. Excel Export).
- Change default doc page to qfq.io/doc.
- Add new specialColumnName: '_noWrap' - skips wrapping of fbeg,fsep,fend.
- First version of 'FullCalendar.io' - new SpecialColumnName will follow in the future.

16.8.2 Features

- #9929 / New keyword '_noWrap' for column names (alias) - skips wrapping of fbeg/fskip/fend.
- #9905 / Keyword 'render' in Report. Final implementation. Doc updated.
- #9959 / Update QFQ Config on the fly.
- #9990 / Describe order of FormElement processing - https://qfq.io/doc/#form-process-order.
- #8658 / FullCalendar.io V3 implemented.
- #9535 / VerticalText new implementation.
- Manual.rst: Add list of icons. Enhance sendmail doc.
- Change color of qfq-info-* from blue to light-blue. Add qfq-primary, qfq-danger.

16.8.3 Bug Fixes

- #10010 / FE.type=sendmail will now be fired together with fe.type=after* (not after).
- #5869 / Table names not properly escaped.
- #9638 / TextArea: Autosize - broken when using clipboard,
- Fixed problem with border showing when qfq-color-white is set.
- Fix selenium tests, remove chromedriver from npm.
- Log problem that crashes qfq when calendar dependencies are missing.
- Fixed gruntfile problem.

16.9 Version 20.1.1

Date: 13.01.2020

16.9.1 Bug Fixes

- #7705 / Fix problem with wrong value after save and form update.
- #8587 / A form triggers a save only, if there are real table columns.

16.10 Version 20.1.0

Date: 09.01.2020

16.10.1 Notes

- Deprecated: Form.parameter.mode. Use Form.parameter.formModeGlobal

16.10.2 Features

- #9805 / Form.parameter.activateFirstRequiredTab.
- #9858 / Form.parameter: replace 'mode' by 'formModeGlobal'
- #9860 / SQL function qmore(): change text '...' to '[...]'.
- Update Developer doc for record locking.
- Mockup for error handling.

16.10.3 Bug Fixes

- #7925 / Error in split PDF file during upload. Fix the cwd error in Logger.
- #9789 / Record lock release to early on 'leave page'. QfqJS: Moved release lock to before unload.
- #9861 / Fix problem with broken sql.log filename.

- #8851 / Revert implementation: LogMode 'modify' vs. 'modifyAll'.
- #9859 / Database Update: check for 'Update specialColumnName needed' breaks new QFQ install.
- #9813 / During QFQ database update, skip errors like 'Error 1060 - Duplicate Column'.
- Manual.rst: Fix various broken table layouts.

16.11 Version 19.12.0

Date: 17.12.2019

16.11.1 Notes

- Switch the whole homepage to readonly: FormModeGlobal and STORE_USER

16.11.2 Features

- TinyMCE: grey out controls when readonly.
- Mockup:
  - Update files to get CSS & JS files from their own directory structure (not an installed QFQ extension).
  - Add fontawesome, tablesorter to mockup 'formCheckbox.html'.
- CI: Download selenium logs (only failed) under artifacts.
- Dev: .gitignore: exclude some docker & selenium.
- Merge Selenium Python Checks into Master.
- #9686 / html decode and sanitize an export filename to become the 'save as'-filename.
- #9666 / min-width for extraButtonInfo.
- FormEditor: optimize minWidth for 'rowLabelInputNote' field.

16.11.3 Bug Fixes

- #9720 / Checkbox dynamic update varoious settings:
  - Multi Plain Vertical & Horizontal, Checkbox Multi BS.
  - Fixed that label of 'checkbox' are bold and label of 'checkbox-inline' are normal.
- #7974 / TinyMCE: ReadOnly.
- #9424 / modeSql: skip if it starts with '#'.
- #9674 / Select Required Dynamic Update.
- #9678 / textarea now trigger DynamicUpdate.
- #9679 / FormModeGlobal: add STORE_USER - system wide readonly.
- #9690 / Select Required.
- #9691 / Checkbox: dynamic update > readonly. HTML ID for checkbox elements. Dynamic update switch 'readonly' for 'checkbox plain multi' and 'radio plain multi'.

- #9692 / Keyboard Select Checkbox.
- #9720 / Checkbox: Various setups with dynamic update.
- #9733 / Identify different tabs. Record lock for same tab will always be granted.
- #9734 / Fix 'dirty lock release' - leaving a dirty form without closing, leaves a stale lock record. Added a releaselock() before window.unload. Dirty remove on goBack.
- #9735 / File Delete: no dirty trigger.
- Download / PDF merge: skip leading errors, interpret only 'Could not merge encrypted files'.
- DragAndDrop broken: after refactoring Support.php, the dragAndDrop was broken - missed init of '$store'.

16.12 Version 19.11.3

Date: 29.11.2019

16.12.1 Notes

16.12.2 Features

- #8886 / Check pattern: after focus lost.
- #9655 / Checkboxes and radios now defined with a min-width in horizontal plain mode.
- #9617 / formModeGlobal:
  - Two modes of 'formModeGlobal' available: 'requiredOff' and 'requiredOffButMark'.
  - 'requiredOffButMark':
    - Renamed temporary 'skipRequiredCheck' to 'requiredOffButMark'.
    - Keep required marks after save.
    - Stop hiding helpblocks per default, set with class qfq-only-active-error.
- Radio: new class 'qfq-disabled' if readonly is set. Softer blue. Mark disabled - changed hover. Text in darker orange. Simple-error renamed to qfq-notify - removed box around error.
- New default class for 'form' tag: 'qfq-notify'.
- Manual.rst: Add info for 'letter-no-break'.
- Add validator.js to list of used packages.

16.12.3 Bug Fixes

- #9670 / If *qpdf* fails to decrypt a PDF, try *gs*.
- #3995 / Implemented partly: CheckBox and Radio can now be locked.
- #8091 / Checkbox required:
  - If radio or checkbox is required and empty on submit, form save brings the element to front.
  - Fix radio plain vertical.
  - Fix label2 not to be bold.

  - Checkbox Plain Vertical: forces 'font-weight: 400;'.
  - Updated colors for checkboxes/radios.
- #9638 / Textarea Sizing: Now also listens for paste.
- #7891 / Added missing 'type="button"' to button element.
- Remove ' style="font-size: 1em;"' in extraButton - this causes extra space between multiple extraButton inline in one input element.

16.13 Version 19.11.2

Date: 25.11.2019

16.13.1 Notes

- Enhance formModeGlobal=requiredOff/-ButMark (temporarily skipRequiredCheck): fill's '{{allRequiredGiven:V}}' before save to 1 (all given) else 0. Offers user to save form, even if not all required data are given and offers application logic to check easily if all required fields has been filled.

16.13.2 Features

- #9526 / Mark required fields more visible.
- #9617 / Improve 'formModeGlobal=requiredOff'.
  - Feature *Form.formModeGlobal* implemented - STORE_SIP overwrites form definition.
  - New STORE_VAR variable 'allRequiredGiven'. Becomes '1' if all required fields are given, else 0.
- Add param 'L' & 'type' automatically to form save.
- Manual.rst: Procedure to find an irreproducible error.
- Change definition of QFQ system tables for 'modified' and 'created'. Use DATETIME instead of TIMESTAMP.

16.13.3 Bug Fixes

- #7639 / subrecord drag n drop:
  - *orderInterval* has not been respected.
  - Update Manual.rst.
  - Fake STORE_SIP so it can be used during processing sql1.
  - The record, currently loaded into form, is available via STORE_RECORD.
  - Check for id/_id and ord/_ord.
  - Throw meaningful exception if missing 'id' or 'ord'.
- Fixes bug that no mime_type_content is called if there is on file.
- Fix broken regex101 url.

16.14 Version 19.11.1

Date: 11.11.2019

16.14.1 Bug Fixes

- #9532 / 'Advanced Upload' broken - slaveId/sqlUpdate/... have been processed two times, after multiform code changes.

16.15 Version 19.11.0

Date: 08.11.2019

16.15.1 Notes

New CSS Class:

- 'qfq-sticky' - to make an element sticky.

Update/new stored procedures:

- QMORE: change symbols from '..more' / '..less' to '...' / '<<'.
- QIFEMPTY: add empty detection for a) date, b) datetime, c) '0'.
- QDATE_FORMAT(timestamp). Return text in 'dd.mm.yyyy hh:mm' format or '-' if timestamp is empty.

16.15.2 Features

- #9521 / Textarea auto-grow. Update Manual.rst. New class qfq-auto-grow for textareas.
- Update FormEditor to 80,1,350.
- Alerts a) changed overlay to blur and white, b) optimized to better reading for developer - ':hover' inside of error messages: black text and blue highlight.
- Add new CSS class 'qfq-sticky-element' - to stick individual elements.

16.15.3 Bug Fixes

- #9354 / Missing border around form if there are no pills / no title
- #9053 / Using the tablesorter filter in form triggers dirty detection for save. Set qfq-skip-dirty on all tablesorter-filter.
- #2720 / Formelement 'radio' - required failed.
- #9512 / PDF merge fails on encrypted PDFs. Try to use qpdf to decrypt.
- #6232 / Missing required: Pill/Input not 'bring to front' - Textarea, Editor, Radio.
- #9300 / Fix Name restMethod-FormElement.
- Add '@' to various PHP/IO function to suppress generic OS exception - now QFQ will report customized messages.

- FormEditor / error message: fix missing formId in STORE_SIP on calling FE directly from error message.
- Manual.rst: Tip for wkhtml added. Add description for 'fileButtonText'. Adjustments for 'slaveId' concept.

16.16 Version 19.10.0

Date: 17.10.2019

16.16.1 Notes

- Background color of popups & error messages changed.
- slaveId/sqlInsert/sqlUpdate/sqlDelete are now available for all FormElements.
- New MySQL stored procedure strip_tags() function.

16.16.2 Features

- #5695 / First implementation of multiform.
- #7495 / Removed dirty flag when 'enable-save-button' is set.
- Add MySQL strip_tags() function.

16.16.3 Bug Fixes

- #9329 / Fabric annotations. Fixed Scaling Problem on Static Canvas instances.
- #9298 / Fix timeout of file_get_contents. Extend timeout for downloadPage to 10min.
- #9274 / PHP 7.3 reports: switch statement with 'continue 2'.
- #9269 / fillStoreForm now fired two times in API calls. The first time during loadFormDefinition and the second time after the STORE_TYPO3 has been faked via SIP.
- Manual.rst: Add '!' to fillStoreVar Query
- Increase z-index in CSS 'dropdown-menu' class to 1060. The 'tablesorter > columnselector' will be hidden with 'qfq-sticky' by values < 1000.
- Reduce z-index in CSS 'qfq-sticky' class from 9999 to 1000. The 'tablesorter > columnselector' will be hidden with values > 1060.

16.17 Version 19.9.1

Date: 21.09.2019

16.17.1 Notes

- Use the CSS class 'qfq-sticky' in <*thead*> to make a table header 'sticky' - on long pages such tables headers are always visible.

- The page with the list of all forms: a new best practice report includes useful statistics *FormEditor with usage* - best is to replace the old code.

16.17.2 Features

- #9203 / Pin the header of table (CCS 'sticky'), to make it always visible even if the page scrolls down.
- #9172 / AutoCron: new colum 'autoGenerated', 'xId'
- #9089 / Move Stored Procedure to SECURITY=INVOKER
- Manual.rst: Best practice *FormEditor with usage*
- Reduce BS legend.font-size from 21 to 17.
- Change doc of tablesorter to use 'sorter-false' as class.

16.17.3 Bug Fixes

- #9074 / QFQ query with nested QFQ query failed, if the outer QFQ query is a multi column query (='{{!'). Fixed.

16.18 Version 19.9.0

Date: 09.09.2019

16.18.1 Notes

- Size of input elements now might be specified dynamically (with min and max height).
- Twig converts json objects to an array.

16.18.2 Features

- Report.php: Twig, convert json object into associative array.
- Test SQL stored procedure in Form report.
- qfq-bs.css.less: a) reduce 'qfq-note' padding-top from 7 to 2 px, b) reduce 'legend' margin-bottom from 25 to 0.
- Debug output sendmail redirect all: addresses now always space delimited.
- FormEditor:
  - Remove FormElement 'tabindex'.
  - Change FormElement.class from 'Select' to 'Radio'.
- Manual.rst:
  - Clean syntax highlight.
  - Update realtime log file QFQ code.
  - Add 'Best Practice' code to show QFQ log files in realtime. Reformat some content.

- #7849 / New option 'fileTrash' and 'fileTrashText'.
- #7682 / 'Input textarea auto height'
- #4434 / Special column names now have to start with underscore. Earlier it was recognised even if there was no underscore.

16.18.3 Bug Fixes

- #7860 / Special column name 'mailto' no handles text correctly if multi byte encoded. Fixed.
- #7849 / Missed filename after the file chooser selected an file. Fixed.
- #1201 / FE.parameter option 'tabindex' has never been implemented. Removed. According https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/tabindex only the values '0' and '-1' are good.
- Add table html-id to subrecords, created by QFQ. Needed for tablesorter.
- Add TinyMCE to list of 'Software distributed together with QFQ'

16.19 Version 19.8.0

Date: 28.08.2019

16.19.1 Notes

16.19.2 Features

- Manual.rst:
  - Update tablesorter *class="sorter-false"*.
  - Add note to include font-awesome.
  - QFQ_LOG documentation.
  - Enhance wkhtml debug info.
  - Add link to https://mariadb.com/kb/en/library/stored-routine-privileges/
- Increase Length of column Form.forwardPage limit from 255 to 511.
- Index.rst: Add Rafi, Elias, Nicola as further contributors.
- Add SQL stored procedure QIFEMPTY().
- #8806 / Add SQL stored procedure QNL2BR().
- #8851 / Add sqlLogMode modifyAll and restrict sqlLogMode modify.
- #8859 / FormEditor: Filter + ColumnSelector.
- #8856 / AutoCron: SQL logMode should be defined separately.

16.19.3 Bug Fixes

- #8933 / Broken record lock: lock extend does not work - fixed
- #8846 / FormEditor: subrecord of FormElement might be wider than Form.
- #8853 / fe.class=action: container should be hidden - hidden is wrong - FE-action might be assigned to templateGrou Container. New solution: if FE.class='action' only templateGroups are in the 'container' list.
- #6656 / DragnDrop' into 'master'
- Update compatibility: API change in T3 makes QFQ incompatible with Typo3 >=9.4. Change compatibility back to <9.3. Problem: sys_language_uid
- Update QFQ from earlier/equal version than 0.19.2 fails to create table 'Cron'.

16.20 Version 19.7.1

Date: 17.07.2019

16.20.1 Notes

- TWIG integration

16.20.2 Features

- TWIG:
  - Add Twig Extension to parse QFQ Links
  - Pass assoc-array of query result as context
  - allow to pass template as string
- Database.php: extend function sql error message
- #8179 / extraButtonLock and extraButtonPassword might be specified without a value or with 0 or with 1.

16.20.3 Bug Fixes

- Database.php: fix CodeException
- Fabric: Readonly now displays annotations again.

16.21 Version 19.7.0

Date: 02.07.2019

16.21.1 Notes

- Settings for tablesorter can now be saved.
- Selenium docker based is integrated now.

16.21.2 Features

- #7284 / tablesorter save sort order
- #8592 / Fixed button display, added Alert instead of disabled button.
- #8204 / position required mark
- Excel.php: Delete superficial autoloader require
- Cleanup: hashPassword.php moved from Api to External.
- Cleanup unwanted git tracked files: Documentation-develop/jsdocl. . .
- Selenium + Docker
- Update .gitignore - documentation directory moved several days ago.

16.21.3 Bug Fixes

- #6917 / Fix Monitor does not work.
- #8098 / Fix Language specific retype label or note has not been set correctly.
- Fix undefined index and potential undefined index.
- Fix Autoloader for External/AutoCron.php.
- Small fixes Manual.rst.
- Fix broken Manual: unwanted '<' after '{{!' lead to fully broken Manual.
- Fix path Source/api to Source/Api.

16.22 Version 19.6.2

Date: 21.06.2019

16.22.1 Notes

The Rest API path changed:

- old: typo3conf/ext/qfq/Source/api/rest.php
- new: typo3conf/ext/qfq/Classes/Api/rest.php

QFQ is now PSR4 compliant.

16.22.2 Features

- #8520 / Switch to psr4 composer.
- #8272 / Disable 'save' button as long as an upload is running.
- #7529 / Guarantee Sip Action plays only once.
- #8577 / Documentation enhanced for behaviour of SELECT '_test'.

16.22.3 Bug Fixes

- #7903 / Delete with 'table' as parameter doesn't work anymore.
- #8571 / Primary Key: report error if lower/uppercase don't match - in general if given primary key is not found in table definition.
- #8465 / Var substitute: '{{name:R:::0}}' default not used if it is '0'.

16.23 Version 19.6.1

Date: 16.06.2019

16.23.1 Notes

- 'pdftk' is no longer used.
- Please install 'poppler-utils' - the command 'pdfunite' is the replacement of 'pdftk'.

16.23.2 Features

- #8558 / Split error message in toUser and toDeveloper. Add error code.
- #8562 / Replace pdftk by pdfunite. 'pdftk' is outdated. It's hard to install on Ubuntu 18. It fails for recent PDFs. 'pdfunite' is based on poppler. URLs are preserved. Orientation and size are preserved.

16.23.3 Bug Fixes

- Fix uniqIdQfq() - returned always badcaffeee1234
- Refactor function.sql to function.sql. Implement constants for Version numbers.
- Update NewDoc.md

16.24 Version 19.6.0

Date: 12.06.2019

16.24.1 Notes

- Add Marc Egger as Developer

16.24.2 Bug Fixes

- #8523 / htmlid: remove spaces (war #8460). Replace uniqid() by uniqIdQfq() which delivers a constant id in case of running unit tests
- #8430 / QFQ Doc mising.
  - Index.rst: customize URL for 'report a problem'.

  - Sitemap.rst: new created to be included by Index.rst.
  - Reformat Index.rst according https://github.com/TYPO3-Documentation/TYPO3CMS-Example-ExtensionManual.git.
  - Move doc to docker generation. Update ReST Syntax URLs.
  - Remove 'extension/Documentation.outdated'.
  - Update .gitignore to not commit rendered doc.
- Manual.rst: Excel Export / PDF export replace 'uid:<int>' with 'uid:<tt-content record id>'
- Update SQL functions to hanlde errors more efficient.

16.25 Version 19.5.1

Date: 22.5.19

16.25.1 Notes

- New dropdown menu, fully dynamic via '... AS _link' incl. SIP generation.
- New SQL stored procedure for use directly in SQL queries:
  - QMORE(text, length) - limits a text to 'length' characters, click on 'more' shows complete text.
  - QBAR(text) - escapes the character '|'. That one is heavily used in format strings for '... AS _link'.

16.25.2 Features

- #8391 / Additional line 'Website: ...' in sendmail redirect all.
- #8270 / SQL Stored procedure QMORE(text,length): Limit long text to 'length' characters. Show button 'more'.
- #8270 / SQL Stored procedure QBAR(text): Escape the character '|'.
- #8348 / Dynamic Dropdown menu via '... AS _link'

16.25.3 Bug Fixes

- #8116 / Error-Dialog - Button for FE element is broken. First problem: replacing 'n' with '
' failed for FE link. Second problem: FormEditor fixed in case missing {{formId:S}}.
- #8315 / FE Datetime: 'd.m.yy hh:mm' thows an error
- #8278: Too long GET var crashes QFQ - Bug 1 - check GET Vars but try to report problem from POST var. Bug 2: writing log message crashed due to not initialized QFQ-config.
- Html2Pdf.php: Change logfile from sql.log to qfq.log
- formEditor.sql: Add specific pattern message for Form.name allowed characters
- Config.php: more precise error message

16.26 Version 19.5.0

Date: 03.05.2019

16.26.1 Notes

- QFQ is now T3 V9.5 compatible.
- Minimal required version increased to T3 V7.
- QFQ description in ext_emconf.php updated > will be displayed on TER.

16.26.2 Features

- #5103 / Upload any file type: `accept=*` or `accept=*/*` or `accept=*.*`.
- Manual.rst: update all links to bootstrap to fixed version 3.4: https://getbootstrap.com/docs/3.4/...

16.26.3 Bug Fixes

- Manual.rst: Update BS glyphicon URL
- Manual.rst: replace '{{feUser:Y}}' by '{{feUser:T}}'
- #8109 / Change email pattern and disable sanitize for FORM_UPDATE
- #8149 / Excel Export cutoff at Column AA
- #8113 / Fehler in Dokumentation für Default value

16.27 Version 19.3.2

Date: 18.03.2019

16.27.1 Notes

- New Escape/Action Class:
  - 'X': Throw an exception if a variable isn't found. Custom message can be defined.
  - 'S': Stop replacing nested variables.
- Form forward:
  - 'url-sip': Call a target page with SIP encoded parameter
  - 'url-sip-skip-history': Call a target page with SIP encoded parameter, do not add current page to browser history.
- Report: new token 'fskipwrap' - comma separated list of column indexes. Suche columns won't be rendered with 'fbeg, fend'.
- Upload max file size:
  - Default changed:

        * old: 10MB
        * new: System limit
    - Definition via a) system configuration, b) qfq setup, c) per Form, d) per FormElement

16.27.2 Features

- #8043 / New Escape/Action Class "Exception" if variable is not found.
- #8012 / New Escape/Action Class "Stop replace" of nested variables.
- #8067 / url-sip / url-sip-skip-history - Form forwarding can now be used with SIP encoded parameter.
- #8072 / New token for Report: fskipwrap - wrapping of fields can be disabled per column.
- #8041 / Upload: maxFileSize based on system maximum, qfq config, form config, formElement config
- Manual.rst / Add more detailed description for '... AS _exec'. Add notes for 'fileSplit'.

16.27.3 Bug Fixes

- #8035 / Missing sql.log: throws an error - fixed.
- #8077 / PDF/fileSplit: uploaded PDF file with missing extension '.pdf' have not been recognized as PDF files - fixed.
- #8076 / splitPdf: using 'fileSplit=jpeg' and uploading a PDF with only one page, results in filename 'split.jpg' (missing index '-0'). New: split.jpg is renamed internally to split-0.jpg to provide a unified naming scheme.
- #8075 / Catch exception on filesize() - fixed: now return '-' for non existing files.

16.28 Version 19.3.1

Date: 15.03.2019

16.28.1 Bug Fixes

- 8058 / Form > fillStoreVar: broken for TemplateGroup - Form.fillStoreVar not available during fillStoreForm().
- 8048 / A retype FE should not be checked for 'required' during save.

16.29 Version 19.3.0

Date: 05.03.2019

16.29.1 Notes

- New FormElement 'datalist' for fe.type='select'.
- FormElement 'annotate':
    - Now supports 'ReadOnly'.

  - 'grafic': Undo/Redo

16.29.2 Features

- #7729 / Select as datalist
- #7783 / FormElement 'annotate' - ReadOnly mode is now supported for grafic/text.
- FormElement 'annotate/grafic' History for undo / redo
- FormEditor: by creating a new FormElement, the feIdContainer is now preselected based on the last choice.
- Manual.rst: started to use more predefined Sphinx formatting styles.

16.29.3 Bug Fixes

- #7978 / mail.log: not written in some cases - fixed
- #7949 / Table MailLog: missing column 'cc,bcc'
- Manual.rst: correct parameter 'mode' for special column name '_sendMail'

16.30 Version 19.2.3

Date: 22.02.2019

16.30.1 Notes

New: QFQ REST interface implemented.

16.30.2 Features

- Rest Implementation with GET,PUT,POST,DELETE and authorization token
- REST.md: add
- Manual.rst: reformat all sql code to use tyoposcript - it seems mysql does not work
- Manual.rst: Add some admonitions

16.30.3 Bug Fixes

- #7925 / Change CWD during split reduced to splitting only.
- #7925 / Fixed problem in mkDirParent() with absolute paths.
- #7925 / Make logger independent of CWD.
- #7925 / Fixed problem with `mktemp --tmpdir` (difference Ubuntu 16 / 18) by using PHP function again.
- #7925 / Fixed some 'undefined index' problems.

16.31 Version 19.2.2

Date: 19.02.2019

16.31.1 Notes

- QFQ now offers a basic REST API. Check https://docs.typo3.org/p/IMATHUZH/qfq/master/en-us/Manual.html#rest

16.31.2 Features

- 7910 / Check for double form names
- 7904 / REST api export. Manual.rst: describe QFQ REST API
- Latest phpStorm IDE complains about missing ext-json in composer.json. Added.
- Manual.rst: Example how to use 'password' escape class.

16.31.3 Bug Fixes

16.32 Version 19.2.1

Date: 18.02.2019

16.32.1 Notes

- New default session timeout: if nothing special is needed, leave the config.sessionTimeout empty. If there is an old value best is to remove it.
- Variables with escape class='p' now hashes the content to be used as T3 FE passwords. This let's QFQ handle FE User registration or password reset.

16.32.2 Features

- Manual.rst: update pathFilename to pathFileName an all places.
- Apply padding-toplbotttom to fieldset via qfq-fieldset class.
- F7165 / fe user registration. New escape type 'p' for T3 passwords.

16.32.3 Bug Fixes

- #7634 / Session Timeout too short. Annoying 'session expired' message removed. Default timeout now takes the system defautl.
- #7864 / 'required'-FE Elements, deactivated via formModeGlobal=requiredOff missed the read marker. Class 'required-field' is now always assigned. The final 'required' mode is still temporarily disabled.
- #7848 / extraColumName 'paged' - easier handling in case 'r=0' or empty 'U:...' - fixes #7848

16.33 Version 19.2.0

Date: 07.02.2019

16.33.1 Bug Fixes

- #7714 / autocron fails to open logfiles - adjust CWD based on argv(0).

16.34 Version 19.1.3

Date: 28.1.2019

16.34.1 Notes

- If a variable violates a sanitize class, the substituted result can now be configured: a) !!<class>!!, b) '0', c) '', d) '<custom message>'.
- Alerts (based on _link class), might now show only 'ok' (alone, without 'cancel').
- Excel Import - three new options: importNamedSheetsOnly, importSetReadDataOnly, importListSheetNames

16.34.2 Features

- SQL Error / underlining in exception dialog: Add two SQL errors to be underlined in exceptions. Extend to "... in 'order clause'"
- Extend allowed SQL commands in QFQ vars (have been already subscribed in that way in Manual.rst).
- New escape mode 'C' - escapes ':' by '.' - useful for variables in variable definition.
- fillStoreVar: Replace setStore() with appendToStore()
- #6914 / Customized typeMessageViolation. Incl. unit tests.
- #7743 / Move error messages to Constants.php. Unit tests use those constants now. 'data-pattern-error' only delivered if a 'pattern' is given. 'required' attribute only delivered is set. Detection of 'pattern error' on per QFQ default, custom instance wide, per form or per FormElement - per FormElement overwrites other. Move default pattern to constants. Make default error text more specific (only if default error text is not explicit set in config, form or form-element)
- #7747 / New options to import Excel files: importNamedSheetsOnly, importSetReadDataOnly, importListSheetNames
- #7684 / Optional hide second button (cancel) in link/question alerts.

16.34.3 Bug Fixes

- #7743 / Form-Element: Explicit given '0' for MIN or MAX has been interpreted as 'not set'.
- #7702 / Form,Form-Element: Unnecessary evaluation of column 'noteInternal' / 'adminNote'.
- #7695 / Form/URL Forward: pageAlias not substituted.
- #7686 / FormAction/sendmail: uninitialised sendMailAttachment.

- #7685 / Open FormElement from QFQ error message and save modified record: report error about missing {{formId:F}}.
- FormElement.type=select: fixed problem with dynamic update and mode=readonly - list was still selectable.
- Hint to skip leading zeros in version number.

16.35 Version 19.1.2

Date: 17.01.2019

16.35.1 Notes

- Align FormElement labels left/center/right.
- Cleanup FormEditor.
- All SQL commands now are allowed.

16.35.2 Features

- #7620 / Label align left/center/right. Defined by config/form/formelement.
- #7647 / Preparation for Selenium Tests - Implement new token 'A' to add any custom attribute to '... AS _link'
- Cleanup FormEditor: FormElement as second pill. Hide MultiForm as long as it not active. Rename Various to Layout.
- Manual.rst: Add some tips. Add note how to extend TinyMCE valid_elements.
- Delete composer in Resources dir, run phpunit tests in gitlab pipeline.
- Database.php: allow all SQL commands.

16.35.3 Bug Fixes

- #7671 / On FormElements, 'fillStoreVar' is now detected and fired at first.
- Fix for checkbox-inline / radio-inline.
- Check for non unique FormElements: multiple empty FE.name (e.g. for FE.type=subrecord) are allowed now.

16.36 Version 19.1.1

Date: 04.01.2019

16.36.1 Bug Fixes

- #7600 / Path to sendEmail has changed and is updated now.
- #7603 / Fix problem: formEditor.sql broken - missing semicolon. QFQ updates did not played formEditor.sql. Unit test for DatabaseUpdate(). Especially that formEditor.sql is running fine.

- #7594 / FE.type=extra: don't name it 'type' or 'id' or 'L' - more detailed error message and an explanation in Manual.rst.
- Config.php: error message about to high session timeout now reports the PHP settings.

16.37 Version 19.1.0

Date: 03.01.2019

16.37.1 Notes

16.37.2 Features

- Session expired: report details about session timestamps
- File not found in FE.type=file: Show more clearly that the pathfilename is only shown when ShowDebugInfo=on.

16.37.3 Bug Fixes

- #7553 / If a pill is hidden, FE on that one should not be processed during save/update.
- #7573 / Upload: Do FillStoreVar before slaveId.
- #7551 / General error: Access to undefined index.
- #7544 / General error: Download.php / Line: 175.
- #7543 / General error: QuickFormQuery.php / Line: 1364. Happened during upload.
- Download Excel (and all other download types): Content-Disposition header delivered/suppressed in the opposite meaning as it should be. Seems to be fixed now.

16.38 Version 18.12.3

Date: 25.12.2018

16.38.1 Features

- Form: Add text 'Record id/Created/Modified' to tooltip of save button.

16.38.2 Bug Fixes

- #7540 / Form: Upload broken. HelperFile.php: correctRelativePathFileName() broken after refactor of qfq paths.
- #7514 / Report: Broken defaults in _pdf, _file, _link.
- #7538 / Report: Excel.php - access to undefined index.
- #7289 / Report: {{<level>.line.insertId}} - missing for altsql.

- #7539 / Report: Copy to clipboard not reliable. 'Direct' content now correctly encoded. 'Copy file content' fully implemented for text files.

16.39 Version 18.12.2

Date: 24.12.2018

16.39.1 Notes

- Version skipped. Build problems in CI queue.

16.40 Version 18.12.1

Date: 22.12.2018

16.40.1 Notes

- Existing installations: update QFQ extension config form-layout.formBsColumns/formBsLabelColumns/formBsInputColumns,for
  old: '12', new: 'col-md-12 col-lg10' resp. smaller values for individual columns.
- New config values:
  - Config/flagProduction: yes/now - differentiate between development und production system. Will be used for 'throwExceptionGeneralError' too.
  - Debug/throwExceptionGeneralError - shows/hide exception of general errors.
- Renamed config values:
  - SITE_PATH >> sitePath
  - EXT_PATH >> extPath
  - _dbName >> dbName
- Record locking: revert to old behaviour, that a locked record can't be modified by another form, even if the second form has modeRecordLock=NONE.
- Autocron: update the system crontab entry to the new path (old 'qfq', new 'Source'):

  . . . /typo3conf/ext/qfq/Classes/External/autocron.php

16.40.2 Features

- #7228 / Show error if form element with same name and class already exists.
- #7494 / Exception 'General Error': disable/enable per config.
- Adapt class paths in composer.json to the newly refactored folder names.
- Add 'col-lg-10' to notes section.
- Added Alert Manager to have better control over the Alerts.
- Config.php: make 'reading dbname' Typo3 version dependent.

- Delete two old composer.json and call new composer in Makefile.
- Excel.php: change autoload.php path to new composer folder.
- Manual.rst: several places where the bs-column description are updated with the new way 'col-md-...' instead of '12'. Replace ''{{pageId' with '{{pageAlias'. Replace '... AS _Page' by '... AS _page'. Add 'tablesorter tablesorter-filter' to FormEditor example page.
- Move bootstrap.php and BindParamTest.php due to refactoring.
- phpunit.xml: implement const 'PHPUNIT_QFQ'. Store.php: set self::$phpUnit on const 'PHPUNIT_QFQ'
- Refactor: 'extension/qfq/qfq/...' to 'extension/Classes/Core/...'
- Refactor: Manual.rst update config variables (reorder), add 'qfqLog'. Support.php: formSubmitLog hardcoded to fileadmin/protected/log. DOCUMENTATION_QFQ > SYSTEM_DOCUMENTATION_QFQ. Remove config var 'logDir'.
- Refactor: SITE_PATH >> sitePath, EXT_PATH >> extPath, SYSTEM_PATH_EXT >> SYSTEM_EXT_PATH
- Remove report/Define.php, report/Error.php.

16.40.3 Bug Fixes

- #3464 / Checkboxes now disabled (readonly), even when rendered as Bootstrap. Fixes missing readonly for Template Groups.
- #6467 / Sanitizing a hidden field makes the form unsubmittable. Updated elementupdate for pattern, added "false" as an alternative to null. 'element-update' now get's 'pattern=<pattern>|false' on element-update.
- #7089 / FE.type=extra: value already set in SIP store.
- #7223 / Add "-" as allowed characters in filenames.
- #7455 / phpunit: Remove outdated report syntax. Catch exception on trying to open a non existing logfile.
- #7461 / Bug in Doku.
- #7464 / DragAndDrop - Undefined index: ord. Bug in subrecord fixed.
- AbstractException.php: fixed problem with empty $match in sql syntax highlighting.
- Block screen stuck seems fixed.
- Check not to try to number_format() empty string.
- config.qfq.example.php: add missing '>'.
- Fixed broken init in phpunit run. Fixed access to uninitialized var. Throw exception if dndTable or form is missing.
- Fixed missing $formElement[FE_DECIMAL_FORMAT]. Add 'missing primary record' check. Fixed missing fe['id'].
- Fixed problem with '?:' - implemented in 2016.
- Fixed problem with resultset in 'altsql'.
- formEditor.sql, Manual.rst: renamed '{{_dbName...:Y}}' to '{{dbName...:Y}}'.
- Manual.rst: Unify 'dbName*' documentation..
- phpunit: Update fixtures table 'Form', 'FormElement' & 'Dirty' definition. Update LDAP Test - surrounding spaces seems to be escaped now by '20'. Create tests dir outside of source, create phpunit.xml, move some tests and make them work. Rename tests directory to Tests

- Record Lock. Revert change in cb2e2a70cfe5c251cffbfce65bdc0899da75a9c5: if a record lock exist, another form, with record lock mode=NONE, can't get write access to it. This is what the manual describes.
- Sanitize.php: fixes "Uninitialized string offset: 0" error.
- Save.php: Fixed exception (file size, mime type) for non-existing uploads.
- Store.php: fixed problem with wrong config varname for DB.
- Timeout check throws error since php.ini cookie lifetime is zero during unit test. Fix filepath relative to Typo3 dir when running unit test.

16.41 Version 18.12.0

Date: 10.12.2018

16.41.1 Notes

- Config.qfq.php: the variable T3_DB_NAME is not necessary anymore.
- Following SYSTEM_STORE variables renamed. Old: '_dbNameQfq' ,'_dbNameData'. New: 'dbNameQfq' ,'dbNameData'
- New: Bootstrap 'col-lg-10' is defined on every form. On screens greater than 'md' the forms won't be expanded to 100% anymore.

16.41.2 Features

- #3992 / STORE_SYSTEM: dbNameQfq, dbNameData, dbNameT3
- Config.php: read 'dbNameT3' from TYPO3_CONF_VARS or from T3 config file.
- Download.php: get dbNameT3 now from STORE_SYSTEM
- #4906 / Php Session Timeout: can be specified globally in configuration and per form. Affects only logged in FE User.
- #6866 / On logout destroy STORE_USER. Session.php: check if _COOKIE['fe_typo_user'] has changed - yes: clear STORE_USER. Store.php: Rearrange functions.
- #6999 / Bootstrap/Form: define columns for desktop 'col-lg-10'
- #7138 / PDF / single source: deliver without converting
- #7293 / Implement new logging for file upload.
- #7406 / dbinit might contain multiple sql statements now.
- #7407 / MariaDB / Ubuntu 18 complains about missing values if column of type TEXT isn't explicit specified in INSERT. New default for database.init=SET sql_mode = "NO_ENGINE_SUBSTITUTION"
- #7431 / FE.type=afterSave (FE Action): SQL won't report the causing FE.name/id
- #7434 / FE.type=beforeLoad / sqlValidate: Validation message not shown to user
- FormEditor.sql: Switch off 'MySQL strict setting of default values'
- Logger.php: remove UserAgent - that one is logged in FormSubmitLog Table. Add 'cookie' to filter individual actions.

- New css classes for icons: .icon-flipped (mirrors icon), .icon-spin (icon spins once on hover), .icon-spin-reverse (mirror of icon spin).

16.41.3 Bug Fixes

- #7001 / Error message: if 'modeSql' fails, error message does not contain a reference to the causing FE.
- ErrorHandler.php: raise an error as an exception has been stopped in mid 2017 - reactivated now.
- Fix problem in upload elements: a) in exception FE_ID was not reported, b) fillStoreVar was not fired.
- FormAction.php: throw exception if 'fillStoreVar' selects more than one row.

16.42 Version 18.10.3

Date: 29.10.2018

16.42.1 Notes

- QFQ now recommends ImageMagick instead of GraphicsMagic, due to much better 'auto orient' (JPEG) capabilities.

16.42.2 Features

- #4901 / Implement 'delete split files' if primary file is deleted (record) or replaced against a new one.
- #4901 / Implement 'fileSplitOptions'.
- #4901 / Implement PDF split based on IM 'convert' (jpeg).
- #7012 / Memory limit for file download - replace file_get_content() by readfile().
- #7112 / fabric: configure default color.
- thumbnail: empty path filename is no ignored, instead of throwing an exception. This is the same behaviour as building links - no definition means 'no link', and not 'error'.
- Refactor chmod(), unlink(), rename(), rmdir(), copy() to use by QFQ version which throws an exception.

16.42.3 Bug Fixes

- #3613 - Revert due to bug in dynamic show/hide (Revision 0bb99fd, Revision 77096ca7) -

16.43 Version 18.10.2

Date: 13.10.2018

16.43.1 Features

- #2509 / Render encrypted mailto as single link
- #3281 / Trim form inputs
- #4649 / Add sqlBefore & sqlAfter to sendmail FE, move validate() before sqlBefore()/sqlAfter().
- #4922 / Some minor improvements to the excel import
- #5112 / Add incompatibility warning for encode specialchar and checkType allbut. . .
- #5450 / MySQL Exception: underline faulty area
- #6596 / uid ExcelExport / PDF
- #6944 / Add double comma SQL Hint

16.43.2 Bug Fixes

- #3529 / No double urldecode() of GET parameters
- #3613 / Label input note layout
- #3850 / Change set input maxlength
- #4545 / After delete: reload page with original parameters
- #4751 / Add bulleted und numbered lists back to tinyMCE Editor
- #4765 / Extend tooltip visibility for checkboxes and radio buttons
- #6911 / Only fire afterInsert on new record (also fix afterUpdate, beforeInsert,. . .
- #6929 / Treat single file argument like several file argument (savePdf: copy,. . .

16.44 Version 18.10.1

Date: 12.10.2018

16.44.1 Features

- #5578 / Safari only handles one filetype in upload dialog.
- #6991 / Optional process 'readonly' FE during save. New FE parameter 'processReadOnly = 0|1'.

16.44.2 Bug Fixes

- #6880 / Fixed Exceptions with too many details to end user.
- 'Drag and drop' failed due to fillStoreForm requests {{form:S}} which was not necessary for drag and drop.
- Upload: rename 'chmod' to 'chmodFile'. Implement 'chmodDir'. Permissions applied for all new created directories.
- Upload: replace 'rename' with 'copy/unlink'

16.45 Version 18.10.0

Date: 04.10.2018

16.45.1 Features

- #6894 / Upload: chmod for file creation.
- #6886 / Upload: Auto Orient - implementation.
- #6721 / Log switching {{feUser:U}} to qfq.log. Log {{feUser:U}} to sql.log.
- #5458 / Add '{{feUser:U}}' to be shown on exception.
- Manual.rst: Fix missing single tick in special column name '_=<var>'
- Report / Variables copied to STORE_USER via "... AS '_=<varname>'" are now available in STORE_RECORD by {{<varname>:R}}

16.45.2 Bug Fixes

- #6902 / Drag and drop in subrecords: expect 1 row got nothing.

16.46 Version 18.9.2

Date: 16.09.2018

16.46.1 Notes

- To use the new 'tablesorter' feature, please add 'tablesorter-bootstrap.css', 'jquery.tablesorter.combined.min.js', 'jquery.tablesorter.pager.min.js', 'widget-columnSelector.min.js' in your Typo3 template record. See https://docs.typo3.org/p/IMATHUZH/qfq/master/en-us/Manual.html#setup-css-js
  - *Existing* QFQ installations: update your CSS/JS includes! The new tablesorter jquery plugin might break (JS errors seen on the console) your installation, if it isn't included!
  - If you use the extension 'UZH Corporate Design Template':
    - Update to the latest version 18.9.0.
    - Add constants in the Typo3 template record 'ext: main':

```
cd.extra.css.file5 = typo3conf/ext/qfq/Resources/Public/Css/tablesorter-
↪bootstrap.css

cd.extra.js.file10 = typo3conf/ext/qfq/Resources/Public/JavaScript/
↪jquery.tablesorter.combined.min.js
cd.extra.js.file11 = typo3conf/ext/qfq/Resources/Public/JavaScript/
↪jquery.tablesorter.pager.min.js
cd.extra.js.file12 = typo3conf/ext/qfq/Resources/Public/JavaScript/
↪widget-columnSelector.min.js
```

- STORE_USER: check the examples - great new feature to temporary save user settings.

16.46.2 Features

- #6721 / STORE_USER: variables per browser session
- #6690 / Tablesorter for subrecords

16.47 Version 18.9.1

Date: 15.09.2018

16.47.1 Notes

- Report
  - Type of nesting delimiter in 'report' now limited to '{', '[', '(', '['
  - Hide/reuse report content later: *10.content=hide* and later *1000.head = {{10.content}}*
- New Excel import - copy Excel files directly into a DB table.
- Forms with named primary key different than 'id' are now supported.

16.47.2 Features

- #1261 / Tablesorter, incl. saved sort, combined column sort, filters, pagination
- #3129 / suggestion for subrecord title design
- #4922 / Excel import
- #6300 / Disable preview button for requiredNew forms
- #6314 / HTML Mode & sendMailSubjectHtmlEntity for Forms.
- #6481 / Add custom primary key option to Forms.
- #6645 / better error message for incomplete download as link
- #6650 / Report hide content (line) - use later via a variable. First version - problem with enclosed ticks.
- #6653 / Add save button class/glyphicon/tooltip for submit button
- Thesis code correction
- Update QFQ download URL

16.47.3 Bug Fixes

- AbtractBuildForm.php: fix problem with subrecords in MultiDB Environment.
- #2340 / Report: Problematic Bracket - form now on, only ''{[(<" will change the nesting token.
- #3333 / Fixed subquery recognition in reports '10.sql, 10.1.sql ...'
- #4837 / Don't display hidden pills
- #5467 / Fill Record Store when evaluating min/max parameters
- #6621 / Fix shifted subrecord head

- #6646 / Fix broken extraInfoButton for some FEs

16.48 Version 18.9.0

Date: 07.09.2018

16.48.1 Features

- #6357 / Save pdf on server
- #5381 / Stored procedures can be called from QFQ Reports
- #4996 / Log QFQ Update with timestamp.
- #6255 / Inline Report Editing - now with SIP and save.php api

16.48.2 Bug Fixes

- #6465 / Allow newlines in form action queries (e.g. sqlInsert)
- #4654 / Better FE color highlighting (UX)
- #5689 / Default BS Columns for FormElement match Form setting
- #6484 / Download Links mit css class
- #6576 / download buttons are now rendered disabled with render mode r:3
- Cookie Sitepath: wrong detected in case of API calls.

16.49 Version 18.8.2

Date: 28.8.18

16.49.1 Features

- #6563 / Accept 0 as required.

16.49.2 Bug Fixes

- DatabaseUpdateData.php: add missed 'on the fly' update for Form.title, changed in FormEditor.sql in 18.8.1
- 6562 / sendmail: redirect all mail - the sender is replaced too.
- Manual.rst: several typos fixed

16.50 Version 18.8.1

Date: 26.08.2018

16.50.1 Features

- #4432 / Every 'form submit' will be logged with raw data.
- #4763 / Render vertical text more stable: '... AS _vertical'
- #4996 / Log QFQ Version update
- #5403 / Tooltip on pills are now supported
- #5876 / Subrecord title of column 'Edit' & 'Delete' are now customizable.
- #6249 / Subrecords can now be reordered via drag and drop.
- #6333 / Add to qfq.log: IP Address, User Agent, QFQ Cookie, FE User

16.50.2 Bug Fixes

- #6401 / Handle Backticks in sendmail
- #6452 / Empty form title: no title row will be rendered anymore.

16.51 Version 18.8.0

Date: 25.08.2018

16.51.1 Notes

- Excel export
- Copy to clipboard

16.51.2 Features

- #4922 / Excel Export - create Excel sheets from scratch or based on a template.
- #3294 / Improve Typo3 QFQ backend layout. Add sparql syntax highlighting.
- #5878 / Formelement.type=note with #!report - whitespace is trimmed.
- #6314 / HTML Mails enabled by specifying flag 'mode=html'.
- Import/Merge form: A new form 'copyFormFromExt' (see file *copyFormFromExt.sql*) offers a one click import of external QFQ forms (incl. renumbering of id's).
- formEditor.sql: resized Form.title from 255 to 511 (requested by IK Tool)
- Drag and Drop now offers the possibility to show the renumbered values.
- Manual.rst: security hints, T3 Setup best practice, text input retype, charactercountwrap.
- Config.qfq: central defaults for DATA_MATCH, DATA_ERROR
- Bootstrap QFQ development: switched from bower to npm only.

16.51.3 Bug Fixes

- #5843 / File upload: limitation to file extensions are no case insensitive.
- #6247 / Replace deprecated each function
- #6281 / FormElement / column 'note': token '#!report' - STORE_RECORD does not work.
- #6331 / File Upload: Wrong error message if filesize is much too big.
- #6229 / Add QFQ icon to content element and content element wizard
- AbstractException.php: fixed problem with htmlEntities() on link to 'Edit Form' and 'Edit FormElement'.

16.52 Version 18.6.1

Date: 21.06.2018

16.52.1 Notes

- Configuration QFQ: form-config.formDataPatternError. New behaviour: If this field is empty, a more specific default message is shown (instead of one message for all situations). Best is to clear this field.

16.52.2 Features

- sqlHint / Note if a query fails and contains some not replaced variables.
- #4438 / Log attack detected: will be logged now to fileadmin/protected/log/qfq.log.
- #4041 / Subrecord: Spalte 'id' automatisch mit '' wrappen.
- #5885 / show 'sql.log' in FE.
- #6121 / Formular: ID per Default in Titel.

16.52.3 Bug Fixes

- #6283 / Form: hide title frame if empty.
- #4299 / HiddenSelect' into 'master'.
- #6276 / default data-required-error moved to central Config.php.
- #5884 / sql.log by default public - protect against access.
- #6276 / Default check_type messages not shown.
- #6233 / Alert 'Form incomplete' - stays until click - auto disappear would be better.

16.53 Version 18.6.0

Date: 13.06.2018

16.53.1 Notes

- config.qfq.ini migrated to config.qfq.php - the old config.qfq.ini get's *chmod 000*.
- Most of config.qfq.ini migrated to Typo3 / Extension Manager - all but the DB /LDAP credentials.
- Keep in config.qfq.ini:

```
# Rename DB credentials from DB_<key> to DB_1_<key>, with key =
↪'NAME|HOST|USER|PASSWORD'
DB_1_USER = ...
DB_1_SERVER = ...
DB_1_PASSWORD = ...
DB_1_NAME = ...
```

- NEW: Drag and drop to sort elements! Check the Manual.
- *URL forwardMode*
- *client* renamed to *auto*.
- *close* added.

16.53.2 Features

- #6100 / Url Forward Auto: Update Manual.rst. The F.parameter.saveAndClose has been removed again. Mode 'close' can be assigned statically or dynamic.
- #6178 / Input: Step: New option 'step' for FE.parameter.
- Download.php: references to non existing files now reported as missing file, not 'wrong mimetype' anymore.
- #4918 / Drag'n'Drop reorder elements DRAGANDDROP.md, PROTOCOL.md: Doc for "drag'n' drop" implementation.
- dragAndDrop.php: API endpoint DragAndDrop.php: Class for implementing drag'n' drop functionality.
- Link.php: implement new renderMode=8 - returning only the sip. QuickFormQuery.php: New entry point for processing "drag'n' drop".
- #3971 / Form title: new design from form title.

16.53.3 Bug Fixes

- #5077 / Dynamic Update & FE.type=required: Server fixed -

a) dynamic calculated modeSql respected,

b) formModeGlobal=requiredOff respected,

c) dynamic FE with mode='hidden' are not saved anymore.

- #6176 / Icon not aligned when error text: Buttons now wrapped in one 'input-group'.
- Manual.rst: reformat autocron QFQ code.
- #5880 / Skip Error Message during dynamicUpdate.
- #5870 / Missing file config.qfq.ini: Clean QFQ message.
- #5924 / config.qfq.ini/LocalConfiguration.php: several places in formEditor.sql still contained the 'dbIndex...'.

- #6168 Configuration language setting ignored: Form and FormElement editor still used uppercase config values for language configuration. Updated to the new camel case notation.
- #5890 / config.qfq.ini is public readable. Renamed file to config.qfq.php. Implement a basic migration assistant to copy DB credentials to new config.qfq.php. All other values have to be copied to extmanager/qfq-configuration manually.
- #6216 / Oops, an error occurred! Code - unhandled exception will be caught now.

16.54 Version 18.4.4

Date: 28.04.18

16.54.1 Bug Fixes

- Fix broken ext_emconf.php

16.55 Version 18.4.3

Date: 28.04.18

16.55.1 Bug Fixes

- Version Number . . . 04. . . not supported by TE. Changing naming scheme to omit leading zero.

16.56 Version 18.04.1

Date: 28.04.2018

16.56.1 Bug Fixes

- config: broken dbIndexQfq, dbIndexData.

16.57 Version 18.04.0

Date: 26.04.2018

16.57.1 Notes

- QFQ marked as 'stable'
- New version numbering: Year.Month.Index
- Manual.rst:
  - AutoCron documentation enhanced.

  - Replace '{{form:S}}' against '{{form:SE}}'.
  - Check list for 'new installations'.
  - Description for config variables enhanced.
  - Details 'how record locking' is done.
  - Details: extraButtonInfo.
  - Replace config.qfq.ini on most places with 'configuration'.
- Path of 'sql.log' / 'mail.log' are now relative to <site path> (not <ext path> as before).

16.57.2 Features

- formEditor.sql: update table cron.
- AutoCron.php: allow https connections with invalid certificate (e.g. 'localhost' is not listed as a valid hostname).
- ext_conf_template.txt: Extension manager configuration setup.

16.57.3 Bug Fixes

- AutoCron:
  - Update form 'cron' to load/save records in DB_INDEX_QFQ.
  - Fix problem with array in checkForOldJobs().
  - Implement check that re-trigger asynchronous cron jobs are handled correctly.

16.58 Version 0.25.15

Date: 20.03.2018

16.58.1 Features

- Fabric Read Only mockup.

16.58.2 Bug Fixes

- #5706 / Fixed that problematic characters in 'fileDestination' has not been sanatized.
- Fixed problem with buttons clipping trough alert.
- Client: wrong variable, updated CSS for long errors.

16.59 Version 0.25.14a

Date: 15.03.2018

16.59.1 Features

- Change getMimeType() in Report in case file is missing or *file* beaks: instead to throw an exception, an empty string is returned.
- Updated protocol.md with Alert description.
- Update Status message for save/delete.
- Makefile: 1) remove sonar, add dependency to let update-qfq-doc run. 2) do qfq doc commit inside of the Makefile.
- Client: Changed save timeout from 1500 to 3000.
- Client: removing the blackout screen when modal gets dismissed.
- Client: modal alerts are now blocking everything.
- Manual.rst: fix RST syntax errors.

16.59.2 Bug Fixes

- #5677-TinyMCE broken - fixed.

16.60 Version 0.25.14

Date: 14.03.2018

16.60.1 Features

- Change notification from 'save: success' to 'Save' and 'delete: success' to 'Delete'.
- DB update: write intermediate QFQ version after every step.

16.60.2 Bug Fixes

- #5652 / TypeAheadSql: destroyed SQL statement. Fixed broken compare and missing init of $sqlTest.
- #5668 / Fix Broken SIP after login.

16.61 Version 0.25.13

Date: 08.03.18

16.61.1 Features

- AutoCron: Added doc for autocron. Extend AutoCron.php to be MultiDB aware. Update der AutoCron form.
- #4720 / Separate Database for Form & FormElement - Multi DB - fixed problem that 'Quick Edit Form / FormElement' has been broken in MultiDB Setup.
- #5603 / Report: final value of report columns (special column name).

- Fabric / delete now triggers form.changed / emojis work again.
- #5571 / File Upload: save filesize and mimetype automatically in 'upload mode simple',if those columns exist.
- #5423 / two new column names 'filesize', 'mimetype'.
- #5571 / File Upload: save filesize and mimetype.
  - STORE_VARS contains now 'mimeType' and 'fileSize'.
  - sqlBefore and sqlAfter will be fired in Upload Advanced and new in Upload Simple as well.
  - STORE_VARS contains now *filenameOnly*. It can be used in downloadButton=....

16.61.2 Bug Fixes

- Fabric: Corrected resizing with changed width in editor.
- #5640 / UTF8 encoded strings: MAX LENGTH wrong.

16.62 Version 0.25.12

Date: 18.02.2018

16.62.1 Notes

- New
  - FE.parameter:
    - timeIsOptional
    - enterAsSubmit
  - FE.checkType: Auto
  - Thumbnail rendering. Public or secure.
- Update
  - Multi DB Support: Form & Report

16.62.2 Features

- #5064 / Throw user form exception on invalid date.
- #5308 / TimeIsOptional parameter.
- #5318 / Allow sendmail speaking word token, adjust documentation and fix some typos.
- #5347 / Error Message (Exception): BS colored box for report error messages. Hide technical informations, show it on click.
- #5392 / Violate message with expected date format.
- #5414 / Add checkType Auto, refactor setDefault methods, add smart detection of defaults, extend documentation and rules.
- #3470 / Enter As Submit= on/off - implemented.

- #4437 / violate sanitize message.
- #4542 / input-type-decimal' into 'master'.
- #5298 / Update docs for HTML mails.
- #5333 / Thumbnail: implementation.
- #5425 / Thumbnail: render mode 7 - implemented, rewrite - secure thumbnails are now rendered on first access, not when 'AS _thumbnail' is called.
- Implemented $dbIndex for Report.
- Implemeted two new STORE_SYSTEM variables: '_dbNameData' and '_dbNameQfq' - those will be automatically filled qfq during instantiation QuickFormQuery(). They can be used in Report to easily access the needed DB.
- Increased Formelement.label from 255 to 511.
- Make DB_INIT in config.qfq.ini set by default.
- Notes how to optimize PDF thumbnailing.
- Reformat manual for config.qfq.ini. Copy config.qfq.example.ini to MANUAL.rst. Migrate config defaults from setIfNotSet() to array_merge().
- Security: hide $SQL in error messages to regular user.
- New FE.parameter 'inputType'. Can optional be given by webmaster. Additional, the 'type="number"' will be automatically set, if the column is of type 'int' or if 'min' and 'max' is numerically.

16.62.3 Bug Fixes

- #3192 / Fill STORE_RECORD before loading table title.
- #5285 / Make typeAheadPedantic the default.
- #5348 / Exception/Report: level key missing.
- #5367 / Error Report: reworked alerts, updated css for alerts, 'full level' missing, content too much escaped: Fixed too much escaping. Form / FormElement Links in error messages now with BS Buttons.
- #5382 / Double quotes in tooltips are now escaped with ".
- #5390 / input validation decimal broken. fixed.
- #5430 / Add unique ID to each radio button for dynamic update.
- Form: 'FormElement' > 'Container' - relied on '{{formId:S}}' even if the FE record already exist - fixed.
- Subrecord Title - now wrapped with <label class='control-label'>.

16.63 Version 0.25.11

Date: 31.01.2018

16.63.1 Notes

- Violating a sanitize class now returns '!!<sanitize class>!!' instead of an empty string.

16.63.2 Features

- #5022 / Variable violates sanatize class: 'msg' instead of empty string - new identifier "!!<sanitize class>!!".
- #4813 / Exception during form load: show 'form edit link' if editor is logged in.
- formEditor.sql: Increas size of Form.title to give more room for SQL statements in.
- Manual.rst: enhance debug tipps.
- #5321 / Plain Link - render mode- only url - implemented.
- Add regex101 link to checkPattern FormEditor.

16.63.3 Bug Fixes

- Fixed some broken help links in formEditor.sql.
- #5306 / Exception: tt_content_uid wrong - fixed.
- #4303 / Download von doc/docx-Dateien / Download.php - Mime type wird nicht mehr an Dateiname angehängt.
- #5316 / Help on how to send an E-Mail is wrong - several places fixed.
- #5311 / Error Msg SLQ_RAW != SQL_FINAL: Debug message shows outdated SQL_RAW.
- #5309 / min/max broken for date fields. Add min/max attributes to input and date input tag.
- Fabric now detects 'dirty'.
- Manual.rst: Remove broken link to W3C file upload.

16.64 Version 0.25.10

Date: 26.01.2018

16.64.1 Notes

- PROTOCOL.md: update notes.
- Form / Upload: new option 'downloadButton' - if given renders a download button instead of showing the pathFileName.

16.64.2 Features

- #5023 / Fabric: Cut, rotate and enhance uploaded images. Update Manual.
- All FE 'typeahead' fields are set to 'autocomplete="off"'. Respect user setting for 'autocomplete' - if none given (mostly), set it for FE 'typeahead' to 'off'.
- #5295 / Upload: check if given QFQ 'maxFileSize' is higher than php.in post_max_size, upload_max_filesize.
- FE.Subrecord: rearranged column order, start columns with uppercase letter.
- New CSS class 'qfq-full-width-left': especially for buttons to become full width.
- New CSS class 'qfq-table-100' - 100% width, with auto width per column. FE.subrecord changed to 'qfq-table-100'.

- #5302 / remove CSS class 'internal / external'.

16.64.3 Bug Fixes

- #5189 / BCC SendMail Problem - fixed missing double ticks.
- Manual.rst: Update documentation that the default escape type is 'm'. Remove subrecord/list (have been removed long time ago). Fix enumeration problem FE.type=radio *classButton*. Add short note for typeahead.js. Remove never implemented 'keySemdId...', 'ANREDE'. Fixed typo - replace '' by '' on most places (not in code sections). More generic SQL to extract filename from pathFileName. Fixed several phinx syntax errors. Add example for 'recent list'-records.
- Fixed problem with missing 'if note exits' in CREATE TABLE *Split*.
- #5030 / Manual.rst: Fixed example with XSS vulnerability.
- #5275 / typeahead.bundle.min.js missing in Manual.rst: fixed.
- FormEditor: 'typeahead' for column 'name' fixed. Attention: only succeed if DB_1_NAME is the final DB (mostly given).
- #5048 / Default value NULL in pathFileName breaks uploads.
- #5028 / Links im FormularEditor zeigen ins Leere (Fehlende Ziel-Anker) - fixed.
- Make readonly BS radio buttons non-selectable.

16.65 Version 0.25.9

Date: 17.12.2017

16.65.1 Features

- #5133 / sendmail: subject and body html entity decode: Introduce options for 'subject' and 'body' to switch on/off HTML encoding / decoding
- Manual.rst: Add notes to QFQ installation, wkhtml problems, paragraph on 'sendEmail' Html2Pdf.php: Add error codes and a hint on wkhtml fails.
- Reformat table qfq-letter.css.less: redefined h1, letter-receiver.

16.65.2 Bug Fixes

- Bug in sendEmail: invalid SSL_version specified at /usr/share/perl5/IO/Socket/SSL.pm line 575. Patch for sendEmail (see https://unix.stackexchange.com/a/68952).

16.66 Version 0.25.8

Date: 11.12.2017

16.66.1 Features

- #5080 / Dynamic PDF Letter.
- #5083 / Bodytext / Report: join lines without spaces.

16.66.2 Bug Fixes

- Fix problem with commit from 8.12.17 / Store.php: appendToStore.php stopped working - 'report' failed to replace '{{<column>:R}}'.
- Store.php: fix problem with empty 'appendToStore()' call.

16.67 Version 0.25.7

Date: 07.12.2017

16.67.1 Notes

- Report: parameter in '... AS _sendmail' needs token now - position dependent is removed now.
- Report: parameter 'a:' in '... AS _sendmail' replaced by 'F:' to be compatible with downloads. Do not separate files by comma.
- Manual: most occurences of 'U:' replaced by 'p:' - same meaning.

16.67.2 Features

- #4255 / Attachments for emails implemented.

16.67.3 Bug Fixes

- Bug - PHP Warning: Declaration of qfqBuildFormTable::head() should be compatible with qfqAbstractBuildForm::head($mode = qfqFORM_LOAD) - fixed.

16.68 Version 0.25.6

Date: 03.12.2017

16.68.1 Notes

Bigger changes in update form after save/dynamic update.

16.68.2 Bug Fixes

- #4865 / Pill Dynamic Updates Show / Hide.
- #5031 / Missing details in DbException: New definition of SYSTEM_SHOW_DEBUG_INFO: even after config.qfq.ini is parsed and SIP Infos has been read - if there is no BE User logged in, the value stays on 'auto' (earlier it has been replaced to 'no'). Staying on 'auto' keeps the information that replacing is still open and not replaced means 'no'-BE User logged in.
- #5016 / Loose checkbox value on save - Dirty workaround - better solution necessary.
- #5017 / STORE_RECORD used in FormElement and via '#!report' - save & restore STORE_RECORD.
- #5004 / FormElement with state 'ReadOnly' will be saved with empty value - existing values will be overwritten - fixed.
- 'element-update' for type 'UPLOAD seems to make trouble. Exclude it like 'SELECT'.

16.69 Version 0.25.5

Date: 23.11.17

16.69.1 Bug Fixes

- #4771: Workaround which switches off updates to SELECT lists, if they are part of a Multi-FE-Row.

16.70 Version 0.25.4

Date: 22.11.2017

16.70.1 Notes

- New keywords / features in report:
  - *altsql*: Fire the query if there is no record selected in *sql*. Shown after *althead*.
  - *shead*: Static head - will always be shown (before *head*), independent of sql selects records or not.
  - *stail*: Static tail - will always be shown (after *tail*), independent of sql selects records or not.

16.70.2 Features

- #2948 / altsql, shead, stail - new directives in Report.
- #4255 / Attachments fuer 'Email'. Static files can be attached to mails.

16.70.3 Bug Fixes

- #4980 / Variables in Report: a) nested not replaced, b) 'rbeg' not replaced, c) missing unit tests.

16.71 Version 0.25.3

Date: 19.11.2017

16.71.1 Notes

- Report:
  - Special column name 'sendmail': the old way of position dependent parameter are deprecated. Instead use the new defined token. See https://docs.typo3.org/p/IMATHUZH/qfq/master/en-us/Manual.html#column_sendmail
  - Every row is now merged in STORE_RECORD. Inner SQL statement can now retrieve outer values via STORE_RECORD. E.g. *{{column:R}}*. No more level keys!
- The config.qfq.ini directive *VAR_ADD_BY_SQL* is replaced by *FILL_STORE_SYSTEM_BY_SQL_?*. Up to 3 statements are possible.

16.71.2 Features

- Report / sendmail: control via token.
- #4967 / config.qfq.ini: Rename 'VAR_ADD_BY_SQL' to 'FILL_STORE_SYSTEM_BY_SQL_1'. Handle up to 3 FILL_STORE_SYSTEM_SQL_x. Implement an optional error message together with a full stop.
- #4766 / Set STORE_RECORD in Report per row.

16.71.3 Bug Fixes

- #4966 / Variable {{feUser:T}} is not available in config.qfq.ini *FILL_STORE_SYSTEM_?* - changed ordering of store initialization. Now: TCY...
- #4944 / Delete: broken when using 'tableName' (instead of form).
- #4904 / Undefined Index: DIRTY_FE_USER - PHP problem that constants cant be replaced inside of single ticks. Fixed.
- #4965: insert path to QFQ cookie/session, to make usage of multiple QFQ installation on one host possible.

16.72 Version 0.25.2

Date: 8.11.2017

16.72.1 Notes

- Starting with this release, the default escape mode is 'm' (mysql_real_escape).

16.72.2 Features

- Default Escape Type changed from 's' to 'm'. DatabaseUpdateData.php: removed the DB update from last commit - not necessary. Config.php: New default 'm' Evaluate.php: Respect EscapeTypeDefault in form definition. QuickFormQuery.php: Replace 'EscapeTypeDefault' in form defintion very early.
- #4049 / QFQ Variables '{{...}}' might now contain a default value.
- If 'pageAlias:T' is empty, take 'pageId:T'.

16.72.3 Bug Fixes

- #4836 / Multiple entries in table after several clicks on save. Created a saveInProgress Variable.
- Replaced latest project homepage URL in Manual.rst.
- Fix example SQL for periodId in config.qfq.ini in Manual.rst.
- Remove multiple header 'RELEASE' - there has to be only one.

16.73 Version 0.25.1

Date: 3.11.2017

16.73.1 Bug Fixes

- #4857 / broken (stale) download: multiple 'u:..' or 'u:...'.
- #4212 / Broken JSON on response to save new record 'Unknown index' fixed by isset().

16.74 Version 0.25.0

Date: 10.10.2017

16.74.1 Notes

- The config.qfq.ini directives DB_USER, DB_NAME, DB_HOST, DB_PASSWORD are replaced by DB_1_USER, DB_1_NAME, DB_1_HOST, DB_1_PASSWORD. The old directives are still used, as long as the new directives does not exist.
- New config.qfq.ini directives: DB_INDEX_DATA, DB_INDEX_QFQ.

16.74.2 Features

- #4720 / Separate database handles for QFQ 'form' and QFQ 'data' - 'Form' might now load/save from forign database/host/user.

16.75 Version 0.24.0

Date: 09.10.2017

16.75.1 Notes

- Change Remove SYSTEM_SECURITY_ABSOLUTE_GET_MAX_LENGTH - makes no sense to hardcode an upper limit.

16.75.2 Features

- Feature Manual.rst: Doc updated for latest subrecord column special names.
- Feature AbstractBuildForm.php: new function subrecordHead(). Replaced several hard coded subrecord column names against constants.
- Feature #4456 / formModeGlobal=requiredOff - update Manual.rst.
- Feature #4606 / _link: qualifier to render bootstrap button - fix unit tests for tooltip. Add tooltip to button/text, even if there is no link. Implement token 'b:...' for link class. Manual is updated. Open: *pageX* should be recoded to use the new 'b:' instead of hardcoed behaviour to render a button.
- Feature: Upload Button - wrapped with Bootstrap Button. New option 'fileButtonText' to specify a button text.
- Feature #3752 / Pills auf mode|modeSql=hidden|readonly setzen - implemented during 'form load' (not dynamic update).
- Feature: Neu wird nach dem Speichern das Formular nochmal komplett geladen. Das ist wichtig um die durch aftersave geaenderten Records in die Formularelemente zu bekommen.
- Feature #4511 / Form: URL Forward - mode dynamic computed - more generic implementation.

16.75.3 Bug Fixes

- Bug #4731 / Dynamic Update: load(post) triggers 'check required' - makes no sense during filling a form - fixed.
- Bug #4730 / InvalidDate-00-00-2000 FE.type=date - detection of empty date was broken for '00.00.0000'.
- Bug Fixed problem in subrecord when no record is selected.
- Bug #4620 / Easy Fix: saveButtonText / closeButtonText Formatierung.

16.76 Version 0.23.1

Date: 23.9.2017

16.76.1 Bug Fixes

- #4620 / Easy Fix: saveButtonText / closeButtonText Formatierung.

16.77 Version 0.23.0

Date: 17.09.2017

16.77.1 Features

- #3752 / Pills auf mode|modeSql=hidden|readonly setzen - implemented during 'form load' (not dynamic update).

16.77.2 Bug Fixes

- #4548 /Template Group: 'form-update' broken - Broken Redirect after Save - Broken same HTML ID for FE copies in a template group.
- #4548 /Template Group: 'form-update' broken - max tg element value/index shown after save instead of last user supplied value, but save is ok. Neu wird nach dem Speichern das Formular nochmal komplett geladen. Das ist wichtig um die durch aftersave geaenderten Records in die Formularelemente zu bekommen.

16.78 Version 0.22

Date: 14.09.2017

16.78.1 Notes

- Form Editor: element 'forwardPage' is static again (no dynamic update) - see features in #4511.

16.78.2 Features

- #4511 / Form: URL Forward - mode dynamic computed.

16.78.3 Bug Fixes

- #4512 | SIP URL does not respect anker token '#'- fixed PLUS: L and type _GET Params included in links which contain a SIP (regular links still open).
- #4508 / Form: during Save with FE with 'report'-Note/Values an exception is thrown - report does not expect, to be called without typo3 - but this is the case during save and generating the JSON.
- #4021 / "required" asterik does not handle multi column labels correctly.
- #4423 / Date inputs with readonly: label is grey.
- Empty date might create '2001-00-00'.
- #4504 / Upload Button: required asterik missing after save - seems to be a problem for every element - should be fixed now.

16.79 Version 0.21.0

Date: 10.09.2017

16.79.1 Notes

- The Form-Editor now has two 'requiredParamter' fields: one for 'New' record and one 'Edit'. Existing settings will be automatically copied to both.
- The FormElement-Editor field 'Note' is not anymore a TinyMCE Editor. Instead a regular 'textarea' is used. Main reason are incompatibilities between TinyMCE HTML mode and the neede CR/LF linebreaks needed for 'Report' Syntax in the 'note' column.

16.79.2 Features

- #4431 / FE.type=note: QFQ Report Syntax in 'FE.value' and 'FE.note'.
- #4456 / formModeGlobal=requiredOff - Switches FormElement.mode=required to 'show' for all FE of the current Form.
- #4356 / Form: required parameter - split between 'New' & 'Edit'.

16.80 Version 0.20.0

16.80.1 Changes

- New configuration value EXTRA_BUTTON_INFO_POSITION in config.qfq.ini.

16.80.2 Features

- #4386 Fuer GRC: Optional Info Button bei 'input' wie bei 'textarea' - EXTRA_BUTTON_INFO_POSITION=below.
- #4429 / subrecord: new FE parameter 'subrecordTableCass' - a custom class for the subrecord table might be specified.
- #4428 / subrecord: mode=readonly.
- #4421 / subrecord: column of the sql1 row should go into the edit link - implemented.
- #4399 / Do not render '_pdf' when r:5 or empty string.

16.80.3 Bug Fixes

- #4396 / FE: Justify DATE and TIME in case it's DATETIME on a non primary table.
- #2414 / Deaktivieren von Option 'new' bei subrecord hat keine Folge.
- #4426 / Subrecord: mode=hidden - still shown.
- #4425 / Subrecords: Table head is not wrapped in <thead>.
- #4331 / SQL Statement 'REPLACE' not fired - Keyword missing in list of SQL Keywords.

16.81 Version 0.19.7

16.81.1 Changes

- #4306 / Update Text Subrecord: Please save this record first.

16.81.2 Features

16.81.3 Bug Fixes

- #4278 / Language: Check that language settings are respectet inside of container / pill / fieldset / templateGroup.
- #4310 / Fixed error where custom values wouldn't be saved, nor not found for non pedantic.
- #4311 / Record Lock: expired lock wird nicht geloescht bei form reload.
- #4309 / typeahead: allow free entry.

16.82 Version 0.19.6

16.82.1 Features

- #4299 / HTML Element 'Select': Placeholder.
- Changes to the alert generation and added btn-group for multiple buttons.
- Should only show reload button and be modal when the conflict is mandatory.
- #4144 / Close/New: bei acquireLock=false anschliessend keine Nachfrage ob gespeichert werden soll.
- #4120 / Removed Timeout from Dirty Alert Message.
- #4283 / FE.parameter=emptyMeansNull.

16.82.2 Bug Fixes

- #4281 Prevent save from being clicked multiple times. Save no turns orange when saving.

16.83 Version 0.19.5

16.83.1 Features

- #3790 / Multilanguage: German/ English/ ...

16.83.2 Bug Fixes

- #4274 / ItemList: escape ',' ':'

16.84 Version 0.19.4

16.84.1 Features

- Feature: Form Paste Records - skip columns during copy if they do not exist on the source side.

16.84.2 Bug Fixes

- #4266 / FormElement Type=Editor: value not saved - fixed.
- #4253 / Record Lock not deleted, when window closes without save.

16.85 Version 0.19.3

16.85.1 Changes

- Changing buttons for the dirty Events depending on status.

16.85.2 Bug Fixes

- #4257 / Dynamic update broken - after changing JSON data structure, update load.php has been missed.

16.85.3 Open

- #4253 / Record Lock not deleted when window closes without save.

16.86 Version 0.19.2

16.86.1 Features

- #4250 / autocron: sending mails.
- #4248 / FormElement: TypeAhead fuer den Spaltennamen - Implemented.
- #4144 / Close/New: bei acquireLock=false anschliessend keine Nachfrage ob gespeichert werden soll.
- #4120: Removed Timeout from Dirty Alert Message.

16.87 Version 0.19.1

16.87.1 Features

- #4172 / record locking: Bob tries to delete a record and get 'status=error': Client should disable 'delete' button.
- #4185 / Detect modified record.
- #4143 / New alert removes old alert(s).

- #4173 / Form: User open's a new tab and press close - alert to inform user that he has to close the tab.
- #1930, #3980 / Client: Bei Form Submit den Status 'submit_reason=savelsave,close' mitsenden.
- Implemented: New > Close (save) now closes correctly the current page. Addtional, #1930 has been solved implizit.

16.87.2 Bug Fixes

- Bug #4174 / record locking: error message if delete fails due to record locking.
- Bug: SQL 'CREATE' implemented as a valid command.

16.88 Version 0.19.0

16.88.1 Changes

- bower.json: change bootstrap version number from micro to minor.
- Sip.php: Guarantee that uniqid() is unique at least for the current user.
- Makefile: change installation of phpDocumentor to –alldeps and remove 'phpdoc/'.

16.88.2 Features

- #3881 / Variables: Ex 'keySemId', New 'periodId' (System Store).
- AbstractBuildForm.php: if a datetime / timestamp has the string 'CURRENT_TIMESTAMP' it will be replaced by the current date/time.
- Add new STORE_TYPO3 vars: pageAlias, pageTitle.
- Config.php: cleanup of checking GET variables.
- #3981 / Record Locking.
- Manual.rst: add documentation for record locking.
- Manual.rst: more details about QFQ variables.
- plantuml: sequence diagrams for record locking.

16.88.3 Bug Fixes

- Bug #4158 / Delete Button im Form fehlen die SIP Parameter.
- Bug #4159 / missing htmlspecialchar_decode() for FE.value supplied content.
- Bug Makefile: fixed unwanted removing of whole 'doc' by 'maintainer-clean' - doc nowadays contains QFQ related manually created documentation.
- Bug typeahead.php: An exception catched in typeahead.php has been assigned as array element, instead of a whole array. Fixed.

16.89 Version 0.18.7

16.89.1 Changes

- Makefile: 'make bootstrap' udpates all JS Lib packages. Double npm install removed.

16.89.2 Features

- #3947 / Attack detectect: logout current user (only QFQ, FE User still logged in).
- #3959 / Class _link: implement 'any' Bootstrap glyphicons. New token 'G:' for '_link'.

16.89.3 Bug Fixes

- #3953 / Radio buttons: Auswahl nicht angezeigt, wenn per itemList definiert.
- #3982 / Filename Sanatize: remove spaces. Filename not properbly enclosed by double ticks.

16.90 Version 0.18.6

16.90.1 Features

- #3460 / Report: new column types '_striptags, '_htmlentities', '_+Tag'.

16.91 Version 0.18.5

16.91.1 Features

- QuickFormQuery.php: added function to check if there are copyForm paste records.
- Manual.rst, formEditor.sql: add several links in Form 'FormEditor' to the online documentation. The FormEditor now contains links to the Online Documentation. Add missing explanations: Required Parameter, Forward.

16.91.2 Bug Fixes

- #3912 / templateGroup: max. 5 instances are saved.
- Manual.rst: Fixed missing '{{' and '%' in examples.
- #3925 / templateGroup / non primary / delete records: only one at a time.
- Makefile: Artifactory builds fails at chromedriver - temporary remove npm install of chromedriver.

16.92 Version 0.18.4

16.92.1 Bug Fixes

- #3910 / 'submitButtonText' not shown.

16.93 Version 0.18.3b

16.93.1 Features

- #3906 / Mark required inputs with an asterik.

16.93.2 Bug Fixes

- #3903 / Copy/Paste form: references inside a record are not updated at all.

16.94 Version 0.18.3a

16.94.1 Changes

- Copy / Paste form: take care that there is now Form.id=3. This is now the reserved formId for the 'copyForm'.

  UPDATE *FormElement* SET formId=100 WHERE formId=3 UPDATE *Form* SET *id* = '100' WHERE *Form*.'id' = 3;

16.94.2 Bug Fixes

16.95 Version 0.18.3

16.95.1 Features

- #3899 / Copy/Paste
  - DatabaseUpdate.php: New table Clipboard, New FE.type='paste', New Form.forwardMode='url-sip' - will be applied for 0.18.3.
  - Store.php: New member in STORE_CLIENT 'CLIENT_COOKIE_QFQ' - might be used to identify current user.
  - formEditor.sql: New form 'copyForm'. New table 'Clipboard'.

16.96 Version 0.18.2

16.96.1 Changes

- #3891 / Dynamic Update: Multiple Elements in a row - single show/hide - First implementation: Show / Hide via dynamicUpdate for a second element, wrapped in an own 'col-md' div.

16.96.2 Bug Fixes

- #3875 / FormElement 'extra': Fehler bei neuen Records.
- #3647 / Dynamic Update: Multiple Elements in a row not updated properly.

- #3890 / upload-FormElement: mode 'hide' without effect.
- #3876 / upload-FormElement: verschwindet bei dynamicUpdate.

16.97 Version 0.18.1

- Update unit tests

16.98 Version 0.18.0

16.98.1 Changes

- New defaults to FormElement.type='file' (=Upload).
- *access* = *application/pdf*
- *maxFileSize* = *10485760* (10MB)

16.98.2 Features

- Make the image shown for ExtraButtonInfo configureable. Manual.rst: added documentation for new config.qfq.ini variables GFX_EXTRA_BUTTON_INFO_INLINE, GFX_EXTRA_BUTTON_INFO_BELOW.
- Manual.rst: description added for *extraButtonLock* ,'extraButtonPassword', *extraButtonInfo*. Example on how to reuse custom vars defined in *config.qfq.ini*. Add three more examples to _pdf. How to access local online documentation. Add note about wkhtml only on linux boxes. More wkhtml explanations.
- #3773 / Button: Info / Unlock / ShowPassword: FE_PARAMETER: extraButtonLock / extraButtonPassword / extraButtopnInfo.
- #3769 / Allow specific GET variables longer than SECURITY_GET_MAX_LENGTH. config.php: implemented special handling of GET vars, named with '..._<num>'.
- #3766 / SQL_LOG per tt_content record einstellbar machen. Add *sqlLog* and *sqlLogMode* to QFQ tt-content records. Add mode 'error' and *none* to sqlLogMode. Manual.rst: Added explanations for SQL_LOG, SQL_LOG_MODE, and tt-content pendants sqlLog, sqlLogMode. Update config.qfq.ini to latest attributes.
- Config.php: Set defaults for config.qfq.ini SQL_LOG and SQL_LOG_MODE.
- config.qfq.example.ini: Remove DB_NAME_TEST, Add some details about SQL_LOG, add example for TECHNICAL_CONTACT.
- Session.php: Activate cookie_httponly for QFQ cookies.
- Html2Pdf.php: recode setting of $this->logFile.
- config.qfq.ini: new syntax for SHOW_DEBUG_INFO - [yes,no,auto][,download].
- #3711 / Upload Button opens the camera on a smartphone. New attribute 'capture'.
- Database.php: Add SQL keyword 'DROP'.
- #3706 / Check File Upload: a) mime type, b) max file size. Implemented: file upload check for mime type and max file size.
- New 'pageForward'-Mode: 'url-skip-history'.
- FormEditor:

  - Most radio FEs changed from HTML standard to Bootstrap Design.
  - FE 'feIdContainer' will only be shown if there is at least one FE container element.
  - Use dynamicUpdate to hide/show FE 'forwardPage'.
  - FE 'subrecordOption' shown only for FE.type==subrecord.
  - FE 'checkPattern' shown only for FE.checkType=='pattern|min%'.
  - In form 'FormElement' the FE 'formId' removed - not necessary.
- #3568 / Form: fuer alle Buttons (save, close, new, delete) eine optionale class & text konfigurierbar.
- #3569 / Input Optional '0' unterdruecken.
- #3863 / New config var 'DB_UPDATE' in config.qfq.ini.

16.98.3 Bug Fixes

- #3770 / Attack Delay: merge processing to one codeplace. Config.php: new function attackDetectedExitNow(). Sip.php: replace local sleep(PENALTY_TIME_BROKEN_SIP) with central function attackDetectedExitNow().
- #3768 / log to sql.log: ip, formname, feuser.
- Store.php: Fix problem that an empty SQL_LOG will be added to SYSTEM_PATH_EXT. Logger.php: do nothing if there is no file defined.
- #3751 / download FE_GROUP protected pages failed. Implement additional 'SHOW_DEBUG_INFO = download' to track down problems with 'session forwarding'.
- Allow spaces in value when selection <radio> and <select> tags.
- #3812 / extraButtonInfo (extraButtonLock, extraButtonPassword) - Problems in TemplateGroups.
- #3832 / Dynamic Update und Radio buttons - leerer value ('') kann nicht angewählt werden.
- #3612 / Konflikt typeAheadLdap mit dynamic modesql: inputs with typeahead lacks 'dynamicUpdate' via 'modeSql'.
- #3853 / New > Save: Reload des Forms mit neuer SIP und neu erstellter recordId.
- #3854 / Wrong final page: a) New > Save > Close, b) New > Save > Delete, c) New > New formEditor.sql: update table 'Form.forwardMode' to ('client', 'no', 'url', 'url-skip-history').
- #2337 / Checkbox: checked/unchecked parameters genügen nicht.
- #2542 / FormElement-Typ 'note' funktioniert nicht mit dynamic update.
- #3863 / DB Update Fails: Expected no record, got 2 rows: SHOW TABLE STATUS WHERE Name='Form'.

16.99 Version 0.17.0

16.99.1 Changes

- ALTER TABLE *FormElement* ADD *encode* ENUM('none', 'specialchar') NOT NULL DEFAULT 'specialchar' AFTER *subrecordOption* ;
- UPDATE *FormElement* SET encode='none' WHERE class='native' AND type='editor'

- ALTER TABLE *Form* ADD *escapeTypeDefault* ENUM(‘’, ‘s’, ‘d’, ‘l’, ‘L’, ‘m’, ‘-‘) NOT NULL DEFAULT ‘’ AFTER *permitEdit* ;
- In order to not break functionality of existing forms, it might be necessary (bad for security, good for stability) to leave existing forms untouched: *UPDATE Form SET escapeTypeDefault=’-‘*
- Play formEditor.sql

16.99.2 Features

- New security option *escapeTypeDefault*: will be defined 1) sytem wide in config.qfq.ini, or 2) more specific per Form or 3) individually per variable. The later has priority.
- #3544 / Form: view current form - It’s now possible to direct view a form, which is currently loaded/edited in the FormEditor: Button ‘eye’ near left of button ‘save’.
- #3552 / typeAheadLdapSearchPerToken - webpass kann nicht gleichzeitig nach Vornamen und Nachnamen suchen. Added option typeAheadLdapSearchPerToken to split search value in token and OR-combine every search with the individual tokens.
- Download latest QFQ builds and releases: https://w3.math.uzh.ch/qfq/.
- #3218, #3600 / download.php / export: QFQ is now able to create PDFs and ZIPs on the fly. The sources might be uploaded PDFs or Websites (local or remote) which will be converted to PDFs.
- Implement ‘encode=specialchar’ - new option per FormElement which is now the default for every FormElement.
- Sanatize.php: New function urlDecodeArr(). Decode all _GET vars.
- Implemented max GET parameter lenght. Default: 50.
- Implemented new escape class ‘mysql’ (realEscapeString).
- LICENSE.txt: Add GPLv3.
- Html2Pdf.php: Add SIP support wkhtmltopdf URLs. Move cookies for wkhtmltopdf from commandline arguments to filebased.
- SessionCookie.php: New class to save current cookies in a file.
- Html2Pdf.php: implemented session forwarding to wkhtmltopdf.
- Session.php: introduced close(). This will unlock the current session. Take care on subsequent calls to reopen primary session again.
- Database.php: Set charset to real_escape_string() functions properly. Proxy for mysqli::real_escape_string().
- Implement honeypot variables to detect bots.
- HTML special char encode all URL GET parameter. This can’t be skipped.

16.99.3 Bug Fixes

- Sip.php: Parameter XDEBUG_SESSUIB_START excluded from GET parameter copied to SIP.
- Manual.rst: add libxrender1 to install by using wkhtmltopdf.
- Download.php: Skip ‘pdftk’ if there is only one PDF file to concatenate.
- #3615 / download.php: Das Popup schliesst nicht automatisch bei ZIP, im FF, Warnung in der Console, hourglass wobbles.

- Split PHP 'print.php' in a pure API file 'print.php' and a class 'Html2Pdf.php' - the class will be reused by Download.php.
- #3573 / TypeaheadLdap: Prefetch funktioniert nicht.
- #3547 / FE of type 'note' causes writing of empty fields.
- #3546 / Throw of a UserFormException with wrong parameter. Fixed.
- #3545 / Errormessages via API/JSON not displayed.
- #3536 / a) Datum (datetime / timestamp) werden nicht angezeigt, b) Angezeigte Datumsformat String und aktzeptierte Eingabe matchen nicht.
- #3533 / afterSave: sqlUpdate auf child-record ändert xId von Hauptrecord auf 0.

16.100 Version 0.16

16.100.1 Changes

- Play:

  ALTER TABLE *FormElement* ADD INDEX *feIdContainer* (*feIdContainer*); ALTER TABLE *FormElement* ADD INDEX *ord* (*ord*); ALTER TABLE *FormElement* ADD INDEX *feGroup* (*feGroup*);

  ALTER TABLE *FormElement* ADD *adminNote* TEXT NOT NULL AFTER *note*;

- Play formEditor.sql

16.100.2 Features

- formEditor.sql:
  - Added 'on update current timestamp'.
  - Add three indexes to formEditor.sql.
  - Column FormElement.adminNote added.
  - Change default width on Form for subrecord FormElement.name.
  - Changed input height of 'parameter of FormEditor and FormElementEditor to 8 lines.
- Enlarge placeholder value in FormElement. Old 512, New 2048..
- #3466 / Input Typeahead: optional only allow specified input. Mode: typeAheadPedantic.
- #3465 / Save button: optional 'active after form load': *Form.parameter.saveButtonActive* - if this attribute is set, the save button will be enabled directly on form load.
- #3463 / form.mode=readonly. Make a form complete *readonly*. This can be done statically or dynamically via variable (e.g. SIP).
- #3447 / Icons das man im FrontEnd direkt das gewaehlte FormElement im Formulareditor bearbeiten kann. Add checkbox left. to the 'EditForm'-Button, to toogle the 'FormElemnt'-Icons. Like the 'Form Edit'-Pencil, the 'FormElement Checkbox' is only displayed if the user is logged in BE.
- #3456 / LDAP: with Credentials (e.g. to access 'webpass'). Updated doc for a) config.qfq.ini: LDAP_1_RDN, LDAP_1_PASSWORD, b) Form.parameter|FormElement.parameter: ldapUseBindCredentials.
  - ErrorHandler.php: removed details - the end user should not too many details.

  - FormAction.php, Ldap.php, QuickFormQuery.php: implement 'ldapUseBindCredentials'.
  - Ldap.php: set_error_handler() to catch ldap_bind() problems. Always set LDAP_OPT_PROTOCOL_VERSION=3 - this might cause problems with som LDAP Servers - we will see.

16.100.3 Bug Fixes

- 3509 / SELECT Element: value wird nicht selektiert.
- 3502 / TemplateGroups: Checkboxen werden beim ersten Speichern (insert) nicht geschrieben - ein anschliessendes Update ist ok.
- 3385 / templateGroup: insert/update/delete non primary records.
  - Non primary record leads to a problem that the default values, given as an array, are not replaced by scalar values. fixed.
  - Update doc how to insert/update/delete non primary templateGroup records.
  - Removed $templateGroupIndex - solved implicit by defining a LIMIT on 'slaveId' . Implemented '%D' (one below %d). Implemented FE_SQL_HONOR_FORM_ELEMENTS - reduces unecassary SQL queries.
  - Fill STORE_RECORD during Formload - to read templateGroup records very early. Local copy of *getNativeFormElements()*, new *explodeTemplateGroupElements()*
- TypeAhead.js: Handle <ENTER> key properly.
- #3462 / FormElement.parameter: requiredList not ok for non numeric content. STORE_FORM had been called without 'sanatize class'. Therefore, all non numeric values has been sanatized by default. New: SANATIZE_ALLOW_ALL.
- Corrected error message to use 'itemList' instead of 'itemValues'. Renamed constant too.
- #2542 / FormElement-Typ 'note' funktioniert nicht mit dynamic update. 'Label' and 'note' are fixed - 'value' is still not updated, open.

16.101 Version 0.15

16.101.1 Changes

- Play formEditor.sql.
- Form 'FormElement' failed to display the formtitle of the current form in case of a new FE.
- Updated subrecord in 'Form' for 'FormElements' - columns 'size' and 'sql1' removed and 'dyn' inserted. Play formEditor.sql.
- #3431, Parameter keyword 'typeAheadLdapKeyPrintf' changed to 'typeAheadLdapIdPrintf'.:

```
UPDATE FormElement SET parameter = REPLACE(parameter, 'typeAheadLdapKeyPrintf',
↪'typeAheadLdapIdPrintf')
```

- Size 'placeholder' increased:

```
ALTER TABLE `FormElement` CHANGE `placeholder` `placeholder` VARCHAR( 2048 )␣
↪CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT '';
```

16.101.2 Features

- Introduce new config.qfq.ini setting 'EDIT_FORM_PAGE' - by default set to 'form' - the wrench symbol on every Form will direct to that page. Fix #3420 / Quicklink 'editform' on form: directs to the current T3 page which might be insufficient.
- Form 'subrecord' columns: the default width limit of 20 chars are disabled if 'nostrip' is specifed.
- #3431 / typeAheadSql: columnname 'key' is a reserverd SQL statement - replace by 'id'. Additional the parametername 'typeAheadLdapKeyPrintf' renamed to 'typeAheadLdapIdPrintf'. By using 'id' instead of 'key' the quoting of the columnname is not necessary anoymore.

16.101.3 Bug Fixes

- #3419 / typeAheadSql: Array with only one column or non standard columnnames are not handeld properbly. Detection of missing LIMIT implemented.
- #3425 / Form.parameter, FormElement.parameter: comment handling, trailing & leading spaces Manual.rst: commented handling of 'comment character' and 'escaping of leading/trailing spaces' Support.php: new funtion handleEscapeSpaceComment().
- Evaluate.php: parse all F|FE.parameter via handleEscapeSpaceComment(). A leading '#' or ' ' might be escaped by ''.
- Saving 'extra' FE in STORE_SIP has been done with inappropiate FE_NAME. Correct is the pure FE_NAME, without any extension like recordId. Unessary and broken decoding removed.
- #3426 / Dynamic Update: Inputs loose the new content and shows the old value.
- Through fix #2064 the FE.checkType has not been used anymore. This is fixed now.
- #3433 / templateGroup on primary Record: Values of removed copies are not deleted. The new implementation creates empty fake instances of all copies of templateGroup FormElements. Those are empty. Before save, the submitted form values will be expanded with the empty fake templateGroup FormElements and such empty values will be saved.

16.102 Version 0.14

16.102.1 Changes

- Play formEditor.sql.
- All Form & FormEditor input elements now have a maxlength definition of 0, which means take the column definition value.
- Drop-down list of container assignment:
- Display 'type' ('pill', 'fieldset', 'templategroup') instead of 'class' (always 'container').
- Display 'name' (internal name) instead of 'label' (shown on the website and might not so usefull as 'name' which is nowhere else used than in that drop-down.
- FormElement.placeholder colum width extended to 512:

  ALTER TABLE *FormElement* CHANGE *placeholder placeholder* VARCHAR(512) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT ''

- New class Ldap.php.

16.102.2 Features

- Typeahead for SQL and LDAP Datasources implemented.
- formEditor.sql: Changed width of column FormElement.placeholder from 255 to 512. Removed hardcoded 'size' in FormElement 'placeholder'.
- Character Count: Display a *counter* on input or textarea fields, activated by specifying the formElement.parameter 'characterCountWrap'.
- Evaluate.php: Two new escape options 'l' and 'L'. Backport of ldap_escape() for PHP <5.6. Multiple escaping for one value now possible.
- Manual.rst: add some example for TypeAhead and for saving LDAP value.
- Load foreign values in templatGroups - saving is not implemented yet.
- Manual: Added howto prevent <p>-wrap in TinyMCE.
- TemplateGroup: Add button now disabled if max. number of copies reached.
- #3414 / QuickFormQuery.php: wrap whole form in 'col-md-XX' - User controls the width of an QFQ form.

16.102.3 Bug Fixes

- Dynamic Update has been broken since implementing of 'element-update' (#3180). Now both methods, 'element-update' and 'form-update' should be fine.
- qfq-bs.css.less: Fixed problem with 'typeahead input elements' not expanded to Bootstrap column width. Changed Layout/Design Typeahead drop-down box. Add hoover for the drop-down box with a blue background.
- AbstractBuildForm.php: #3374 - textarea elements now contains 'maxlength' attribute.
- BuildFormBootstrap.php: wrapping of optional 'submitButtonText' now done with the 'per form' values.
- typeahead.php: if there is an exception, the message body is sent as regular 'content' for the drop-down box. At the moment this is the only way to transmit any error messages.
- formEditor.sql: removed all 'maxLength' string values for 'Form' and 'FormElement' forms.
- Save button becomes active if a templateGroup copy is removed.
- #3413 / Form ohne Pill hat kein padding am Rand. Fix: if there are no pills, an additinal col-md-12 will be rendered.

16.103 Version 0.13

16.103.1 Changes

- Play formEditor.sql.
- formEditor.sql:
  - Checktype of *Form.name* restricted to *alnumx* (prior *all*).
  - Changed *access* for Form *form* & '*ormElement* from *always* to *sip*.
- Table *FormElement*:
  - Modified column: *checkType* - new value *numerical*.

ALTER TABLE FormElement MODIFY COLUMN checkType ENUM('alnumx','digit','numerical','email','n
'pattern','allbut','all') NOT NULL DEFAULT 'alnumx'

- Example Report for *forms* extended by a delete button per row.

16.103.2 Features

- print.php: offers 'print page' for any local page - create a PDF on the fly (printout is then browser independent).
  - Install *wkhtmltopdf* on the webserver (http://wkhtmltopdf.org/).
  - In config.qfq.ini setup:

    BASE_URL_PRINT=http://www.../ WKHTMLTOPDF=/opt/wkhtmltox/bin/wkhtmltopdf
- Check and error report if 'php_intl' is missing.
- New Checktype 'allow numerical'.
- Documentation: example for 'radio' with no pre selection.
- #3063 / Radios and checkboxes optional rendered in Bootstrap layout.
- Added 'help-box with-errors'-DIV after radios and checkboxes.
- Respect attribute *data-class-on-change* on save buttons.

16.103.3 Bug Fixes

- #2138 / digit sanitize: new class 'numerical' implemented.
- Fixed recursive thrown exception.
- #2064 / search of a default value for a non existing tablecolumn returns 'false'.
  - Fixed setting of STORE_SYSTEM / showDebugInfo during API call.
- #2081, #3180 / Form: Label & note - update via *DynamicUpdate*.
- #3253 / if there is no STORE_TYPO3 (calls through .../api/ like save, delete, load): use SIP / CLIENT_TYPO3VARS.
- qfq-bs.css:
  - Alignment of checkboxes and radios optimized.
  - CSS class 'qfq-note' for 'notes' (third column in a form).

16.104 Version 0.12

16.104.1 Changes

- Table 'FormElement' * New column: rowLabelInputNote:

  ALTER TABLE *FormElement* ADD *rowLabelInputNote* set('row','label','/label','input','/input','note','/note','/row') NOT NULL DEFAULT 'row,label,/label,input,/input,note,/note,/row' AFTER *bsNoteColumns* ;
  - Modified column: 'type' - new value 'templateGroup':

ALTER TABLE *FormElement* CHANGE *type type* ENUM('checkbox', 'date', 'datetime', 'dateJQW', 'datetimeJQW', 'extra', 'gridJQW', 'text', 'editor', 'time', 'note', 'password', 'radio', 'select', 'subrecord', 'upload', 'fieldset', 'pill', 'templateGroup', 'beforeLoad', 'beforeSave', 'beforeInsert', 'beforeUpdate', 'beforeDelete', 'afterLoad', 'afterSave', 'afterInsert', 'afterUpdate', 'afterDelete', 'sendMail') CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'text'

- formEditor.sql: Added HTML 'placeholder' in FormEditor for bs*Columns.
  - PLAY 'formEditor.sql'.
- User Input will be UTF8 normalized.
  - INSTALL 'php5-intl' or 'php7.0-intl' on Webserver.
- Add globalize.js to be included. Needed by jqx-all.js.
  - UPDATE EXISTING TypoScript TEMPLATES of QFQ Installation.
- Name of variable '_filename' (used in field 'parameter') has changed. Old: '_filename', New: 'filename'.
  - UPDATE *FormElement* SET parameter = REPLACE(parameter, "_filename", "filename")

16.104.2 Features

- User input will be UTF8 normalized.
- config.qfq-ini:
  - New configuration values: FORM_BS_LABEL_COLUMNS / FORM_BS_INPUT_COLUMNS / FORM_BS_NOTE_COLUMNS.
  - **Comment empty variables - the new default setting is, that empty parameter in config.qfq.ini means EMPTY** (=parameter is set and will not be overwritten by internal default), not UNDEFINED (overwritten by internal default).
- FileUpload:
  - Implemented new Formelement.parameter: fileReplace=always - will replace existing files.
  - Multiple / Advanced Upload: new logic implements slaveId, sqlInsert, sqlUpdate, sqlDelete.
- FormElement.parameter: sqlBefore / sqlAfter fired during 'Form' save for action elements.
- STORE FORM: variable 'filename' moved to STORE VAR - sanatize class needs no longer specified.
- STORE VAR: two new variables 'filename' and 'fileDestination' valid during processing of current upload FormElement.
- Default store priority list changed. Old: 'FSRD', New: 'FSRVD'.
- CODING.md: update doc for FormElement 'upload' and general 'Form' rendering & save (recursive rendering).
- User manual:
  - Described form layout options: description for bsLabelColumn, bsInputColumn, bsNoteColumn.
  - Update 'file-upload' doc.
  - Described 3 examples for upload forms.
- Administrator manual:
  - Add description page.meta...
- New FormElement (type= 'container') added: 'templateGroup'.

  - FormElement.parameter.tgAddClass | tgAddText | tgRemoveClass | tgRemoveText | tgClass.
  - FormElement.maxSize: max number of duplicates.
  - #3230 / templateGroup: margin between copies. 'tgClass' implemented.
- Native FormElements:
  - FormElement.parameter.htlmlBefore|htmlAfter - add the specified HTML code before or after the element (outside of any wrapping).
  - #3224, #3231 / Html Tag <hr> als FormElement. >> htmlBefore | htmlAfter.
  - FormElement.parameter.wrapLabel | wrapInput | wrapAfter | wrapRow - if specified, any default wrapping is omitted.
  - FormElement.bsNoteColumns | bsInputColumns | bsNoteColumns - a '0' will suppress the whole rendering of the item.
  - FormElement.rowLabelInputNote - switch on/off rendering of the corresponding system wrapping items.
- #3232 / Define custom 'on-change' color - used for the save button: Form.parameter.buttonOnChangeClass=...
- Form.parameter & FormElement.parameter: Lines starting with '#' are treated as comments and will not be parsed.

16.104.3 Bug fixes

- User manual:
  - Fixed double include of validator.js in T3 Typoscript template example.
  - Fixed wrong store name SYSTEM: S > Y.
  - Fixed wrong STORE_FORM variable names.
  - Reformat FormElement.parameter description.
  - Styling errors fixed.
- Use of 'decryptCurlyBraces()' to get better error messages.
- Skip unwanted parameter expansion during save.
- Fixed bug with uninitialized FE_SLAVE_ID.
- formEditor.sql: * The defintion as 'editor' (not text) for FormElement 'note' has been lost - reinserted. * Fixed problem while playing SQL query - deleting old FormElements of Formeditor deleted also FormElements of other forms.
- #3066 / help-text with-error - CSS class 'hidden' will be rendered by default (as long there is no error).
- Labels are skipped, if FormElement.bsLabelColumns=0.
- Respect attribute *data-class-on-change* on save buttons.

16.105 Version 0.11

16.105.1 Features

- Added STORE_BEFORE, #3146 - Mainly used to compare old and new values during a form 'save' action.
- Added 'best practice' for defining and using of 'Central configure values' in UserManual.

- Added accent characters to sanatize class 'alnumx', #3183.
- Set default all QFQ send mails to 'auto-submit'.
- Added possibility to customize error messages ('data-pattern-error', 'data-rquired-error', 'data-match-error', 'data-error') if validation fails. Customization can be done on global level (config.qfq.ini), per Form or per FormElement.
- *FormElement*: Double an input element and validate that the input match: FormElement.parameter.retype=1.
- Autofocus in Forms is now supported. By default the first Input Element receives the focus. Can be customized.
- Added a timestamp in shown exceptions. Usefull for screenshots, send by customer, to find the problem in SQL logfiles.

16.105.2 Bug fixes

- Fixed missing docutmentation for FormElement 'note'.
- Failed SQL queries will now always be logged, even if they do not modify some data.

16.106 Version 0.10

16.106.1 Features

- Implemented Parameter 'extraDeleteForm' for 'forms' and 'subrecords'. Update doc.

16.106.2 Bug fixes

- Suppress rendering of form title during a 'delete' call. No one will see it and required parameters are not supplied.
- In case of broken SQL queries, print them in ajax error message.
- Remove parameter 'table' from Delete SIP URLs. ToolTip updated.

16.107 Version 0.9

16.107.1 Features

- FormEditor: * design update - new default background color: grey. * per form configureable background colors. * Optional right align of all form element labels. * Added config.qfq.ini values CSS_CLASS_QFQ_FORM_PILL, CSS_CLASS_QFQ_FORM_BODY, CSS_CLASS_QFQ_CONTAINER.

16.107.2 Bug fixes

- BuildFormBootstrap.php: added new class name 'qfq-label' to form labels - needed to assign 'qfq-form-right' class. Changed wrapping of formelements from 'col-md-8' (wrong) to 'col-md-12'.
- QuickFormQuery.php: Set default for new F_CLASS_PILL & F_CLASS_BODY.
- formEditor.sql: New default background color for formElements is blue.

- qfq-bs.css.less: add classes qfq-form-pill, qfq-form-body, form-group (center), qfq-color-. . . , qfq-form-right.
- Index.rst: Add note to hierachy chars. Fixed uncomplete doc to a) bs*Columns, showButton. Add classPill, classBody. Rewrote form.paramter.class.
- QuickFormQuery.php: Button save/ close/ delete/ new - align to right border of form.
- UsersManual/index.rst: renamed chapter for formelements. Cleanup formelement types. Wrote chapter 'Detailed concept'.
- QuickFormQuery.php, FormAction.php: '#2931 / afterSave Hauptrecord xId nicht direkt verfügbar' - load master record again, after 'action'-elements has been processed.
- UsersManual/index.rst: Startet FAQ section.
- config.qfq.example.ini: Added comment where to save config.qfq.ini.
- UsersManual/index.rst: Rewrite of 'action'-FormElement definition.
- #2739 / beforeDelete / afterDelete.
- PROTOCOL.md: update 'delete' description.
- delete.php: fixed unwanted loose of MSG_CONTENT.
- Report.php: Fixed double '&&' in building UrlParam.
- FormAction.php: In case of 'AFTER_DELETE', do not try to load primary record - that one is already deleted.
- Sip.php: Do not skip SIP_TARGET_URL as parameter for the SIP.
- #3001 / Report: delete implementieren.
- Index.rst, Constants.php: reverted parameter '_table' in delete links back to 'table' - Reason: 'form' needs to be 'form' (instead of '_form') due to many used places already.
- Sip.php: move SIP_TARGET_URL back to stored inside SIP - it's necessary for 'delete'-links.
- Report.php, Constants.php: Remove code to handle unecessary 'p:' tag for delete links.
- Link.php: Check paged / Paged that the parameter r, table and form are given in the right combination.
- Link.php, Report.php: New '_link' token 'x'. '_paged' and '_Paged' are rendered via Link() class, Link() class now supports delete links.
- QuickFormQuery.php: for modeForm='Form Delete' the 'required param' are not respected - this makes sense, cause these parameters typically filled in newly created records.
- #3076 / Delete Button bei Subrecords erzeugt sporadisch Javascript Exceptions (Webkit: Chrome / Vivaldi) - kein loeschen moeglich.

CHAPTER 17

License

- QFQ is licensed under GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007

17.1 Software distributed together with QFQ

- Bootstrap - http://getbootstrap.com
- Chart.js - https://github.com/nnnick/Chart.js.git
- Fabric.js - https://github.com/fabricjs/fabric.js/tree/master
- Font Awesome - https://github.com/FortAwesome/Font-Awesome
- jQuery - http://jquery.com
- jQWidgets - https://www.jqwidgets.com
- PhpSpreadsheet - https://github.com/PHPOffice/PhpSpreadsheet
- sendEmail - https://github.com/mogaal/sendemail
- Tablesorter - https://mottie.github.io/tablesorter/docs/index.html
- TinyMCE - https://github.com/tinymce/tinymce
- Twig - https://twig.symfony.com
- Twitter typeahead JS - https://twitter.github.io/typeahead.js/
- bootstrap-validator.js - https://github.com/1000hz/bootstrap-validator
- Event Emitter - https://git.io/ee
- FullCalendar - https://fullcalendar.io/

CHAPTER 18

Sitemap

CHAPTER 19

Searching Documentation

This page was copied from: https://docs.readthedocs.io/en/stable/guides/searching-with-readthedocs.html

Read the Docs uses /server-side-search to power our search. This guide explains how to add a "search as you type" feature to your documentation, and how to use advanced query syntax to get more accurate results.

You can find information on the search architecture and how we index documents in our Search docs.

Table of contents

19.1 Search query syntax

Read the Docs uses the Simple Query String feature from Elasticsearch. This means that as the search query becomes more complex, the results yielded become more specific.

19.1.1 Exact phrase search

If a query is wrapped in " (double quotes), then only those results where the phrase is exactly matched will be returned.

Example queries:

- https://docs.rea>dthedocs.io/?rtd_search=%22custom%20css%22

- https://docs.readthedocs.io/?rtd_search=%22adding%20a%20subproject%22
- https://docs.readthedocs.io/?rtd_search=%22when%20a%20404%20is%20returned%22

19.1.2 Exact phrase search with slop value

`~N` (tilde N) after a phrase signifies slop amount. It can be used to match words that are near one another.

Example queries:

- https://docs.readthedocs.io/?rtd_search=%22dashboard%20admin%22~2
- https://docs.readthedocs.io/?rtd_search=%22single%20documentation%22~1
- https://docs.readthedocs.io/?rtd_search=%22read%20the%20docs%20story%22~5

19.1.3 Prefix query

`*` (asterisk) at the end of any term signifies a prefix query. It returns the results containing the words with specific prefix.

Example queries:

- https://docs.readthedocs.io/?rtd_search=API%20v*
- https://docs.readthedocs.io/?rtd_search=single%20v*%20doc*
- https://docs.readthedocs.io/?rtd_search=build*%20and%20c*%20to%20doc*

19.1.4 Fuzzy query

`~N` after a word signifies edit distance (fuzziness). This type of query is helpful when the exact spelling of the keyword is unknown. It returns results that contain terms similar to the search term as measured by a Levenshtein edit distance.

Example queries:

- https://docs.readthedocs.io/?rtd_search=reedthedcs~2
- https://docs.readthedocs.io/?rtd_search=authentation~3
- https://docs.readthedocs.io/?rtd_search=configurtion~1

19.1.5 Build complex queries

The search query syntaxes described in the previous sections can be used with one another to build complex queries.

For example:

- https://docs.readthedocs.io/?rtd_search=auto*%20redirect*
- https://docs.readthedocs.io/?rtd_search=abandon*%20proj*
- https://docs.readthedocs.io/?rtd_search=localisation~3%20of%20doc*