TYPO3 CMS 7 LTS – What's New

TSconfig & TypoScript

Created by:
Patrick Lobacher and Michael Schams

10/November/2015

TYPO3

Introduction

TYPO3 CMS 7 LTS – What's New

The following slides focus on a specific topic. Depending on your role, the following topics might also be important for you:

	BE User Interface	TypoScript	In-Depth Changes	Extbase/Fluid	Deprecated/Removed	Sys.Administration
Editors	X					
Integrators		X	X		X	
Developers			X	X	X	
SysAdmins						X

Download all versions of the **What's New Slides** from typo3.org

TSconfig & TypoScript

TSconfig & TypoScript

TypoScript is used to define information in a hierarchical structure to *configure* a TYPO3 CMS instance. Many properties and options have been added, changed, marked deprecated or removed in TYPO3 CMS 7 LTS.
Open Graph attributes are now supported out-of-the-box and the integrity of externally hosted JavaScript files can be verified by using a SRI hash, to name just two new features.
The slides **Deprecated/Removed Functions** provide an overview of removed TypoScript options.

TSconfig & TypoScript

TSConfig Available to Link Checkers

- TSconfig configuration is read
  - either from the backend (if Linkvalidator is used)
  - or from the scheduler task configuration
- Example: TSconfig, which can be read by Linkchecker:

```
mod.linkvalidator.mychecker.myvar = 1
```

- TSconfig is then available as `$this->tsConfig`

TSconfig & TypoScript

Linkcheck: Report Deleted Records

- In TYPO3 CMS < 7.0, linkhandler warned about links to non-existing or deleted records only
- Since TYPO3 CMS >= 7.0, the following TSconfig setting enables a warning, if links point to disabled records:

```
mod.linkvalidator.linkhandler.reportHiddenRecords = 1
```

TSconfig & TypoScript

RTE: Multiple CSS Classes Per Style

- Modern frameworks such as Twitter Bootstrap require multiple CSS classes per HTML tag

  For example: `<a class="btn btn-danger">Alert</a>`
- Multiple CSS classes are now supported, which means, editors need to select one style only

```
RTE.classes.[ *classname* ] {
  .requires = list of CSS classes
}
```

TSconfig & TypoScript

RTE: Configure CSS Class As Not-Selectable

- It is now possible to configure CSS classes as "not-selectable"

```
// value "1" means, class is selectable
// value "0" makes it not-selectable
RTE.classes.[ *classname* ] {
  .selectable = 1
}
```

TSconfig & TypoScript

RTE: Include Multiple CSS Files

- It is now possible to include multiple CSS files

```
RTE.default.contentCSS {
  file1 = fileadmin/rte_stylesheet1.css
  file2 = fileadmin/rte_stylesheet2.css
}
```

- Without defining any CSS stylesheet files the default is:
  `typo3/sysext/rtehtmlarea/res/contentcss/default.css`

TSconfig & TypoScript

Exception Handling When cObjects Are Rendered (1)

- In TYPO3 CMS < 7.0, if an error occurred during the rendering process of content objects (e.g. `USER`), the error broke the whole frontend
- Since TYPO3 CMS >= 7.0, an exception handling has been implemented, which allows the display of a message instead of the failed cObject

TSconfig & TypoScript

Exception Handling When cObjects Are Rendered (2)

```
# default exception handler (activated in context "production")
config.contentObjectExceptionHandler = 1

# configuration of a class for the exception handling
config.contentObjectExceptionHandler =
  TYPO3\CMS\Frontend\ContentObject\Exception\ProductionExceptionHandler

# customised error message (show random error code)
config.contentObjectExceptionHandler.errorMessage = Oops an error occurred. Code: %s

# configuration of exception codes, which are not dealt with
tt_content.login.20.exceptionHandler.ignoreCodes.10 = 1414512813

# deactivation of exception handling for a specific plugins or content objects
tt_content.login.20.exceptionHandler = 0

# ignoreCodes and errorMessage can be configured globally...
config.contentObjectExceptionHandler.errorMessage = Oops an error occurred. Code: %s
config.contentObjectExceptionHandler.ignoreCodes.10 = 1414512813

# ...or locally for individual content objects
tt_content.login.20.exceptionHandler.errorMessage = Oops an error occurred. Code: %s
tt_content.login.20.exceptionHandler.ignoreCodes.10 = 1414512813
```

TSconfig & TypoScript

StdWrap for `page.headTag`

- TypoScript setting `page.headTag` has `stdWrap` functionality now

```
page = PAGE
page.headTag = <head>
page.headTag.override = <head class="special">
page.headTag.override.if {
     isInList.field = uid
     value = 24
}
```

TSconfig & TypoScript

Include JavaScript files asynchronously

- JavaScript files can be loaded asynchronously

```
page {
  includeJS {
    jsFile = /path/to/file.js
    jsFile.async = 1
  }
}
```

- This affects:
  - `includeJSlibs` / `includeJSLibs`
  - `includeJSFooterlibs`
  - `includeJS`
  - `includeJSFooter`

TSconfig & TypoScript

HMENU item selection via `additionalWhere`

- TypoScript cObject `HMENU` features a new property `additionalWhere`
- This allows for a more specific database query (e.g. filtering)
- Example:

```
lib.authormenu = HMENU
lib.authormenu.1 = TMENU
lib.authormenu.1.additionalWhere = AND author!=""
```

TSconfig & TypoScript

Additional properties for `HMENU` browse menus

- Two new properties for cObject `HMENU` (option "`special=browse`") to select menu items more fine-grained:
  - `excludeNoSearchPages`
  - `includeNotInMenu`
- Example:

```
lib.browsemenu = HMENU
lib.browsemenu.special = browse
lib.browsemenu.special.excludeNoSearchPages = 1
lib.browsemenu.includeNotInMenu = 1
```

TSconfig & TypoScript

Multiple HTTP headers

- HTTP headers can be set as an array (`config.additionalHeaders`)
- This allows for the configuration of multiple headers at the same time

```
config.additionalHeaders {
  10 {
    # header string
    header = WWW-Authenticate: Negotiate

    # (optional) replace previous headers with the same name (default: 1)
    replace = 0

    # (optional) force HTTP response code
    httpResponseCode = 401
  }
  # set second additional HTTP header
  20.header = Cache-control: Private
}
```

TSconfig & TypoScript

Option "auto" added for `config.absRefPrefix`

- TypoScript setting `config.absRefPrefix` can be used to allow URL rewriting. As an alternative to `config.baseURL` (to configure a specific domain), `absRefPrefix` can detect the site root automatically:

  ```
  config.absRefPrefix = auto

  # ...instead of:
  [ApplicationContext = Production]
  config.absRefPrefix = /

  [ApplicationContext = Testing]
  config.absRefPrefix = /my_site_root/
  ```

  Note: The new option is also safe for multi-domain environments to avoid duplicate caching mechanism.

TSconfig & TypoScript

Two-letter ISO code for `sys_language` (1)

- The handling of languages is done by records stored in DB table `sys_language`, which are usually referenced via `sys_language_uid`
- In TYPO3 CMS 7.1, the ISO 639-1 two-letter code has been introduced:
  - New DB field: `sys_language.language_isocode`
  - New TypoScript option: `sys_language_isocode`

Note: **ISO 639** is a set of standards by the International Organization for Standardization. List of ISO 639-1 codes is available at:
http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

TSconfig & TypoScript

Two-letter ISO code for sys_language (2)

- Example:

```
# Danish by default
config.sys_language_uid = 0
config.sys_language_isocode_default = da

[globalVar = GP:L = 1]
  # ISO code stored in table sys_language (uid 1)
  config.sys_language_uid = 1
  # overwrite ISO code as required
  config.sys_language_isocode = fr
[GLOBAL]

page.10 = TEXT
page.10.data = TSFE:sys_language_isocode
page.10.wrap = <div class="main" data-language="|">
```

TSconfig & TypoScript

Custom TypoScript Conditions in Backend

- Support of custom conditions for the **frontend** has been introduced in TYPO3 CMS 7.0 already
- Since TYPO3 CMS 7.1, it is also possible to use custom conditions in the **backend**
- The condition must be derived from `AbstractCondition` and implement method `matchCondition()`
- Example usage in TypoScript:

```
[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition]

[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition = 7]

[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition = 7, != 6]

[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition = {$mysite.myconstant}]
```

TSconfig & TypoScript

Customize icons via PageTSconfig

- Value/label pairs of select fields can be configured by PageTSconfig option `addItems` already
- It is also possible to influence the **icon** of these fields now
  - Option 1: by using `addItems` and sub-property `.icon`
  - Option 2: by using `altIcons` (all items in general)
- Example:

```
TCEFORM.pages.doktype.addItems {
  10 = My Label
  10.icon = EXT:t3skin/icons/gfx/i/pages.gif
}
TCEFORM.pages.doktype.altIcons {
  10 = EXT:myext/icon.gif
}
```

TSconfig & TypoScript

Extend element browser with mount points

- New UserTSconfig option `.append` allows administrators to **add** mount points, rather than replacing the configured database mount points of the user
- Example:

```
options.pageTree.altElementBrowserMountPoints = 20,31
options.pageTree.altElementBrowserMountPoints.append = 1
```

TSconfig & TypoScript

Label override of checkboxes and radio buttons

- Labels of radio buttons and checkboxes can be overwritten now
- Example:

```
// field with a single checkbox (use ".default")
TCEFORM.pages.hidden.altLabels.default = new label
TCEFORM.pages.hidden.altLabels.default = LLL:path/to/languagefile.xlf:individualLabel

// field with multiple checkboxes (0, 1, 2, 3...)
TCEFORM.pages.l18n_cfg.altLabels.0 = new label of first checkbox
TCEFORM.pages.l18n_cfg.altLabels.1 = new label of second checkbox
TCEFORM.pages.l18n_cfg.altLabels.2 = new label of third checkbox
...
```

TSconfig & TypoScript

Miscellaneous (1)

- Width and height of the Element Browser can be configured using UserTSconfig:

  ```
  options.popupWindowSize = 400x900
  options.RTE.popupWindowSize = 200x200
  ```

- PageTSconfig: new RTE configuration property can be used to configure a default target for links of a given type:

  ```
  buttons.link.[type].properties.target.default
  ```

  Where `[type]` can be `page`, `file`, `url`, `mail` or `spec`
  (extensions may provide further types)

TSconfig & TypoScript

Miscellaneous (2)

- Section headlines of indexed search results are links by default. It is now possible to disable these links and display sections as simple texts

```
plugin.tx_indexedsearch.linkSectionTitles = 0
```

- `getData` can access `field` data now (not only arrays such as `GPVar` and `TSFE`):

```
10 = TEXT
10.data = field:fieldname|level1|level2
```

- TypoScript setting `config.pageTitle` has `stdWrap` functionality now

```
# make value of <title> upper case
page = PAGE
page.config.pageTitle.case = upper
```

TSconfig & TypoScript

Flexible Preview URL Configuration (1)

- It is now possible to configure the preview link generated for the "save & view" button in the backend.
- A common use case is to have previews for blog or news records, but you can also define different preview pages for normal content elements.

```
TCEMAIN.preview {
  <table name> {
    previewPageId = 123
    useDefaultLanguageRecord = 0
    fieldToParameterMap {
      uid = tx_myext_pi1[showUid]
    }
    additionalGetParameters {
      tx_myext_pi1[special] = HELLO
    }
  }
}
```

TSconfig & TypoScript

Flexible Preview URL Configuration (2)

- `previewPageId`:
  UID of the page to use for preview
  (if this setting is omitted the current page will be used)
- `useDefaultLanguageRecord`:
  defines if translated records will use the UID of the default record
  (this is activated by default, value: 1)
- `fieldToParameterMap`:
  a mapping which allows to select fields of the record to be included as GET-parameters
- `additionalGetParameters`:
  allows to add arbitrary GET-parameters and even to override others

TSconfig & TypoScript

RTE Configuration: Default Target

- RTE configuration property can be used in PageTSconfig to configure a default target for links of a given type

  `buttons.link.[` *type* `].properties.target.default = ...`

- Possible link types are:
  (further types may be provided by extensions)
  - `page`
  - `file`
  - `url`
  - `mail`
  - `spec`

TSconfig & TypoScript

Strip Empty HTML Tags in HTMLparser

- A new functionality has been implemented in the HTMLparser that allows the stripping of empty HTML tags

```
stdWrap {
   // this removes all empty HTML tags
   HTMLparser.stripEmptyTags = 1
   // this removes empty h2 and h3 tags only
   HTMLparser.stripEmptyTags.tags = h2, h3
}

RTE.default.proc.entryHTMLparser_db {
   stripEmptyTags = 1
   stripEmptyTags.tags = p
   stripEmptyTags.treatNonBreakingSpaceAsEmpty = 1
}
```

Note: HTMLparser strips all unknown tags by default.
Therefore it might be useful to retain these:
`HTMLparser.keepNonMatchedTags = 1`

TSconfig & TypoScript

Miscellaneous

- New property `buttons.abbreviation.removeFieldsets` may be used in PageTSconfig to configure the abbreviation dialog

```
# Possible values are:
# acronym, definedAcronym, abbreviation, definedAbbreviation
buttons.abbreviation.removeFieldsets = acronym,definedAcronym
```

- Property `inlineLanguageLabel` of object `PAGE` can handle `LLL:` references now

TSconfig & TypoScript

New stdWrap Function `strtotime`

- New TypoScript stdWrap property `strtotime` allows for conversion of formatted dates to Unix timestamps, e.g. to perform date calculations
- Valid values are `1` or any time string that is used as the first argument of the PHP function `strtotime()`

```
date_as_timestamp = TEXT
date_as_timestamp {
  value = 2015-04-15
  strtotime = 1
}

next_weekday = TEXT
next_weekday {
  data = GP:selected_date
  strtotime = + 2 weekdays
  strftime = %Y-%m-%d
}
```

TSconfig & TypoScript

`GPmerged` in Conditions

- Using GP in TypoScript conditions only returns the POST variable, if the request contains both, POST and GET variables
- New option `GPmerged` merges both methods and returns the result

```
[globalVar = GPmerged:tx_demo|foo = 1]
  page.90 = TEXT
  page.90.value = DEMO
[global]
```

TSconfig & TypoScript

New Options for `stdWrap.case`

- Options `uppercamelcase` and `lowercamelcase` have been added to `stdWrap.case`
- Example:

```
tt_content = CASE
tt_content {
  key.field = CType
  my_custom_ctype =< lib.userContent
  my_custom_ctype {
    file = EXT:site_base/Resources/Private/Templates/SomeOtherTemplate.html
    settings.extraParam = 1
  }
  default =< lib.userContent
  default {
    file = TEXT
    file.field = CType
    file.stdWrap.case = uppercamelcase
    file.wrap = EXT:site_base/Resources/Private/Templates/|.html
  }
}
```

TSconfig & TypoScript

Property `integrity` Added for JavaScript Files (1)

- Property `integrity` has been added to JavaScript file inclusions in order to specify a SRI hash to allow resource verification (SRI: Sub-Resource Integrity, see next slide)
- This affects the TypoScript PAGE properties `page.includeJSLibs`, `page.includeJSFooterlibs`, `includeJS` and `includeJSFooter`
- Example:

```
page {
  includeJS {
    jQuery = https://code.jquery.com/jquery-1.11.3.min.js
    jquery.external = 1
    jQuery.disableCompression = 1
    jQuery.excludeFromConcatenation = 1
    jQuery.integrity = sha256-7LkWEzqTdpEfELxcZZlS6wAx5Ff13zZ83lYO2/ujj7g=
  }
}
```

TSconfig & TypoScript

Property `integrity` Added for JavaScript Files (2)

- SRI is a W3C specification that allows web developers to ensure that resources hosted on third-party servers have not been tampered with
- Generate integrity hashes:
    - Option 1: https://srihash.org
    - Option 2: use the following shell command

```
cat FILENAME.js | openssl dgst -sha256 -binary | openssl enc -base64 -A
```

- Read more:
    - http://www.w3.org/TR/SRI/

TSconfig & TypoScript

Data-Provider for Backend Layouts (1)

- It is now possible to define backend layouts via page TSconfig and also store them in files. For example:

```
mod {
  web_layout {
    BackendLayouts {
      exampleKey {
        title = Example
        config {
          backend_layout {
            colCount = 1
            rowCount = 2
            rows {
              1 {
                columns {
                  1 {
                    name = LLL:EXT:frontend/ ... /locallang_ttc.xlf:colPos.I.3
                    colPos = 3
                    colspan = 1
                  }
                }
              }
[...]
```

TSconfig & TypoScript

Data-Provider for Backend Layouts (2)

- (continued)

```
[...]
       2 {
        columns {
         1 {
          name = Main
          colPos = 0
          colspan = 1
         }
        }
       }
      }
     }
    }
    icon = EXT:example_extension/Resources/Public/Images/BackendLayouts/default.gif
   }
  }
 }
}
```

TSconfig & TypoScript

Meta Tags Extended

- Option `page.meta` supports Open Graph attribute names now

```
page {
  meta {
    X-UA-Compatible = IE=edge,chrome=1
    X-UA-Compatible.attribute = http-equiv
    keywords = TYPO3
    # <meta property="og:site_name" content="TYPO3" />
    og:site_name = TYPO3
    og:site_name.attribute = property
    description = Inspiring people to share
    og:description = Inspiring people to share
    og:description.attribute = property
    og:locale = en_GB
    og:locale.attribute = property
    og:locale:alternate {
      attribute = property
      value.1 = fr_FR
      value.2 = de_DE
    }
    refresh = 5; url=http://example.com/
    refresh.attribute = http-equiv
  }
}
```

TSconfig & TypoScript

`languageField` Set by Default

- TypoScript option `select` (used in cObject CONTENT for example) required to set `languageField` explicitly
- This is not required anymore, as the setting is now fetched from the TCA information structure automatically

```
config.sys_language_uid = 2
page.10 = CONTENT
page.10 {
  table = tt_content
  select.where = colPos=0

  # the following line is not required anymore:
  #select.languageField = sys_language_uid

  renderObj = TEXT
  renderObj.field = header
  renderObj.htmlSpecialChars = 1
}
```

TSconfig & TypoScript

Individual Content Caching

- Caching of content parts using `stdWrap.cache` exists since TYPO3 CMS 4.7. The cache property is now available for all cObjects and `stdWrap` support has been added to key, lifetime and tags

```
page = PAGE
page.10 = COA
page.10 {
  cache.key = coaout
  cache.lifetime = 60
  #stdWrap.cache.key = coastdWrap
  #stdWrap.cache.lifetime = 60
  10 = TEXT
  10 {
    cache.key = mycurrenttimestamp
    cache.lifetime = 60
    data = date : U
    strftime = %H:%M:%S
    noTrimWrap = |10: | |
  }
[...]
```

```
[...]
  20 = TEXT
  20 {
    data = date : U
    strftime = %H:%M:%S
    noTrimWrap = |20: | |
  }
}
```

TSconfig & TypoScript

Count Elements in a List

- A new property `returnCount` has been added to the stdWrap property `split`
- This allows to count the number of elements in a comma-separated list
- The following code returns `9` for example:

```
1 = TEXT
1 {
  value = x,y,z,1,2,3,a,b,c
  split.token = ,
  split.returnCount = 1
}
```

TSconfig & TypoScript

Sort Order of Tables in List View

- New TSconfig option `mod.web_list.tableDisplayOrder` has been added to the "List" module
- With this option, the order in which tables are displayed is configurable
- Keywords `before` and `after` can be used to specify an order relative to other table names

Syntax:

```
mod.web_list.tableDisplayOrder {
  <tableName> {
    before = <tableA>, <tableB>, ...
    after = <tableA>, <tableB>, ...
  }
}
```

For example:

```
mod.web_list.tableDisplayOrder {
  be_users.after = be_groups
  sys_filemounts.after = be_users
  pages_language_overlay.before = pages
  fe_users.after = fe_groups
  fe_users.before = pages
}
```

TSconfig & TypoScript

Content-Language in HTTP Header

- HTTP header `Content-language: XX` is sent by default, where "XX" is the ISO code of the `sys_language_content` configuration
- By using `config.disableLanguageHeader = 1`, this feature can be disabled (do not send the `Content-language` header at all)

TSconfig & TypoScript

Recursive Option for File Collections

- Folder-based file collections have an option to fetch all files recursively in the given folder now
- The option is also available in the TypoScript Object FILES

```
filecollection = FILES
filecollection {
  folders = 1:images/
  folders.recursive = 1
  renderObj = IMAGE
  renderObj {
    file.import.data = file:current:uid
  }
}
```

TSconfig & TypoScript

Extension `.ts` for Static Templates

- In TYPO3 CMS < 7.4, only the following file names are allowed as static TypoScript templates:
  - `constants.txt`
  - `setup.txt`
  - `include_static.txt`
  - `include_static_files.txt`
- For `constants` and `setup`, file extension `.ts` is also allowed now
- In this context, `.ts` is prioritised over `.txt`

TSconfig & TypoScript

Save & view Button

- Button "save & view" is configurable via TSconfig now
- TSconfig `TCEMAIN.preview.disableButtonForDokType` accepts a comma-separated list of "doktypes"
- Default value is "254, 255, 199" (which is: Storage Folder, Recycler and Menu Separator)
- As a consequence, the "save & view" button is not shown in folders and recycler pages by default anymore

TSconfig & TypoScript

stdWrap for `treatIdAsReference`

- For object `getImgResource` the option `treatIdAsReference` exists, which can be used to define that UIDs are treated as UIDs of `sys_file_reference` rather than `sys_file`.
- Option `treatIdAsReference` received stdWrap functionality now

TSconfig & TypoScript

Introducing Data Processors

- The following Data Processors have been introduced, which allow for a flexible processing of comma-separated lists, arrays, files, etc.:
  - `SplitProcessor`
  - `CommaSeparatedValueProcessor`
  - `FilesProcessor`
  - `GalleryProcessor`
  - `DatabaseQueryProcessor`
- See: `TYPO3\CMS\Frontend\DataProcessing`

TSconfig & TypoScript

The `SplitProcessor`

- The "SplitProcessor" allows to split values separated by a delimiter into an array

```
page.10 = FLUIDTEMPLATE
page.10.file = EXT:site_default/Resources/Private/Template/Default.html
page.10.dataProcessing.2 = TYPO3\CMS\Frontend\DataProcessing\SplitProcessor
page.10.dataProcessing.2 {
  if.isTrue.field = bodytext
  delimiter = ,
  fieldName = bodytext
  removeEmptyEntries = 1
  filterIntegers = 1
  filterUnique = 1
  as = keywords
}
```

- Possible usage in Fluid:

```
<f:for each="{keywords}" as="keyword">
  <li>Keyword: {keyword}</li>
</f:for>
```

TSconfig & TypoScript

The `CommaSeparatedValueProcessor` (1)

- The "CommaSeparatedValueProcessor" splits values separated by a delimiter in a two-dimensional array:

```
page.10 = FLUIDTEMPLATE
page.10.file = EXT:site_default/Resources/Private/Template/Default.html
page.10.dataProcessing.4 = TYPO3\CMS\Frontend\DataProcessing\CommaSeparatedValueProcessor
page.10.dataProcessing.4 {
  if.isTrue.field = bodytext
  fieldName = bodytext
  fieldDelimiter = |
  fieldEnclosure =
  maximumColumns = 2
  as = table
}
```

- Useful to process CSV files for example or `tt_content` data sets of CType "table"

  See an example usage in Fluid on the following slide

TSconfig & TypoScript

The `CommaSeparatedValueProcessor` (2)

- Possible usage in Fluid:

```
<table>
  <f:for each="{table}" as="columns">
    <tr>
      <f:for each="{columns}" as="column">
        <td>
          {column}
        </td>
      </f:for>
    <tr>
  </f:for>
</table>
```

TSconfig & TypoScript

The `FilesProcessor` (1)

- The "FilesProcessor" resolves file references, files, or files inside a folder or collection to be used for output in the frontend

```
tt_content.image.20 = FLUIDTEMPLATE
tt_content.image.20 {
  file = EXT:myextension/Resources/Private/Templates/ContentObjects/Image.html
  dataProcessing.10 = TYPO3\CMS\Frontend\DataProcessing\FilesProcessor
  dataProcessing.10 {
    references.fieldName = image
    references.table = tt_content
    files = 21,42
    collections = 13,14
    folders = 1:introduction/images/,1:introduction/posters/
    folders.recursive = 1
    sorting = description
    sorting.direction = descending
    as = myfiles
  }
}
```

See an example usage in Fluid on the following slide

TSconfig & TypoScript

The `FilesProcessor` (2)

- Possible usage in Fluid:

```
<ul>
  <f:for each="{myfiles}" as="file">
    <li>
      <a href="{file.publicUrl}">{file.name}</a>
    </li>
  </f:for>
</ul>
```

TSconfig & TypoScript

The GalleryProcessor

- The "GalleryProcessor" calculates the max asset size of file sets

```
tt_content.text_media.20 = FLUIDTEMPLATE
tt_content.image.20 {
  file = EXT:myextension/Resources/Private/Templates/ContentObjects/Image.html
  dataProcessing {
    10 = TYPO3\CMS\Frontend\DataProcessing\FilesProcessor
    20 = TYPO3\CMS\Frontend\DataProcessing\GalleryProcessor
    20 {
      filesProcessedDataKey = files
      mediaOrientation.field = imageorient
      numberOfColumns.field = imagecols
      equalMediaHeight.field = imageheight
      equalMediaWidth.field = imagewidth
      maxGalleryWidth = 1000
      maxGalleryWidthInText = 1000
      columnSpacing = 0
      borderEnabled.field = imageborder
      borderWidth = 0
      borderPadding = 10
      as = gallery
    }
  }
}
```

TSconfig & TypoScript

The `DatabaseQueryProcessor` (1)

- The "DatabaseQueryProcessor" can be used to fetch data from the database

```
tt_content.mycontent.20 = FLUIDTEMPLATE
tt_content.mycontent.20 {
  file = EXT:myextension/Resources/Private/Templates/ContentObjects/MyContent.html
  dataProcessing.10 = TYPO3\CMS\Frontend\DataProcessing\DatabaseQueryProcessor
  dataProcessing.10 {
    if.isTrue.field = records
    table = tt_address
    colPos = 1
    pidInList = 13,14
    as = myrecords
    dataProcessing {
      10 = TYPO3\CMS\Frontend\DataProcessing\FilesProcessor
      10 {
        references.fieldName = image
      }
    }
  }
}
```

See an example usage in Fluid on the following slide

TSconfig & TypoScript

The `DatabaseQueryProcessor` (2)

- Possible usage in Fluid:

```
<ul>
  <f:for each="{myrecords}" as="record">
    <li>
      <f:image image="{record.files.0}" ></f:image>
      <a href="{record.data.www}">{record.data.first_name} {record.data.last_name}</a>
    </li>
  </f:for>
</ul>
```

TSconfig & TypoScript

Conditions for TypoScript Includes

- `INCLUDE_TYPOSCRIPT` has an extra (optional) property "condition" now, which includes the file/directory only, if the condition is met

```
// only include TypoScript, if current user is logged in:
<INCLUDE_TYPOSCRIPT: source="FILE:EXT:my_extension/Configuration/TypoScript/feuser.ts"
  condition="[loginUser = *]">

// include TypoScript depending on application context:
<INCLUDE_TYPOSCRIPT: source="FILE:EXT:my_extension/Configuration/TypoScript/staging.ts"
  condition="applicationContext = /^Production\\/Staging\\/Server\\d+$/">
```

TSconfig & TypoScript

TCA-Option: Display Date Offset

- TCA option `disableAgeDisplay` disables the display of the age (for example: `"2015-08-30 (-27 days)"`)

  ```
  $GLOBALS['TCA']['tt_content']['columns']['date']['config']['disableAgeDisplay'] = true;
  ```

- As a precondition, `type` of the field has to be `input` and `eval` has to be set to `date`

TSconfig & TypoScript

Inline Language Label Files with TypoScript (1)

- XLF language files can be read and converted into an inline array
- This enables accessing language labels in JavaScript for example
- The following three optional parameters are supported:
  - `selectionPrefix`:
    only label keys that start with this prefix will be included
  - `stripFromSelectionName`:
    string that will be removed from any included label key
  - `errorMode`:
    error mode if file could not be found:
    0: syslog entry (default), 1: ignore, 3: throw an exception

Inline Language Label Files with TypoScript (2)

- Example:

```
page = PAGE
page.inlineLanguageLabelFiles {
  someLabels = EXT:myExt/Resources/Private/Language/locallang.xlf
  someLabels.selectionPrefix = idPrefix
  someLabels.stripFromSelectionName = strip_me
  someLabels.errorMode = 2
}
```

- Output:

```
<script type="text/javascript">
/*<![CDATA[*/
  var TYPO3 = TYPO3 || {};
  TYPO3.lang = {"firstLabel":[{"source":"first Label","target":"erstes Label"}],
  "secondLabel":[{"source":"second Label","target":"zweites Label"}]};
/*]]>*/
</script>
```

TSconfig & TypoScript

Workspace Preview by TSconfig

- TYPO3 CMS generates preview links only for tables `tt_content`, `pages` and `pages_language_overlay` by default
- This can be configured using PageTSconfig now:

```
# use page 123 for previewing workspaces records (in general)
options.workspaces.previewPageId = 123

# use the pid field of each record for previewing (in general)
options.workspaces.previewPageId = field:pid

# use page 123 for previewing workspaces records (for table tx_myext_table)
options.workspaces.previewPageId.tx_myext_table = 123

# use the pid field of each record for previewing (or table tx_myext_table)
options.workspaces.previewPageId.tx_myext_table = field:pid
```

TSconfig & TypoScript

Image Quality of `sourceCollection`

- Image quality of each `sourceCollection` entry can be configured now
- This setting takes precedence over configuration in Install Tool (as stored in file `LocalConfiguration.php`)
- Example:

```
# for small retina images
tt_content.image.20.1.sourceCollection.smallRetina.quality = 80

# for large retina images
tt_content.image.20.1.sourceCollection.largeRetina.quality = 65
```

TSconfig & TypoScript

Count Elements in a List

- A new property `returnCount` has been added to the stdWrap property `split`
- This allows to count the number of elements in a comma-separated list
- The following code returns 9 for example:

```
1 = TEXT
1 {
  value = x,y,z,1,2,3,a,b,c
  split.token = ,
  split.returnCount = 1
}
```

TSconfig & TypoScript

Handling of Backend Layouts (1)

- Handling of backend layouts has been simplified for the frontend
- New option `pagelayout` can be used in TypoScript
- Example:

```
page.10 = FLUIDTEMPLATE
page.10 {
  file.stdWrap.cObject = CASE
  file.stdWrap.cObject {
    key.data = pagelayout
    default = TEXT
    default.value = EXT:sitepackage/Resources/Private/Templates/Home.html
    3 = TEXT
    3.value = EXT:sitepackage/Resources/Private/Templates/1-col.html
    4 = TEXT
    4.value = EXT:sitepackage/Resources/Private/Templates/2-col.html
  }
}
```

(continue on next page)

TSconfig & TypoScript

Handling of Backend Layouts (2)

- ...where `key.data = pagelayout` replaces the following code:

```
field = backend_layout
ifEmpty.data = levelfield:-2,backend_layout_next_level,slide
ifEmpty.ifEmpty = default
```

TSconfig & TypoScript

Miscellaneous

- stdWrap-Funktion `bytes` has been introduced in TYPO3 CMS 7.4
- The ability to set the `base` has been added in TYPO3 CMS 7.5, which allows to define whether to use a base of 1000 or 1024 to calculate with

```
bytes.labels = " | K| M| G"
bytes.base = 1000
```

TSconfig & TypoScript

indexed_search: Parameters

- The following TypoScript properties can now be configured for `indexed_search`:

```
titleCropAfter = 50
titleCropSignifier = ...
summaryCropAfter = 180
summaryCropSignifier =
hrefInSummaryCropAfter = 60
hrefInSummaryCropSignifier = ...
markupSW_summaryMax = 300
markupSW_postPreLgd = 60
markupSW_postPreLgd_offset = 5
markupSW_divider = ...
```

- Keys can be:
  - `plugin.tx_indexedsearch.results.`
  - `plugin.tx_indexedsearch.settings.results.`
- Every property has `stdWrap`-functionality

TSconfig & TypoScript

`indexed_search`: Configurable Path Separator

- New TypoScript configuration option `breadcrumbWrap` has been added
- This allows to configure page path separator in `indexed_search` results
- This option supports the TypoScript **option split** syntax.
  Default configuration is "/":

  ```
  plugin.tx_indexedsearch.settings.breadcrumbWrap = / || /
  ```

TSconfig & TypoScript

indexed_search: Configurable no_cache Parameter

- New TypoScript configuration option has been added: `forwardSearchWordsInResultLink.no_cache`
- This controls whether the `no_cache` parameter should be added to page links for `indexed_search`

```
// for Indexed Search Extbase plugins
plugin.tx_indexedsearch.settings.forwardSearchWordsInResultLink.no_cache = 1

// for plugins based on AbstractPlugin
plugin.tx_indexedsearch.forwardSearchWordsInResultLink.no_cache = 1
```

Sources and Authors

Sources and Authors

Sources and Authors

Sources

TYPO3 News:

- http://typo3.org/news

Release Infos:

- https://wiki.typo3.org/Category:ReleaseNotes/TYPO3_7.x
- INSTALL.md and ChangeLog
- typo3/sysext/core/Documentation/Changelog/*

TYPO3 Bug-/Issuetracker:

- https://forge.typo3.org/projects/typo3cms-core

TYPO3 Git Repositories:

- https://git.typo3.org/Packages/TYPO3.CMS.git
- https://git.typo3.org/Packages/TYPO3.Fluid.git

Sources and Authors

TYPO3 CMS What's New Slides:

Patrick Lobacher
(Research, Information Gathering and German Version)

Michael Schams
(Project Leader and English Version)

Translations and Contributions by:
Andrey Aksenov, Paul Blondiaux, Pierrick Caillon, Sergio Catalá,
Ben van't Ende, Jigal van Hemert, Sinisa Mitrovic, Michel Mix, Angeliki Plati,
Nena Jelena Radovic and Roberto Torresani

http://typo3.org/download/release-notes/whats-new