TYPO3 CMS 7 LTS – What's New

TSconfig & TypoScript

Patrick Lobacher und Michael Schams

10/November/2015

TYPO3

Introduction

TYPO3 CMS 7 LTS – What's New

Die folgenden Slides sind auf ein spezielles Thema fokussiert. Abhängig von der Rolle, können die folgenden Themen ebenfalls interessant sein:

	BE User Interface	TypoScript	In-Depth Changes	Extbase/Fluid	Deprecated/Removed	Sys.Administration
Redakteure	X					
Integratoren		X	X		X	
Entwickler			X	X	X	
SysAdmins						X

Download aller **What's New Slides** unter typo3.org

TSconfig & TypoScript

TypoScript wird verwendet, um hierarchische Informationen in einer Struktur zu verwalten um damit die TYPO3 CMS Instanz zu konfigurieren. In TYPO3 CMS 7 LTS wurden viele Eigenschaften und Optionen zugefügt, geändert bzw. auch als veraltet markiert oder entfernt.
Open Graph Attribute werden nun out-of-the-box unterstützt und die Integrität von extern gehosteten JavaScript-Dateien kann nun via SRI-Hash überprüft werden, um nur zwei der neuen Features zu nennen.
Entfernte TypoScript-Optionen sind in den Slides **Deprecated/Removed** zu finden.

TSconfig & TypoScript

TSconfig für Linkvalidator inkludieren

- TSconfig Konfiguration für den Linkvalidator wird gelesen...
  - entweder aus dem aktiven TSconfig des Backends,
  - oder aus der Konfiguration des Scheduler-Tasks
- Das folgende TSconfig kann vom Linkchecker ausgelesen werden

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

- Dort steht das TSconfig dann als `$this->tsConfig` zur Verfügung

TSconfig & TypoScript

Links zu deaktivierten Datensätzen im Linkhandler melden

- Bisher hat der Linkhandler lediglich gewarnt, wenn es Links zu gelöschten oder nicht existierenden Datensätzen gab
- Über die folgende TSconfig Einstellung kann nun auch eine Warnung aktiviert werden, wenn der Link auf einen deaktivierten Datensatz zeigt:

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

TSconfig & TypoScript

RTE: mehrere CSS Klassen

- Um das Handling mit komplexen CSS Frameworks wie Twitter Bootstrap zu handhaben, muss es möglich sein, mehrere Klassen an ein Element zu vergeben
- Mit diesem neuen Feature muss der Autor nur noch einen Style auswählen, um dies zu erreichen, und nicht mehrere

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

TSconfig & TypoScript

RTE: nicht-selektierbare Klassen

- Man kann nun Klassen als "nicht-selektierbar" im Style-Selektor des RTE konfigurieren

```
// der Wert "1" bedeutet, class ist auswaehlbar
// der Wert "0" bedeutet, class ist nicht auswaehlbar
RTE.classes.[ *classname* ] {
  .selectable = 1
}
```

TSconfig & TypoScript

RTE: mehrere CSS-Dateien einbinden

- Man kann nun mehrere CSS-Dateien in den RTE laden

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

- Gibt man keine CSS Datei an, so wird die Default-Datei geladen:
  `typo3/sysext/rtehtmlarea/res/contentcss/default.css`

TSconfig & TypoScript

Exception während Rendering erzeugen – Teil 1

- Sobald Fehler im Rendering von einzelnen Inhaltselementen (Content Objects) (z.B. mittels `USER`) auftreten, wird eine Fehlermeldung erzeugt, die in TYPO3 CMS < 7.0 die gesamte Ausgabe zerstört
- In TYPO3 CMS 7.0 wurde ein Exception-Handling eingeführt, welches eine Fehlermeldung in die Ausgabe an der Stelle integriert, an der das Rendering stattgefunden hat

TSconfig & TypoScript

Exception während Rendering erzeugen – Teil 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 für `page.headTag`

- TypoScript Option `page.headTag` hat nun `stdWrap`-Funktionalität

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

TSconfig & TypoScript

JavaScript-Dateien asynchron laden

- JavaScript-Dateien könen nun asynchron geladen werden

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

- Das gilt für:
  - `includeJSlibs` / `includeJSLibs`
  - `includeJSFooterlibs`
  - `includeJS`
  - `includeJSFooter`

TSconfig & TypoScript

HMENU Eigenschaft mit `additionalWhere`

- TypoScript cObject `HMENU` erhält eine neue Eigenschaft `additionalWhere`
- Jenes erlaubt eine spezifischere DB Abfrage (z.B. Filterung)
- Beispiel:

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

TSconfig & TypoScript

Zusätzliche Eigenschaften für `HMENU` Browse-Menü

- Zwei neue Eigenschaften für das cObject `HMENU` (Option "`special=browse`"), um detaillierter definieren zu können, welche Seiten im Menü erscheinen sollen:
  - `excludeNoSearchPages`
  - `includeNotInMenu`
- Beispiel:

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

TSconfig & TypoScript

Mehrere HTTP-Header

- HTTP Header können nun mittels `config.additionalHeaders` als Array gesetzt werden
- Das ermöglicht es, mehreren Header-Zeilen auf einmal zu konfigurieren

```
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" für `config.absRefPrefix`

- TypoScript Konfiguration `config.absRefPrefix` kann verwendet werden, um der URL einen Prefix bei relativen Pfaden zu geben. Als Alternative zu `config.baseURL` (um eine bestimmte Domain zu spezifizieren), erkennt `absRefPrefix` die Site-Root automatisch:

```
config.absRefPrefix = auto

# anstelle von:
[ApplicationContext = Production]
config.absRefPrefix = /

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

Hinweis: diese Option ist "Multi-Domain"-sicher und mehrfaches Caching der selben Daten wird verhindern.

TSconfig & TypoScript

Zwei-Zeichen ISO Code für `sys_language` (1)

- Die Behandlung von Sprachen wird durch Einträge in DB Tabelle `sys_language` vorgenommen, die durch `sys_language_uid` referenziert werden
- In TYPO3 CMS 7.1 wurden ISO 639-1 Zwei-Zeichen Codes implementiert:
  - Neues DB Feld: `sys_language.language_isocode`
  - Neue TypoScript-Option: `sys_language_isocode`

Hinweis: bei **ISO 639** handelt es sich um eine Sammlung von Standards der "International Organization for Standardization". Eine List der ISO 639-1 Codes ist hier abrufbar:
http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

TSconfig & TypoScript

Zwei-Zeichen ISO Code für `sys_language` (2)

- Beispiel:

```
# 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

Eigene Conditions im Backend

- Eigene Conditions für das **Frontend** wurden bereits mit TYPO3 CMS 7.0 eingeführt
- Seit TYPO3 CMS 7.1 ist es nun auch möglich, eigene Conditions für das **Backend** zu implementieren
- Die Condition muss von `AbstractCondition` ableiten und die Methode `matchCondition` bereitstellen
- Anwendungsbeispiel in TypoScript:

```
[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition]

[BigCompanyName\TypoScriptLovePackage\MyCustomTypoScriptCondition = 7]

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

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

TSconfig & TypoScript

Zufügen von Icons in TCEFORM via PageTSconfig

- Eigene Werte und Labels von Select-Feldern können bereits mit der PageTSconfig Option `addItems` vergeben werden
- Nun können auch Icons für diese Felder definiert werden
  - Option 1: mittels `addItems` und der Eigenschaft `.icon`
  - Option 2: mittels `altIcons` (generell für alle Felder)
- Beispiel:

```
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

Element Browser: Mountpoints hinzufügen

- Neue UserTSconfig Option `.append` erlaubt es Administratoren Mountpoints **hinzuzufügen**, anstatt die Liste der konfigurierten DB Mountpoints eines Benutzers neu zu schreiben
- Beispiel:

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

TSconfig & TypoScript

Überschreiben der Labels von Radio-Buttons und Checkboxen

- Labels von Radio-Buttons und Checkboxen können nun überschrieben werden
- Beispiel:

```
// 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

Diverses (1)

- Breite und Höhe des Element-Browsers können nun per UserTSconfig festgelegt werden

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

- PageTSconfig: mit einer neue RTE-Konfiguration kann das Standard-Ziel von Links beeinflusst werden

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

Wobei `[type]` zum Beispiel `page`, `file`, `url`, `mail` or `spec` sein kann (Extensions können weitere Typen zur Verfügung stellen)

TSconfig & TypoScript

Diverses (2)

- Standardmäßig sind Section-Headlines der Indexed-Search Resultate verlinkt. Das kann nun mittels TypoScript deaktiviert werden

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

- `getData` kann jetzt auch `field`-Daten abfragen (nicht nur Arrays, wie beispielsweise `GPVar` und `TSFE`)

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

- TypoScript Konfiguration `config.pageTitle` hat jetzt `stdWrap`-Funktionalität

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

TSconfig & TypoScript

Konfigurierbarer Vorschau-Link (1)

- Es ist nun möglich, die URL zur Vorschau einer Seite zu definieren, die durch den Button "Speichern & Vorschau" aufgerufen wird.
- Damit kann man unterschiedliche Links für Blog- oder News-Datensätze, aber auch für Inhaltselemente generieren lassen.

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

TSconfig & TypoScript

Konfigurierbarer Vorschau-Link (2)

- `previewPageId`:
  UID der Seite, die für den Preview verwendet werden soll
  (ohne Angabe wird die aktuelle Seite verwendet)
- `useDefaultLanguageRecord`:
  definiert, ob übersetzte Datensätze die UID des Default-Datensatzes verwenden
  (standardmäßig ist jenes aktiviert, default: 1)
- `fieldToParameterMap`:
  Mapping (Key = Value) von Feldern des Datensatzes, die als GET-Parameter an den Link angehängt werden
- `additionalGetParameters`:
  wie `fieldToParameterMap`, aber für beliebige Paramater

TSconfig & TypoScript

RTE Konfiguration: Default-Target

- Das Default-Target in der RTE Konfiguration ist nun im PageTSconfig abhängig vom Typ einstellbar

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

- Als "*type*" sind folgende Werte zulässig:
  (weitere können via Extensions eingebracht werden)
  - `page`
  - `file`
  - `url`
  - `mail`
  - `spec`

TSconfig & TypoScript

Leere HTML-Tags im HTMLparser löschen

- Es ist nun möglich, leere HTML-Tags im HTMLparser zu löschen

```
stdWrap {
   // Hier werden alle leeren HTML-Tags entfernt
   HTMLparser.stripEmptyTags = 1
   // Hier werden nur leere h2 und h3 Tags entfernt
   HTMLparser.stripEmptyTags.tags = h2, h3
}

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

- Da der HtmlParser unbekannte Tags grundsätzlich entfernt, ist es ratsam, diese zunächst zu behalten:
  `HTMLparser.keepNonMatchedTags = 1`

TSconfig & TypoScript

Diverses

- Der Button für "Abkürzung" (engl. *abbreviation*) im RTE kann nun in der PageTSconfig ausgeblendet werden (da nicht mehr HTML5 konform):

```
# moegliche Wert sind:
# acronym, definedAcronym, abbreviation, definedAbbreviation
buttons.abbreviation.removeFieldsets = acronym,definedAcronym
```

- Die Eigenschaft `inlineLanguageLabel` des Objekts `PAGE` kann nun auch mit `LLL:`-Referenzen umgehen

TSconfig & TypoScript

stdWrap Funktion `strtotime`

- Es gibt nun eine stdWrap Funktion `strtotime`, welche es ermöglicht, formatierte Datum-Angaben in einen Timestamp umzuwandeln

```
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

- Prüft man in Conditions nur mittels GP so wird beim gleichzeitigen Vorhandensein von POST- und GET-Variablen (z.B. `tx_demo_demo[...]=...`), lediglich die POST-Variable zurückgegeben
- Mit der neuen Option `GPmerged` werden beide Variablen zusammengeführt und dann zurückgegeben

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

TSconfig & TypoScript

Weitere Werte für die Funktion `stdWrap.case`

- Die stdWrap Funktion `case` ist um die beiden Werte `uppercamelcase` und `lowercamelcase` ergänzt worden
- Beispiel:

```
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

Eigenschaft `integrity` für JavaScript-Dateien (1)

- Es wurde die Eigenschaft `integrity` zugefügt, um einen SRI Hash zum JavaScript-Markup hinzuzufügen, mit dem die Quelle verifiziert werden kann (SRI: Sub-Resource Integrity)
- Dies betrifft die Eigenschaften `page.includeJSLibs`, `page.includeJSFooterlibs`, `includeJS` und `includeJSFooter`
- Beispiel:

```
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

Eigenschaft `integrity` für JavaScript-Dateien (2)

- SRI ist eine Spezifikation des W3C, die es ermöglicht zu verifizieren, ob extern gehostete Dateien manipuliert worden sind
- Erstellen von Integrity Hashes:
  - Option 1: https://srihash.org
  - Option 2: durch folgende Kommandos

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

- Weitere Informationen:
  - http://www.w3.org/TR/SRI/

TSconfig & TypoScript

Data-Provider für Backend Layouts (1)

- Backend-Layouts können jetzt per PageTSconfig definiert und damit auch in Dateien ausgelagert werden. Zum Beispiel:

```
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
                  }
                }
              }
[...]
```


TYPO3

TSconfig & TypoScript

Data-Provider für Backend Layouts (2)

- (Fortsetzung)

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

TSconfig & TypoScript

Erweiterung der Option `page.meta`

- Die Option `page.meta` unterstützt nun auch Open Graph Attributnamen

```
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` wird automatisch gesetzt

- In der TypoScript-Option `select` (die beispielsweise beim cObject CONTENT verwendet wird) musste man bisher das `languageField` explizit setzen
- Jenes wird nun automatisch gesetzt und kann daher weglassen werden

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

  # Die nachfolgende Zeile ist nicht notwendig:
  #select.languageField = sys_language_uid

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

TSconfig & TypoScript

Individuelles Content Caching

- Es gibt nun ein individuelles Content Caching, welches im Gegensatz zu `stdWrap.cache` auch mit COA-Objekten funktioniert (ähnlich dem "Magento Block Caching")

```
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

Zähler für listNum

- Es gibt eine neue Eigenschaft `returnCount` für die stdWrap Eigenschaft `split`
- Damit kann die Anzahl der Elemente in einer kommaseparierten Liste ermittelt werden
- Das folgende Beispiel gibt `9` zurück:

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

TSconfig & TypoScript

Sortierung von Tabellen im Backend

- Über die TSconfig Option `mod.web_list.tableDisplayOrder` kann eingestellt werden, wie die Tabellen im List-Modul sortiert werden
- Dafür werden die Schlüsselworte `before` und `after` verwendet

Anwendung:

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

Zum Beispiel:

```
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 im HTTP Header

- Es wird nun standardmäßig `Content-language: XX` im HTTP Response Header an den Client gesendet, wobei "XX" dem ISO-Code entspricht, der via `sys_language_content` konfiguriert wurde
- Dabei kann `sys_language_content` unterschiedlich zu `sys_language_uid` sein, wenn der Inhalt von der Fallback-Sprache ermittelt wird
  (jenes hängt von der Einstellung `sys_language_mode` ab)
- Über die Einstellung `config.disableLanguageHeader = 1` kann der Header bei Bedarf auch deaktiviert werden

TSconfig & TypoScript

Rekursive Option für ordner-basierte File Collections

- Ordner-basierte File Collections haben nun eine Option um rekursiv alle Dateien für einen gegebenen Ordner zu ermitteln
- Die Option ist ebenfalls für das TypoScript Objekt `FILES` verfügbar

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

TSconfig & TypoScript

Extension `.ts` für Static Templates

- Bislang waren für statische TypoScript Templates nur die folgenden Dateinamen zugelassen:
  - `constants.txt`
  - `setup.txt`
  - `include_static.txt`
  - `include_static_files.txt`
- Als Extension kann nun auch `.ts` verwendet werden
- Dabei hat `.ts` Vorrang vor `.txt`

TSconfig & TypoScript

save & view Button

- Der "save & view" Button ist nun via TSconfig konfigurierbar
- Der folgende Schlüssel nimmt eine kommaseparierte Liste an "doktypes" auf: `TCEMAIN.preview.disableButtonForDokType`
- Der Standardwert ist "254, 255, 199" (Storage Folder, Recycler und Menu Seperator)
- In Foldern und Recycler-Seiten ist der "save & view" Button daher standardmäßig nicht mehr sichtbar

TSconfig & TypoScript

stdWrap für `treatIdAsReference`

- Für das Objekt `getImgResource` existiert die Option `treatIdAsReference`, die ggf. definiert, dass die angegebenen UIDs als UIDs von `sys_file_reference`, anstatt von `sys_file` gelten
- Die Option `treatIdAsReference` besitzt nun stdWrap Funktionalität

TSconfig & TypoScript

Conditions für TypoScript-Include

- Der `INCLUDE_TYPOSCRIPT` Tag besitzt nun das optionale Attribut "condition", welches es ermöglicht, die Datei (oder das Verzeichnis) nur dann zu inkludieren, wenn die Condition erfüllt ist:

```
// TypoScript nur laden, wenn Benutzer eingeloggt ist:
<INCLUDE_TYPOSCRIPT: source="FILE:EXT:my_extension/Configuration/TypoScript/feuser.ts"
  condition="[loginUser = *]">

// TypoScript nur laden, wenn ApplicationContext gesetzt ist:
<INCLUDE_TYPOSCRIPT: source="FILE:EXT:my_extension/Configuration/TypoScript/staging.ts"
  condition="applicationContext = /^Production\\/Staging\\/Server\\d+$/">
```

TSconfig & TypoScript

TCA-Option, um Datum Feldweise auszublenden

- Es gibt nun eine TCA-Option `disableAgeDisplay`, um die Anzeige des Datums auszublenden
  Voraussetzung hierfür ist, dass der Typ des Feldes `input` ist, und `eval` auf `date` gesetzt ist

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

TSconfig & TypoScript

Inline Sprachlabels mit TypoScript (1)

- Man kann nun Sprachdateien mittels TypoScript auslesen und als Inline-Array in den Quelltext schreiben, um z.B. per JavaScript darauf zuzugreifen
- Folgende Optionen sind möglich:
  - `selectionPrefix`:
    nur Schlüssel, die mit diesem Prefix anfangen, werden ermittelt
  - `stripFromSelectionName`:
    String, der von jedem Schlüssel entfernt wird
  - `errorMode`:
    Mode, wenn die Sprachdatei nicht gefunden wird
    (0: Eintrag im Syslog vornehmen, 1: ignorieren, 3: Exception generieren)

TSconfig & TypoScript

Inline Sprachlabels mit TypoScript (2)

- Beispiel:

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

- Ausgabe:

```
<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 per TSconfig

- Standardmäßig erzeugt TYPO3 lediglich Vorschau-Links für die Tabellen `tt_content`, `pages` und `pages_language_overlay`
- Dies kann nun per PageTSconfig angepasst werden:

```
# Verwendung der Seite 123 fuer Workspace Preview (fuer alle Tabellen)
options.workspaces.previewPageId = 123

# Verwendung des Feldes pid (fuer alle Tabellen)
options.workspaces.previewPageId = field:pid

# Verwendung der Seite 123 fuer Workspace Preview (fuer die Tabelle tx_myext_table)
options.workspaces.previewPageId.tx_myext_table = 123

# Verwendung des Feldes pid fuer Workspace Preview (fuer die Tabelle tx_myext_table)
options.workspaces.previewPageId.tx_myext_table = field:pid
```

TSconfig & TypoScript

Bildqualität kann per SourceCollection gesetzt werden

- Die Bildqualität jeder sourceCollection kann nun konfiguriert werden
- Dies überschreibt die Einstellungen, die im Install Tool gemacht wurden und in der Datei `LocalConfiguration.php` gespeichert sind

```
# fuer kleine Retina Bilder
tt_content.image.20.1.sourceCollection.smallRetina.quality = 80

# fuer groessere Retina Bilder
tt_content.image.20.1.sourceCollection.largeRetina.quality = 65
```

TSconfig & TypoScript

Count für Split hinzugefügt

- Es wurde eine neue Eigenschaft `returnCount` zur stdWrap-Funktion `split` hinzugefügt, die die Anzahl der Elemente nach dem Split enthält

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

# result: 9
```

TSconfig & TypoScript

Handling von Backend-Layouts vereinfacht (1)

- Das Handling, um Backend-Layouts mit Templates für die Frontend-Ausgabe zu versehen, wurde vereinfacht, indem die Option `pagelayout` eingeführt wurde
- Beispiel:

```
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
  }
}
```

(Fortsetzung auf nächster Seite)

TSconfig & TypoScript

Handling von Backend-Layouts vereinfacht (2)

- `pagelayout` ersetzt dabei den folgenden Code:

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

TSconfig & TypoScript

Diverse

- Für die mit TYPO3 CMS 7.4 eingeführte stdWrap-Funktion `bytes` kann nun die Basis (z.B. 1000 oder 1024) gesetzt werden:

  ```
  bytes.base = 1000
  ```

TSconfig & TypoScript

Parameter für `indexed_search`

- Einige Parameter für `indexed_search`, die bisher hart-kodiert waren, können nun konfiguriert werden

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

- Dabei können folgende Schlüssel angesprochen werden:
  - `plugin.tx_indexedsearch.results.`
  - `plugin.tx_indexedsearch.settings.results.`
- Alle Optionen besitzen außerdem `stdWrap`-Funktionalität

TSconfig & TypoScript

Konfiguration des Path-Seperators bei `indexed_search`

- Es wurde eine neue TypoScript-Option `breadcrumbWrap` hinzugefügt, mit dem Path-Seperators bei `indexed_search` konfiguriert werden kann
- Darüber wird der Pfad für die Breadcrumb bei Suchergebnissen angezeigt
- Die Option verfügt über Option-Split und ist standardmäßig auf "/" konfiguriert

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

TSconfig & TypoScript

`no_cache` Parameter-Konfiguration für `indexed_search`

- Es wurde eine neuen TypoScript-Option hinzugefügt: `forwardSearchWordsInResultLink.no_cache`
- Damit kann eingestellt werden, ob der `no_cache` Paramater an die Seiten-Links innerhalb von `indexed_search` hinzugefügt wird

```
// Fuer Extbase-Plugins
plugin.tx_indexedsearch.settings.forwardSearchWordsInResultLink.no_cache = 1

// Fuer eingefuegte Plugins
plugin.tx_indexedsearch.forwardSearchWordsInResultLink.no_cache = 1
```

Quellen und Autoren

Sources and Authors

Quellennachweis

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

TYPO3

Sources and Authors

TYPO3 CMS What's New Slides:

Patrick Lobacher
(Recherche, Informationsdokumentation und deutsche Version)

Michael Schams
(Project Leader und englische Version)

Übersetzungen und Mitwirkung von:
Andrey Aksenov, Paul Blondiaux, Pierrick Caillon, Sergio Catalá,
Ben van't Ende, Jigal van Hemert, Sinisa Mitrovic, Michel Mix, Angeliki Plati,
Nena Jelena Radovic und Roberto Torresani

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