TYPO3 CMS 7 LTS – What's New

Änderungen im System

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

Änderungen im System

Durch die Überarbeitung aller Komponenten und der Entfernung veralteter Technologien, ist das Backend von TYPO3 CMS 7 LTS nun deutlich schneller geworden im Vergleich zu älteren Versionen.
Der Performance-Boost ist aber nicht die einizige Änderung im Backend - viele weitere Änderungen wurden im Core durchgeführt. PHP Code wurde aufgeräumt, einige Komponenten in Extbase neu geschrieben und viele neue Features wurden mit Hilfe von aktuallen Technologien und Programmier-Paradigmen implementiert.

Änderungen im System

Integration von jQuery UI 1.11.2

- jQuery UI 1.11 unterstützt AMD (Asynchronous Module Definition), mit welcher JavaScript Dateien erst geladen werden, wenn sie benötigt werden (Geschwindigkeitsvorteil)
- jQuery UI 1.11 ersetzt jQuery UI 1.10 + Scriptaculous in TYPO3 CMS 7.0
- Es werden nur der Core und Interaction Components inkludiert, die notwendig sind, um ExtJS und Scriptaculous zu ersetzen.
- Widgets werden nicht inkludiert, sondern über Bootstrap realisiert (z.B. DatePicker, Spinner, Dialog, Buttons, Tabs, Tooltip)

Änderungen im System

Registry für die File Rendering Classes

- Um verschiedene Dateiformate zu rendern, wird eine Registry benötigt, bei der sich FileRenderer registrieren können. Dies geschieht für "Name" (z.B. Video, MPEG, AVI, WAV, ...) wie folgt:

```
<?php
namespace ...;

class NameTagRenderer implements FileRendererInterface {
  protected $possibleMimeTypes = array('audio/mpeg', 'audio/wav', ...);
  public function getPriority() {
    return 1; // priority: the higher, the more important (max: 100)
  }
  public function canRender(FileInterface $file) {
    return in_array($file->getMimeType(), $this->possibleMimeTypes, TRUE);
  }
  public function render(FileInterface $file, $width, $height, array $options = array(),
        $usedPathsRelativeToCurrentScript = FALSE) {
    ...

    return 'HTML code';
  }
}
```

Änderungen im System

Evaluierungsfunktion "email" für das TCA

- Für das TCA wurde eine neue Evaluierungsfunktion "email" hinzugefügt, welche serverseitig überprüft, ob ein eingegebener Wert eine gültige Email ist. Im Fehlerfall wird eine Flash-Message ausgegeben
- Example:

```
'emailaddress' => array(
  'exclude' => 1,
  'label' => 'LLL:EXT:myextension/Resources/Private/Language/locallang_db
      .xlf:tx_myextension
    'config' => array(
      'type' => 'input',
      'size' => 30,
      'eval' => 'email,trim'
  ),
)
```

Änderungen im System

Einführung einer abstrakten Condition in TypoScript

- Es gibt nun eine AbstractCondition, von der eigene Conditions ableiten können

```
class TestCondition
  extends \TYPO3\CMS\Core\Configuration\TypoScript\ConditionMatching\AbstractCondition {

  public function matchCondition(array $conditionParameters) {
      if ($conditionParameters[0] === '= 7' && $conditionParameters[1] === '!= 6') {
      throw new TestConditionException('All Ok', 1411581139);
    }
  }
}
```

- Die Verwendung in TypoScript sieht wie folgt aus:

```
[Vendor\Package\TestCondition]
[Vendor\Package\TestCondition = 7]
[Vendor\Package\TestCondition = 7, != 6]
```

- Welche Operatoren in der Condition zur Verfügung stehen, wird in der Klasse selbst festgelegt

Änderungen im System

Signal zur Manipulation des HTML-Tags in IconUtility

- Signal für IconUtility zur HTML-Tag Manipulation

```
dispatch(
  'TYPO3\\CMS\\Backend\\Utility\\IconUtility',
  'buildSpriteHtmlIconTag',
  array($tagAttributes, $innerHtml, $tagName)
);
```

- Wird aufgerufen in:

```
TYPO3\CMS\Backend\Utility\IconUtility\buildSpriteHtmlIconTag
```

Änderungen im System

Signal Slots für SoftReferenceIndex

- Zwei neue Signal Slot Dispatch Calls innerhalb von SoftReferenceIndex:

```
protected function emitGetTypoLinkParts(
  $linkHandlerFound, $finalTagParts, $linkHandlerKeyword, $linkHandlerValue) {
  return $this->getSignalSlotDispatcher()->dispatch(
    get_class($this),
    'getTypoLinkParts',
    array($linkHandlerFound, $finalTagParts, $linkHandlerKeyword, $linkHandlerValue)
  );
}
protected function emitSetTypoLinkPartsElement(
  $linkHandlerFound, $tLP, $content, $elements, $idx, $tokenID) {
  return $this->getSignalSlotDispatcher()->dispatch(
    get_class($this),
    'setTypoLinkPartsElement',
    array($linkHandlerFound, $tLP, $content, $elements, $idx, $tokenID, $this)
  );
}
```

- Wird augerufen in:

```
TYPO3\CMS\Core\Database\SoftReferenceIndex->findRef_typolink
TYPO3\CMS\Core\Database\SoftReferenceIndex->getTypoLinkParts
```

Änderungen im System

Signal Slots für afterPersistObjects

- Bislang gab es nur ein Signal, wenn ein Objekt im Repository (Extbase) aktualisiert wurde: `afterUpdate`
- Da aber ein Aggregate Root beispielsweise Subobjekte ebenfalls persistiert, benötigt es ein Signal, welches erst dann emmitiert wird, wenn alle Objekte eines Aggregate Roots persistiert wurden:

```
protected function emitAfterPersistObjectSignal(DomainObjectInterface $object) {
  $this->signalSlotDispatcher->dispatch(__CLASS__, 'afterPersistObject', array($object));
}
```

- Wird aufgerufen in:

```
TYPO3\CMS\Extbase\Persistence\Generic\Backend->persistObject
```

Änderungen im System

Signal Slots für loadBaseTca

- Mit diesem Signal kann das gesamte TCA (anstelle von Teilen) gecacht werden

```
protected function emitTcaIsBeingBuiltSignal(array $tca) {
  list($tca) = static::getSignalSlotDispatcher()->dispatch(
    __CLASS__,
    'tcaIsBeingBuilt',
    array($tca)
  );
  $GLOBALS['TCA'] = $tca;
}
```

- Wird audgerufen in:

```
TYPO3\CMS\Core\Utility\ExtensionManagementUtility\Backend->
     buildBaseTcaFromSingleFiles
```

Änderungen im System

API um gecachte TCA Änderungen zuzufügen

- PHP Dateien in `extkey/Configuration/TCA/Overrides/` werden ausgeführt, direkt nachdem der TCA-Cache aufgebaut wurde
- Die Dateien dürfen nur Code enthalten, der das TCA manipuliert, wie beispielsweise: `addTCAColumns` oder `addToAllTCATypes`
- Sobald Extensions dieses Features verwenden, ist mit einem deutlichen Geschwindigkeitsschub von Backend Requests zu rechnen

Änderungen im System

Nur-lesender Zugriff auf File Mounts

- File Mounts können (wieder) mit nur-lesenden Zugriff konfiguriert werden ("read-only")
- Jenes war bereits in TYPO3 CMS 4.x möglich, wurde aber in TYPO3 CMS 6.x entfernt
- Zum Beispiel: Ordner "test" vom storage UID 3 als Mount mit nur lesenden Zugriff in der Dateiliste und dem Element Browser:

```
options.folderTree.altElementBrowserMountPoints = 3:/test
```

Falls kein Storage angegeben wird, geht man davon aus, dass sich der Ordner im Default Storage befindet.

Änderungen im System

Sonstiges

- jQuery wurde von Version 1.11.0 auf Version 1.11.1 aktualisiert
- Datatables wurde von Version 1.9.4 zu Version 1.10.2 aktualisiert
- Einige alte, nicht mehr benutzte Variablen wurden von `EM_CONF` entfernt
- Extension Icons können nun auch im SVG Bildformat vorliegen (`ext_icon.svg`)
- Das Senden eines unbekannten eID Identifikators im Request resultiert nun in eine Exception

Änderungen im System

TCA: Maximum chars in text element

- TCA-Typ `text` unterstützt nun das HTML5-Attribut `maxlength`, um die maximale Anzahl der einzugebenden Zeichen zu beschränken (Hinweis: Zeilenumbrüche zählen hierbei als zwei Zeichen)

```
'teaser' => array(
  'label' => 'Teaser',
  'config' => array(
    'type' => 'text',
    'cols' => 60,
    'rows' => 2,
    'max' => '30' // <-- maxlength
  )
),
```

Es ist zu beachten, dass nicht alle Browser dieses Attribut unterstützen.

Siehe: Browserübersicht

Änderungen im System

New SplFileInfo implementation

- Neue Klasse: `TYPO3\CMS\Core\Type\File\FileInfo`
- Diese erweitert `SplFileInfo`, die wiederrum Meta-Informationen von Dateien ermittelt

```
$fileIdentifier = '/tmp/foo.html';
$fileInfo = GeneralUtility::makeInstance(
  \TYPO3\CMS\Core\Type\File\FileInfo::class,
  $fileIdentifier
);
echo $fileInfo->getMimeType(); // output: text/html
```

- Entwickler können über folgenden Hook auf die Funktionalität zugreifen:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']
  [\TYPO3\CMS\Core\Type\File\FileInfo::class]['mimeTypeGuessers']
```

Änderungen im System

UserFunc in TCA Display Condition

- userFunc `displayCondition` ermöglicht es auf jeden erdenklichen Status und jede Condition zu prüfen
- Sollte irgendeine Situation nicht mit den existierenden Checks abgefangen werden können, ist es auch möglich, eigene Funktionen zu schreiben (diese müssen lediglich `TRUE/FALSE` zurückgeben, um das entsprechende TCA Field sichtbar zu machen oder zu verbergen)

```
$GLOBALS['TCA']['tt_content']['columns']['bodytext']['displayCond'] =
  'USER:Vendor\\Example\\User\\ElementConditionMatcher->
    checkHeaderGiven:any:more:information';
```

Änderungen im System

API für Twitter Bootstrap Modals (1)

- Zwei neue API Methoden um Modal Popups zu erzeugen/entfernen:
  - `TYPO3.Modal.confirm(title, content, severity, buttons)`
  - `TYPO3.Modal.dismiss()`
- Optionen `title` und `content` sind mindestens erforderlich
- Optionen `buttons.text` und `buttons.trigger` sind erforderlich, wenn `buttons` verwendet wird
- Beispiel 1:

```
TYPO3.Modal.confirm(
  'The title of the modal',          // title
  'This the the body of the modal', // content
  TYPO3.Severity.warning             // severity
);
```

Änderungen im System

API für Twitter Bootstrap Modals (2)

- Beispiel 2:

```
TYPO3.Modal.confirm('Warning', 'You may break the internet!',
  TYPO3.Severity.warning,
  [
    {
      text: 'Break it',
      active: true,
      trigger: function() { ... }
    },
    {
      text: 'Abort!',
      trigger: function() {
        TYPO3.Modal.dismiss();
      }
    }
  ]
);
```

Änderungen im System

JavaScript Storage API (1)

- Mittels JavaScript kann auf die BE User Konfiguration zugegriffen werden (`$BE_USER->uc`, einfache Key-Value Paare)
- Zusätzlich kann nun auch der HTML5 Standard localStorage verwendet werden, um Daten (Client-seitig) im Browser des Benutzers zu speichern und auszulesen
- Zwei neue global TYPO3 Objekte:
  - `top.TYPO3.Storage.Client`
  - `top.TYPO3.Storage.Persistent`
- Jedes Objekt hat folgende API Methoden:
  - `get(key)`: Daten holen
  - `set(key,value)`: Daten schreiben
  - `isset(key)`: Prüfen, ob `key` genutzt wird
  - `clear()`: Löschen des Speichers

Änderungen im System

JavaScript Storage API (2)

- Beispiel:

```
// get value of key 'startModule'
var value = top.TYPO3.Storage.Persistent.get('startModule');

// write value 'web_info' as key 'start_module'
top.TYPO3.Storage.Persistent.set('startModule', 'web_info');
```

Änderungen im System

Inline Rendering von Checkboxes

- Die Konfiguration `inline` sorgt bei "cols" dafür, dass Checkboxen nebeneinander dargestellt werden, um Platz im Backend User Interface zu sparen

```
'weekdays' => array(
  'label' => 'Weekdays',
  'config' => array(
    'type' => 'check',
    'items' => array(
      array('Mo', ''),
      array('Tu', ''),
      array('We', ''),
      array('Th', ''),
      array('Fr', ''),
      array('Sa', ''),
      array('Su', '')
    ),
    'cols' => 'inline'
  )
),
...
```

Änderungen im System

Content Object Registration

- Es wurde eine neue globale Option eingeführt, um cObjects wie `TEXT` zu registrieren bzw. zu erweitern
- Eine Liste aller verfügbaren cObjects ist verfügbar als:

```
$GLOBALS['TYPO3_CONF_VARS']['FE']['ContentObjects']
```

- Beispiel: ein neues cObject `EXAMPLE` registrieren

```
$GLOBALS['TYPO3_CONF_VARS']['FE']['ContentObjects']['EXAMPLE'] =
  Vendor\MyExtension\ContentObject\ExampleContentObject::class;
```

- Die registrierte Klasse muss von der folgenden Klasse ableiten:
  `TYPO3\CMS\Frontend\ContentObject\AbstractContentObject`
- Idealerweise speichert man seine Datei im Verzeichnis
  `typo3conf/myextension/Classes/ContentObject/`
  um für zukünftige Autoload-Funktionen vorbereitet zu sein

Änderungen im System

Hooks und Signals (1)

- Neuer Hook wurde am Ende von `PageRepository->init()` hinzugefügt, mit dem die Sichtbarkeit von Seiten beeinflusst werden kann
- Der Hook kann wie folgt registriert werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']
   [\TYPO3\CMS\Frontend\Page\PageRepository::class]['init']
```

- Die Hook-Klasse muss das folgende Interface implementieren:

```
\TYPO3\CMS\Frontend\Page\PageRepositoryInitHookInterface
```

Änderungen im System

Hooks und Signals (2)

- Neuer Hook wurde zu PageLayoutView hinzugefügt, um das Rendering des Footers von Inhaltselementen im Backend manipulieren zu können
- Beispiel:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']
  ['cms/layout/class.tx_cms_layout.php']['tt_content_drawFooter'];
```

- Die Hook-Klasse muss das folgende Interface implementieren:

```
\TYPO3\CMS\Backend\View\PageLayoutViewDrawFooterHookInterface
```

Änderungen im System

Hooks und Signals (3)

- Es wurde ein Hook als Post-Prozessor zu `BackendUtility::countVersionsOfRecordsOnPage` hinzugefügt
- Dieser wird z.B. verwendet, um Workspace-Zustände im Seitenbaum zu visualisieren
- Der Hook kann wie folgt registriert werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']
  ['t3lib/class.t3lib_befunc.php']['countVersionsOfRecordsOnPage'][] =
  'My\Package\HookClass->hookMethod';
```

Änderungen im System

Hooks und Signals (4)

- Neues Signal wurde am Ende der Methode `DataPreprocessor::fetchRecord()` hinzugefügt
- Jenes kann dazu verwendet werden, um das Array `regTableItems_data` zu manipulieren, damit die manipulierten Daten in TCEForms angezeigt werden können

```
$this->getSignalSlotDispatcher()->dispatch(
  \TYPO3\CMS\Backend\Form\DataPreprocessor::class,
  'fetchRecordPostProcessing',
  array($this)
);
```

Änderungen im System

Hooks und Signals (5)

- Neues Signal wurde eingeführt, um zusätzlichen Code bei der Registrierung des Mailer-Objekts auszuführen (z.B. Swift Mailer Plugins)

```
$signalSlotDispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(
  \TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class
);

$signalSlotDispatcher->connect(
  \TYPO3\CMS\Core\Mail\Mailer::class,
  'postInitializeMailer',
  \Vendor\Package\Slots\MailerSlot::class,
  'registerPlugin'
);
```

Änderungen im System

Multiple UID in `PageRepository::getMenu()`

- Methode `PageRepository::getMenu()` kann nun auch ein Array aufnehmen, um meherer Root-Seiten zu definieren

```
$pageRepository = new \TYPO3\CMS\Frontend\Page\PageRepository();
$pageRepository->init(FALSE);
$rows = $pageRepository->getMenu(array(2, 3));
```

Änderungen im System

SVG Support im Core

- Der Core unterstützt nun SVG-Bilder ("Scalable Vector Graphics")
- Wenn ein SVG-Bild skaliert wird, wird kein prozessiertes Bild abgelegt, dafür aber die neuen Größenangaben in einem Datensatz `sys_file_processedfile` gespeichert
  (außer, das Bild wird - z.B. durch Cropping - weiterverarbeitet).
- Zudem wurde ein zusätzlicher Fallback eingebaut, falls ImageMagick/GraphicsMagick nicht in der Lage sein sollte, die Dimensionen zu berechnen: in diesem Fall wird das XML ausgelesen.
- SVG wurde außerdem zur Liste der zulässigen Bildtypen hinzugefügt:
  `$GLOBALS['TYPO3_CONF_VARS']['GFX']['imagefile_ext']`

Änderungen im System

Erweiterung der FAL-Treiber

- Um die Performance der Dateiliste bei (Remote-)Storages innerhalb von FAL zu erhöhen, ist es notwendig, die Sortierung und das Ermitteln der Anzahl direkt im Treiber zu erledigen. Dafür wurden zwei neue Parameter `sort` und `sortRev` eingebracht:

```
public function getFilesInFolder($folderIdentifier, $start = 0, $numberOfItems = 0,
  $recursive = FALSE, array $filenameFilterCallbacks = array(), $sort = '', $sortRev = FALSE);

public function getFoldersInFolder($folderIdentifier, $start = 0, $numberOfItems = 0,
  $recursive = FALSE, array $folderNameFilterCallbacks = array(), $sort = '', $sortRev = FALSE);
```

- Außerdem wurden zwei neue Methoden eingeführt:

```
public function getFilesInFolderCount($folderIdentifier, $recursive = FALSE,
  array $filenameFilterCallbacks = array());

public function getFoldersInFolderCount($folderIdentifier, $recursive = FALSE,
  array $folderNameFilterCallbacks = array());
```

Änderungen im System

Backend Routing API (1)

- Der Core enthält nun eine Backend Routing API, welche die Entry Points ins Backend verwaltet
- Die Routing API wurde vom Symfony Routing Framework inspiriert und ist weitgehend kompatibel mit dieser
  (auch wenn für TYPO3 zur Zeit nur ca. 20% genutzt werden)
- Grundsätzlich existieren hierfür drei Klassen:
  - `class Route`: enthält Angaben zum Pfad und Optionen
  - `class Router`: API, um die Route zu matchen
  - `class UrlGenerator`: generiert die URL

Änderungen im System

Backend Routing API (2)

- Routen werden dabei in folgender Datei in der entsprechenden Extension definiert: `Configuration/Backend/Routes.php` (siehe Systemextension `backend` als Beispiel)
- Weitere Informationen zur Backend Routing API: http://wiki.typo3.org/Blueprints/BackendRouting

Änderungen im System

Neue Systemextension für Media Inhaltselemente

- Neue Systemextension "`mediace`" enthält folgende cObjects:
  - `MULTIMEDIA`
  - `MEDIA`
  - `SWFOBJECT`
  - `FLOWPLAYER`
  - `QTOBJECT`
- Die Inhaltselemente `media` und `multimedia` wurden ebenfalls in die Systemextension verschoben, ebenso der "Media Wizard Provider"
- Die Extension ist standardmäßig **nicht** installiert!

Änderungen im System

Fremd-Bibliotheken an neuer Stelle

- Sämtliche Fremd-Bibliotheken werden nicht mehr in `Packages/Libraries`, sondern in `typo3/contrib/vendor` abgelegt
- Grundsätzlich ist dafür die Installation der Bibliotheken mittels `composer install` notwendig
- Probleme gibt es beim Upgrade einer Installation, wenn dort `phpunit` verwendet wurde! Dies kann wie folgt behoben werden:

```
# cd htdocs/
# rm -rf typo3/contrib/vendor/ bin/ Packages/Libraries/ composer.lock
# composer install
```

Änderungen im System

API für JavaScript Notifikationen

- Neue API, um JavaScript Notifikationen zu erzeugen:

```
// Bisheriger (veralteter) Weg:
top.TYPO3.Flashmessages.display(TYPO3.Severity.notice)

// Neuer Weg:
top.TYPO3.Notification.notice(title, message)
```

- Es existieren folgende API-Funktionen:
  (Parameter `duration` ist optional und standardmäßig auf 5s eingestellt)
  - `top.TYPO3.Notification.notice(title, message, duration)`
  - `top.TYPO3.Notification.info(title, message, duration)`
  - `top.TYPO3.Notification.success(title, message, duration)`
  - `top.TYPO3.Notification.warning(title, message, duration)`
  - `top.TYPO3.Notification.error(title, message, duration)`

Änderungen im System

Systeminformationen (1)

- Das Dropdown mit Systeminformationen kann über folgenden Slot erweitert werden:

```
$signalSlotDispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(
  \TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class);

$signalSlotDispatcher->connect(
  \TYPO3\CMS\Backend\Backend\ToolbarItems\SystemInformationToolbarItem::class,
  'getSystemInformation',
  \Vendor\Extension\SystemInformation\Item::class,
  'getItem'
);
```

Änderungen im System

Systeminformationen (2)

- Zur Ansprache benötigt man die Klasse `Item` und dazugehörig die Methode `getItem()` innerhalb einer Extension

  `EXT:extension\Classes\SystemInformation\Item.php`:

```
class Item {
  public function getItem() {
    return array(array(
      'title' => 'The title shown on hover',
      'value' => 'Description shown in the list',
      'status' => SystemInformationHookInterface::STATUS_OK,
      'count' => 4,
      'icon' => \TYPO3\CMS\Backend\Utility\IconUtility::getSpriteIcon(
    'extensions-example-information-icon')
    ));
  }
}
```

Änderungen im System

Systeminformationen (3)

- Das Icon `extensions-example-information-icon` wird in der Datei `ext_localconf.php` registriert:

```
\TYPO3\CMS\Backend\Sprite\SpriteManager::addSingleIcons(
  array(
    'information-icon' => \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::extRelPath(
      $_EXTKEY) . 'Resources/Public/Images/Icons/information-icon.png'
    ),
   $_EXTKEY
);
```

Änderungen im System

Systeminformationen (4)

- Nachrichten werden am unteren Ende des Dropdowns angezeigt
- Über den folgenden Slot können eigene Nachrichten eingebracht werden:

```
$signalSlotDispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(
  \TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class);

$signalSlotDispatcher->connect(
  \TYPO3\CMS\Backend\Backend\ToolbarItems\SystemInformationToolbarItem::class,
  'loadMessages',
  \Vendor\Extension\SystemInformation\Message::class,
  'getMessage'
);
```

Änderungen im System

Systeminformationen (5)

- Zur Ansprache benötigt man die Klasse `Message` und dazugehörig die Methode `getMessage()` innerhalb einer Extension

  `EXT:extension\Classes\SystemInformation\Message.php`:

```
class Message {
  public function getMessage() {
    return array(array(
      'status' => SystemInformationHookInterface::STATUS_OK,
      'text' => 'Something went wrong. Take a look at the reports module.'
    ));
  }
}
```

Änderungen im System

Einstellungen für Bild-Manipulation (1)

- Folgende Einstellungen können über TypoScript getätigt werden:

```
# Cropping fuer alle Bilder deaktivieren
tt_content.image.20.1.file.crop =

# Ueberschreiben/Setzen der Cropping Eigenschaften
# offsetX,offsetY,width,height
tt_content.image.20.1.file.crop = 50,50,100,100
```

Änderungen im System

Einstellungen für Bild-Manipulation (2)

- Das Cropping kann auch in Fluid verwendet werden:

```
# Cropping fuer alle Bilder deaktivieren
<f:image image="{imageObject}" crop="" ></f:image>

# Ueberschreiben/Setzen der Cropping Eigenschaften
# offsetX,offsetY,width,height
<f:image image="{imageObject}" crop="50,50,100,100" ></f:image>
```

Änderungen im System

Einstellungen für Bild-Manipulation (3)

- Im TCA wird das Image-Cropping wie folgt zur Verfügung gestellt:
  - Column Type: `image_manipulation`
  - Config `file_field`: string (default: `uid_local`)
  - Config `enableZoom`: boolean (default: `FALSE`)
  - Config `allowedExtensions`: string
    (default: `$GLOBALS['TYPO3_CONF_VARS']['GFX']['imagefile_ext']`)
  - Config `ratios`: array, default:

```
array(
  '1.7777777777777777' => '16:9',
  '1.3333333333333333' => '4:3',
  '1' => '1:1',
  'NaN' => 'Free'
)
```

Änderungen im System

Zusätzliche Parameter für HTMLparser userFunc

- Die userFunc im HTMLparser kann nun zusätzliche Parameter aufnehmen:

```
myobj = TEXT
myobj.value = <a href="/" class="myclass">MyText</a>
myobj.HTMLparser.tags.a.fixAttrib.class {
  userFunc = Tx\MyExt\Myclass->htmlUserFunc
  userFunc.myparam = test
}
```

- Diese können in einer Extension wie folgt abgerufen werden:

```
function htmlUserFunc(array $params, HtmlParser $htmlParser) {
  // $params['attributeValue'] enthaelt den Wert der
  // verarbeiteten Eigenschaft - hier also "myclass"
  // $params['myparam'] enthaelt den Wert "test"
  ...
}
```

Änderungen im System

Locking API (1)

- Es wurde eine neue Locking-API eingeführt, welche verschiedene Locking-Methoden (SimpleFile, Semaphore, ...) zur Verfügung stellt
- Eine Locking-Methode muss dabei das `LockingStrategyInterface` implementieren

```
$lockFactory = GeneralUtility::makeInstance(LockFactory::class);
$locker = $lockFactory->createLocker('someId');
$locker->acquire() || die('Could not acquire lock.');
...
$locker->release();
```

Änderungen im System

Locking API (2)

- Man kann außerdem "non-blocking" Locks realisieren:

```
$lockFactory = GeneralUtility::makeInstance(LockFactory::class);
$locker = $lockFactory->createLocker(
  'someId',
  LockingStrategyInterface::LOCK_CAPABILITY_SHARED |
    LockingStrategyInterface::LOCK_CAPABILITY_NOBLOCK
);
try {
  $result = $locker->acquire(LockingStrategyInterface::LOCK_CAPABILITY_SHARED |
        LockingStrategyInterface::LOCK_CAPABILITY_NOBLOCK);
  catch (\RuntimeException $e) {
  if ($e->getCode() === 1428700748) {
    // einige Prozesse haben noch ein Lock
    // daher sollte etwas in der Zwischenzeit gemacht werden
    ...
  }
}
if ($result) {
  $locker->release();
}
```

Änderungen im System

Signal nach Installation von Extensions

- In der Methode

  `\TYPO3\CMS\Extensionmanager\Utility\InstallUtility::install()`

  wurde ein Signal eingebaut, welches emmitiert wird, sobald eine Extension fertig installiert ist und alle Imports/Updates durchgelaufen sind

```
// Aufruf
$this->emitAfterExtensionInstallSignal($extensionKey);

// Methode
protected function emitAfterExtensionInstallSignal($extensionKey) {
  $this->signalSlotDispatcher->dispatch(
    __CLASS__,
    'afterExtensionInstall',
    array($extensionKey, $this)
  );
}
```

Änderungen im System

Registry für Text-Extraktoren

- Der Core enthält nun eine Registry um Text-Extraktoren anzumelden
- Dabei prüft `canExtractText()` ob eine Extrahierung möglich ist und `extractText()` führt diese durch
- Die Registrierung erfolgt in `ext_localconf.php`:

```
$textExtractorRegistry = \TYPO3\CMS\Core\Resource\TextExtraction\TextExtractorRegistry::
      getInstance();
$textExtractorRegistry->registerTextExtractor(
  \TYPO3\CMS\Core\Resource\TextExtraction\PlainTextExtractor::class
);
```

- Die Verwendung erfolgt folgendermaßen:

```
$textExtractorRegistry = \TYPO3\CMS\Core\Resource\TextExtraction\TextExtractorRegistry::
      getInstance();
$extractor = $textExtractorRegistry->getTextExtractor($file);
if($extractor !== NULL) {
  $content = $extractor->extractText($file);
}
```

Änderungen im System

Diverses

- Alle Web-Bibliotheken (wie z.B. Twitter Bootstrap, jQuery, Font Awesome usw.) verwenden nun "Bower" (`http://bower.io`) zur Installation, und sind nicht mehr im TYPO3 Core Git enthalten

  `bower install` führt eine Installation durch

  `bower update` führt ein Update durch

  (die zugehörige Datei `bower.json` befindet sich im Verzeichnis `Build`)
- Ein laufender Scheduler Task kann nun in der Kommandozeile mit der Option `-s` wieder gestoppt werden
- Der "Processing" Ordner eines Storages kann nun auch außerhalb von diesem liegen (z.B. bei read-only Storages)
- Man kann nun auf die ID der ursprünglich angefragten Seite über das TSFE zugreifen: `$TSFE->getRequestedId()`

Änderungen im System

Symfony/Console im CommandController (1)

Der CommandController verwendet nun intern Symfony/Console und stellt damit verschiedene Methoden zur Verfügung:

- TableHelper
  - `outputTable($rows, $headers = NULL)`
- DialogHelper
  - `select($question, $choices, $default = NULL, $multiSelect = false, $attempts = FALSE)`
  - `ask($question, $default = NULL, array $autocomplete = array())`
  - `askConfirmation($question, $default = TRUE)`
  - `askHiddenResponse($question, $fallback = TRUE)`
  - `askAndValidate($question, $validator, $attempts = FALSE, $default = NULL, array $autocomplete = NULL)`
  - `askHiddenResponseAndValidate($question, $validator, $attempts = FALSE, $fallback = TRUE)`

Änderungen im System

Symfony/Console im CommandController (2)

- ProgressHelper
  - `progressStart($max = NULL)`
  - `progressSet($current)`
  - `progressAdvance($step = 1)`
  - `progressFinish()`

(siehe Code-Beispiel auf den folgenden Slides)

Änderungen im System

Symfony/Console im CommandController (3)

```
<?php
namespace Acme\Demo\Command;
use TYPO3\CMS\Extbase\Mvc\Controller\CommandController;

class MyCommandController extends CommandController {
  public function myCommand() {

    // Table rendern
    $this->output->outputTable(array(
      array('Bob', 34, 'm'),
      array('Sally', 21, 'f'),
      array('Blake', 56, 'm')
    ),
    array('Name', 'Age', 'Gender'));

    // Selektieren
    $colors = array('red', 'blue', 'yellow');
    $selectedColorIndex = $this->output->select('Please select one color', $colors, 'red');
    $this->outputLine('You choose the color %s.', array($colors[$selectedColorIndex]));

    [...]
```

Änderungen im System

Symfony/Console im CommandController (4)

```
    [...]
    // Abfrage
    $name = $this->output->ask('What is your name?' . PHP_EOL, 'Bob', array('Bob', 'Sally', 'Blake'));
    $this->outputLine('Hello %s.', array($name));

    // Prompt
    $likesDogs = $this->output->askConfirmation('Do you like dogs?');
    if ($likesDogs) {
      $this->outputLine('You do like dogs!');
    }

    // Progress
    $this->output->progressStart(600);
    for ($i = 0; $i < 300; $i ++) {
      $this->output->progressAdvance();
      usleep(5000);
    }
    $this->output->progressFinish();
  }
}
?>
```


TYPO3

Änderungen im System

Backend Login API (1)

- Das Backend-Login wurde nun komplett als API realisiert und lässt sich damit selbst per Programmierung ansprechen
- Grundsätzlich muss man dazu einen sogenannten Login-Provider in der Datei `ext_localconf.php` registrieren:

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['backend']['loginProviders'][1433416020] = [
  'provider' => \TYPO3\CMS\Backend\LoginProvider\UsernamePasswordLoginProvider::class,
  'sorting' => 50,
  'icon-class' => 'fa-key',
  'label' => 'LLL:EXT:backend/Resources/Private/Language/locallang.xlf:login.link'
];
```

Änderungen im System

Backend Login API (2)

- Die Optionen sind wie folgt definiert:
  - `provider`:
    Name der Login-Provider Klasse, die das Interface `TYPO3\CMS\Backend\LoginProvider\LoginProviderInterface` implementiert
  - `sorting`:
    Sortierung, der die Reihenfolge der Links zum Login-Provider festlegt
  - `icon-class`:
    Font-Awesome Icon-Name für den Link auf dem Login-Screen
  - `label`:
    Label für den Link auf dem Login-Screen

Änderungen im System

Backend Login API (3)

- Das `LoginProviderInterface` enthält lediglich die Methode
  `public function render(StandaloneView $view, PageRenderer $pageRenderer, LoginController $loginController);`
- Die Parameter sind wie folgt definiert:
  - `$view`:
    Der Fluid-StandaloneView, welcher den Login-Screen rendert. Hier muss das Template-File gesetzt und die Variablen übergeben werden, die man benötigt.
  - `$pageRenderer`:
    PageRenderer-Instanz, welche z.B. JavaScript-Ressourcen einbringen kann.
  - `$loginController`:
    Instanz des Login-Controllers.

Änderungen im System

Backend Login API (4)

- Das Template muss `<f:layout name="Login">` und `<f:section name="loginFormFields">` (für die Formular-Felder) enthalten:

```
<f:layout name="Login" />
<f:section name="loginFormFields">
  <div class="form-group t3js-login-openid-section" id="t3-login-openid_url-section">
    <div class="input-group">
      <input type="text" id="openid_url"
        name="openid_url"
        value="{presetOpenId}"
        autofocus="autofocus"
        placeholder="{f:translate(key: 'openId', extensionName: 'openid')}"
        class="form-control input-login t3js-clearable t3js-login-openid-field" />
      <div class="input-group-addon">
        <span class="fa fa-openid"></span>
      </div>
    </div>
  </div>
</f:section>
```

Änderungen im System

`CategoryRegistry` mit neue Optionen

- Die Methode `CategoryRegistry->addTcaColumn` hat Optionen erhalten, um sowohl `l10n_mode` als auch `l10n_display` zu setzen:

```
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::makeCategorizable(
  $extensionKey,
  $tableName,
  'categories',
  array(
    'l10n_mode' => 'string (keyword)',
    'l10n_display' => 'list of keywords'
  )
);
```

Änderungen im System

Sprites in Backend Modulen

- Backend Module (z.B. Hauptmodule wie "Web" sowie Untermodule wie "Filelist") können nun Sprites als Icons verwenden
  (nur Sprites, die TYPO3 bekannt sind!)
- Beispiel:

```
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addModule(
  'web',
  'layout',
  'top',
  \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::extPath($_EXTKEY) . 'Modules/Layout/',
  array(
    'script' => '_DISPATCH',
    'access' => 'user,group',
    'name' => 'web_layout',
    'configuration' => array('icon' => 'module-web'),
    'labels' => array(
      'll_ref' => 'LLL:EXT:cms/layout/locallang_mod.xlf',
    ),
  )
);
```

Änderungen im System

FormEngine NodeFactory API (1)

- Es ist nun möglich, neue Nodes zu registrieren und bestehende zu überschreiben

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['formEngine']['nodeRegistry'][1433196792] = array(
  'nodeName' => 'input',
  'priority' => 40,
  'class' => \MyVendor\MyExtension\Form\Element\T3editorElement::class
);
```

- Dies registriert eine neue Klasse, die den TCA-Type `input` rendert und das `NodeInterface` implementieren muss
- Als Array-Key wird der Unix-Timestamp des Zeitpunkts verwendet, wenn man das Element zufügt

Änderungen im System

FormEngine NodeFactory API (2)

- Wenn mehrere Registry-Elemente für den selben Typ registriert werden, gewinnt das, mit der höchsten Priorität (0-100)
- Ein neuer TCA-Type wird wie folgt registriert:

TCA

```
'columns' => array(
  'bodytext' => array(
    'config' => array(
      'type' => 'text',
      'renderType' => '3dCloud',
    ),
  ),
)
```

ext_localconf.php

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['formEngine']['nodeRegistry'][1433197759] = array(
  'nodeName' => '3dCloud',
  'priority' => 40,
  'class' => \MyVendor\MyExtension\Form\Element\ShowTextAs3dCloudElement::class
);
```

Änderungen im System

Signal `postProcessMirrorUrl` wurde verschoben

- Die Klasse für das Signal `postProcessMirrorUrl` wurde geändert

BREAKING CHANGE!

- Mit folgendem Code kann man die Klasse je nach TYPO3-Version ansprechen:

```
$signalSlotDispatcher->connect(
  version_compare(TYPO3_version, '7.0', '<')
    ? 'TYPO3\\CMS\\Lang\\Service\\UpdateTranslationService'
    : 'TYPO3\\CMS\\Lang\\Service\\TranslationService',
  'postProcessMirrorUrl',
  'Vendor\\Extension\\Slots\\CustomMirror',
  'postProcessMirrorUrl'
);
```

Änderungen im System

Driver Interface

- Zum `DriverInterface` wurden die folgenden Methoden hinzugefügt:
  - `getFolderInFolder`
  - `getFileInFolder`
- Jeder eigene FAL-Driver muss daher diese beiden Methoden nachimplementieren:

```
public function getFoldersInFolder(
  $folderIdentifier,
  $start = 0,
  $numberOfItems = 0,
  $recursive = FALSE,
  array $folderNameFilterCallbacks = array(),
  $sort = '',
  $sortRev = FALSE
);
```

```
public function getFileInFolder(
  $fileName,
  $folderIdentifier
);
```

BREAKING CHANGE!

Änderungen im System

Unterstützung von IEC/SI-Keywords für Größen

- Die Formatierung von Größen unterstützt nun zwei Keywords, um die Einheiten festzulegen:
  - `iec` (default)
    (Basis: 2, Labels: `| Ki| Mi| Gi| Ti| Pi| Ei| Zi| Yi`)
  - `si`
    (Basis: 10, Labels: `| k| M| G| T| P| E| Z| Y`)
- Gesetzt werden kann die Formatierung z.B. via TypoScript:
  `bytes.labels = iec`

```
echo GeneralUtility::formatSize(85123);
// => Vorher "83.1 K"
// => Nachher "83.13 Ki"
```

Änderungen im System

Dependency Ordering Service (1)

- Oftmals ist es notwendig eine sortierte Liste an Items zur Verfügung zu stellen, deren Einträge einerseits Abhängigkeiten haben und andererseits dazu verwendet werden, um Aktionen in eben dieser Reihenfolge auszuführen.
- Im Core findet jenes beispielsweise Verwendung bei:
  - Reihenfolge der Hook-Ausführung,
  - Ladereihenfolge von Extensions,
  - Reihefolge der Anzeige von Menü-Einträgen,
  - usw.
- Eine Überarbeitung des bisherigen `DependencyResolver` stellt nun den `DependencyOrderingService` zur Verfügung

Änderungen im System

Dependency Ordering Service (2)

- Anwendung:

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['someExt']['someHook'][<some id>] = [
  'handler' => someClass::class,
  'runBefore' => [ <some other ID> ],
  'runAfter' => [ ... ],
  ...
];
```

- Zum Beispiel:

```
$hooks = $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['someExt']['someHook'];
$sorted = GeneralUtility:makeInstance(DependencyOrderingService::class)->orderByDependencies(
  $hooks, 'runBefore', 'runAfter'
);
```

Änderungen im System

Hooks und Signals (1)

- Ein neuer Hook wurde am Ende von `InlineRecordContainer::checkAccess` hinzugefügt, mit dem der Zugriff von Inline-Records geprüft werden kann
- Der Hook kann wie folgt registriert werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_tceforms_inline.php']
  ['checkAccess'][] = 'My\\Package\\HookClass->hookMethod';
```

Änderungen im System

Hooks und Signals (2)

- Ein neuer Hook wurde am Ende von `AbstractUserAuthentication::checkAuthentication` hinzugefügt, mit dem man fehlgeschlagene Anmeldeversuche verarbeiten kann
- Standardmäßig wartet der Prozess 5 Sekunden nachdem eine Anmeldung fehlgeschlagen ist
- Über den Hook kann ein anderes Verhalten implementiert werden (z.B. zur Abwehr von Brute Force Angriffen)
- Der Hook kann wie folgt registriert werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_userauth.php']
  ['postLoginFailureProcessing'][] = 'My\\Package\\HookClass->hookMethod';
```

Änderungen im System

Hooks und Signals (3)

- Das neue Signal `recordMarkedAsMissing` wird ausgesendet, wenn der FAL Indexer auf einen `sys_file` Eintrag stößt, dessen Datei im Dateisystem aber nicht auffindbar ist. Dabei wird die `sys_file` UID übermittelt.
- Jenes kann in Extensions verwendet werden, die Dienste rund um das Datei-Management anbieten (wie beispielsweise Versionierung, Synchronisation, Recovery, usw.)
- Das Signal `afterMappingSingleRow` wird ausgesendet, wann immer der DataMapper ein Objekt erstellt

Änderungen im System

HTML in TypoLink-Titeln

- Anführungszeichen in TypoLink-Titeln werden nun automatisch "escaped"
- Ein eventuell bereits existierendes Escaping wird daher nun falsch dargestellt:
  Aus `'Some &quot;special&quot; title'`
  wird `'Some &amp;quot;special&amp;quot; title'`
- Es wird empfohlen, hier auf Escaping komplett zu verzichten, da sich TYPO3 nun darum kümmert

BREAKING CHANGE!

Änderungen im System

Diverse Änderungen (1)

- Mit `Files->replace` gibt eine neue Berechtigung für Backend Benutzer, um Dateien im Modul Dateiliste zu ersetzen
- Der Dateinamen des Logfiles, welches der FileWriter schreibt, ändert sich wie folgt:
  - bisher: `typo3temp/logs/typo3.log`
  - neu: `typo3temp/logs/typo3_<hash>.log`

  (der Wert `<hash>` berechnet sich aus dem Encryptionkey)

Änderungen im System

Diverse Änderungen (2)

- Die in Hooks verwendeten Klassen müssen ab sofort dem Autoloading-Mechanismus folgen
- Daher kann die Hook-Definition auch verkürzt werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['tce']['formevals']
  [\TYPO3\CMS\Saltedpasswords\Evaluation\FrontendEvaluator::class] = '';
```

BREAKING CHANGE!

Änderungen im System

Fluid-basierte Inhaltselemente (1)

- Es wurde eine Alternative zur Extension *CSS Styled Content* geschaffen: **"Fluid-based Content Elements"**
- Hier werden anstelle von TypoScript Fluid-Templates für das Rendering von Inhalten verwendet
- Dazu müssen die folgenden beiden static-Templates eingebunden werden:
  - Content Elements (`fluid_styled_content`)
  - Content Elements CSS (optional) (`fluid_styled_content`)

Änderungen im System

Fluid-basierte Inhaltselemente (2)

- Zusätzlich muss das PageTSconfig Template `Fluid-based Content Elements fluid_styled_content` in den Seiteneigenschaften eingebunden werden, damit der New-Content-Element Wizard entsprechend angepasst wird
- Eigene Fluid-Templates können wie folgt festgelegt werden:

```
lib.fluidContent.templateRootPaths.50 = EXT:site_example/Resources/Private/Templates/
lib.fluidContent.partialRootPaths.50 = EXT:site_example/Resources/Private/Partials/
lib.fluidContent.layoutRootPaths.50 = EXT:site_example/Resources/Private/Layouts/
```

Änderungen im System

Fluid-basierte Inhaltselemente (3)

- Um eine Installation auf die neue Struktur zu migrieren, kann man wie folgt vorgehen:
  - Deinstallieren der Extension `css_styled_content`
  - Installieren der Extension `fluid_styled_content`
  - Nun ist ein "Upgrade Wizard" im Install Tool verfügbar, der die Migration der Inhaltselemente `text`, `image` und `textpic` in `textmedia`, durchführt

Änderungen im System

SELECTmmQuery Methode für Datenbank-Zugang

- Bislang enthielt die Datenbank-Klasse die Methode `exec_SELECT_mm_query`, die die Datenbank-Abfrage direkt ausführte
- Nun wurde die Generierung des Queries (*Query-Building*) und Ausführung getrennt, indem die Methode `SELECT_mm_query` hinzugefügt wurde

```
$query = SELECT_mm_query('*', 'table1', 'table1_table2_mm', 'table2', 'AND table1.uid = 1',
'', 'table1.title DESC');
```

Änderungen im System

Scheduler Task zur Datenbank-Optimierung

- Es wurde ein Scheduler Task implementiert, der die Datenbank via MySQL-Kommando `OPTIMIZE TABLE` optimiert
- Optimiert werden können lediglich Tabellen vom Typ `MyISAM`, `InnoDB` und `ARCHIVE`
- DBAL wird nicht unterstützt

Änderungen im System

Online Medien Unterstützung (1)

- Der Core wurde um eine externe Medien-Unterstützung erweitert (exemplarisch für YouTube- und Vimeo-Videos)
- Diese kann (z.B. im Inhaltselement **"Text & Media"**) als URL eingeben werden. Anschließend wird die Resource wie eine interne Datei integriert.

Änderungen im System

Online Medien Unterstützung (2)

Folgende YouTube/Vimeo URLs sind möglich:

youtu.be/<code>
www.youtube.com/watch?v=<code>
www.youtube.com/v/<code>
www.youtube-nocookie.com/v/<code>
www.youtube.com/embed/<code>

vimeo.com/<code>
player.vimeo.com/video/<code>

Änderungen im System

Online Medien Unterstützung (3)

- Der Zugriff per Fluid kann z.B. wie folgt durchgeführt werden:

```
<!-- enable js api and set no-cookie support for YouTube videos -->
<f:media file="{file}" additionalConfig="{enablejsapi:1, 'no-cookie': true}" ></f:media>

<!-- show title and uploader for YouTube and Vimeo before video starts playing -->
<f:media file="{file}" additionalConfig="{showinfo:1}" ></f:media>
```

- Für YouTube existieren folgende Optionen:

  `autoplay`, `controls`, `loop`, `enablejsapi`, `showinfo`, `no-cookie`

- Für Vimeo existieren folgende Optionen:

  `autoplay`, `loop`, `showinfo`

Änderungen im System

Online Medien Unterstützung (4)

- Für einen eigenen Media-Service benötigt man eine `OnlineMediaHelper` Klasse, welche das `OnlineMediaHelperInterface` implementiert, sowie eine `FileRenderer` Klasse, die das `FileRendererInterface` implementiert

```
// Registrierung eines eigenen Online-Video-Services
$GLOBALS['TYPO3_CONF_VARS']['SYS']['OnlineMediaHelpers']['myvideo'] =
  \MyCompany\Myextension\Helpers\MyVideoHelper::class;

$rendererRegistry = \TYPO3\CMS\Core\Resource\Rendering\RendererRegistry::getInstance();
$rendererRegistry->registerRendererClass(
  \MyCompany\Myextension\Rendering\MyVideoRenderer::class
);

// Registrierung eines eigenen Mime-Types
$GLOBALS['TYPO3_CONF_VARS']['SYS']['FileInfo']['fileExtensionToMimeType']['myvideo'] =
  'video/myvideo';

// Registrierung einer eigenen Datei-Extension
$GLOBALS['TYPO3_CONF_VARS']['SYS']['mediafile_ext'] .= ',myvideo';
```

Änderungen im System

Backend Routing

- Es wurde eine neue Routing Komponente zum TYPO3-Backend hinzugefügt, welche verschiedene Aufrufe handhaben kann (z.B. `http://www.example.com/typo3/document/edit`)
- Die Routen werden in folgender Datei definiert:

  `Configuration/Backend/Routes.php`

```
return [
  'myRouteIdentifier' => [
    'path' => '/document/edit',
    'controller' => Acme\MyExtension\Controller\MyExampleController::class . '::methodToCall'
  ]
];
```

- Die Methode erhält das Response- und Request-Objekt:

```
public function methodToCall(ServerRequestInterface $request, ResponseInterface $response) {
  ...
}
```

Änderungen im System

Autoload Definition in `ext_emconf.php`

- Zusätzlich zur Datei `composer.json` können nun Autoload-Definitionen in der Datei `ext_emconf.php` hinterlegt werden
- Das hat den Vorteil, dass nicht die gesamte Extension nach Klassen gescannt wird

```
$EM_CONF[$_EXTKEY] = array (
  'title' => 'Extension Skeleton for TYPO3 CMS 7',
  ...
'autoload' =>
  array(
    'psr-4' => array(
      'Helhum\\ExtScaffold\\' => 'Classes'
    )
  )
);
```

Änderungen im System

Neue Icon-Factory (1)

- Die Logik, um mit Icons, Größen und Overlays zu arbeiten, wurde in die neue `IconFactory` ausgelagert
- Es gibt drei "IconProvider": `BitmapIconProvider`, `FontawesomeIconProvider` und `SvgIconProvider`
- Die Registrierung eines Icons erfolgt folgendermaßen:

```
IconRegistry::registerIcon($identifier, $iconProviderClassName, array $options = array());
```

Änderungen im System

Neue Icon-Factory (2)

- Anwendung:

```
$iconFactory = GeneralUtility::makeInstance(IconFactory::class);
$iconFactory->getIcon(
  $identifier,
  Icon::SIZE_SMALL,
  $overlay,
  IconState::cast(IconState::STATE_DEFAULT)
)->render();
```

- Zulässige Werte für `Icon::SIZE_...` sind:
  `SIZE_SMALL`, `SIZE_DEFAULT` und `SIZE_LARGE`

- Zulässige Werte für `Icon::STATE_...` sind:
  `STATE_DEFAULT` und `STATE_DISABLED`

Änderungen im System

Neue Icon-Factory (3)

- Der Core stellt einen eigenen ViewHelper zur Verfügung, um Icons anzuzeigen:

```
{namespace core = TYPO3\CMS\Core\ViewHelpers}

<core:icon identifier="my-icon-identifier"></core:icon>

<!-- use the "small" size if none given ->
<core:icon identifier="my-icon-identifier"></core:icon>
<core:icon identifier="my-icon-identifier" size="large"></core:icon>
<core:icon identifier="my-icon-identifier" overlay="overlay-identifier"></core:icon>

<core:icon identifier="my-icon-identifier" size="default" overlay="overlay-identifier">
</core:icon>

<core:icon identifier="my-icon-identifier" size="large" overlay="overlay-identifier">
</core:icon>
```

Änderungen im System

Hooks und Signals (1)

- Es wurde ein Signal im LinkValidator zugefügt, welches die zusätzliche Verarbeitung eines Eintrages möglich macht
  (z.B. um Daten aus der Plugin-Konfiguration zu ermitteln o.ä.).
- Der Hook kann wie folgt in der Datei `ext_localconf.php` registriert werden:

```
$signalSlotDispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(
  \TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class
);
$signalSlotDispatcher->connect(
  \TYPO3\CMS\Linkvalidator\LinkAnalyzer::class,
  'beforeAnalyzeRecord',
  \Vendor\Package\Slots\RecordAnalyzerSlot::class,
  'beforeAnalyzeRecord'
);
```

Änderungen im System

JumpUrl als System-Extension (1)

- Die Erzeugung und das Handling von JumpURLs wurde aus der Frontend-Extension entfernt und zur neuen System-Extension `jumpurl` verschoben
- Hook zur Manipulation von **URLs** in der Datei `ext_localconf.php`:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['urlProcessing']['urlHandlers']
  ['myext_myidentifier']['handler'] = \Company\MyExt\MyUrlHandler::class;

// Die Klasse muss das UrlHandlerInterface implementieren
class MyUrlHandler implements \TYPO3\CMS\Frontend\Http\UrlHandlerInterface {
  ...
}
```

BREAKING CHANGE!

TYPO3

Änderungen im System

JumpUrl als System-Extension (2)

- Handling von **Links** in der Datei `ext_localconf.php`:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['urlProcessing']['urlProcessors']
  ['myext_myidentifier']['processor'] = \Company\MyExt\MyUrlProcessor::class;

// Die Klasse muss das UrlProcessorInterface implementieren
class MyUrlProcessor implements \TYPO3\CMS\Frontend\Http\UrlProcessorInterface {
  ...
}
```

BREAKING CHANGE!

Änderungen im System

Kommandozeilenaufruf (CLI)

- Sollte es beim Aufruf von `typo3/cli_dispatch.phpsh` zu Fehlern kommen, so werden diese farbig dargestellt
- CommandController können nun auch in Unterordnern liegen
- Beispiel:

  Controller in Datei:

  `my_ext/Classes/Command/Hello/WorldCommandController.php`

  ...kann im CLI wie folgt aufgerufen werden:

  `typo3/cli_dispatch.sh extbase my_ext:hello:world <arguments>`

TYPO3

Änderungen im System

Diverse Änderungen (1)

- Die Verschieben-Buttons beim TCA-Type `group` können nun mit der TCA-Option `hideMoveIcons = TRUE` deaktiviert werden
- Die Funktion `makeCategorizable()` kann nun überschrieben werden, sofern diese vorher bereits aufgerufen wurde (z.B. für `tt_content`).
- Beispiel:

```
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::makeCategorizable(
  'css_styled_content', 'tt_content', 'categories', array(), TRUE
);
```

Der letzte Parameter steuert das Überschreiben (hier: `TRUE`).

Standardmäßig ist der Wert `FALSE`.

Änderungen im System

Diverse Änderungen (2)

- Es gibt nun eine Funktion, um eine Unique-ID zu erzeugen

```
$uniqueId = \TYPO3\CMS\Core\Utility\StringUtility::getUniqueId('Prefix');
```

- Als Plaintext Dateiendung wurde `typoscript` hinzugefügt
- Es gibt nun eine neue Konfigurations-Option, die regelt, welche Dateiendungen als Media-Dateien interpretiert werden:

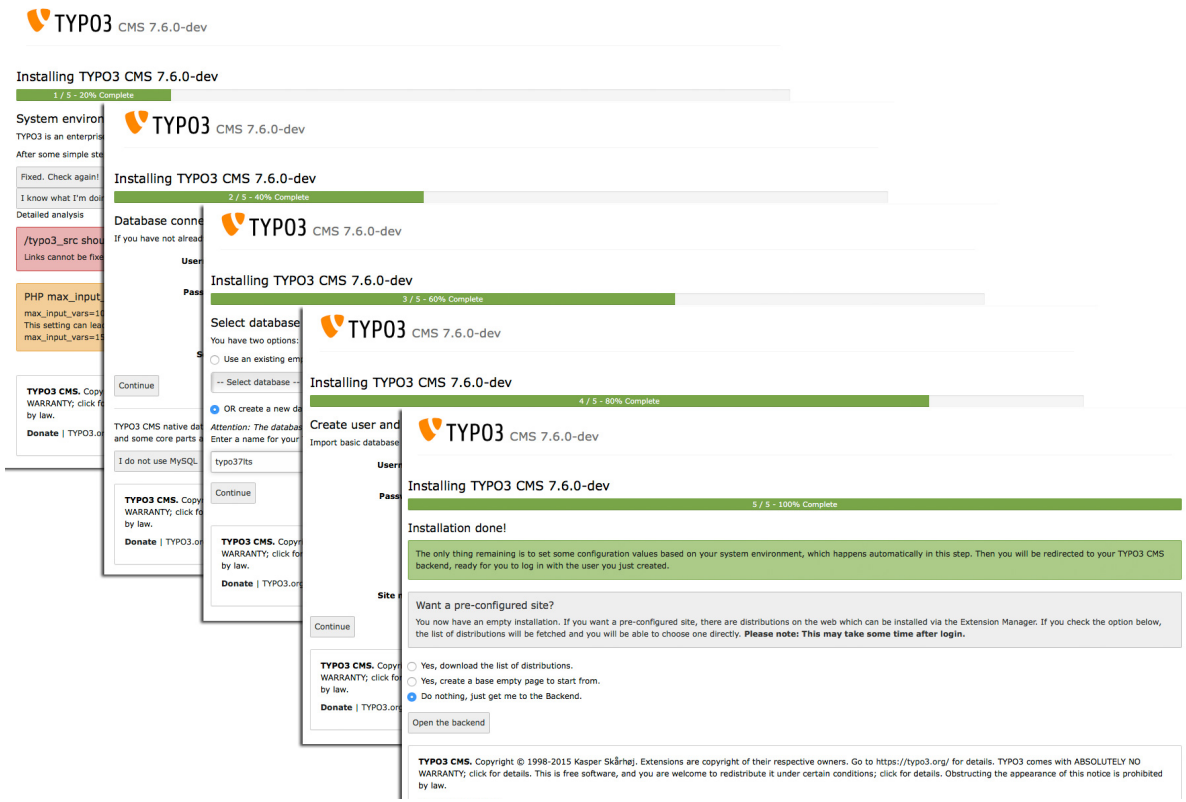
```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['mediafile_ext'] =
  'gif,jpg,jpeg,bmp,png,pdf,svg,ai,mov,avi';
```

BREAKING CHANGE!

Änderungen im System

Bootstrap für Install Tool(1)

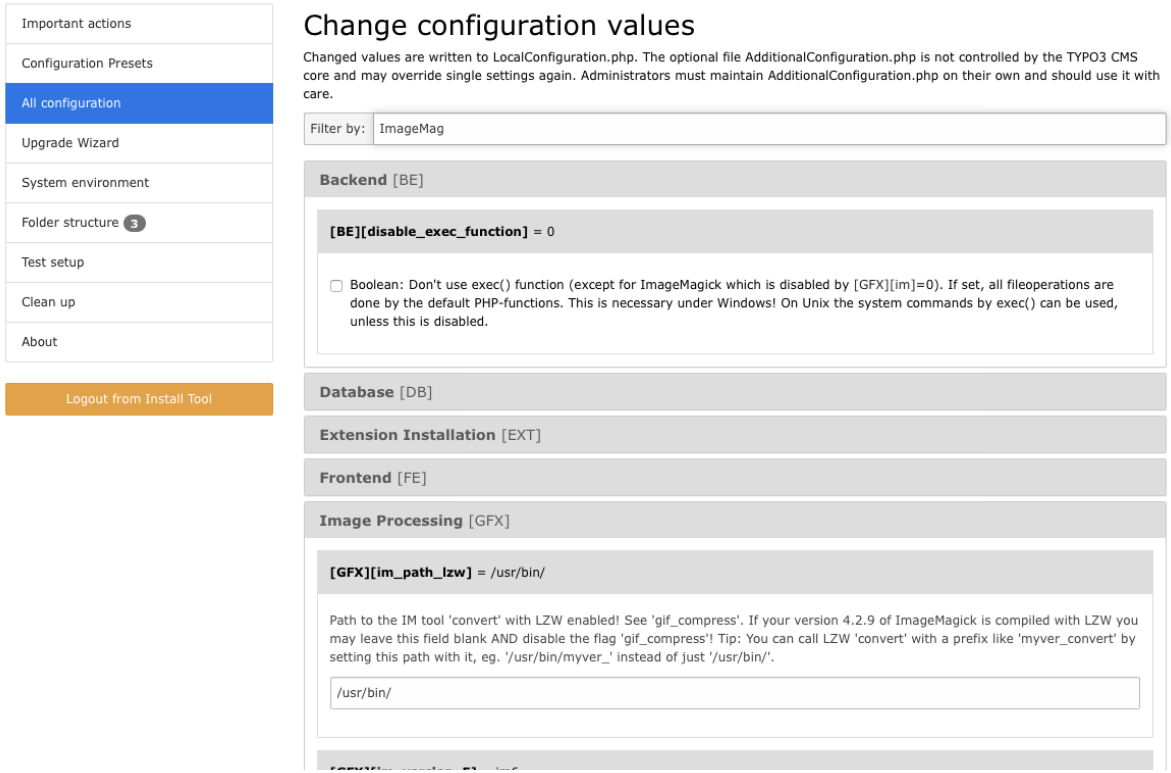
- Das Install Tool basiert nun komplett auf Bootstrap: Sowohl für die Installation...



Änderungen im System

Bootstrap für Install Tool(2)

- Das Install Tool basiert nun komplett auf Bootstrap: ...wie auch für die Konfiguration



Änderungen im System

CSRF Schutz für eigene Plugins

- Frontend Plugins müssen nun selbst für einen CSRF-Schutz sorgen:

```
$formToken = \TYPO3\CMS\Core\FormProtection\FormProtectionFactory::get()->getFormProtection()->
      generateToken('news', 'edit', $uid);
if (
  $dataHasBeenSubmitted
  && \TYPO3\CMS\Core\FormProtection\FormProtectionFactory::get()->validateToken(
    \TYPO3\CMS\Core\Utility\GeneralUtility::_POST('formToken'), 'User setup', 'edit')) {
  // alles in Ordnung
}
else {
  // ungueltiger Token!
}
```

Änderungen im System

Neue Tabs für LinkBrowser (1)

- Mit diesem Feature kann der LinkBrowser um neue Tabs erweitert werden
- Jeder Tab wird über einem sogenannten "LinkHandler" gesteuert, welcher das folgende Interface implementieren muss:
  `\TYPO3\CMS\Recordlist\LinkHandler\LinkHandlerInterface`
- Die LinkHandler werden über PageTSconfig registriert:

```
file {
  handler = TYPO3\\CMS\\Recordlist\\LinkHandler\\FileLinkHandler
  label = LLL:EXT:lang/locallang_browse_links.xlf:file
  displayAfter = page
  scanAfter = page
  configuration {
    customConfig = passed to the handler
  }
}
```

Änderungen im System

Neue Tabs für LinkBrowser (2)

- Die Optionen `displayBefore` und `displayAfter` geben die Anzeigeposition der Tabs an
- Die Optionen `scanBefore` und `scanAfter` regeln die Reihenfolge der Ausführung

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['LinkBrowser']['hooks'][1444048118] = [
  'handler' => \Vendor\Ext\MyClass::class,
  'before' => [], // optional
  'after' => [] // optional
];
```

Änderungen im System

Neue Module Template API (1)

- Es wurde eine Module Template API integriert, um die Erstellung der DocHeader zu vereinheitlichen
- Beispiel 1: Button hinzufügen

```
$openInNewWindowButton = $this->moduleTemplate->getDocHeaderComponent()->getButtonBar()
  ->makeLinkButton()
  ->setHref('#')
  ->setTitle($this->getLanguageService()->sL(
    'LLL:EXT:lang/locallang_core.xlf:labels.openInNewWindow', TRUE
    ))
  ->setIcon($this->iconFactory->getIcon('actions-window-open', Icon::SIZE_SMALL))
  ->setOnClick($aOnClick);

$this->moduleTemplate->getDocHeaderComponent()->getButtonBar()
  ->addButton($openInNewWindowButton, ButtonBar::BUTTON_POSITION_RIGHT);
```

Änderungen im System

Neue Module Template API (2)

- Beispiel 2: Menü hinzufügen

```
$languageMenu = $this->moduleTemplate->getDocHeaderComponent()
  ->getModuleMenuRegistry()->makeMenu()
  ->setIdentifier('_langSelector')
  ->setLabel($this->getLanguageService()->sL(
    'LLL:EXT:lang/locallang_general.xlf:LGL.language', TRUE
  ));

$menuItem = $languageMenu->makeMenuItem()
  ->setTitle($lang['title'] . $newTranslation)
  ->setHref($href);

if((int)$lang['uid'] === $currentLanguage) {
  $menuItem->setActive(TRUE);
}

$languageMenu->addMenuItem($menuItem);
$this->moduleTemplate->getDocHeaderComponent()->getModuleMenuRegistry()->addMenu($languageMenu);
```

Änderungen im System

PSR-7 Routing für Backend AJAX Requests

- Um eine Route für einen AJAX-Request zuzufügen, erstellt man eine Datei `Configuration/Backend/AjaxRoutes.php` mit folgendem Inhalt in der eigenen Extension:

```
return [
  // do something
  'unique_route_name' => [
    'path' => '/toolcollection/some-action',
    'target' => \Vendor\Controller\SomeController::class . '::myAction',
  ]
];
```

Änderungen im System

OpenID Hook `getUserRecord`

Es wurden zwei Hooks zur Verarbeitung von OpenID hinzugefügt (1/2)

- Hook 1:

  ```
  $GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['openid']['getUserRecord']
  ```

  - Modifiziert den Benutzer-Datensatz, nachdem dieser ermittelt wurde, oder:
  - Legt einen neuen Datensatz an, wenn keine gefunden wurde
  - Es werden die Parameter `record`, `response` und `authInfo` an den Hook übermittelt

Änderungen im System

OpenID Hook `authRequest`

Es wurden zwei Hooks zur Verarbeitung von OpenID hinzugefügt (2/2)

- Hook 2:

  ```
  $GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['openid']['authRequest']
  ```

  - Modifiziert den Authentifizierungs-Request bevor dieser abgesendet wird
  - Damit können z.B. zusätzliche Attribute, wie der Nickname vom OpenID-Server angefordert werden
  - Es werden die Parameter `authRequest` und `authInfo` an den Hook übermittelt

Änderungen im System

Hooks und Signals (1)

- Es ist nun möglich, das Verzeichnis, welches von `BackendUserAuthentication::getDefaultUploadFolder()` zurückgegeben wird, via Hook zu verändern
- Dazu muss folgende Konfiguration in die Datei `ext_localconf.php` eingetragen werden:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_userauthgroup.php']
  ['getDefaultUploadFolder'][] =
  \Vendor\MyExtension\Hooks\DefaultUploadFolder::class . '->getDefaultUploadFolder';
```

Änderungen im System

Hooks und Signals (2)

Beispiel:

```
<?php
namespace Vendor\MyExtension\Hooks;
use TYPO3\CMS\Core\Authentication\BackendUserAuthentication;
use TYPO3\CMS\Core\Resource\Folder;

/**
 * Class DefaultUploadFolder
 */
class DefaultUploadFolder {

  /**
   * Get default upload folder
   * If there is a folder present with the same name as the last part of the table name use that folder.
   * @param array $params
   * @param BackendUserAuthentication $backendUserAuthentication
   * @return Folder
   */
   public function getDefaultUploadFolder($params, BackendUserAuthentication $backendUserAuthentication)
          {
    [...]
```

Änderungen im System

Hooks und Signals (3)

Beispiel (Fortsetzung):

```
    [...]

    /** @var Folder $uploadFolder */
    $uploadFolder = $params['uploadFolder'];
    $pid = $params['pid'];
    $table = $params['table'];
    $field = $params['field'];

    $matches = [];
    if (!empty($uploadFolder) && preg_match('/_([a-z]+)$/', $table, $matches)) {
      $folderName = $matches[1];
      if ($uploadFolder->hasFolder($folderName)) {
        $uploadFolder = $uploadFolder->getSubfolder($folderName);
      }
    }
    return $uploadFolder;
  }
}
```

Änderungen im System

Diverse Änderungen

- Der TCA-Typ `select` muss nun mit einer Option `renderType` versehen werden
- Folgende Werte sind hierbei zulässig:

```
'renderType' => 'selectMultipleSideBySide',
'renderType' => 'selectCheckBox',
'renderType' => 'selectSingle',
'renderType' => 'selectSingleBox',
'renderType' => 'selectTree',
```

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