TYPO3 CMS 8.0 – What's New

Übersicht der neuen Funktionen, Änderungen und Verbesserungen

Patrick Lobacher und Michael Schams

30/March/2016

TYPO3

TYPO3 CMS 8.0 – What's New

Kapitelübersicht

Introduction

Die Fakten

Introduction

TYPO3 CMS 8.0 – The Facts

- Veröffentlichungsdatum: 22. März 2016
- Releasetyp: Sprint Release
- Vision: Start your engines

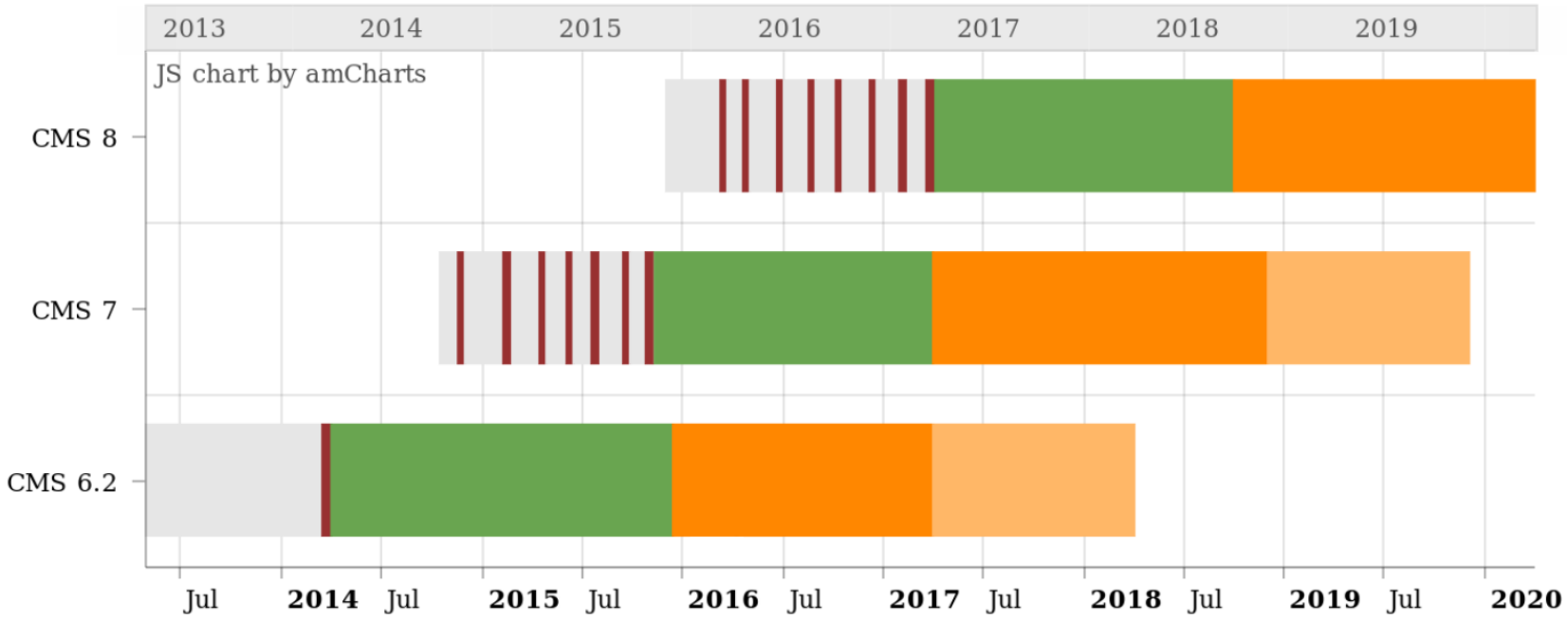
Introduction

Systemvoraussetzungen

- PHP: v7.0.0
- MySQL: v5.5.x - v5.7.x
- Festplattenplatz: mindestens 200 MB
- PHP settings:
  - `memory_limit` >= 128M
  - `max_execution_time` >= 240s
  - `max_input_vars` >= 1500
  - compilation option `--disable-ipv6` must not be used
- Das Backend benötigt einen Microsoft Internet Explorer 11 oder später, Microsoft Edge, Google Chrome, Firefox, Safari oder jeden anderen modernen Browser

Introduction

Release Zyklus



Introduction

TYPO3 CMS Roadmap

Voraussichtliche Veröffentlichungen und deren Hauptfokus:

- v8.0 22/Mar/2016 Adding last minute things
- v8.1 03/Mai/2016 Cloud Integration
- v8.2 05/Jul/2016 Rich Text Editor
- v8.3 30/Aug/2016 Frontend Editing on Steroids
- v8.4 18/Okt/2016 *to be determined*
- v8.5 20/Dez/2016 Integrator Support
- v8.6 14/Feb/2017 *to be determined*
- v8.7 04/Apr/2017 LTS Preparation

https://typo3.org/typo3-cms/roadmap/

https://typo3.org/news/article/kicking-off-typo3-v8-development/

Introduction

Installation

- Empfohlene Installationsschritte unter Linux/Mac OS X (DocumentRoot ist beispielsweise `/var/www/site/htdocs`):

```
$ cd /var/www/site
$ wget --content-disposition get.typo3.org/8.0
$ tar xzf typo3_src-8.0.0.tar.gz
$ cd htdocs
$ ln -s ../typo3_src-8.0.0 typo3_src
$ ln -s typo3_src/index.php
$ ln -s typo3_src/typo3
$ touch FIRST_INSTALL
```

- Symbolische Links unter Microsoft Windows:
  - unter Windows XP/2000 kann `junction` benutzt werden
  - unter Windows Vista und Windows 7 kann `mklink` benutzt werden

Introduction

Upgrade zu TYPO3 CMS 8.x

- Upgrades sind nur von TYPO3 CMS 7.6 LTS möglich
- TYPO3 CMS < 7.6 LTS sollte man zuerst auf TYPO3 CMS 7.6 LTS aktualisieren
- Upgrade-Anleitung:

  http://wiki.typo3.org/Upgrade#Upgrading_to_8.0
- Official TYPO3 guide "TYPO3 Installation and Upgrading":

  http://docs.typo3.org/typo3cms/InstallationGuide
- Generelles Vorgehen:
  - Prüfen, ob Mindestvoraussetzungen erfüllt sind (PHP, MySQL, etc.)
  - Das **deprecation_*.log** der TYPO3 Instanz durchsehen
  - Sämtliche Extensions auf den aktuellsten Stand bringen
  - Neuen TYPO3 Quellcode entpacken und im Install Tool den Upgrade Wizard ausführen

Introduction

PHP Version 7

- PHP 7.0 ist die minimal mögliche Version für TYPO3 CMS 8.x
- TYPO3 wird kontinuierlich weitere PHP 7 Releases unterstützen, sobald diese veröffentlicht werden
- Diese Version beschleunigt das gesamte System signifikant
- Nicht nur Backend-Redakteure werden das deutlich beschleunigte Interface bemerken, auch der Aufruf des Caches im Frontend ist nun unter 7ms möglich, was ein Geschwindigkeitswachstum von 40% gegenüber PHP 5.5 bedeutet
- Zeitgleicht wurden neue PHP 7 Features in den Core integriert, wie beispielsweise die Verwendung der kryptografischen Pseudo-Zufalls-Generatoren

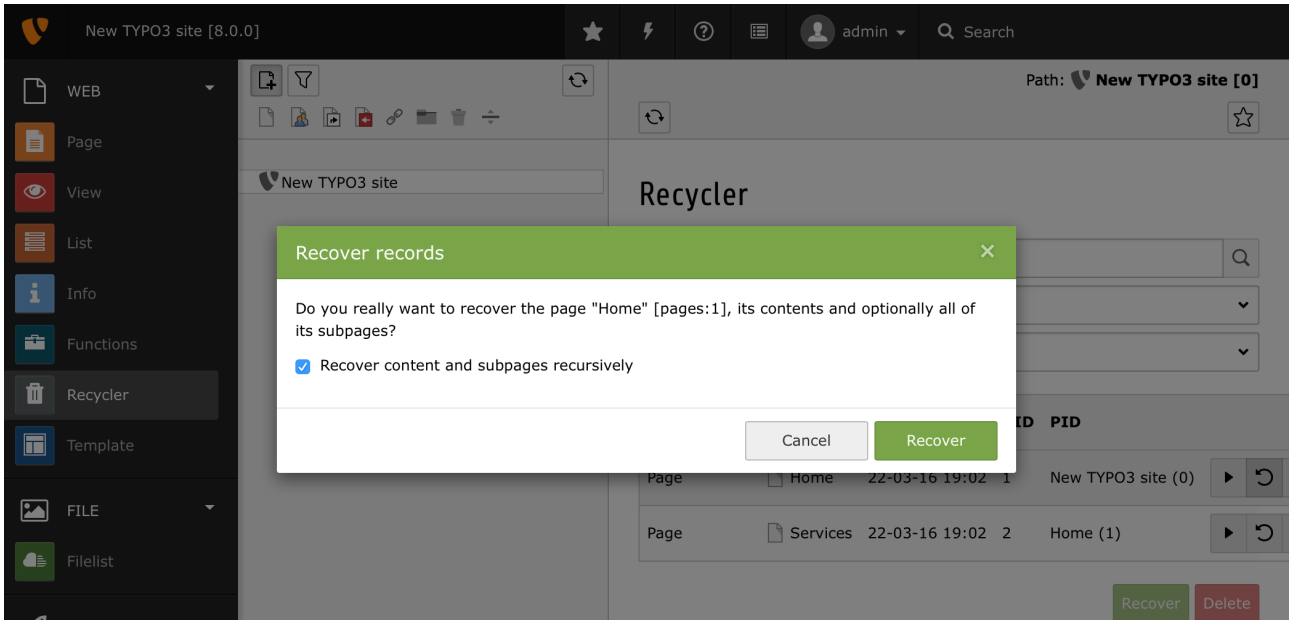
Kapitel 1:

Backend User Interface

Backend User Interface

Rekursive Wiederherstellung von Seiten

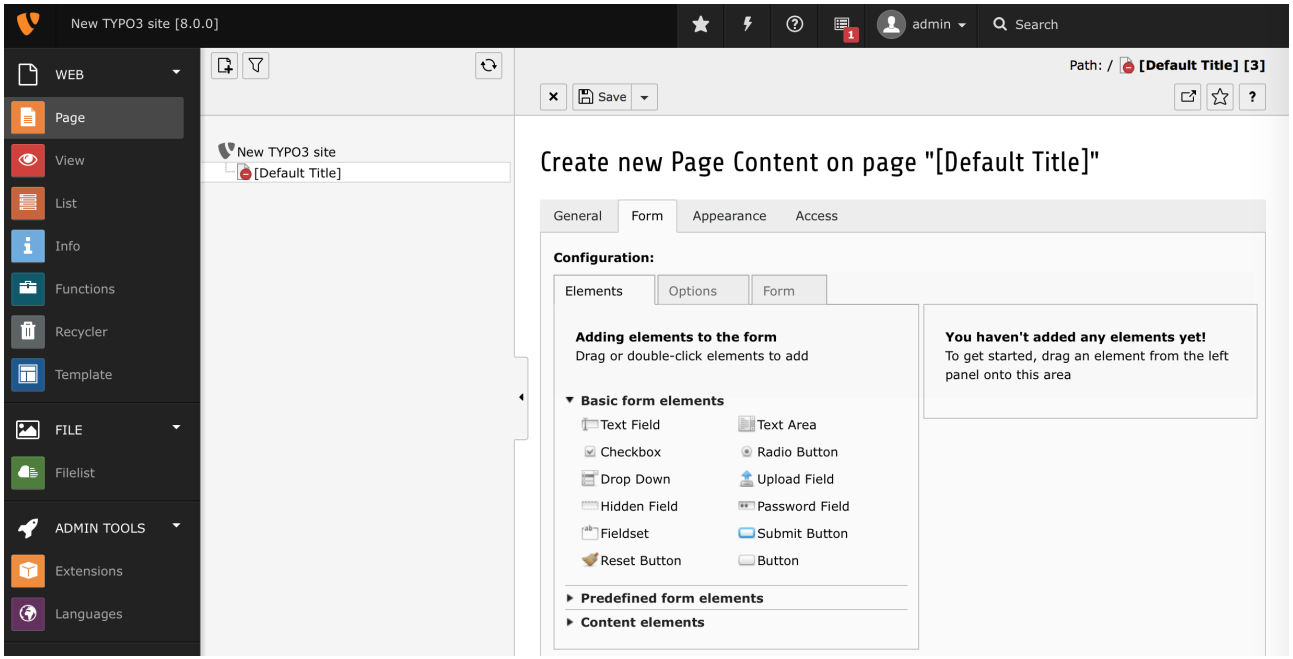
Das Modul "Recycler" unterstützt nun die rekursive Wiederherstellung von gelöschten Seiten zurück bis zum ersten Level der Rootline.
Dieses Feature ist auschließlich für Admin-User verfügbar, da hierfür spezielle Rechte vorhanden sein müssen.



Backend User Interface

Form-Assistent als Inline-Wizard

Der Assistent der Extension `EXT:form` wird nun direkt inline geladen. Früher musste man hierzu das Content-Element speichern und Neuladen, um den Assistenten zu verwenden.



Backend User Interface

Setzen eines alternativen Backend-Logos via Extension Manager

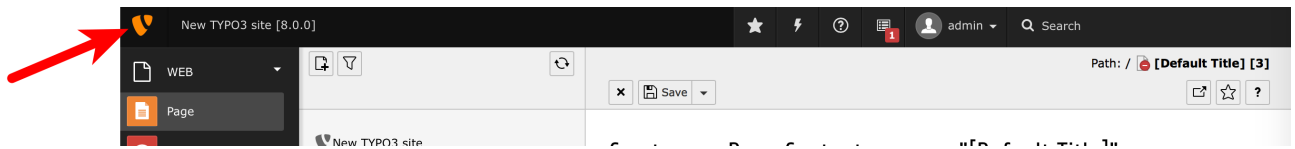
Das Backend-Logo in der oberen linken Ecke kann nun über die Extension-Konfiguration von `EXT:backend` im Extension Manager konfiguriert werden.
Die Konfigurations-Optionen sind:

- Angabe der Ressource als relativer Pfad

  z.B. "`fileadmin/images/my-background.jpg`"
- Angabe der Ressource als Extension-Pfad

  z.B. "`EXT:my_theme/Resources/Public/Images/my-background.jpg`"
- Angabe als externe Ressource

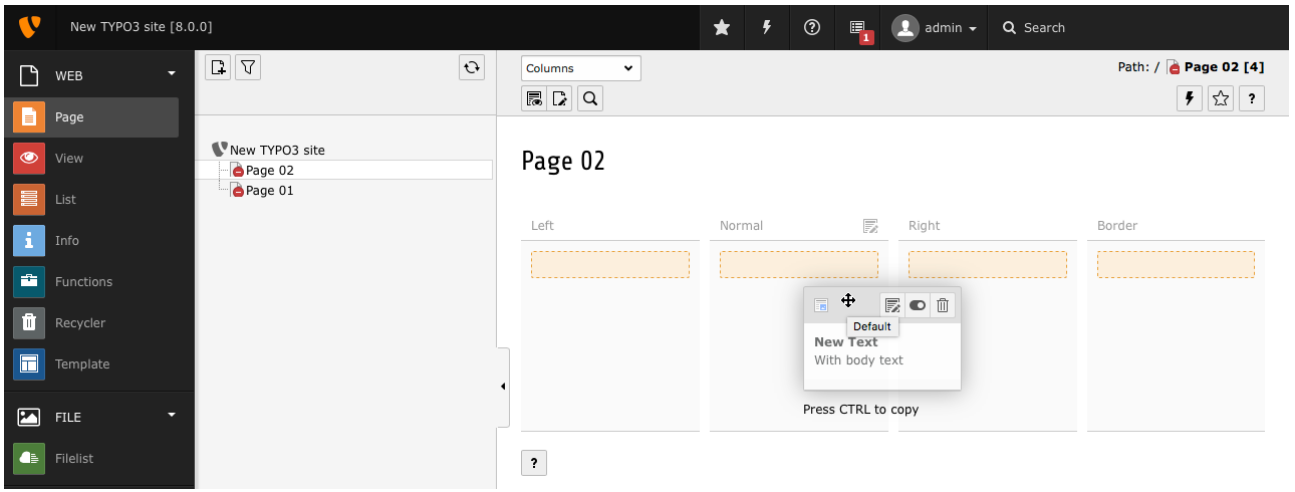
  e.g. "`//example.com/my-background.png`"



Backend User Interface

Kopieren von Seiten per Drag & Drop

Zusätzlich zum Drag & Drop Feature im Seitenmodul (womit man Inhaltselemente *verschieben* konnte), ist es nun möglich diese auch zu kopieren, indem man zusätzlich die CTRL-Taste drückt. Nach der Operation lädt das Seitenmodul neu, damit alle Informationen aktualisiert werden.



Kapitel 2:

TSconfig & TypoScript

TSconfig & TypoScript

Sortierung der Tabs im "New content element wizard"

- Es ist nun möglich die Sortierung der Tabs im "New content elementwizard" zu verändern, indem die Optionen `before` und `after` im Page TSconfig gesetzt werden:

```
mod.wizards.newContentElement.wizardItems.special.before = common
mod.wizards.newContentElement.wizardItems.forms.after = common,special
```

TSconfig & TypoScript

HTMLparser.stripEmptyTags.keepTags

- Es wurde eine neue Option für die HTMLparser.stripEmptyTags Konfiguration hinzugefügt, welche es ermöglicht, die Tags anzugeben, die behalten werden sollen
- Vorher war es nur möglich, anzugeben, welche Tags entfernt werden sollen
- Das folgende Beispiel entfernt alle leeren Tags, **außer** tr und td Tags:

```
HTMLparser.stripEmptyTags = 1
HTMLparser.stripEmptyTags.keepTags = tr,td
```

Wichtig: wenn diese Einstellung verwendet wird, hat die Konfiguration stripEmptyTags.tags keinen Effekt mehr. Man kann nur eine Option zur selben Zeit verwenden.

TSconfig & TypoScript

EXT:form - Integration von vordefinierten Formularen (1)

- Das Content-Element von EXT:form erlaubt nun die Integration von vordefinierten Formularen
- Der Integrator kann Formulare definieren (z.B. innerhalb eines Site Packages), indem der Schlüssel plugin.tx_form.predefinedForms verwendet wird
- Der Redakteur kann das neue Content-Element mailform auf einer Seite platzieren und dort aus einer Liste vordefinierter Formulare wählen
- Integratoren können eigene Formulare über TypoScript erstellen, wodurch mehr Optionen zur Verfügung stehen, als es im Form-Wizard möglich wäre (z.B. durch die Verwendung von stdWrap)

TSconfig & TypoScript

EXT:form - Integration von vordefinierten Formularen (2)

- Es gibt für Redakteure keine Notwendigkeit mehr, den Form-Wizard zu verwenden - diese können aus vordefinierten Formularen wählen, welche Layout-technisch optimiert sind
- Formulare können überall wiederverwendet werden
- Formulare können außerhalb der Datenbank gespeichert und somit versioniert werden
- Damit man die Formulare im Backend auswählen kann, müssen diese über PageTS registriert werden:

```
TCEFORM.tt_content.tx_form_predefinedform.addItems.contactForm =
  LLL:EXT:my_theme/Resources/Private/Language/locallang.xlf:contactForm
```

TSconfig & TypoScript

EXT:form: Integration von vordefinierten Formularen (3)

- Beispiel-Formular:

```
plugin.tx_form.predefinedForms.contactForm = FORM
plugin.tx_form.predefinedForms.contactForm {
  enctype = multipart/form-data
  method = post
  prefix = contact
  confirmation = 1
  postProcessor {
    1 = mail
    1 {
      recipientEmail = test@example.com
      senderEmail = test@example.com
      subject {
        value = Contact form
        lang.de = Kontakt Formular
      }
    }
  }
  10 = TEXTLINE
  10 {
    name = name
...
```

Kapitel 3:

Änderungen im System

Änderungen im System

Unterstützung von PECL-memcached im MemcachedBackend

- Es wurde eine Unterstützung für das PECL Modul "memcached" zum MemcachedBackend des Caching-Frameworks hinzugefügt
- Wenn beide ("memcache" und "memcached") installiert sind, wird "memcache" verwendet, um einen Breaking Change zu vermeiden
- Als Integrator kann man die Option `peclModule` setzen, um das bevorzugte PECL-Modul auszuwählen:

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['caching']['cacheConfigurations']['my_memcached'] = [
  'frontend' => \TYPO3\CMS\Core\Cache\Frontend\VariableFrontend::class
  'backend' => \TYPO3\CMS\Core\Cache\Backend\MemcachedBackend::class,
  'options' => [
    'peclModule' => 'memcached',
    'servers' => [
      'localhost',
      'server2:port'
    ]
  ]
];
```

Änderungen im System

Native Unterstützung der Symfony-Konsole (1)

- TYPO3 unterstützt nun out-of-the-box die Symfony-Konsole in dem es ein CLI-Skript in `typo3/sysext/core/bin/typo3` zur Verfügung stellt. Auf Instanzen, die mittels Composer installiert wurden, wird das Binary in das `bin`-Verzeichnis verlinkt, z.B. `bin/typo3`.
- Das neue Binary unterstützt die existierenden CLI-Kommandos als Fallback, wenn kein entsprechender Symfony Konsolen-Befehl gefunden wurde
- Um einen Befehl zu registrieren, der via `typo3` CLI aufgerufen werden soll, muss man eine entsprechende Datei `Configuration/Commands.php` innerhalb der eigenen Extension anlegen

Änderungen im System

Native Unterstützung der Symfony-Konsole (2)

- Sobald man ein Kommando regirstriert, muss man zwingend die Eigenschaft `class` angeben. Optional kann der Parameter `user` angegeben werden, mit dem angegeben wird, als welcher User das Kommando im Backend ausgeführt wird
- Eine beispielhafte Datei `Configuration/Commands.php` könnte so aussehen:

```
return [
  'backend:lock' => [
    'class' => \TYPO3\CMS\Backend\Command\LockBackendCommand::class
  ],
  'referenceindex:update' => [
    'class' => \TYPO3\CMS\Backend\Command\ReferenceIndexUpdateCommand::class,
    'user' => '_cli_lowlevel'
  ]
];
```

Änderungen im System

Native Unterstützung der Symfony-Konsole (3)

- Ein exemplarischer Aufruf könnte wie folgt lauten:

```
bin/typo3 backend:lock http://example.com/maintenance.html
```

- Bei Nicht-Composer Installationen:

```
typo3/sysext/core/bin/typo3 backend:lock http://example.com/maintenance.html
```

Änderungen im System

Kryptografisch sicherer Pseudo-Zufallszahlen-Generator

- Es wurde ein neuer kryptografisch sicherer Pseudo-Zufallszahlen-Generator (CSPRNG) in den Core integriert. Dieser verwendet die neue CSPRNG-Funktion in PHP 7.
- Die API hierzu befindet sich in der Klasse `\TYPO3\CMS\Core\Crypto\Random`
- Beispiel:

```
use \TYPO3\CMS\Core\Crypto\Random;
use \TYPO3\CMS\Core\Utility\GeneralUtility;

// Retrieving random bytes
$someRandomString = GeneralUtility::makeInstance(Random::class)->generateRandomBytes(64);

// Rolling the dice..
$tossedValue = GeneralUtility::makeInstance(Random::class)->generateRandomInteger(1, 6);
```

Änderungen im System

Wizard Komponente (1)

- Es wurde eine neue Wizard Komponente (Assistent) zugefügt. Dieser Assistent kann für User-unterstützte Interaktionen verwendet werden
- Das RequireJS-Module kann durch die Inklusion von `TYPO3/CMS/Backend/Wizard` verwendet werde
- Der Assistent unterstützt bislang nur direkte Actions (und keine Verbindungen untereinander)
- Die Wizard-Komponente besitzt die folgenden öffentlichen Methoden:

```
addSlide(identifier, title, content, severity, callback)
addFinalProcessingSlide(callback)
set(key, value)
show()
dismiss()
getComponent()
lockNextStep()
unlockNextStep()
```

Änderungen im System

Wizard Komponente (2)

- Der Event `wizard-visible` wird abgefeuert, wenn das Rendering des Wizards fertig gestellt wurde
- Wizards können über das Abfeuern des `wizard-dismiss` Events geschlossen werden
- Wizards feuern den `wizard-dismissed` Event, wenn diese geschlossen wurden
- Eigene Listener können über `Wizard.getComponent()` integriert werden

Änderungen im System

Generierte Asset-Dateien wurden verschoben

- Die Ordner-Struktur innerhalb von `typo3temp` wurde verändert, um Assets zu separieren, die von temporär erzeugten Dateien öffentlichen Zugriff benötigen (so benötigt z.B. zum Zwecke des Cachings oder Locking lediglich der Server Zugriff auf deren Dateien)
- Diese Assets werden nun nicht mehr in den folgenden Ordnern gespeichert:
  `_processed_`, `compressor`, `GB`, `temp`, `Language`, `pics`
  sondern in Zukunft in den nachfolgenden:
  - `typo3temp/assets/js/`
  - `typo3temp/assets/css/`,
  - `typo3temp/assets/compressed/`
  - `typo3temp/assets/images/`

Änderungen im System

ImageMagick/GraphicsMagick Änderungen (1)

- Die Settings für den Graphic-Prozessor (Image- oder GraphicsMagick) wurden umbenannt (Datei: `LocalConfiguration.php`).
  ALT: `im_`
  NEU: `processor_`
- Negative Benennung - wie `noScaleUp` - wurden in positive Entsprechungen geändert
- Zusätzlich wurden Referenzen zu spezifischen Versionen von ImageMagick/GraphicsMagick entfernt

Änderungen im System

ImageMagick/GraphicsMagick Änderungen (2)

- Die nicht benutzte Konfigurationsoption `image_processing` wurde ohne Ersatz entfernt
- Die Prozessor-spezifische Konfigurationsoption `colorspace` wurde mit einem *Namespace* unterhalb der `processor_` Hierarchie eingeordnet

Änderungen im System

Hooks und Signals (1)

- Ein zusätzlicher Hook wurde zur Methode `BackendUtility::viewOnClick()` zugefügt, um die Preview-URL zu verarbeiten
- Die Registrierung einer Hook-Klasse implementiert eine Methode `postProcess`:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_befunc.php']['viewOnClickClass'][] =
  \VENDOR\MyExt\Hooks\BackendUtilityHook::class;
```

Änderungen im System

Hooks und Signals (2)

- Vor TYPO3 CMS 7.6, war es möglich, ein Record-Overlay innerhalb von `Web -> List` zu überschreiben. Ein neuer Hook in TYPO3 CMS 8.0 stellt die alte Funktionalität wieder zur Verfügung
- Der Hook wird mit der folgenden Signatur aufgerufen:

```
/**
 * @param string $table
 * @param array $row
 * @param array $status
 * @param string $iconName
 * @return string the new (or given) $iconName
 */
function postOverlayPriorityLookup($table, array $row, array $status, $iconName) { ... }
```

- Die Registrierung einer Hook-Klasse implementiert eine Methode `postOverlayPriorityLookup`:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS'][IconFactory::class]['overrideIconOverlay'][] =
  \VENDOR\MyExt\Hooks\IconFactoryHook::class;
```

Änderungen im System

Hooks und Signals (3)

- Es wurde ein neues Signal implementiert, welches vor der Initialisierung eines Resource Storage ausgesendet wird
- Die Registrierung der Klasse erfolgt in der Datei `ext_localconf.php`:

```
$dispatcher = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(
  \TYPO3\CMS\Extbase\SignalSlot\Dispatcher::class);
$dispatcher->connect(
  \TYPO3\CMS\Core\Resource\ResourceFactory::class,
  ResourceFactoryInterface::SIGNAL_PreProcessStorage,
  \MY\ExtKey\Slots\ResourceFactorySlot::class,
  'preProcessStorage'
);
```

- Die Methode wird mit den folgenden Argumenten aufgerufen:
  - int `$uid` - UID des Datensatzes
  - array `$recordData` - alle Daten als Array
  - string `$fileIdentifier` - Datei-Identifier

Änderungen im System

Password Hashing Algorithmus: PBKDF2

- Es wurde ein neuer Password Hashing Algorithmus "PBKDF2" zur System-Extension "saltedpasswords" hinzugefügt
- PBKDF2 steht für: Password-Based Key Derivation Function 2
- Der Algorithmus wurde entworfen, um Brute Force Passwort-Cracken deutlich zu erschweren

Kapitel 4:

Extbase & Fluid

Extbase & Fluid

Standalone Fluid

- Die Fluid Rendering Funktionalität von TYPO3 CMS wurde durch eine Standalone Variante ersetzt, welche nun per Composer Dependecy inkludiert ist
- Die alte Fluid Extension wurde in einen sogenannten *Fluid Adapter* umgewandelt, welcher es erlaubt, dass TYPO3 CMS das Standalone Fluid verwenden kann
- Es wurden neue Features/Möglichkeiten in nahezu allen Bereichen von Fluid hinzugefügt
- Wichtig: Eine Vielzahl von Fluid Komponten, welche früher intern waren und nicht ersetzt werden konnten, sind nun so flexibel, dass diese sowohl ersetzt werden können, wie auch per öffentloicher API anprechbar sind

Extbase & Fluid

Standalone Fluid: RenderingContext (1)

- Der wichtigste neue Teil der öffentlichen API ist der "RenderingContext"
- Der bislang interne RenderingContext von Fluid wurde erweitert und ist verantwortlich für ein neues Fluid Feature: **Provisionierung der Implementierung**
- Dies befähigt Entwickler eine Vielzahl von Klasses zu ändern, die für Parsing, Auflösung, Caching und ähnliches zuständig sind
- Dies kann erreicht werden, indem entweder ein eigener RenderingContext hinzugefügt wird oder der standardmäßige RenderingContext über öffentliche Methoden manipuliert wird

Extbase & Fluid

Standalone Fluid: Rendering Context (2)

- Die folgenden (neuen) Features können durch Veränderung des RenderingContext aktiviert werden – hierzu muss lediglich eine einfache Methode aufgerufen werden:

```
$view->getRenderingContext()->setLegacyMode(false);
```

Extbase & Fluid

Standalone Fluid: ExpressionNodes (1)

- ExpressionNodes sind eine neue Fluid Syntax Struktur, welche ausschließlich innerhalb von geschweiften Klammern eingesetzt werden kann

```
$view->getRenderingContext()->setExpressionNodeTypes(array(
    'Class\Number\One',
    'Class\Number\Two'
));
```

- Entwickler können eigene, zusätzliche ExpressionNode Typen zufügen
- Jeder Typ besteht aus einem Matching-Pattern und Methoden um die Matches zu verarbeiten
- Jeder bestehender ExpressionNode Typ kann als Referenz verwendet werden

Standalone Fluid: ExpressionNodes (2)

ExpressionNode Typen stellen folgende neue Syntax zur Verfügung:

- **CastingExpressionNode**
  erlaubt das Casting von Variablen zu einem bestimmten Typ, um beispielsweise sicherzustellen, dass die Variable zu Integer oder Boolean umgewandelt wird. Die Verwendung erfolgt einfach mit dem `as` Keyword: `{myStringVariable as boolean}` oder `{myBooleanVariable as integer}` u.s.w. Versucht man ein Casting einer Variable zu einem inkompatiblen Typ, erhält man eine Fluid Fehlermeldung.
- **MathExpressionNode**
  erlaubt es, mathematische Operationen auf Variablen durchzuführen, z.B. `{myNumber + 1}`, `{myPercent / 100}` oder `{myNumber * 100}` u.s.w. Ein unmöglicher Ausdruck gibt eine leeren Ausgabe zurück

Standalone Fluid: ExpressionNodes (3)

ExpressionNode Typen stellen folgende neue Syntax zur Verfügung:

- **TernaryExpressionNode**
  erlaubt einen ternären Operator auf Variablen anzuwenden. Ein typischer Use-Case ist: "wenn dies, dann das, ansonsten jenes". Dies wird wie folgt verwendet:
  `{myToggleVariable ? myThenVariable : myElseVariable}`
  Achtung: hier kann kein verschachtelter Ausdruck oder ein InlineViewHelper verwendet werden, sondern nur standard Variablen

Extbase & Fluid

Standalone Fluid: Erweiterbare Namespaces (1)

- Fluid erlaubt nun die Erweiterung jedes Namespace Alias (wie z.B. `f:`) durch Hinzufügen von PHP-Namespaces
- PHP Namespaces werden hinsichtlich dem Vorhandensein von ViewHelper Klassen überprüft
- Das bedeutet insbesondere, dass Entwickler die bestehenden ViewHelper mit eigenen Klassen überschreiben können, während trotzdem der `f:` Namespace verwendet wird
- Weiterhin sind Namespaces nun nicht mehr monadisch.
  Sobald man beispielsweise `{namespace f=My\Extension\ViewHelpers\}`
  verwendet, bekommt man keinen "namespace already registered" Fehler mehr. Fluid addiert diesen PHP-Namespace stattdessen und schaut dort ebenfalls nach ViewHelpern

Extbase & Fluid

Standalone Fluid: Erweiterbare Namespaces (2)

- Zusätzliche Namespaces werden von unten nach oben überprüft – dies erlaubt es, eigene zusätzliche Namespaces einzubringen, welche die bestehenden überschreiben können
- Beispielsweise: `f:format.nl2br` kann durch `My\Extension\ViewHelpers\Format\Nl2brViewHelper`, überschrieben werden

Extbase & Fluid

Standalone Fluid: Rendering mittels `f:render` (1)

Zugefügt wurden optionale Default-Inhalte beim `f:render` ViewHelper:

- Wann immer `f:render` verwendet wird und das Flag `optional = TRUE` gesetzt ist, resultiert das Rendering einer nicht vorhandenen Sektion in einem leeren Output
- Anstelle des Rendern eines leeren Inhalts, kann das neue Attribut `default` (mixed) verwendet werden, welches zu einem Fallback mit Default-Content führt
- Alternativ wird der Tag-Content verwendet

Extbase & Fluid

Standalone Fluid: Rendering mittels `f:render` (2)

Weitergabe des Tag-Inhalts von `f:render` zum/zur Partial/Section:

- Dies erlaubt eine neue Strukturierung des Fluid Template Renderings
- Partials und Sections können als "Wrapper" für beliebigen Template-Code verwendet werden
- Beispiel:

```
<f:section name="MyWrap">
  <div>
    <!-- more HTML, using variables if desired -->
    <!-- tag content of f:render output: -->
    {contentVariable -> f:format.raw()}
  </div>
</f:section>

<f:render section="MyWrap" contentAs="contentVariable">
  This content will be wrapped. Any Fluid code can go here.
</f:render>
```

Extbase & Fluid

Standalone Fluid: Komplexe Bedingungen

- Fluid unterstützt nun auch komplexe Bedingungen mit Gruppierungen und Verschachtelungen:

```
<f:if condition="({variableOne} && {variableTwo}) || {variableThree} || {variableFour}">
  // Done if both variable one and two evaluate to true,
  // or if either variable three or four do.
</f:if>
```

- Zusätzlich wurde `f:else` um ein "elseif"-ähnliches Verhalten erweitert:

```
<f:if condition="{variableOne}">
  <f:then>Do this</f:then>
  <f:else if="{variableTwo}">
    Do this instead if variable two evals true
  </f:else>
  <f:else if="{variableThree}">
    Or do this if variable three evals true
  </f:else>
  <f:else>
    Or do this if nothing above is true
  </f:else>
</f:if>
```

Extbase & Fluid

Standalone Fluid: Dynamische Variablennamen (1)

- Es ist nun möglich, dynamische Variablen-Referenzen zu verwenden, wenn man auf Variablen zugreift. Wenn man beispielsweise folgende Fluid Template Variablen als Array hat:

```
$mykey = 'foo'; // or 'bar', set by any source
$view->assign('data', ['foo' => 1, 'bar' => 2]);
$view->assign('key', $mykey);
```

- Kann man nun wie folgt darauf zugreifen:

```
You chose: {data.{key}}.
(output: "1" if key is "foo" or "2" if key is "bar")
```

Extbase & Fluid

Standalone Fluid: Dynamische Variablennamen (2)

- Der selbe Ansatz kann verwendet werden, um dynamisch Teile des Variablennamens zu generieren:

```
$mydynamicpart = 'First'; // or 'Second', set by any source
$view->assign('myFirstVariable', 1);
$view->assign('mySecondVariable', 2);
$view->assign('which', $mydynamicpart);
```

- Der Zugriff erfolgt gleichermaßen:

```
You chose: {my{which}Variable}.
(output: "1" if which is "First" or "2" if which is "Second")
```

Extbase & Fluid

Standalone Fluid: Neue ViewHelper

- Standalone Fluid wurde mit einigen neuen ViewHelpern ausgestattet:
  - `f:or`
    Damit kann man verkettete Bedingungen realisieren. Der ViewHelper unterstützt die folgende Syntax. Hier wird jede Variable geprüft und ausgegeben, sobald diese nicht leer ist:

    ```
    {variableOne -> f:or(alternative: variableTwo) -> f:or(alternative: variableThree)}
    ```

  - `f:spaceless`
    Mit diesem ViewHelper wird redundater Whitespace und Leerzeilen eliminiert

Extbase & Fluid

Standalone Fluid: ViewHelper Namespace Erweiterung durch PHP

- Durch Zugriff auf den ViewHelperResolver des RenderingContext, können Entwickler auf die Inklusion des ViewHelper Namespace (auf per View Basis) zugreifen:

```
$resolver = $view->getRenderingContext()->getViewHelperResolver();
// equivalent of registering namespace in template(s):
$resolver->registerNamespace('news', 'GeorgRinger\News\ViewHelpers');
// adding additional PHP namespaces to check when resolving ViewHelpers:
$resolver->extendNamespace('f', 'My\Extension\ViewHelpers');
// setting all namespaces in advance, globally, before template parsing:
$resolver->setNamespaces(array(
  'f' => array(
    'TYPO3Fluid\\Fluid\\ViewHelpers', 'TYPO3\\CMS\\Fluid\\ViewHelpers',
    'My\\Extension\\ViewHelpers'
  ),
  'vhs' => array(
    'FluidTYPO3\\Vhs\\ViewHelpers', 'My\\Extension\\ViewHelpers'
  ),
  'news' => array(
    'GeorgRinger\\News\\ViewHelpers',
  );
));
```

Extbase & Fluid

Standalone Fluid: ViewHelper mit beliebigen Argumenten (1)

- Mit diesem Feature können ViewHelper mit beliebigen zusätzlichen Argumenten ausgestattet werden
- Dies funktioniert so, dass die Argumente in zwei Gruppen aufgeteilt werden - die, die per `registerArgument` deklariert werden und die restlichen
- Die, die nicht deklariert wurden, werden zu der speziellen Methode `handleAdditionalArguments` der ViewHelper-Klasse weitergeleitet, welche diese verarbeiten kann. Per Default wird ein Fehler ausgegeben.

Extbase & Fluid

Standalone Fluid: ViewHelper mit beliebigen Argumenten (2)

- Durch Überschreiben der Methode im eigenen ViewHelper kann dieses Verhalten angepasst werden
- Dieses Feature ist auch dafür zuständig, dass TagBasedViewHelper beliebige `data-` Argumente akzeptiert
- Bei TagBasedViewHelper sorgt die `handleAdditionalArguments` Methode per Default dafür, dass neue Attribute zum Tag hinzugefügt werden und wirft einen Fehler, wenn zusätzliche Argumente weder registriert, noch mit dem Prefix `data-` versehen wurden

Extbase & Fluid

Argument "allowedTags" für f:format.stripTags

- Das Argument `allowedTags` enthält eine Liste von HTML-Tags welche beim `f:format.stripTags` ViewHelper nicht entfernt werden
- Die Syntax der Tag-Liste ist identisch zum zweiten Paramater der PHP-Funktion `strip_tags` (siehe: http://php.net/strip_tags)

Extbase & Fluid

Zugriff auf ObjectStorage als Array

- Dieses Feature erstellt einen Alias von `toArray()`, welches es erlaubt, dass die Methode als `getArray()` über `ObjectAccess::getPropertyPath` aufgerufen werden kann
- Beispiel: Ermittlung des 4. Elements

```
// in PHP:
ObjectAccess::getPropertyPath($subject, 'objectstorageproperty.array.4')

// in Fluid:
{myObject.objectstorageproperty.array.4}
{myObject.objectstorageproperty.array.{dynamicIndex}}
```

Chapter 5:

Veraltete und entfernte Funktionen

Veraltete/Entfernte Funktionen

Verschiedenes

- Die folgenden Konfigurations-Optionen wurden entfernt (hier findet eine Autodection statt und es wird per Default `mbstring` gewählt):
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_utils']`
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_convMethod']`
- Die veraltete TypoScript-Eigenschaft `page.includeJSlibs` wurde entfernt. Stattdessen muss man nun die TypoScript-Eigenschaft `page.includeJSLibs` (mit großem "L") verwenden
- Die TypoScript-Option `config.renderCharset` wurde entfernt

Chapter 6:

Quellen und Autoren

Quellen und Autoren

Quellennachweis

TYPO3 News:

- http://typo3.org/news

Release Infos:

- http://wiki.typo3.org/TYPO3_CMS_8.0.0
- INSTALL.md and ChangeLog
- typo3/sysext/core/Documentation/Changelog/8.0/*

TYPO3 Bug-/Issuetracker:

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

TYPO3 und Fluid Git Repositories:

- https://git.typo3.org/Packages/TYPO3.CMS.git
- https://github.com/TYPO3Fluid/Fluid

Quellen und Autoren

TYPO3 CMS What's New Team:

Patrick Lobacher
(Recherche, Informationsdokumentation, englische und deutsche Version)

Michael Schams
(Project Leader und englische Version)

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