TYPO3 CMS 7 LTS – What's New

Extbase & Fluid

Created by:
Patrick Lobacher and Michael Schams

10/November/2015

TYPO3

Introduction

TYPO3 CMS 7 LTS – What's New

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

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

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

Extbase & Fluid

Extbase and Fluid are the de facto standards when it comes to modern technologies in the TYPO3 universe today. Although Extbase has received a few new features, most of the changes in TYPO3 CMS 7 LTS are Fluid-related. A number of ViewHelpers have been updated, improved or newly introduced. For example the **MediaViewHelper**, which can be used to render video, audio and other file types in the frontend.
Additionally the system extension `form` has been adapted to support the Extbase/Fluid MVC stack.

Extbase & Fluid

Template Path Fallback

- `Fluid Standalone View` as well as the TypoScript object `FLUIDTEMPLATE` support template fallback paths now

```
page.10 = FLUIDTEMPLATE
page.10.file = EXT:myextension/Resources/Private/Templates/Main.html
page.10.partialRootPaths {
  10 = EXT:myextension/Resources/Private/Partials
  20 = EXT:fallback/Resources/Private/Partials
}
```

- If new and old option is used (e.g. `partialRootPaths` and `partialRootPath`), the path stated by the option is at the first position (index = 0)

Extbase & Fluid

Typolink ViewHelper

- A new Typolink ViewHelper can parse and analyse the `typolink` string, created by the Link-Wizard and RTE

```
<f:link.typolink parameter="{link}" target="_blank" class="ico-class" title="some title"
      additionalAttributes="{type:'button'}">
```

`link` could contain:

```
42 _blank - "This is the link title" &foo=bar
```

Output:

```
<a href="index.php?id=42&foo=bar" title="This is the title" target="_blank" class="ico-class"
      type="button">
```

Note: only `parameter` is required, rest is optional

Extbase & Fluid

Generic data-* Attribute

- All ViewHelpers, which output HTML tags, support the HTML5 `data-*` attribute now
- An array passed as `data` is transformed and the key/value pair builds the attribute: `data-key="value"`

Example:

```
<f:form.textfield data="{foo: 'bar', baz: 'foos'}" />
```

Output:

```
<input data-foo="bar" data-baz="foos" ... />
```

Extbase & Fluid

Class Tag Values Via Reflection

- Extbase Reflection Service can return tags and annotations which have been added to a class

Example:

```
/**
 * @SomeClassAnnotation A value
 */
class MyClass {
}
```

Annotation can be accessed by:

```
$service = new \TYPO3\CMS\Extbase\Reflection\ReflectionService();

// Returns all tags and their values the specified class is tagged with
$classValues = $service->getClassTagsValues('MyClass');

// Returns the values of the specified class tag
$classValue = $service->getClassTagValue('MyClass', 'SomeClassAnnotation');
```

Extbase & Fluid

PaginateViewHelper

- Since TYPO3 CMS 7.1, PaginateViewHelper accepts input collections of the following types:
  - `QueryResultInterface`
  - `ObjectStorage`
  - `ArrayAccess`
  - `array`
- Example:

```
<f:widget.paginate objects="{blogs}" as="paginatedBlogs">
  <f:for each="{paginatedBlogs}" as="blog">
    <h4>{blog.title}</h4>
  </f:for>
</f:widget.paginate>
```

Extbase & Fluid

ContainerViewHelper loads RequireJS modules

- ContainerViewHelper can load RequireJS modules via the `includeRequireJsModules` attribute
- Example:

```
<f:be.container pageTitle="Extension Module" loadJQuery="true"
  includeRequireJsModules="{
    0:'TYPO3/CMS/Extension/Module1',
    1:'TYPO3/CMS/Extension/Module2',
    2:'TYPO3/CMS/Extension/Module3',
    3:'TYPO3/CMS/Extension/Module4'
  }" >
```

Extbase & Fluid

Method `has()` in ObjectAccess

- For the usage in Fluid, `object.property` and `object.isProperty` already support the following methods:
  - `isProperty()`
  - `getProperty()`
- New since TYPO3 CMS 7.1: `hasProperty()`
- This calls method `$object->hasProperty()` if `object.hasProperty` is used in Fluid

Extbase & Fluid

Upload multiple files with FormUpload-ViewHelper

- FormUpload-Viewhelper supports new attribute `multiple`, that provides the option to upload multiple files at a time

```
<f:form.upload property="files" multiple="multiple" />
```

Note: developers need to prepare the incoming value for the property mapping by writing their own TypeConverter!

Extbase & Fluid

Callouts vs. FlashMessages

- Callouts (content info boxes) replace FlashMessages in several places in the backend to display context information
- This required a new Fluid ViewHelper `be.infobox`:

```
<f:be.infobox title="Message title">
   your box content
</f:be.infobox>

<f:be.infobox
   title="Message title"
   message="your box content"
   state="-2"
   iconName="check"
   disableIcon="TRUE" />
```

Extbase & Fluid

format.case ViewHelper

- New ViewHelper `format.case` changes casing of strings:
  - `upper`: transforms a string to "UPPERCASE"
  - `lower`: transforms a string to "lowercase"
  - `capital`: transforms a string to its first letter upper-cased
  - `uncapital`: transforms a string to its first letter lower-cased
- For example:

```
// transforms to "SOME TEXT WITH MIXED CASE"
<f:format.case>Some TeXt WiTh miXed cAse</f:format.case>

// transforms to "SomeString"
<f:format.case mode="capital">someString</f:format.case>
```

Extbase & Fluid

Miscellaneous

- Parameter `cHash` is not added to action URIs if the current request is not cached and the target action is not cacheable

Extbase & Fluid

ActionMenuItemGroupViewHelper (1)

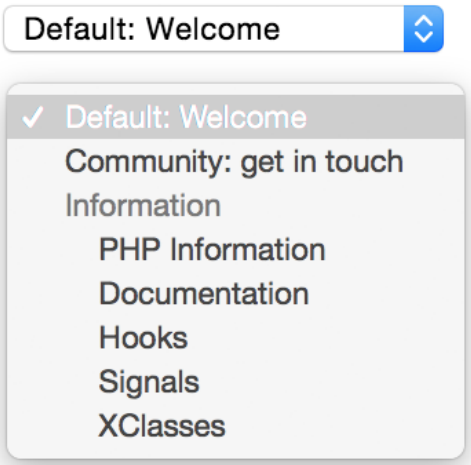
- Using this ViewHelper, option groups can be used in the backend select field, which controls which action is selected
- Example:

```
<f:be.menus.actionMenu>
  <f:be.menus.actionMenuItem label="Default: Welcome" controller="Default" action="index" />
  <f:be.menus.actionMenuItem label="Community: get in touch" controller="Community"
    action="index" />
  <f:be.menus.actionMenuItemGroup label="Information">
    <f:be.menus.actionMenuItem label="PHP Information" controller="Information"
      action="listPhpInfo" />
    <f:be.menus.actionMenuItem label="Documentation" controller="Information"
      action="documentation" />
    <f:be.menus.actionMenuItem label="Hooks" controller="Information" action="hooks" />
    <f:be.menus.actionMenuItem label="Signals" controller="Information" action="signals" />
    <f:be.menus.actionMenuItem label="XClasses" controller="Information" action="xclass" />
  </f:be.menus.actionMenuItemGroup>
</f:be.menus.actionMenu>
```

Extbase & Fluid

ActionMenuItemGroupViewHelper (2)

- Example on previous slide results in the following output:



Extbase & Fluid

Template Support for FlashMessagesViewHelper

- The `FlashMessagesViewHelper` supports templates now
- New attribute `as` allows to specify a variable name, which can be used within the ViewHelper's child elements to access the flash messages
- Example:

```
<f:flashMessages as="flashMessages">
  <ul class="myFlashMessages">
    <f:for each="{flashMessages}" as="flashMessage">
      <li class="alert {flashMessage.class}">
        <h4>{flashMessage.title}</h4>
        <span class="fancy-icon">{flashMessage.message}</span>
      </li>
    </f:for>
  </ul>
</f:flashMessages>
```

- Note: option `renderMode` is deprecated now

Extbase & Fluid

New Properties of cObject FLUIDTEMPLATE (1)

- cObject FLUIDTEMPLATE has been extended with templateRootPaths and templateName
- It is possible to set a template name and when rendering the template this name is used together with the set format to find the template in the given templateRootPaths
- templateRootPaths features the same fallback logic as layoutRootPath and partialRootPath
  - templateName: string/stdWrap
  - templateRootPaths: array of file paths with "EXT:" prefix support

Extbase & Fluid

New Properties of cObject FLUIDTEMPLATE (2)

- TypoScript Example:

```
lib.stdContent = FLUIDTEMPLATE
lib.stdContent {
  templateName = TEXT
  templateName.stdWrap {
    cObject = TEXT
    cObject {
      data = levelfield:-2,backend_layout_next_level,slide
      override.field = backend_layout
      split {
        token = frontend__
        1.current = 1
        1.wrap = |
      }
    }
    ifEmpty = Default
  }
  templateRootPaths {
    10 = EXT:frontend/Resources/Private/Templates
    20 = EXT:sitemodification/Resources/Private/Templates
  }
}
```

Extbase & Fluid

Removal of `xmlns`-Attributes and HTML Tags (1)

- With the introduction of using `xmlns:*` attributes to include ViewHelpers, it is possible to have IDE support for Fluid templates. The Problem is that the `xmlns:*` attributes and the corresponding tag will also be rendered, which is usually not desired.
- The workaround is to use sections, but this solution is counter-intuitive and not available in layouts. It also causes extra processing overhead.
- `xmlns:*` attributes for valid ViewHelper namespaces will now be removed before rendering, if they show the following syntax:
  `http://typo3.org/ns/<phpNamespace>`
  (`xmlns` attributes for non-ViewHelper namespaces are preserved)

Extbase & Fluid

Removal of `xmlns`-Attributes and HTML Tags (2)

- Include ViewHelper namespaces within the HTML tag and the `data-namespace-typo3-fluid="true"` attribute to prevent the rendering of the entire HTML tag

```
<html data-namespace-typo3-fluid="true"
  xmlns:f="http://typo3.org/ns/TYPO3/CMS/Fluid/ViewHelpers"
  xmlns:n="http://typo3.org/ns/GeorgRinger/News/ViewHelpers">

  <f:if condition="{newsItem.title}">
    <f:then>
      <n:titleTag>{newsItem.title}</n:titleTag>
    </f:then>
    <f:else>
      <n:titleTag>News-Detail</n:titleTag>
    </f:else>
  </f:if>

</html>
```

Extbase & Fluid

New Methods in Fluid-StandaloneView

- StandaloneView is extended with `setTemplateRootPaths($templatePaths)` and `setTemplate($templateName, $throwException = TRUE)`
- Same functionality as cObject `FLUIDTEMPLATE`
- Example (render of an email template):

```
$view = GeneralUtility::makeInstance(StandaloneView::class);
$view->setLayoutRootPaths(array(GeneralUtility::getFileAbsFileName(
  'EXT:my_extension/Resources/Private/Layouts')));
$view->setPartialRootPaths(array(GeneralUtility::getFileAbsFileName(
  'EXT:my_extension/Resources/Private/Partials')));
$view->setTemplateRootPaths(array(GeneralUtility::getFileAbsFileName(
  'EXT:my_extension/Resources/Private/Templates')));
$view->setTemplate('Email/Notification');
$emailBody = $view->render();
```

Extbase & Fluid

Data Processing for FLUIDTEMPLATE cObject (1)

- cObject FLUIDTEMPLATE has been extended with dataProcessing
- This setting can be used to add one or multiple processors to manipulate the $data variable of the currently rendered cObject (e.g. tt_content or page)
- Processor must implement the interface FluidTemplateDataProcessorInterface and contain the following method:

```
function process(array &$data, array $processorConfiguration,
  array $configuration, StandaloneView $view) {
    [...]
}
```

Extbase & Fluid

Data Processing for FLUIDTEMPLATE cObject (2)

- Example:

```
my_custom_ctype = FLUIDTEMPLATE
my_custom_ctype {
  templateRootPaths {
    10 = EXT:your_extension_key/Resources/Private/Templates
  }
  templateName = CustomName
  settings {
    extraParam = 1
  }
  dataProcessing {
    1 = Vendor\YourExtensionKey\DataProcessing\MyFirstCustomProcessor
    2 = AnotherVendor\AnotherExtensionKey\DataProcessing\MySecondCustomProcessor
    2 {
      options {
        myOption = SomeValue
      }
    }
  }
}
```

Extbase & Fluid

Anchor for Pagination Widget

- This new feature allows to add a key `section` to the configuration of a Fluid pagination widget
- The anchor gets appended to every link of the pagination widget
- The following code adds an anchor `#archive`:

```
<f:widget.paginate objects="{plantpestWarnings}" as="paginatedWarnings"
  configuration="{section: 'archive', itemsPerPage: 10, insertAbove: 0, insertBelow: 1,
  maximumNumberOfLinks: 10}">

   [...]

</f:widget.paginate>
```

Extbase & Fluid

Attribute `base` for DateViewHelper

- DateViewHelper has been extended by an optional attribute named `base`
- The attribute can be used to calculate relative time specification for dates
- If the date is a DateTime object, `base` is ignored
- The following example returns "2016", if `dateObject` is a date in 2017:

```
<f:format.date format="Y" base="{dateObject}">-1 year</f:format.date>
```

(see PHP documentation for a list of valid values)

Extbase & Fluid

Option `dataProcessing` for FLUIDTEMPLATE

- In TYPO3 CMS 7.3, option `dataProcessing` for cObject `FLUIDTEMPLATE` has been introduced
- Its `FluidTemplateDataProcessorInterface` has been refactored to `DataProcessorInterface`, which also affects method `process()`

```
public function process(
  ContentObjectRenderer $cObj,
  array $contentObjectConfiguration,
  array $processorConfiguration,
  array $processedData
);
```

BREAKING CHANGE!

TYPO3

Extbase & Fluid

Severity Filtering for FlashMessageQueue

- In TYPO3 CMS < 7.5, all messages of the FlashMessageQueue can be fetched and/or removed only
- In TYPO3 CMS >= 7.5, this can be done for a specific severity:

```
FlashMessageQueue::getAllMessages($severity);
FlashMessageQueue::getAllMessagesAndFlush($severity);
FlashMessageQueue::removeAllFlashMessagesFromSession($severity);
FlashMessageQueue::clear($severity);
```

Extbase & Fluid

Query Support for "`between`" added

- Support for `between` has been added to the Extbase Query object
- There is no performance benefit due to the fact that the DBMS converts "`between`" internally anyway: `min <= expr AND expr <= max`
- The new Extbase feature replicates the DBMS' behavior by building a logical `AND` condition, so this works across all DBMS

```
$query->matching(
  $query->between('uid', 3, 5)
);
```

Extbase & Fluid

Multiple FlashMessage Queues

- Is it now possible to implement multiple FlashMessageQueues:

```
$queueIdentifier = 'myQueue';
$this->controllerContext->getFlashMessageQueue($queueIdentifier);
```

- Access using Fluid works as follows:

```
<f:flashMessages queueIdentifier="myQueue" ></f:flashMessages>
```

Extbase & Fluid

Media ViewHelper (1)

- In order to easily render video, audio and all other file types with a registered Renderer class in the frontend, the `MediaViewHelper` has been implemented
- `MediaViewHelper` first checks if there is a Renderer present for the given file - if not, it falls back and renders an image tag
- Examples:

```
<code title="Image Object">
  <f:media file="{file}" width="400" height="375" ></f:media>
</code>

<output>
  <img alt="alt set in image record" src="fileadmin/_processed_/323223424.png"
    width="396" height="375" />
</output>
```

Extbase & Fluid

Media ViewHelper (2)

- Examples (continued):

```
<code title="MP4 Video Object">
  <f:media file="{file}" width="400" height="375" ></f:media>
</code>

<output>
  <video width="400" height="375" controls>
    <source src="fileadmin/user_upload/my-video.mp4" type="video/mp4">
  </video>
</output>

<code title="MP4 Video Object with loop and autoplay option set">
  <f:media file="{file}" width="400" height="375"
    additionalConfig="{loop: '1', autoplay: '1'}" ></f:media>
</code>

<output>
  <video width="400" height="375" controls loop>
    <source src="fileadmin/user_upload/my-video.mp4" type="video/mp4">
  </video>
</output>
```

Extbase & Fluid

System Extension `form` (1)

- System extension `form` (including the custom data model, controller logic, property validation, views and templating) has been adopted to support the Extbase/Fluid MVC stack
- This allows better customization and control of the generated behavior and markup by simply modifying Fluid templates or utilizing own custom view helper logic
- Each form element uses its own Partial, which can also be configured by the TypoScript option `partialPath = ...`

Extbase & Fluid

System Extension `form` (2)

- The following three new ViewHelpers exist:
  - `AggregateSelectOptionsViewHelper` (for optgroup tags)
  - `SelectViewHelper` (for optgroup tags)
  - `PlainMailViewHelper` (to render plain text mails)
- In addition, there are three Views:
  - `show` (the form itself)
  - `confirmation` (the confirmation page)
  - `postProcessor/mail` (the email)
- Template paths and visibility of fields can be customized for each View individually

Extbase & Fluid

Annotation `@cli`

- By using the new annotation `@cli`, commands in an Extbase CommandController can be marked as CLI-commands only
- These commands are excluded from the scheduler command selection
- Typical use cases are commands such as `extbase:help:help` for example

Extbase & Fluid

Same Table Relations

- It is now possible to use a domain model where an object is connected to another object of the same class directly

```
namespace \Vendor\Extension\Domain\Model;
class A {
  /**
   * @var \Vendor\Extension\Domain\Model\A
   */
  protected $parent;
}

namespace \Vendor\Extension\Domain\Model;
class A {
  /**
   * @var \Vendor\Extension\Domain\Model\B
   */
  protected $x;

  /**
   * @var \Vendor\Extension\Domain\Model\B
   */
  protected $y;
}
```

Extbase & Fluid

Option `absolute` for Image-ViewHelpers

- New option `absolute` forces `ImageViewhelper` and `Uri/ImageViewHelper` to output an **absolute** URL
- Example 1 (`ImageViewhelper`):

```
<f:image image="{file}" width="400" height="375" absolute="1" ></f:image>

// Output
<img alt="alt set in image record"
  src="http://example.com/fileadmin/_processed_/323223424.png"
  width="400" height="375" />
```

- Example 2 (`Uri/ImageViewHelper`):

```
<f:uri.image image="{file}" width="400" height="375" absolute="1" ></f:uri>

// Output
http://example.com/fileadmin/_processed_/323223424.png
```

Extbase & Fluid

Strip Whitespace between HTML Tags

- New ViewHelper `spaceless` removes redundant spaces between HTML tags while preserving the whitespace that may be inside HTML tags:

```
<f:spaceless>
<div>
    <div>
        <div>text

text</div>
</div>
</div>
```

- Output:

```
<div><div><div>text

text</div></div></div>
```


TYPO3

Extbase & Fluid

RootLevel Configuration

- The RootLevel of a table can be configured in TCA now
  (this define where records of a table can be found in the system)
  - `0`: page tree only
  - `1`: on the root page only (PID 0)
  - `-1`: both, root page and page tree
- TCA configuration:

```
$GLOBALS['TCA']['tx_myext_domain_model_record']['ctrl']['rootLevel'] = -1;
```

Sources and Authors

Sources and Authors

Sources

TYPO3 News:

- http://typo3.org/news

Release Infos:

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

TYPO3 Bug-/Issuetracker:

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

TYPO3 Git Repositories:

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

TYPO3

Sources and Authors

TYPO3 CMS What's New Slides:

Patrick Lobacher
(Research, Information Gathering and German Version)

Michael Schams
(Project Leader and English Version)

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

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