TYPO3 CMS 7 LTS - What's New

In-Depth Changes

Created by:
Patrick Lobacher and Michael Schams

10/November/2015

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

In-Depth Changes

By streamlining all components and removing outdated technologies, the backend of TYPO3 CMS 7 LTS is blazingly fast and superior to all prior TYPO3 versions.
But not only the backend received an incredible performance boost, many changes have been made "under the hood". PHP code has been cleaned up, some components even rewritten in Extbase and many new features implemented using the latest technologies and programming paradigms.

In-Depth Changes

Integration of jQuery UI version 1.11.2

- jQuery UI 1.11 supports AMD (Asynchronous Module Definition), which loads JavaScript files only, when they are needed (performance boost)
- jQuery UI 1.11 replaces jQuery UI 1.10 + Scriptaculous in TYPO3 CMS 7.0
- Only core and interaction components are included, which are required to replace ExtJS und Scriptaculous
- Widgets are not included (but those of Twitter Bootstrap are used, such as: DatePicker, Spinner, Dialog, Buttons, Tabs, Tooltip)

TYPO3

In-Depth Changes

Registry for File Rendering Classes

- In order to be able to render all kinds of media files, a file rendering registry has been implemented.
  This happens as follows (e.g. Video, MPEG, AVI, WAV, etc.):

```
<?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';
  }
}
```

In-Depth Changes

TCA: Validate Email Addresses

- New function "email" checks, if value entered is a valid email address
- If check fails, a Flash message appears
- 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'
  ),
)
```

In-Depth Changes

AbstractCondition For Custom TypoScript Conditions

- Custom TypoScript conditions can be derived from an AbstractCondition

```
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);
    }
  }
}
```

- The appropriate TypoScript code as follows:

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

- Operators, which should be available, are defined in the class

In-Depth Changes

Signal for IconUtility HTML Tag Manipulation

- New signal to manipulate the IconUtility HTML tag for sprite icons:

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

- Method call:

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

In-Depth Changes

Signal Slots to SoftReferenceIndex

- Two new signal slot dispatch calls in 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)
  );
}
```

- Called in:

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

In-Depth Changes

afterPersistObjetct Signal Slot

- New afterPersistObject signal slot emits for the aggregate root after persisting all other objects

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

- Called in:

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

- The same signal is emitted in the persistObject method in the AbstractBackend class in Flow

In-Depth Changes

Signal in loadBaseTca

- To improve performance in the backend context, the complete TCA can be cached now (not only parts of it)

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

- Called in:

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

In-Depth Changes

API to Add Cached TCA Changes

- PHP files in `extkey/Configuration/TCA/Overrides/` are executed directly after the TCA cache has been built
- These files may only include code, which manipulates the TCA, such as: `addTCAColumns` or `addToAllTCATypes`
- This feature gives backend requests a performance boost, once extensions start using these files

In-Depth Changes

Read-only File Mounts

- File mounts can be configured as "read only" (again)
- This was already possible in TYPO3 CMS 4.x, but silently dropped in 6.x
- Example: add folder "test" of storage UID 3 as a read-only mount in the File List and Element Browser.

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

If no storage is configured, it is assumed that the folder is in the default storage.

In-Depth Changes

Miscellaneous

- jQuery has been updated from version 1.11.0 to version 1.11.1
- Datatables has been updated from version 1.9.4 to version 1.10.2
- Some old, unused variables have been removed from `EM_CONF`
- Extension icons can be in SVG image format now (`ext_icon.svg`)
- Passing a wrong eID identifier results in an exception now

In-Depth Changes

TCA: Maximum chars in text element

- TCA type `text` now supports the HTML5 attribute `maxlength` to restrict the length of a text (note: line breaks are usually counted as two characters)

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

  Please note, that not every browser supports this attribute.
  See Browser Support List for details.

In-Depth Changes

New SplFileInfo implementation

- New class: `TYPO3\CMS\Core\Type\File\FileInfo`
- This class extends class `SplFileInfo`, which allows fetching meta information from files

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

- Custom implementations can use the following hook:

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

In-Depth Changes

UserFunc in TCA Display Condition

- userFunc `displayCondition` makes it possible to check on any imaginable condition or state
- If a situation can not be evaluated with any of the existing checks, developers can developer their own user function (return `TRUE`/`FALSE` to show/hide appropriate TCA field)

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

In-Depth Changes

API for Twitter Bootstrap modals (1)

- Two new API methods to create/remove modal popups:
  - `TYPO3.Modal.confirm(title, content, severity, buttons)`
  - `TYPO3.Modal.dismiss()`
- Options `title` and `content` are required
- Options `buttons.text` and `buttons.trigger` are also required, if `buttons` is used
- Example 1:

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

In-Depth Changes

API for Twitter Bootstrap modals (2)

- Example 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();
      }
    }
  ]
);
```

In-Depth Changes

JavaScript Storage API (1)

- Accessing the BE user configuration (`$BE_USER->uc`) can be handled in JavaScript by using simple key-value pairs
- Additionally, HTML5's localStorage can be used to store data in the user's browser (client-side)
- Two new global TYPO3 objects:
  - `top.TYPO3.Storage.Client`
  - `top.TYPO3.Storage.Persistent`
- Each object has the following API methods:
  - `get(key)`: fetch data
  - `set(key,value)`: write data
  - `isset(key)`: check, if key is already used
  - `clear()`: empty all storage data

In-Depth Changes

JavaScript Storage API (2)

- Example:

```
// 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');
```

In-Depth Changes

Inline Rendering of Checkboxes

- Checkbox setting `inline` for "cols" can be used to render checkboxes directly next to each other to reduce the amount of space used

```
'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'
  )
),
...
```

In-Depth Changes

Content Object Registration

- New global option to register and/or extend/overwrite cObjects such as `TEXT` has been introduced
- A list of all available cObjects is available as:

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

- Example: register a new cObject `EXAMPLE`

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

- The registered class must be a subclass of `TYPO3\CMS\Frontend\ContentObject\AbstractContentObject`
- Store your class in directory `typo3conf/myextension/Classes/ContentObject/` to be prepared for future autoload mechanisms

In-Depth Changes

Hooks and Signals (1)

- New hook has been added to the end of `PageRepository->init()`, which allows to influence the visibility of pages
- Register the hook as follows:

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

- The hook class must implement the following interface:

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

In-Depth Changes

Hooks and Signals (2)

- New hook has been added to the PageLayoutView to manipulate the rendering of the footer of a content element.
- Example:

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

- The hook class must implement the following interface:

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

In-Depth Changes

Hooks and Signals (3)

- New hook has been added as a post processor of `BackendUtility::countVersionsOfRecordsOnPage`
- This can be used to visualize workspace states in the page tree for example
- Register the hook as follows:

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

In-Depth Changes

Hooks and Signals (4)

- New signal has been added to the end of method `DataPreprocessor::fetchRecord()`
- This can be used to manipulate array `regTableItems_data` for example, in order to display manipulated data in TCEForms

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

In-Depth Changes

Hooks and Signals (5)

- New signal has been added, that allows for additional processing upon initialization of a mailer object, e.g. registering a Swift Mailer plugin

```
$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'
);
```

In-Depth Changes

Multiple UID in `PageRepository::getMenu()`

- Method `PageRepository::getMenu()` accepts arrays now, in order to define multiple root pages

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

In-Depth Changes

SVG Support

- TYPO3 CMS core supports SVG images now ("Scalable Vector Graphics")
- When an SVG image is scaled, a record with the calculated new dimensions is stored in `sys_file_processedfile` rather than creating a processed file
  (except, is image is processed further, e.g. cropping).
- A fallback is added to determine SVG dimensions if ImageMagick/GraphicsMagick can not determine image dimensions. In this case, the contents of the XML file is read.
- SVG has also been added to the list of valid image files:
  `$GLOBALS['TYPO3_CONF_VARS']['GFX']['imagefile_ext']`

In-Depth Changes

Extending FAL Driver

- In order to improve the performance of the file list when showing (remote) storages the FAL driver should take care of sorting, ordering and determining the number of files/folders. Two new parameters `sort` and `sortRev` have been added to allow that:

```
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);
```

- Additionally, two new methods have been implemented:

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

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

In-Depth Changes

Backend Routing API (1)

- A Backend Routing API has been implemented, which manages the backend Entry Points
- Inspired by the Symfony Routing Framework, this API is compatible with it to a large extent
  (however TYPO3 uses only approx. 20% at this point in time)
- Basically three classes implement the functionality:

`class Route`:	contains details about path and options
`class Router`:	API to match the route
`class UrlGenerator`:	generates the URL

In-Depth Changes

Backend Routing API (2)

- Routes are defined in the following file of an extension:
  `Configuration/Backend/Routes.php`
  (see system extension `backend` as an example)
- Further details about the Backend Routing API:
  http://wiki.typo3.org/Blueprints/BackendRouting

TYPO3

In-Depth Changes

New System Extension for Media Content Elements

- New system extension "`mediace`" contains the following cObjects:
    - `MULTIMEDIA`
    - `MEDIA`
    - `SWFOBJECT`
    - `FLOWPLAYER`
    - `QTOBJECT`
- Content elements `media` and `multimedia` have been moved to this system extension, too, as well as the "Media Wizard Provider"
- This extension is **not** installed by default!

In-Depth Changes

Location of Third-party Libraries

- Composer-installed third-party libraries are installed under `typo3/contrib/vendor` now (TYPO3 CMS < 7.2: in folder `Packages/Libraries`)
- This way the packaging process for releasing TYPO3 CMS as tarball or zip can trigger a fully working installation without having to ship `Packages/` for third-party libraries
- Problems may occur with installations if they were set up via composer and use `phpunit` unless composer dependencies have been completely rebuilt. To fix this, execute:

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

In-Depth Changes

JavaScript Notifications

- A new JavaScript Notification API has been implemented:

```
// old and deprecated:
top.TYPO3.Flashmessages.display(TYPO3.Severity.notice)

// new and the only correct way since TYPO3 CMS 7.2:
top.TYPO3.Notification.notice(title, message)
```

- The following API functions exist:
  (parameter `duration` is optional and features a default value of 5 seconds)
  - `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)`

In-Depth Changes

System Information Dropdown (1)

- Custom system information items can be added to the dropdown by creating a slot
- The slot must be registered in file `ext_localconf.php`:

```
$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'
);
```

In-Depth Changes

System Information Dropdown (2)

- Custom system information items can be added to the dropdown by creating a slot
- This requires class `Item` and its method `getItem()` in file `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')
    ));
  }
}
```

In-Depth Changes

System Information Dropdown (3)

- Icon `extensions-example-information-icon` must be registered in `ext_localconf.php`:

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

In-Depth Changes

System Information Dropdown (4)

- Messages are shown at the bottom of the dropdown
- Extensions can provide its own slot to fill messages:

```
$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'
);
```

In-Depth Changes

System Information Dropdown (5)

- Messages are shown at the bottom of the dropdown
- This requires the class `Message` and its method `getMessage()` in file `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.'
    ));
  }
}
```

In-Depth Changes

Image Manipulation Configuration Options (1)

- The following TypoScript configuration options are available:

```
# disable cropping for all images
tt_content.image.20.1.file.crop =

# override or set cropping for all images
# offsetX,offsetY,width,height
tt_content.image.20.1.file.crop = 50,50,100,100
```

- Fluid also supports cropping functions:

```
# disable cropping for all images
<f:image image="{imageObject}" crop="" ></f:image>

# override or set cropping for all images
# offsetX,offsetY,width,height
<f:image image="{imageObject}" crop="50,50,100,100" ></f:image>
```

In-Depth Changes

Image Manipulation Configuration Options (2)

- TCA features the image cropping functionality, too:
  - 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'
)
```

In-Depth Changes

Additional Parameters for HTMLparser userFunc

- Additional parameters can be supplied to a userFunc of the HTMLparser:

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

- Access to these parameters in an extension as follows:

```
function htmlUserFunc(array $params, HtmlParser $htmlParser) {
  // $params['attributeValue'] contains the attribute value "myclass"
  // $params['myparam'] is set to "test" in this example
  ...
}
```

In-Depth Changes

Locking API (1)

- New Locking API has been introduced, which provides various locking methods (SimpleFile, Semaphore, ...)
- A locking method must implement the `LockingStrategyInterface`:

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

In-Depth Changes

Locking API (2)

- Some methods also support non-blocking locks:

```
$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) {
    // some process owns the lock
    // let's do something else meanwhile
    ...
  }
}
if ($result) {
  $locker->release();
}
```

In-Depth Changes

Signal after Extension Installation

- New signal has been implemented in method `\TYPO3\CMS\Extensionmanager\Utility\InstallUtility::install()` which emits as soon as an extension has been installed and all imports/updates finished

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

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

In-Depth Changes

Registry for Text Extraction (1)

- Multiple text extractors can be registered to allow dealing with different file types (e.g. Office, PDF files, etc.)
- TYPO3 core ships with an extractor for plain text files
- Every registered text extractor class needs to implement the `TextExtractorInterface`
- ...and the following methods:
  `canExtractText()`
  checks if text extraction from the given file is possible
  `extractText()`
  returns the file's text content as a string

In-Depth Changes

Registry for Text Extraction (2)

- Text extractor registration in file `ext_localconf.php`:

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

- Usage as follows:

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

In-Depth Changes

Miscellaneous

- Web libraries (such as Twitter Bootstrap, jQuery, Font Awesome, etc.) use "Bower" (`http://bower.io`) and are not part of the TYPO3 core Git repository anymore

`# bower install`	executes an installation
`# bower update`	executes an update

  (file `bower.json` is located in directory `Build/`)
- Scheduler CLI has received option "`-s`" to stop a running task
- The processing folder of a (remote) storage can be outside of the storage (useful in case of a read-only storage for instance)
- It is now possible to retrieve the page ID of the originally requested page: `$TSFE->getRequestedId()`

In-Depth Changes

Symfony/Console Integration into CommandController (1)

The CommandController now makes use of Symfony/Console internally and provides various methods:

- 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)`

In-Depth Changes

Symfony/Console Integration into CommandController (2)

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

(see next slides for code examples)

In-Depth Changes

Symfony/Console Integration into CommandController (3)

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

class MyCommandController extends CommandController {
  public function myCommand() {

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

    // select
    $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]));

    [...]
```

In-Depth Changes

Symfony/Console Integration into CommandController (4)

```
    [...]
    // ask
    $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();
  }
}
?>
```

In-Depth Changes

Backend Login API (1)

- Backend login has been completely refactored and a new API has been introduced
- The OpenID form has been extracted and is now using the new API (makes it independent of the central Core classes)
- The concept of the new backend login is based on "login providers", which can be registered in file `ext_localconf.php` as follows:

```
$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'
];
```

In-Depth Changes

Backend Login API (2)

- Options are defined as follows:
  - `provider`:
    login provider class name, which must implement
    `TYPO3\CMS\Backend\LoginProvider\LoginProviderInterface`
  - `sorting`:
    ordering of the links to the possible login providers on the login screen
  - `icon-class`:
    font-awesome icon name for the link on the login screen
  - `label`:
    label for the login provider link on the login screen

In-Depth Changes

Backend Login API (3)

- The `LoginProviderInterface` only contains method
  `public function render(StandaloneView $view, PageRenderer $pageRenderer, LoginController $loginController);`
- Parameters are defined as follows:
  - `$view`:
    Fluid StandaloneView which renders the login screen. You have to set the template file and you may add variables to the view according to your needs.
  - `$pageRenderer`:
    PageRenderer instance provides possibility to add necessary JavaScript resources.
  - `$loginController`:
    LoginController instance.

In-Depth Changes

Backend Login API (4)

- Template must contain `<f:layout name="Login">` and `<f:section name="loginFormFields">` (for form fields):

```
<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>
```

In-Depth Changes

CategoryRegistry with New Options

- Method CategoryRegistry->addTcaColumn received options to set l10n_mode and l10n_display:

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

In-Depth Changes

Sprites in Backend Modules

- Backend modules (main modules such as "Web" as well as submodules such as "Filelist") can now use sprites as icons
  (only sprite icons known to TYPO3 are available!)
- Example:

```
\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',
    ),
  )
);
```

In-Depth Changes

FormEngine NodeFactory API (1)

- It is now possible to register new nodes and overwriting existing nodes

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

- Example above registers a new class `MyVendor\MyExtension\Form\Element\T3editorElement` as render class for TCA type `input`, which must implement the interface `TYPO3\CMS\Backend\Form\NodeInterface`
- The array key is the Unix timestamp of the date when a registry element is added

In-Depth Changes

FormEngine NodeFactory API (2)

- In cases where multiple registry elements have been registered for the same type, the resolver with the highest priority (0 to 100) is used
- A new TCA type can be registered as follows:

  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
);
```

In-Depth Changes

Signal `postProcessMirrorUrl`

- Signal `postProcessMirrorUrl` has been moved to a new class

BREAKING CHANGE!

- The following code example takes the TYPO3 version into account:

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

In-Depth Changes

Driver Interface

- The following methods have been added to `DriverInterface`:
  - `getFolderInFolder`
  - `getFileInFolder`
- Every FAL driver should implement these new methods:

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

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

BREAKING CHANGE!

In-Depth Changes

IEC/SI Support in File Size Formatting

- File size formatting supports two keywords additionally to the list of labels now:
  - `iec` (default)
    (power of 2, labels: `| Ki| Mi| Gi| Ti| Pi| Ei| Zi| Yi`)
  - `si`
    (power of 10, labels: `| k| M| G| T| P| E| Z| Y`)
- Set formating in TypoScript for example:
  `bytes.labels = iec`

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

In-Depth Changes

Dependency Ordering Service (1)

- In many cases it is necessary to create a sorted list of items from a set of "dependencies". The ordered list is then used to execute actions in the given order.
- Some examples where the TYPO3 core uses this are:
  - hook execution order,
  - extension loading order,
  - listing of menu items,
  - etc.
- The `DependencyResolver` has been re-worked and provides a `DependencyOrderingService` now

In-Depth Changes

Dependency Ordering Service (2)

- Usage:

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

- Example:

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

In-Depth Changes

Hooks and Signals (1)

- Hook to post-process `InlineRecordContainer::checkAccess` results has been added
- `InlineRecordContainer::checkAccess` can be used to check the access to related inline records
- The following code registers the hook:

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

In-Depth Changes

Hooks and Signals (2)

- Hook to post-process login failures in `AbstractUserAuthentication::checkAuthentication` has been added
- Process stalls for 5 seconds in case of a failed login by default
- Using this new hook, alternative solutions could be implemented (e.g. to prevent brute force attacks)
- The following code registers the hook:

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

In-Depth Changes

Hooks and Signals (3)

- New signal `recordMarkedAsMissing` is emitted when the FAL indexer encounters a `sys_file` record which does not have a corresponding filesystem entry and marks it as missing. The signal passes the `sys_file` record UID.
- This is useful in extensions which provide or extend file management capabilities such as versioning, synchronizations, recovery, etc.
- Signal `afterMappingSingleRow` is emitted whenever the DataMapper creates an object

In-Depth Changes

HTML in TypoLink Titles

- Quotes in TypoLink titles are *escaped* automatically now
- This means, instances where HTML codes are already escaped manually, will break the frontend output in TYPO CMS 7.4

  Before: `'Some &quot;special&quot; title'`

  Becomes: `'Some &amp;quot;special&amp;quot; title'`
- It is recommended to avoid escaping, due to the fact that TYPO3 takes care of escaping HTML in TypoLink titles now

BREAKING CHANGE!

In-Depth Changes

Miscellaneous (1)

- By configuring the backend user permission `Files->replace`, users can be allowed to or prevented from replacing Files in module Filelist
- A hash is used in the filename of files, generated by the FileWriter, if no other log file has been configured
  - before: `typo3temp/logs/typo3.log`
  - now: `typo3temp/logs/typo3_<hash>.log`

  (value `<hash>` is calculated based on the encryption key)

In-Depth Changes

Miscellaneous (2)

- Classes used in hooks have to follow the autoloading mechanism
- Therefore the hook definition can be shortened now:

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

BREAKING CHANGE!

In-Depth Changes

Fluid-based Content Elements (1)

- New system extension **"Fluid-based Content Elements"** has been implemented
- Fluid templates are used for the rendering of content elements rather than TypoScript
- Could be an alternative to *CSS Styled Content* at one point in the future
- Include the following static templates in order to use this feature:
  - Content Elements (`fluid_styled_content`)
  - Content Elements CSS (optional) (`fluid_styled_content`)

In-Depth Changes

Fluid-based Content Elements (2)

- In addition, the following PageTSconfig template has to be added to the page properties:

  `Fluid-based Content Elements (fluid_styled_content)`

- Overwrite default templates by adding own paths in TypoScript setup:

```
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/
```

In-Depth Changes

Fluid-based Content Elements (3)

- Migrate from *CSS Styled Content* to *Fluid-based Content Elements*:
    - Uninstall extension `css_styled_content`
    - install extension `fluid_styled_content`
    - Use the Upgrade Wizard in the Install Tool to migrate Content Elements `text`, `image` and `textpic` to `textmedia`

Note: "*Fluid-based Content Elements*" is still in an early stage and breaking changes are possible until TYPO3 CMS 7 LTS. Also some conflicts regarding *CSS Styled Content* possibly still exist.

In-Depth Changes

SELECTmmQuery Method

- New method `SELECT_mm_query` has been added to class `DatabaseConnection`
- Extracted from `exec_SELECT_mm_query` to separate the building and execution of M:M queries.
- This enables the use of the query building in the database abstraction layer

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

In-Depth Changes

Optimize Database Tables in MySQL

- New scheduler task to run the MySQL command `OPTIMIZE TABLE` on selected database tables
- This command reorganizes the physical storage of table data and associated index data to reduce storage space and improve I/O efficiency
- The following types of tables are supported:
  `MyISAM`, `InnoDB` and `ARCHIVE`
- Using this task with DBAL and other DBMS is not supported due to the fact that the commands used are MySQL-specific

Note: optimizing tables is an I/O intensive process. Also in MySQL < 5.6.17 the process locks the tables while it is running, which may impact the website.

In-Depth Changes

Handling of Online Media (1)

- External medias (online media) are supported by default now
- As examples, the support for YouTube and Vimeo videos has been implemented in the core
- Resources can be added as URLs using content element **"Text & Media"** for example
- Matching helper class fetches the meta data and supplies an image that will be used as the preview if available

In-Depth Changes

Handling of Online Media (2)

The following URL syntaxes are possible:

YouTube:

```
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:

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

In-Depth Changes

Handling of Online Media (3)

- Accessing the resources using Fluid can be achieved as follows:

```
<!-- 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>
```

- Custom configuration options for YouTube videos:

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

- Custom configuration options for Vimeo videos:

  `autoplay`, `loop`, `showinfo`

In-Depth Changes

Handling of Online Media (4)

- To register your own online media service, you need an `OnlineMediaHelper` class that implements `OnlineMediaHelperInterface` and a `FileRenderer` class that implements `FileRendererInterface`

```
// register your own online video service (the used key is also the bind file extension name)
$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
);

// register an custom mime-type for your videos
$GLOBALS['TYPO3_CONF_VARS']['SYS']['FileInfo']['fileExtensionToMimeType']['myvideo'] =
  'video/myvideo';

// register your custom file extension as allowed media file
$GLOBALS['TYPO3_CONF_VARS']['SYS']['mediafile_ext'] .= ',myvideo';
```

In-Depth Changes

Backend Routing

- New routing component has been added to the TYPO3 backend which handles addressing different calls/modules inside TYPO3 CMS
- Routes can be defined in the following class:

  `Configuration/Backend/Routes.php`

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

- Called method contains PSR-7 compliant request and response objects:

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

In-Depth Changes

Autoload Definition in `ext_emconf.php`

- Extensions may provide one or more PSR-4 definitions in file `ext_emconf.php` now
- This was already possible in `composer.json`, but with this new feature, extension developers do not need to provide a composer file just for this anymore

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

(this is the new recommended way to register classes in TYPO3)

In-Depth Changes

New Icon Factory (1)

- Logic for working with icons, icon sizes and icon overlays is now bundled into the new class `IconFactory`
- The new icon factory will replace the old icon skinning API step by step
- All core icons will be registered directly in the `IconRegistry` class
- Extensions must use `IconRegistry::registerIcon()` to override existing icons or add additional icons to the icon factory:

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

In-Depth Changes

New Icon Factory (2)

- The TYPO3 CMS core implements three icon provider classes:
  `BitmapIconProvider`, `FontawesomeIconProvider` and `SvgIconProvider`
- Example usage:

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

- Valid values for `Icon::SIZE_...` are:
  `SIZE_SMALL`, `SIZE_DEFAULT` and `SIZE_LARGE`

- Valid values for `Icon::STATE_...` are:
  `STATE_DEFAULT` and `STATE_DISABLED`

In-Depth Changes

New Icon Factory (3)

- The TYPO3 CMS core provides a Fluid ViewHelper which makes it easy to use icons within a Fluid view:

```
{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>
```

In-Depth Changes

Hooks and Signals

- New signal has been added to LinkValidator, which allows for additional processing upon initialization of a specific record (e.g. getting content data from plugin configuration in record)
- Registering the signal in file `ext_localconf.php`:

```
$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'
);
```

In-Depth Changes

JumpUrl as System Extension (1)

- Generation and handling of JumpURLs has been moved to a new system extension `jumpurl`
- New hooks were introduced that allow custom URL generation and handling (see next page)

BREAKING CHANGE!

In-Depth Changes

JumpUrl as System Extension (2)

- Hook 1: manipulating **URLs** during link generation

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

// class needs to implement the UrlHandlerInterface:
class MyUrlHandler implements \TYPO3\CMS\Frontend\Http\UrlHandlerInterface {
  ...
}
```

- Hook 2: handling of **links**

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

// class needs to implement the UrlProcessorInterface:
class MyUrlProcessor implements \TYPO3\CMS\Frontend\Http\UrlProcessorInterface {
  ...
}
```

In-Depth Changes

Command Line Interface (CLI)

- Calling `typo3/cli_dispatch.phpsh` via the command line now shows a colored error message if an invalid or no CLI key as first parameter was given
- Extbase command controllers can now reside in arbitrary subfolders within the `Command` folder
- Example:

  Controller in file:

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

  ...can be called via CLI:

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

In-Depth Changes

Miscellaneous (1)

- The move buttons of the TCA type `group` can now be explicitly disabled by using option `hideMoveIcons = TRUE`
- Method `makeCategorizable` has been extended with a new parameter `override` to set a new category configuration for an already registered table/field combination
- Example:

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

Last parameter (here: `TRUE`) forces override (default value is `FALSE`).

In-Depth Changes

Miscellaneous (2)

- New function generates a unique ID

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

- The file type `.typoscript` has been added to the list of valid plain text file types
- New configuration option defines file extensions of media files

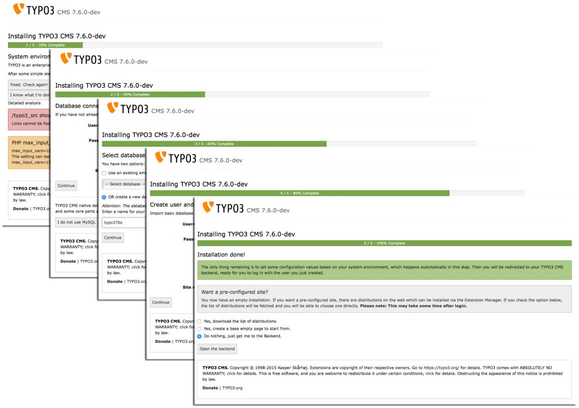
```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['mediafile_ext'] =
  'gif,jpg,jpeg,bmp,png,pdf,svg,ai,mov,avi';
```

BREAKING CHANGE!

In-Depth Changes

Bootstrap for Install Tool (1)

- The Install Tool is now based on Bootstrap – for the installation part:



TYPO3 CMS 7.6.0-dev

Installing TYPO3 CMS 7.6.0-dev

1 / 5 - 20% Complete

Installing TYPO3 CMS 7.6.0-dev

2 / 5 - 40% Complete

Installing TYPO3 CMS 7.6.0-dev

3 / 5 - 60% Complete

Installing TYPO3 CMS 7.6.0-dev

4 / 5 - 80% Complete

Installing TYPO3 CMS 7.6.0-dev

5 / 5 - 100% Complete

Installation done!

The only thing remaining is to set some configuration values based on your system environment, which happens automatically in this step. Then you will be redirected to your TYPO3 CMS backend, ready for you to log in with the user you just created.

Want a pre-configured site?

You now have an empty installation. If you want a pre-configured site, there are distributions on the web which can be installed via the Extension Manager. If you check the option below, the list of distributions will be fetched and you will be able to choose one directly. **Please note: This may take some time after login.**

- Yes, download the list of distributions.
- Yes, create a base empty page to start from.
- [✓ Do nothing, just get me to the Backend.]

Open the backend

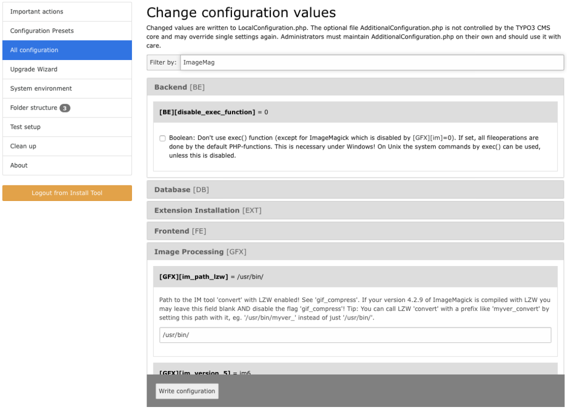
TYPO3 CMS. Copyright © 1998-2015 Kasper Skårhøj. Extensions are copyright of their respective owners. Go to https://typo3.org/ for details. TYPO3 comes with ABSOLUTELY NO WARRANTY; click for details. This is free software, and you are welcome to redistribute it under certain conditions; click for details. Obstructing the appearance of this notice is prohibited by law.

Donate | TYPO3.org

In-Depth Changes

Bootstrap for Install Tool (2)

- The Install Tool is now based on Bootstrap – for the configutation:



Change configuration values

Changed values are written to LocalConfiguration.php. The optional file AdditionalConfiguration.php is not controlled by the TYPO3 CMS core and may override single settings again. Administrators must maintain AdditionalConfiguration.php on their own and should use it with care.

Filter by: ImageMag

Backend [BE]

[BE][disable_exec_function] = 0

Boolean: Don't use exec() function (except for ImageMagick which is disabled by [GFX][im]=0). If set, all fileoperations are done by the default PHP-functions. This is necessary under Windows! On Unix the system commands by exec() can be used, unless this is disabled.

Database [DB]

Extension Installation [EXT]

Frontend [FE]

Image Processing [GFX]

[GFX][im_path_lzw] = /usr/bin/

Path to the IM tool 'convert' with LZW enabled! See 'gif_compress'. If your version 4.2.9 of ImageMagick is compiled with LZW you may leave this field blank AND disable the flag 'gif_compress'! Tip: You can call LZW 'convert' with a prefix like 'myver_convert' by setting this path with it, eg. '/usr/bin/myver_' instead of just '/usr/bin/'.

/usr/bin/

[GFX][im_version_5] = im6

In-Depth Changes

CSRF Protection for Frontend Plugins

- New class allows usage of the FormProtection API in the frontend
- This implements a CSRF protection (Cross-Site Request Forgery)

```
$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')) {
  // processes the data
}
else {
  // invalid token!
}
```

In-Depth Changes

Tabs for LinkBrowser (1)

- This new feature allows to extend the LinkBrowser with new tabs
- Each tab is handled by a so called "LinkHandler", which has to implement the following Interface:
  `\TYPO3\CMS\Recordlist\LinkHandler\LinkHandlerInterface`
- LinkHandlers are registered in PageTSconfig as follows:

```
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
  }
}
```

In-Depth Changes

Tabs for LinkBrowser (2)

- Options `displayBefore` and `displayAfter` define the positions of the tabs
- Options `scanBefore` and `scanAfter` define the order in which handlers are executed when scanning existing links

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

In-Depth Changes

Module Template API (1)

- A new Module Template API aims to normalize the implementation of DocHeaders
- Example 1: adding a button

```
$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);
```

In-Depth Changes

Module Template API (2)

- Example 2: adding a menu with menu items

```
$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);
```

In-Depth Changes

PSR-7 Routing for Backend AJAX Requests

- To add a route for an AJAX request, file `Configuration/Backend/AjaxRoutes.php` can be created with the following content:

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

In-Depth Changes

OpenID Hook `getUserRecord`

Two hooks have been added to the OpenID service (1/2)

- Hook 1:

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

  - Modifies the user record after it has been fetched, or:
  - Create a new record if none was found
  - Parameters `record`, `response` and `authInfo` are passed to the hook

In-Depth Changes

OpenID Hook `authRequest`

Two hooks have been added to the OpenID service (2/2)

- Hook 2:

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

  - Modifies the Authentication Request, before it is sent
  - Can be used to request additional attributes such as a nickname from the OpenID Server for example
  - Parameters `authRequest` and `authInfo` are passed to the hook

In-Depth Changes

Hooks and Signals (1)

- It is now possible to change the upload folder returned by `BackendUserAuthentication::getDefaultUploadFolder()`
- Register the hook in file `ext_localconf.php` as follows:

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

In-Depth Changes

Hooks and Signals (2)

Example:

```
<?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)
          {
    [...]
```

In-Depth Changes

Hooks and Signals (3)

Example (continued):

```
    [...]

    /** @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;
  }
}
```

In-Depth Changes

Miscellaneous

- Using the TCA field type `select` requires to specify an option `renderType`
- Valid values are:

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

Sources and Authors

Sources and Authors

Sources and Authors

Sources

TYPO3 News:

- http://typo3.org/news

Release Infos:

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

TYPO3 Bug-/Issuetracker:

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

TYPO3 Git Repositories:

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

Sources and Authors

TYPO3 CMS What's New Slides:

Patrick Lobacher
(Research, Information Gathering and German Version)

Michael Schams
(Project Leader and English Version)

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

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