

# TYPO3 v8 LTS – What's New

The most important  
new features, changes and improvements in 111 slides

## Introduction

*Quick overview of the facts*

# Introduction

---

## TYPO3 v8 LTS

- Release date: 4 April 2017
- Release type: LTS Release (Long Term Release)
- Development time: 18+ months

# Introduction

---

## System Requirements (1)

- PHP 7.0 is the minimum requirement for TYPO3 v8 LTS
- Subsequent PHP 7 releases will be supported as they are released
- PHP 7 provides a significant performance boost
- Allows usage of new PHP 7 specific features
- Required PHP settings:
  - `memory_limit`  $\geq$  128M
  - `max_execution_time`  $\geq$  240s
  - `max_input_vars`  $\geq$  1500
  - compilation option `--disable-ipv6` must not be used

The logo for PHP 7, featuring the text "php7" in a white, lowercase, sans-serif font inside a blue rounded rectangle.

# Introduction

---

## System Requirements (2)

- TYPO3 v8 LTS uses **Doctrine DBAL**. All database servers supported by this DB abstraction layer are also supported by TYPO3.

For example:



- Minimum disk space required: 200 MB
- The backend requires Microsoft Internet Explorer 11 or later, Microsoft Edge, Google Chrome, Firefox, Safari or any other modern, compatible browser

# Introduction

---

## Development Timeline

### Sprint Releases published:

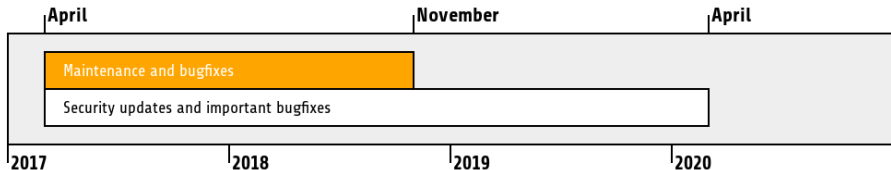
- v8.0 22/Mar/2016 Fluid Standalone Engine
- v8.1 03/May/2016 Cloud Integration
- v8.2 05/Jul/2016 Doctrine Prerequisites
- v8.3 30/Aug/2016 Rich Text Editor
- v8.4 18/Oct/2016 Doctrine Migration + Upgrades
- v8.5 20/Dec/2016 New RTE + Integrator Support
- v8.6 14/Feb/2017 Cropping, Link Handling etc.
- v8.7 04/Apr/2017 LTS Preparation and Release

# Introduction

---

## Long Term Support

Maintenance/support timeline:



- TYPO3 version 8.7 is a LTS Release (Long Term Support)
- Regular maintenance and bugfixes until October 2018
- Security and critical bugfixes until April 2020

## Backend User Interface

*The TYPO3 CMS administration interface got even better*



# Backend User Interface

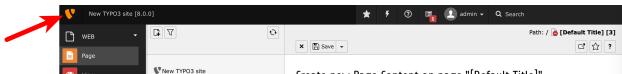
---

## Alternative Logo

The backend logo in the upper left corner can now be configured in the extension configuration of EXT:backend in the Extension Manager.

Configuration options are:

- resource as a relative path of the TYPO3 installation  
e.g. "fileadmin/images/my-background.jpg"
- path to an extension  
e.g. "EXT:my\_theme/Resources/Public/Images/my-background.jpg"
- an external resource  
e.g. "//example.com/my-background.png"

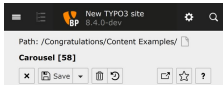


# Backend User Interface

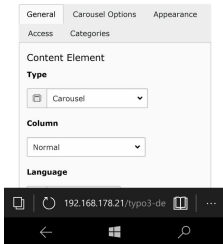
---

## Mobile Responsive TYPO3 Backend

The TYPO3 Backend is fully mobile responsive now.



Edit Page Content "Carousel"  
on page "Carousel"

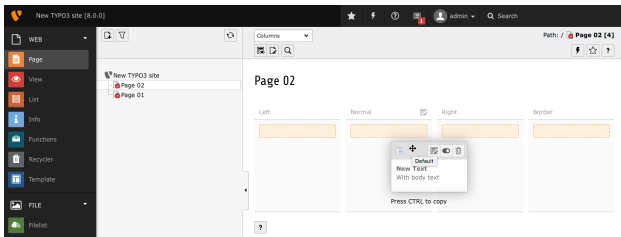


# Backend User Interface

---

## Drag & Drop Can Copy Elements

Additionally to the usual drag and drop feature in the page module (that *moved* content elements), it is now possible to create copies: press the CTRL key while dropping to create a copy of the dragged element. After dropping is complete, the page module will reload to make sure the new element will be generated with all necessary information.



# Backend User Interface

---

## Image Manipulation (1)

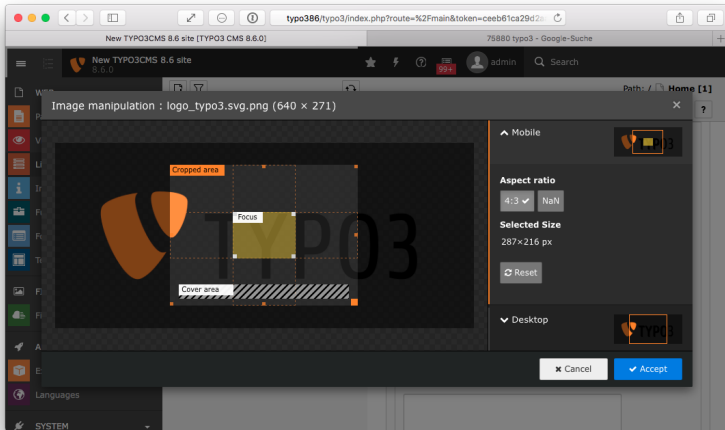
- Multiple aspect ratios can be configured (default ratios: 16:9, 3:2, 4:3 and 1:1)
- Editors can set a focal point (responsive images) (this area of an image is still visible, even if the image is cropped, e.g. on small screens such as on smartphones)
- Integrators/developers can configure *crop variants* (e.g. "mobile", "desktop", etc.) and also use these in TCA and Fluid templates

```
<f:image image="{data.image}" cropVariant="mobile" width="800">  
</f:image>
```

See [docs.typo3.org](https://docs.typo3.org) for further details.

# Backend User Interface

## Image Manipulation (2)



# Backend User Interface

---

## Simplify Cache Clearing

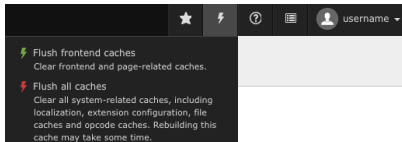
The cache clearing system has been simplified by removing options in cache clear menu and Install Tool.

- **Flush frontend caches:**

Clears frontend and page-related caches, like before.

- **Flush all caches:**

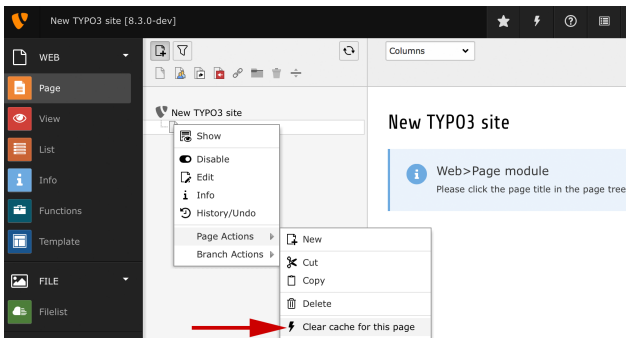
Clears all system-related caches, including the class loader, localization, extension configuration file caches and opcode caches. Rebuilding this cache may take some time.



# Backend User Interface

## "Clear Cache" Entry in Context Menu

A new entry has been added to the context menu of the page tree. The item is located inside "Page Actions" and allows to clear the cache of the selected page.



# Backend User Interface

---

## Reworked Workspaces (1)

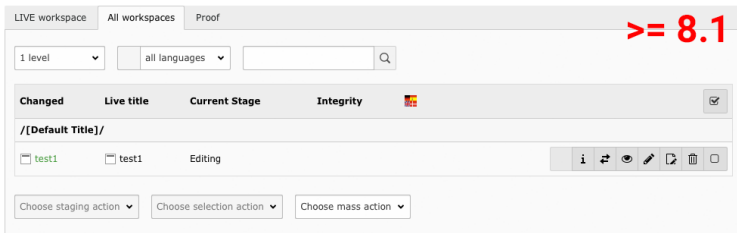
- The workspace module to manage staged content has been rewritten and integrates much better into the visual appearance of the backend now
- Editors will realize straight away, it fits the overall look and feel due to its technical base with Twitter Bootstrap and jQuery
- This change also brings a performance boost and is a huge leap forward to a cleaner and faster TYPO3 backend with less JavaScript



# Backend User Interface

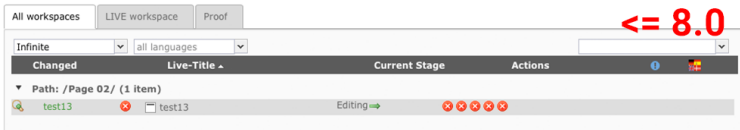
## Reworked Workspaces (2)

Screenshots of the workspace module:



This screenshot shows the workspace module interface for version 8.1 and above. It features a top navigation bar with tabs for "LIVE workspace", "All workspaces", and "Proof". A red label ">= 8.1" is positioned in the top right corner. Below the navigation bar, there are filters for "1 level" and "all languages", along with a search input field. The main content area displays a table with columns for "Changed", "Live title", "Current Stage", and "Integrity". A table row is visible with "test1" in the "Live title" column and "Editing" in the "Current Stage" column. To the right of the table row is a set of icons for actions. Below the table, there are three dropdown menus for "Choose staging action", "Choose selection action", and "Choose mass action".

Legend: • edited • • moved • • created • • hidden • • deleted



This screenshot shows the workspace module interface for version 8.0 and below. It features a top navigation bar with tabs for "All workspaces", "LIVE workspace", and "Proof". A red label "<= 8.0" is positioned in the top right corner. Below the navigation bar, there are filters for "Infinite" and "all languages", along with a search input field. The main content area displays a table with columns for "Changed", "Live-Title", "Current Stage", and "Actions". A table row is visible with "test13" in the "Live-Title" column and "Editing" in the "Current Stage" column. To the right of the table row is a set of icons for actions, including a red 'x' icon.

# Backend User Interface

---

## Position and Order of Elements

- The order and position of certain fields in the backend of TYPO3 has been streamlined
- The aim is to meet users' expectation where to find commonly used options in the user interface
- This is especially important for recurring field definitions and generic categories shared by a lot of records
- Extension authors are encouraged to follow the specified positions and orders of elements in the [official documentation](#)
- *Backend consistency is king! :-)*

## Form Framework

*Creating complex forms becomes a piece of cake*

# Form Framework

---

## Main Facts

- Flexible new framework for building forms was integrated
- Uses jQuery and a modern architecture, ensuring high flexibility and extensibility
- Replaces the legacy *Form Wizard* based on ExtJS
- Forms are stored as templates, which can be re-used across the website
- Configuration stored in YAML files  
(can be exported, adjusted, version controlled, shared, etc.)
- Very intuitive to use (*editors will love it!*)

# Form Framework

---

## Validators and Finishers

### ■ **Validators:**

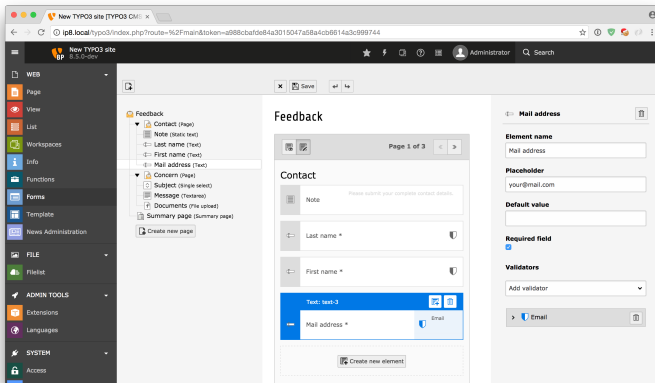
- Data entered can be verified by *Validators*
- Typical Validators are included in TYPO3 CMS (e.g. for fields such as email, alphanumeric, integer, regular expressions, etc.)
- Custom developed Validators can extend this functionality

### ■ **Finishers:**

- Control what should happen with the form data submitted (e.g. trigger an email, redirect to a page, etc.)
- Custom developed Finishers can extend this functionality

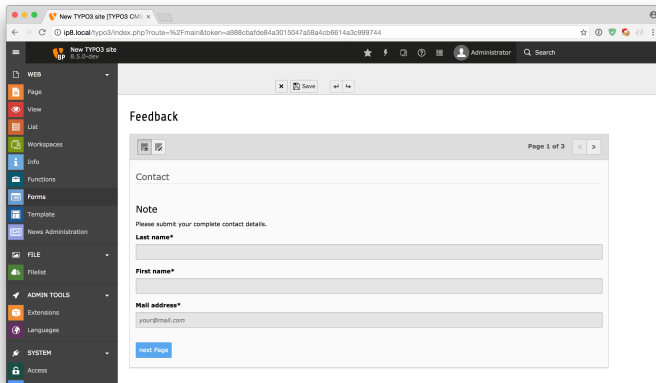
# Form Framework

## Screenshot (1)



# Form Framework

## Screenshot (2)



## Rich Text Editor

*Content editing taken a huge step further*



# Rich Text Editor

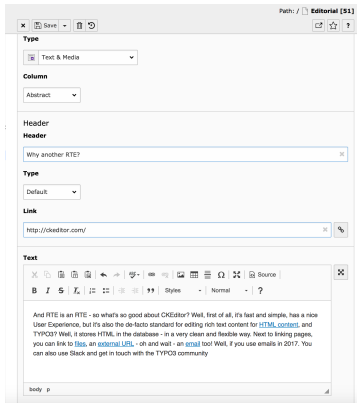
---

## CKEditor Integration

- New Rich Text Editor (RTE) was implemented: **CKEditor**
- Well-known and widely used in the PHP universe, easy to configure
- Replaces "HtmlArea" and lays the foundation for frontend editing in the near future
- HtmlArea is still available as an optional extension in the TYPO3 Extension Repository (TER)
- Crowdfunding campaign initiated by the Swedish web agency [Pixelant AB](#) raised more than 63,000.- Euro (110+ supports worldwide)

# Rich Text Editor

## Screenshots



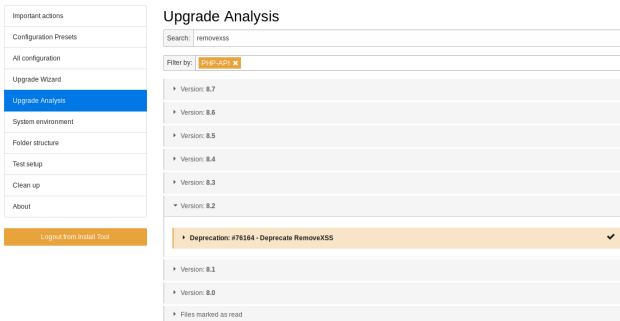
## Install Tool

*The powerful tool for integrators and developers alike*

# Install Tool

## Upgrade Analysis

TYPO3 version upgrades made easy with the new **Upgrade Analysis** tool in the Install Tool (find/filter documented changes between versions).

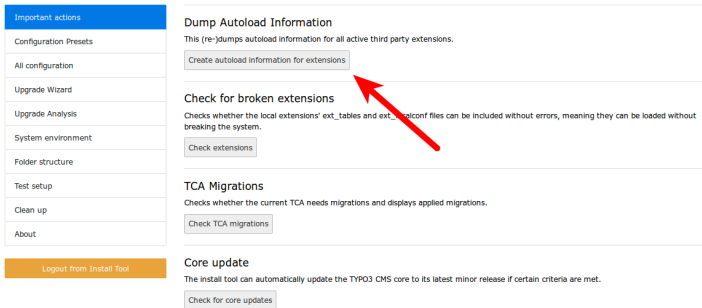


The screenshot displays the 'Upgrade Analysis' tool interface. On the left is a sidebar menu with the following items: 'Important actions', 'Configuration Presets', 'All configuration', 'Upgrade Wizard', 'Upgrade Analysis' (highlighted in blue), 'System environment', 'Folder structure', 'Test setup', 'Clean up', and 'About'. Below the menu is an orange button labeled 'Logout from Install Tool'. The main content area is titled 'Upgrade Analysis' and features a search bar with the text 'removexss' and a filter dropdown set to 'PHP-API'. A list of versions is shown, with 'Version: 8.2' expanded to reveal a deprecation notice: 'Deprecation: #76164 - Deprecate RemoveXSS', which is highlighted in orange and has a checkmark icon to its right. Other versions listed include 8.7, 8.6, 8.5, 8.4, 8.3, 8.1, and 8.0, along with a 'Files marked as read' entry.

# Install Tool

## Dump Autoload Information

In order to re-generate class loading information, a new action has been added to the Install Tool to dump autoload information.



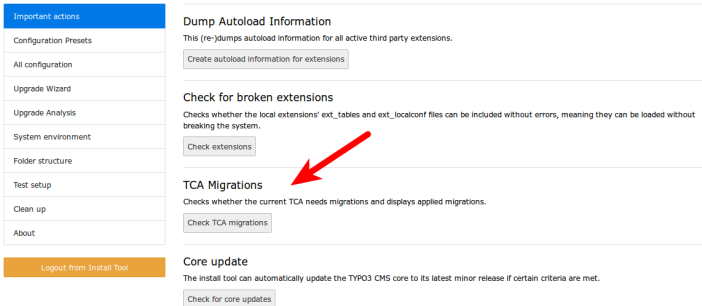
The screenshot displays the TYPO3 Install Tool interface. On the left is a navigation menu with the following items: Important actions (highlighted in blue), Configuration Presets, All configuration, Upgrade Wizard, Upgrade Analysis, System environment, Folder structure, Test setup, Clean up, and About. Below the menu is a 'Logout from Install Tool' button. The main content area is divided into several sections:

- Dump Autoload Information**: This (re-)dumps autoload information for all active third party extensions. It includes a button labeled 'Create autoload information for extensions', which is highlighted by a red arrow.
- Check for broken extensions**: Checks whether the local extensions' ext\_tables and ext\_localconf files can be included without errors, meaning they can be loaded without breaking the system. It includes a button labeled 'Check extensions'.
- TCA Migrations**: Checks whether the current TCA needs migrations and displays applied migrations. It includes a button labeled 'Check TCA migrations'.
- Core update**: The install tool can automatically update the TYPO3 CMS core to its latest minor release if certain criteria are met. It includes a button labeled 'Check for core updates'.

# Install Tool

## TCA Migration Messages

TCA migration message(s) can be checked/listed in the Install Tool now.



The screenshot displays the TYPO3 Install Tool interface. On the left is a vertical navigation menu with the following items: 'Important actions' (highlighted in blue), 'Configuration Presets', 'All configuration', 'Upgrade Wizard', 'Upgrade Analysis', 'System environment', 'Folder structure', 'Test setup', 'Clean up', and 'About'. At the bottom of the menu is a 'Logout from Install Tool' button. The main content area on the right contains several sections:

- Dump Autoload Information**: This (re-)dumps autoload information for all active third party extensions. It includes a button labeled 'Create autoload information for extensions'.
- Check for broken extensions**: Checks whether the local extensions' ext\_tables and ext\_localconf files can be included without errors, meaning they can be loaded without breaking the system. It includes a button labeled 'Check extensions'.
- TCA Migrations**: Checks whether the current TCA needs migrations and displays applied migrations. A red arrow points to this section. It includes a button labeled 'Check TCA migrations'.
- Core update**: The install tool can automatically update the TYPO3 CMS core to its latest minor release if certain criteria are met. It includes a button labeled 'Check for core updates'.

# Install Tool

## Update Wizard

The Update Wizard in the Install Tool lists all tasks marked as *completed*.

Checkboxes and a button "Recheck chosen wizards" allow to re-initiate the updates. The wizard will test if the task needs to be executed again.

### Wizards done

<input type="checkbox"/>	Deprecated RTE properties in Page TSconfig
<input type="checkbox"/>	Rte "acronym" button renamed to "abbreviation"
<input type="checkbox"/>	Move access right parameters configuration to "SYS" section
<input type="checkbox"/>	Update sys_language records to use new ISO 639-1 letter-code field
<input type="checkbox"/>	Update page shortcuts with shortcut type "Parent of selected or current page"
<input type="checkbox"/>	Migrate backend shortcut urls
<input type="checkbox"/>	[Optional] Update sys_file_processedfile records to match new checksum calculation.
<input type="checkbox"/>	Set the "Files:replace" permission for all BE user/groups with "Files:write" set
<input type="checkbox"/>	Migrate the Flexform for CType "table" to regular fields in t_content
<input type="checkbox"/>	Update flexlist user setting "starModule"
<input type="checkbox"/>	Migrate CType's text, image and textpic to textmedia and move file relations from "image" to "asset_references"
<input type="checkbox"/>	Migrate the workspaces notification settings to the enhanced schema
<input type="checkbox"/>	Migrate CType's textmedia database field "media" to "assets"
<input type="checkbox"/>	Update backend user setting "starModule"
<input type="checkbox"/>	[Optional] Install extensions "tba" and "adodt" from TER.

Recheck chosen wizards

## TSconfig & TypoScript

*New configuration options, new features, unlimited possibilities*



# TScnfig & TypeScript

---

## Multiple Locale Names for TypeScript `config.locale_all`

- TypeScript option `config.locale_all` now allows to set locale fallbacks as a comma-separated list, as the underlying PHP function `setlocale()` does as well:

```
config.locale_all = de_AT@euro, de_AT, de_DE, deu_deu
```

See <http://php.net/setlocale>

# TSconfig & TypoScript

---

## Configurable Width and Height for Editpanel in EXT:feedit

- Now it is possible to change the width and height of the popup, which is used in the edit panel of EXT:feedit by using User TSconfig:

```
options.feedit.popupHeight = 700  
options.feedit.popupWidth = 900
```

# TScnfig & TypoScript

---

## Access FlexForm Values

- It is now possible to access properties of a FlexForm field:

```
lib.flexformContent = CONTENT
lib.flexformContent {
    table = tt_content
    select {
        pidInList = this
    }

    renderObj = COA
    renderObj {
        10 = TEXT
        10 {
            data = flexform: pi_flexform:settings.categories
        }
    }
}
```

# TSconfig & TypoScript

---

## New Page Creation Wizard

- In previous versions of TYPO3 CMS, it was possible to override the "New Page Creation Wizard" via custom scripts:

```
mod.web_list.newPageWiz.overrideWithExtension = myextension
```

- The new way of handling entry-points and custom scripts is now built via modules/routes and the option listed above has been removed
- The following new TSconfig option can be used instead:  

```
mod.newPageWizard.override = my_custom_module
```
- Instead of setting the option to a certain extension key, a custom module or route needs to be specified

# TScnfig & TypoScript

---

## Number of Search Results

- Maximum number of search results can be configured in TypoScript now:  
`plugin.tx_indexedsearch.settings.blind.numberOfResults`
- This setting stores a list of values
- If number of search results is passed in the request and matches one of the values configured, this number is used
- If number of search results is not passed in the request or does not match any of the values configured, the first value of the list is used
- To keep backward compatibility, the default values are:  
10, 25, 50 and 100

# Tsconfig & TypoScript

---

## Fluid Data Processor for Menus (1)

- Menu processor utilizes HMENU to generate a JSON-encoded menu string that is be decoded again and assigned to FLUIDTEMPLATE
- Additional DataProcessing is supported and applied to each record
- Supported options: as, levels, expandAll, includeSpacer, titleField (see [TyposcriptReference](#) for more options)

# TScnfig & TypoScript

---

## Fluid Data Processor for Menus (2)

### ■ Example TypoScript configuration:

```
10 = TYPO3\CMS\Frontend\DataProcessing\MenuProcessor
10 {
    special = list
    special.value.field = pages
    levels = 7
    as = menu
    expandAll = 1
    includeSpacer = 1
    titleField = nav_title // title
    dataProcessing {
        10 = TYPO3\CMS\Frontend\DataProcessing\FilesProcessor
        10 {
            references.fieldName = media
        }
    }
}
```

# TScnfig & TypoScript

---

## Section Frame for CSS Styled Content Replaced with Frame Class

- The functionality provided by Section Frame has been streamlined with Fluid Styled Content and is now available as Frame Class.
- For this, the TypoScript keys now use the right part from the CSS class `csc-frame-` instead of numbers.

- Before:

```
tt_content.stdWrap.innerWrap.cObject.key.field = section_frame
tt_content.stdWrap.innerWrap.cObject.5 =< tt_content.stdWrap.innerWrap.cObject.default
tt_content.stdWrap.innerWrap.cObject.5.20.10.value = csc-frame csc-frame-ruler-before
```

- After:

```
tt_content.stdWrap.innerWrap.cObject.key.field = frame_class
tt_content.stdWrap.innerWrap.cObject.ruler-before =< tt_content.stdWrap.innerWrap.cObject.default
tt_content.stdWrap.innerWrap.cObject.ruler-before.20.10.value = csc-frame csc-frame-ruler-before
```



## Doctrine DBAL

*State-of-the-art database abstraction layer*

# Doctrine DBAL

---

## PHP Library "Doctrine DBAL" (1)

- The PHP library "[Doctrine DBAL](#)" has been added via composer dependency to work as a powerful database abstraction layer with many features for database abstraction, schema introspection and schema management within TYPO3 CMS
- A TYPO3-specific PHP class called `TYPO3\CMS\Core\Database\ConnectionPool` has been added as a manager for database connections
- All connections configured under `$GLOBALS['TYPO3_CONF_VARS']['DB']['Connections']` are accessible using this manager, enabling the parallel usage of multiple database systems

# Doctrine DBAL

---

## PHP Library "Doctrine DBAL" (2)

- By using the database abstraction options and the QueryBuilder provided SQL statements being built will be properly quoted and compatible with different DBMS out of the box as far as possible
- Existing `$GLOBALS['TYPO3_CONF_VARS']['DB']` options have been removed and/or migrated to the new Doctrine-compliant options
- The `Connection` class provides convenience methods for insert, select, update, delete and truncate statements
- For select, update and delete only simple equality comparisons (like `WHERE "aField" = 'aValue'`) are supported. For complex statements it is required to use the QueryBuilder.

## PHP Library "Doctrine DBAL" (3)

- The `ConnectionPool` class can be used like this:

```
// Get a connection which can be used for multiple operations
/** @var \TYPO3\CMS\Core\Database\Connection $conn */
$conn = GeneralUtility::makeInstance(ConnectionPool::class)->getConnectionForTable('aTable');
$affectedRows = $conn->insert(
    'aTable',
    $fields, // Associative array of column/value pairs, automatically quoted & escaped
);

// Get a QueryBuilder, which should only be used a single time
$query = GeneralUtility::makeInstance(ConnectionPool::class)->getQueryBuilderForTable('aTable');
$query->select('*')
    ->from('aTable')
    ->where($query->expr()->eq('aField', $query->createNamedParameter($aValue)))
    ->andWhere(
        $query->expr()->lte(
            'anotherField',
            $query->createNamedParameter($anotherValue)
        )
    )
$rows = $query->execute()->fetchAll();
```

# Doctrine DBAL

---

## Doctrine in Extbase

- No code updates are required if extension developers use Extbase's standard already
- Direct SQL query functionality also supports QueryBuilder objects and instances of `\Doctrine\DBAL\Statement` as prepared statements
- The following example works in any Extbase repository using native Doctrine DBAL statements:

```
$connection = $this->objectManager->get(ConnectionPool::class)->getConnectionForTable('mytable');  
$statement = $this->objectManager->get(  
    \Doctrine\DBAL\Statement::class,  
    'SELECT * FROM mytable WHERE uid=? OR title=?',  
    $connection  
);  
  
$query = $this->createQuery();  
$query->statement($statement, [$uid, $title]);
```

# Doctrine DBAL

---

## Miscellaneous (1)

- With the migration to Doctrine, hook `buildQueryParameters` has been introduced in class `DatabaseRecordList`.
- This hook replaces the hook `makeQueryArray` from the deprecated method `AbstractDatabaseRecordList::makeQueryArray`.
- Using the new hook allows modifying the parameters used to query the database for records to be shown in the record list view
- The hook can be registered in `ext_localconf.php`:

```
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']  
    [\TYPO3\CMS\Recordlist\RecordList\DatabaseRecordList::class]['buildQueryParameters'][]
```

- ...and implements the public method `buildQueryParametersPostProcess`

# Doctrine DBAL

---

## Miscellaneous (2)

- Extbase's persistence is now also built completely on Doctrine DBAL's QueryBuilder
- `EXT:dba1` and `EXT:adodb` have been removed from the TYPO3 core  
If third party extensions use the old `TYPO3_DB` API to query non-MySQL database tables, these two extensions can be installed from TER.
- `TYPO3_DB` shorthand functionality has been removed for most of the TYPO3 core PHP classes  
(using `$GLOBALS[TYPO3_DB]` is still possible but discouraged)

# In-Depth Changes

---

## In-Depth Changes

*Awesome new features and improvements under the hood*



# In-Depth Changes

---

## Support PECL-memcached in MemcachedBackend

- Support for the PECL module "memcached" has been added to the MemcachedBackend of the Caching Framework
- If both, "memcache" and "memcached" are installed, "memcache" is used to avoid being a breaking change.
- An integrator may set the option `peclModule` to use the preferred PECL module:

```
$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'  
        ]  
    ]  
];
```

# In-Depth Changes

---

## Native support for Symfony Console (1)

- TYPO3 supports the Symfony Console component out-of-the-box now by providing a new command line script located in `typo3/sysext/core/bin/typo3`. On TYPO3 instances installed via Composer, the binary is linked into the `bin`-directory, e.g. `bin/typo3`.
- The new binary still supports the existing command line arguments when no proper Symfony Console command was found as a fallback.
- Registering a command to be available via the `typo3` command line tool works by putting a `Configuration/Commands.php` file into any installed extension. The content of this file is an associative array, which contains the `Symfony/Console/Command` classes. The key is the name of the command to be called as the first argument to `typo3`.

# In-Depth Changes

---

## Native support for Symfony Console (2)

- A required parameter when registering a command is the `class` property. Optionally the `user` parameter can be set so a backend user is logged in when calling the command.
- Typical example of a `Configuration/Commands.php` file:

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

# In-Depth Changes

---

## Native support for Symfony Console (3)

- An example call could look like:

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

- For a non-Composer installation:

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

# In-Depth Changes

---

## Low-level Parameter Changes (1)

- Low-level commands listed below use the Symfony Console now
- New commands behave like the old ones, but allow using certain parameters
  - DeletedRecordsCommand
  - CleanFlexFormsRecordsCommand
  - OrphanRecordsCommand
  - LostFilesCommand
  - MissingFilesCommand
  - MissingRelationsCommand
  - DoubleFilesCommand
  - RteImagesCommand

# In-Depth Changes

---

## Low-level Parameter Changes (2)

- Related PHP classes have been removed  
(e.g. TYPO3\CMS\Lowlevel\DeletedRecordsCommand)
- Executing the command via `cli_dispatch` does not work anymore  
(e.g. `typo3/cli_dispatch lowlevel cleaner deleted`)
- Calling the PHP class results in a fatal PHP error now
- Commands can now be executed via CLI as follows:  
`/typo3/sysext/core/bin/typo3 cleanup:<command>`  
for example:  
`/typo3/sysext/core/bin/typo3 cleanup:deletedrecords`

# In-Depth Changes

---

## Cryptographically secure pseudorandom number generator

- A new cryptographically secure pseudo-random number generator (CSPRNG) has been implemented in the TYPO3 core. It takes advantage of the new CSPRNG functions in PHP 7.
- The API resides in the class `\TYPO3\CMS\Core\Crypto\Random`
- Example:

```
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);
```

# In-Depth Changes

---

## Password hashing algorithm: PBKDF2

- A new password hashing algorithm "PBKDF2" has been added to the system extension "saltedpasswords"
- PBKDF2 stands for: Password-Based Key Derivation Function 2
- The algorithm is designed to be computationally expensive to resist brute force password cracking



# In-Depth Changes

---

## PHP Library "Guzzle" (1)

- The PHP library "[Guzzle](#)" has been added via composer dependency to work as a feature rich solution for creating HTTP requests based on the PSR-7 interfaces already used within TYPO3
- Guzzle auto-detects available underlying adapters available on the system, like cURL or stream wrappers and chooses the best solution for the system
- A TYPO3-specific PHP class called `TYPO3\CMS\Core\Http\RequestFactory` has been added as a simplified wrapper to access Guzzle clients

# In-Depth Changes

---

## PHP Library "Guzzle" (2)

- The RequestFactory class can be used like this:

```
// Initiate RequestFactory

/** @var \TYPO3\CMS\Core\Http\RequestFactory $requestFactory */
$requestFactory = GeneralUtility::makeInstance(
    \TYPO3\CMS\Core\Http\RequestFactory\RequestFactory::class);

$suri = $additionalOptions = [
    // additional headers for this specific request
    'headers' => ['Cache-Control' => 'no-cache'],
    'allow_redirects' => false,
    'cookies' => true
];

// return a PSR-7 compliant response object
$response = $requestFactory->request($url, 'GET', $additionalOptions);

// get the content as a string on a successful request
if ($response->getStatusCode() === 200) {
    if ($response->getHeader('Content-Type') === 'text/html') {
        $content = $response->getBody()->getContents();
    }
}
```

# In-Depth Changes

---

## Add Linkservice for Unified Referencing Syntax (1)

- Resources within TYPO3 have been referenced using multiple, different forms of syntax in the past.
- TYPO3 now supports a modern and future-proof way of referencing resources using an extensible and expressive syntax which is easy to understand.
- The next slides explain the syntax using the following simple page link:  
`t3://page?uid=13&campaignCode=ABC123`

# In-Depth Changes

---

## Add Linkservice for Unified Referencing Syntax (2)

- The syntax consists of three parts:
  - Namespace (`t3://`)

The namespace is fixed to `t3://` to ensure the "LinkService" is executed to parse the URN.
  - Resource handler key (`page`)

The resource handler key identifies the resource in the TYPO3 available handlers list. At the time of writing the following handlers exist: `page`, `file` and `folder`.  
More keys can be configured in an associative array, where the key is the handler key and the value is a class implementing the `LinkHandlerInterface`:

```
$TYPO3_CONF_VARS['SYS']['linkHandler']
```

# In-Depth Changes

---

## Add Linkservice for Unified Referencing Syntax (3)

- ...and the 3rd part:
  - Resource parameters (?uid=13&campaignCode=ABC123)  
These are the specific identification parameters that are used by any handler. Note that these may carry additional parameters in order to configure the behavior of any handler.

# In-Depth Changes

---

## `DebuggerUtility::var_dump` (1)

- The property visibility information has been added to `DebuggerUtility::var_dump()` for each object property in the dump
- If a closure is part of the debugging object, the source code of the closure is rendered, too

*See example on the next slide*

# In-Depth Changes

---

## DebuggerUtility::var\_dump (2)

Extbase Variable Dump

```
TYPO3\CMS\Core\Database\DatabaseConnection prototype object
debugOutput => public 0 (integer)
debug_lastBuiltQuery => public '' (0 chars)
store_lastBuiltQuery => public FALSE
explainOutput => public 0 (integer)
databaseHost => protected 'localhost' (9 chars)
databasePort => protected 3306 (integer)
databaseSocket => protected '' (0 chars)
databaseName => protected 'typo383' (7 chars)
databaseUsername => protected 'root' (4 chars)
databaseUserPassword => protected 'secure7b1' (9 chars)
persistentDatabaseConnection => protected FALSE
connectionCompression => protected FALSE
connectionCharset => protected 'utf8' (4 chars)
initializeCommandsAfterConnect => protected array (empty)
isConnected => protected FALSE
link => protected NULL
default_charset => public 'utf8' (4 chars)
preProcessHookObjects => protected array (empty)
postProcessHookObjects => protected array (empty)
```

# In-Depth Changes

---

## System Status Updates (Reports)

- Results of test in the "System Status Updates (reports)" can be sent via email
- A checkbox has been added to the job configuration to:
  - send an email if the system has warnings or errors
  - always generate an email
- The default is to include warnings and errors only



# In-Depth Changes

---

## Miscellaneous (1)

- SVGs and D3 rendering
  - As part of the ExtJS removal from the TYPO3 core, the tree within the form editing has been re-worked
  - Rendering is based on SVGs and D3 now, which comes with a significant performance boost
  - Re-working the page tree the same way is planned for the near future
- Extension icons can be stored in the following directory now:  
Resources/Public/Icons/<filename> (where <filename> can be: Extension.png, Extension.svg or Extension.gif)
- The new option backendFavicon in the Extension Manager configuration makes it possible to change the favicon of the backend.

# In-Depth Changes

---

## Miscellaneous (2)

- All system information added by `addSystemInformation()` have `InformationStatus::STATUS_NOTICE` as the default value now
- Enumeration constants can be retrieved easily now:
  - `EnumerationClass::getName($value)`;
  - `EnumerationClass::getHumanReadableName($value)`;
- Priorities of core `TypeConverters` have changed from 1, 2, 3,... to 10, 20, 30,... When registering custom `TypeConverter(s)`, make sure they are using the correct priorities.
- [ISO-8601](#) is used to pass date and datetime values between server and client now. Check if your custom `FormEngine` render types need to be updated (`eval=date/datetime`).

## Extbase & Fluid

# Extbase & Fluid

---

## Standalone Fluid

- The Fluid rendering engine of TYPO3 CMS is replaced by the standalone capable Fluid which is now included as a composer dependency
- The old Fluid extension is converted to a so-called *Fluid adapter* which allows TYPO3 CMS to use standalone Fluid
- New features/capabilities have been added in nearly all areas of Fluid (see [docs.typo3.org](https://docs.typo3.org) for further details)
- Most importantly: several of the Fluid components which were completely internal and impossible to replace in the past, are now easy to replace and have been fitted with a public API
- Standalone Fluid also improves backend performance

# Extbase & Fluid

---

## Rendering Context (1)

- The most important new piece of public API is the RenderingContext
- The previously internal-only RenderingContext used by Fluid has been expanded to be responsible for a vital new Fluid feature:  
**implementation provisioning**
- This enables developers to change a range of classes, Fluid uses for parsing, resolving, caching etc.
- This can be achieved by either including a custom RenderingContext or manipulating the default RenderingContext by public methods.

## Rendering Context (2)

- The features on the following slides can be controlled by manipulating the RenderingContext. By default, none of them are enabled - but calling a simple method (via your View instance) allows you to enable them:

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

# Extbase & Fluid

---

## ExpressionNodes (1)

- ExpressionNodes are a new type of Fluid syntax structures which all share a common trait: they only work inside the curly braces

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

- Developers can add their own additional ExpressionNode types
- Each one consists of a pattern to be matched and methods dictated by an interface to process the matches
- Any existing ExpressionNode type can be used as reference

# Extbase & Fluid

---

## ExpressionNodes (2)

ExpressionNodeTypes allow new syntaxes such as:

- **CastingExpressionNode**

allows casting a variable to certain types, for example to guarantee an integer or a boolean. It is used simply with an `as` keyword:

`{myStringVariable as boolean}` or `{myBooleanVariable as integer}` and so on. Attempting to cast a variable to an incompatible type causes a standard Fluid error.

- **MathExpressionNode**

allows basic mathematical operations on variables, for example `{myNumber + 1}`, `{myPercent / 100}` or `{myNumber * 100}` and so on. An impossible expression returns an empty output.



# Extbase & Fluid

---

## ExpressionNodes (3)

ExpressionNodeTypes allow new syntaxes such as:

- **TernaryExpressionNode**

allows an inline ternary condition which only operates on variables. Typical use case is: "if this variable then use that variable else use another variable".

It is used as:

```
{myToggleVariable ? myThenVariable : myElseVariable}
```

Note: does not support any nested expressions, inline ViewHelper syntaxes or similar inside it. It must be used only with standard variables as input.

# Extbase & Fluid

---

## Namespaces are extensible (1)

- Fluid allows each namespace alias (for example `f :`) to be extended by adding an additional PHP namespace to it
- PHP namespaces are also checked for the presence of ViewHelper classes
- This also means that developers can override individual ViewHelpers with custom versions and have their ViewHelpers called when the `f :` namespace is used
- This change also implies that namespaces are no longer monadic. When using `{namespace f=My\Extension\ViewHelpers\}` you will no longer receive an "namespace already registered" error. Fluid will add this PHP namespace instead and look for ViewHelpers there as well.

# Extbase & Fluid

---

## Namespaces are extensible (2)

- Additional namespaces are checked from the bottom up, allowing the additional namespaces to override ViewHelper classes by placing them in the same scope
- For example: `f:format.nl2br` can be overridden by `My\Extension\ViewHelpers\Format\Nl2brViewHelper`, given the namespace registration on previous slide

# Extbase & Fluid

---

## Complex conditional statements

- Fluid now supports any degree of complex conditional statements with nesting and grouping:

```
<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>
```

- In addition, `f:else` has been fitted with an "elseif"-like behavior:

```
<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

---

## Dynamic variable name parts (1)

- Another new feature, likewise backwards compatible, is the added ability to use sub-variable references when accessing your variables. Consider the following Fluid template variables array:

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

- With the following Fluid template:

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

## Dynamic variable name parts (2)

- The same approach can also be used to generate dynamic parts of a string variable name:

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

- With the following Fluid template:

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

# Extbase & Fluid

---

## New ViewHelpers

- A few new ViewHelpers have been added as part of standalone Fluid and as such are also available in TYPO3 from now on:

- **f:or**

This is a shorter way of writing (chained) conditions. It supports the following syntax, which checks each variable and outputs the first one that is not empty:

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

- **f:spaceless**

This can be used in tag-mode around template code to eliminate redundant whitespace and blank lines for example caused by indenting ViewHelper usages

## Add IconForRecordViewHelper

- A new ViewHelper to render icons for records has been added

```
<core:iconForRecord table="sys_template" row="{templateRecord}" ></core:iconForRecord>
```

```
// output:
```

```
<span class="t3js-icon icon icon-size-small icon-state-default icon-mimetypes-x-content-template" data-identifier="mimetypes-x-content-template">  
  <span class="icon-markup">  
      
  </span>  
</span>
```



# Extbase & Fluid

---

## Content for ViewHelper `f:form.select`

- Introduced two new ViewHelpers allowing the manual definition of all options and optgroups for the `f:form.select` as tag content of the select field
  - `OptionViewHelper`
  - `OptgroupViewHelper`

- Example:

```
<f:form.select name="myproperty">
  <f:form.select.option value="1">Option one</f:form.select.option>
  <f:form.select.option value="2">Option two</f:form.select.option>
  <f:form.select.optgroup>
    <f:form.select.option value="3">Grouped option one</f:form.select.option>
    <f:form.select.option value="4">Grouped option twi</f:form.select.option>
  </f:form.select.optgroup>
</f:form.select>
```

# Extbase & Fluid

---

## Global Fluid ViewHelper Namespace

- Global Fluid ViewHelper namespaces are configurable now:  
`$GLOBALS['TYPO3_CONF_VARS']['SYS']['fluid']['namespaces']`
- This allows the namespaces to be manipulated as part of the site configuration
- Benefits:
  - Third party ViewHelper packages can manipulate the global Fluid namespace `f` :
  - Third party ViewHelper packages are able to register new global namespaces as required
  - Template developers can use such global namespaces without importing them first and can use them in all Fluid templates regardless of context

# Extbase & Fluid

---

## ViewHelper namespaces can be extended also from PHP

- By accessing the ViewHelperResolver of the RenderingContext, developers can change the ViewHelper namespace inclusions on a global (read: per View instance) basis:

```
$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

---

## FlashMessageViewHelper

- The FlashMessageViewHelper has been refactored and no longer inherits from the TagBasedViewHelper
- Remove the tag specific attributes and style the default output. If you need custom output use the possibility to render FlashMessages yourself, for example:

```
<f:flashMessages as="flashMessages">
  <dl class="messages">
    <f:for each="{flashMessages}" as="flashMessage">
      <dt>CODE!! {flashMessage.code}</dt>
      <dd>MESSAGE:: {flashMessage.message}</dd>
    </f:for>
  </dl>
</f:flashMessages>
```

# Extbase & Fluid

---

## ViewHelpers can accept arbitrary arguments (1)

- This feature allows your ViewHelper class to receive any number of additional arguments using any names you desire
- It works by separating the arguments that are passed to each ViewHelper into two groups: those that are declared using `registerArgument` (or render method arguments), and those that are not
- Those that are not declared, are passed to a special function `handleAdditionalArguments` on the ViewHelper class, which in the default implementation throws an error if additional arguments exist

# Extbase & Fluid

---

## ViewHelpers can accept arbitrary arguments (2)

- By overriding this method in your ViewHelper, you can change if and when the ViewHelper should throw an error on receiving unregistered arguments
- This feature is also the one allowing TagBasedViewHelpers to freely accept arbitrary data- prefixed arguments without failing
- on TagBasedViewHelpers, the `handleAdditionalArguments` method simply adds new attributes to the tag that gets generated and throws an error if any additional arguments which are neither registered nor prefixed with `data-` are given.

# Extbase & Fluid

---

## Argument "allowedTags" for `f:format.stripTags`

- The argument `allowedTags` containing a list of HTML tags which will not be stripped can now be used on `f:format.stripTags`
- Tag list syntax is identical to second parameter of PHP function `strip_tags` (see: [http://php.net/strip\\_tags](http://php.net/strip_tags))

# Extbase & Fluid

---

## Default Content Element Changed for Fluid Styled Content

- The default content element has been streamlined with CSS Styled Content and has been changed to "Text"
- To restore the configuration you need to set the default content element manually to your preferred choice. You can do this by simply overriding the configuration again in your `Configuration/TCA/Overrides/tt_content.php` file.

```
$GLOBALS['TCA']['tt_content']['columns']['CType']['config']['default'] = 'textmedia';  
$GLOBALS['TCA']['tt_content']['columns']['CType']['config']['default'] = 'header';
```



# Extbase & Fluid

---

## Fluid Debugging Using the Admin Panel

- Admin Panel features a new setting to debug Fluid output:  
**Preview -> Show fluid debug output**
- If enabled, the following details are shown in the frontend:
  - path to the template file of a partial
  - name of a section
- This feature enables integrators to easily locate the correct template and section

# Deprecated/Removed Functions

# Deprecated/Removed Functions

---

## Miscellaneous

- The following configuration options have been removed:
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_utils']`
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_convMethod']`

(functionality is now auto-detected and `mbstring` is used by default if available)

- The deprecated TypoScript property `page.includeJSlibs` has been removed. Use the TypoScript property `page.includeJSLibs` (capital "L") instead
- The TypoScript option `config.renderCharset`, which was used as character set for internal conversion within a frontend request, has been removed

# Deprecated/Removed Functions

---

## Http-related Options and HttpRequest Class Removed (1)

- The following PHP classes have been **removed**:
  - TYPO3\CMS\Core\Http\HttpRequest
  - TYPO3\CMS\Core\Http\Observer\Download
- The following options have been **renamed**:
  - old: `$TYPO3_CONF_VARS[HTTP][userAgent]`  
new: `$TYPO3_CONF_VARS[HTTP][headers][User-Agent]`
  - old: `$TYPO3_CONF_VARS[HTTP][protocol_version]`  
new: `$TYPO3_CONF_VARS[HTTP][version]`

# Deprecated/Removed Functions

---

## Http-related Options and HttpRequest Class Removed (2)

- All proxy-related options are unified within `$TYPO3_CONF_VARS[HTTP][proxy]`
- All redirect-related options (`HTTP/follow_redirects`, `HTTP/max_redirects`, `HTTP/strict_redirects`) are unified within `$TYPO3_CONF_VARS[HTTP][allow_redirects]`
- All options related to SSL private keys (`HTTP/ssl_local_cert`, `HTTP/ssl_passphrase`) are merged into `$TYPO3_CONF_VARS[HTTP][ssl_key]`
- All options related to verify SSL peers are merged into `$TYPO3_CONF_VARS[HTTP][verify]`

# Deprecated/Removed Functions

---

## ExtJS Removal (1)

- As part of the ExtJS removal work package, the following JavaScript methods have been removed from the Backend main frame (defined in file `backend.js`):
  - `TYP03._instances`
  - `TYP03.addInstance`
  - `TYP03.getInstance`
  - `TYP03.helpers.split`

# Deprecated/Removed Functions

---

## ExtJS Removal (2)

- New class `TYPO3\CMS\Workspaces\Controller\AjaxDispatcher` replaces the `ExtDirect` router functionality in `EXT:workspaces`
- The following classes have been moved:
  - `Classes/ExtDirect/AbstractHandler.php`  
now as: `Classes/Controller/Remote/AbstractHandler.php`
  - `Classes/ExtDirect/ActionHandler.php`  
now as: `Classes/Controller/Remote/ActionHandler.php`
  - `Classes/ExtDirect/MassActionHandler.php`  
now as: `Classes/Controller/Remote/MassActionHandler.php`
  - `Classes/ExtDirect/ExtDirectServer.php`  
now as: `Classes/Controller/Remote/RemoteServer.php`

# Deprecated/Removed Functions

---

## Classes `DatabaseConnection` and `PreparedStatement`

- The following classes have been marked as *deprecated*:
  - `TYPO3\CMS\Core\Database\DatabaseConnection`
  - `TYPO3\CMS\Core\Database\PreparedStatement`
- Use Doctrine DBAL in TYPO3 v8 LTS instead (`ConnectionPool` and `QueryBuilder` classes)
- These two classes will be removed in TYPO3 v9



# Deprecated/Removed Functions

---

## JavaScript settings under `TYPO3.configuration`

- The following JavaScript settings have been removed:
  - `TYPO3.configuration.debugInWindow`
  - `TYPO3.configuration.moduleMenuWidth`
  - `TYPO3.configuration.topBarHeight`
- These options were not used by the TYPO3 core anyway

# Deprecated/Removed Functions

---

## Public Properties of FlexFormTools

- Two public properties have been dropped from class `TYPO3\CMS\Core\Configuration\FlexForm\FlexFormTools`:
  - `public $traverseFlexFormXMLData_DS = array();`
  - `public $traverseFlexFormXMLData_Data = array();`
- Accessing those properties will throw a warning now

# Deprecated/Removed Functions

---

## Frameset and frame

- frameset and frame are not supported in HTML5 anymore
- The following TYPOScript objects have been marked as *deprecated*:
  - frameset
  - frame
- The following TYPOScript options have been marked as *deprecated*:
  - `config.frameReloadIfNotInFrameset`
  - `config.doctype = xhtml_frames`
  - `config.xhtmlDoctype = xhtml_frames`
  - `frameSet` (and its options)
  - `FRAME` (and its options)
  - `FRAMESET` (and its options)

# Deprecated/Removed Functions

---

## Extbase Query Cache Removed

- The PHP-based query cache functionality within the Extbase persistence layer has been removed
- The following public methods within the Extbase persistence layer have been removed:
  - `Typo3DbBackend->quoteTextValueCallback()`
  - `Typo3DbBackend->injectCacheManager()`
  - Interface definition in `QuerySettingsInterface->getUseQueryCache`
- The according cache configuration has no effect anymore:  
`$TYPO3_CONF_VARS[SYS][cache][cacheConfigurations][extbase_typo3dbbackend_queries]`

# Deprecated/Removed Functions

---

## Extbase: Prepared Statement Query Option

- The option to use prepared statements within the Extbase persistence has been removed
- The following methods have been removed from the `QuerySettingsInterface`, as the database abstraction layer will take care of prepared statements automatically:
  - `getUsePreparedStatement()`
  - `usePreparedStatement()`

# Deprecated/Removed Functions

---

## Miscellaneous (3)

- The following TCA option has been removed:  
`$TCA[$table][ctrl][versioning_followPages]`
- Adding items to TCA tree with `pageTsConfig addItem` requires an icon identifier from the icon registry now (paths are not supported anymore):  
`TCEFORM.pages.category.addItem.12345.icon = my-registered-icon`
- All XLIFF Language files of `EXT:lang` have been moved to `Resources/Private/Language/`  
This affects all extensions which use labels from `EXT:lang`!  
OLD: `EXT:lang/locallang_alt_doc.xlf`  
NEW: `EXT:lang/Resources/Private/Language/locallang_alt_doc.xlf`

# Deprecated/Removed Functions

---

## TCA in `ext_tables.php`

- Frontend requests no longer load `ext_tables.php` in requests
- This change impacts extensions which configure the TCA in `ext_tables.php` (which is not allowed anyway)
- Install Tool provides a test "TCA `ext_tables` check" to identify such extensions

### TCA in `ext_tables.php` check

Check if an extension changes `$GLOBALS['TCA']` in `ext_tables.php`.

No TCA changes in `ext_tables.php` files. Good job!

Check loaded extensions

# Deprecated/Removed Functions

---

## Removal of Fluid Styled Content Menu ViewHelpers (1/3)

- Fetching data directly in the view is not recommended and the temporary solution of menu ViewHelpers has been replaced by its successor, the menu processor that is based on HMENU.
- Menu ViewHelpers have been moved to the `compatibility7` extension, and are replaced in the core menu content elements.



## Installation and Upgrade

*It's time to check out TYPO3 v8 LTS*

# Installation and Upgrade

---

## Traditional Installation Method

- Official *traditional* installation procedure under Linux/Mac OS X (DocumentRoot for example `/var/www/site/htdocs`):

```
$ cd /var/www/site/  
$ wget --content-disposition get.typo3.org/8.7  
$ tar xzf typo3_src-8.7.0.tar.gz  
$ cd htdocs  
$ ln -s ../typo3_src-8.7.0 typo3_src  
$ ln -s typo3_src/index.php  
$ ln -s typo3_src/typo3  
$ touch FIRST_INSTALL
```

- Symbolic links under Microsoft Windows:
  - Use `junction` under Windows XP/2000
  - Use `mklink` under Windows Vista and Windows 7

# Installation and Upgrade

---

## Installation Using `composer`

- Installation using `composer` under Linux/Mac OS X

```
$ cd /var/www/site/  
$ composer create-project typo3/cms-base-distribution
```

- Alternatively, create your custom `composer.json` file and run:

```
$ composer install
```

An example `composer.json` file can be downloaded at:

[git.typo3.org/TYPO3CMS/Distributions/Base.git/blob/HEAD:/composer.json](https://git.typo3.org/TYPO3CMS/Distributions/Base.git/blob/HEAD:/composer.json)

# Installation and Upgrade

---

## Upgrade to TYPO3 v8 LTS

- Upgrades only possible from TYPO3 v7 LTS
- TYPO3 CMS < 7.6 LTS should be updated to TYPO3 version 7.6 first
- Upgrade instructions:  
[http://wiki.typo3.org/Upgrade#Upgrading\\_to\\_8.7](http://wiki.typo3.org/Upgrade#Upgrading_to_8.7)
- Official TYPO3 guide "TYPO3 Installation and Upgrading":  
<http://docs.typo3.org/typo3cms/InstallationGuide>
- General approach:
  - Check minimum system requirements (PHP, MySQL, etc.)
  - Review **deprecation\_\*.log** in old TYPO3 instance
  - Update all extensions to the latest version
  - Deploy new sources and run Install Tool -> Upgrade Wizard
  - Review startup module for backend users (optionally)

## Sources and Authors

# Sources and Authors

---

## Sources

### TYPO3 News:

- <http://typo3.org/news>

### Release Infos:

- [http://wiki.typo3.org/TYPO3\CMS\\_8.7.0](http://wiki.typo3.org/TYPO3\CMS_8.7.0)
- [INSTALL.md](#) and [Changelog](#)
- `typo3/sysexst/core/Documentation/Changelog/8.7/*`

### TYPO3 Bug-/Issuetracker:

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

### TYPO3 and Fluid Git Repositories:

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

# Sources and Authors

---

## TYPO3 CMS What's New Team:

Pierrick Caillon, Sergio Catala, Richard Haeser, Jigal van Hemert,  
Patrick Lobacher, Michel Mix, Sinisa Mitrovic,  
Nena Jelena Radovic, Michael Schams and Roberto Torresani

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

Licensed under Creative Commons BY-NC-SA 3.0

