TYPO3 CMS 8.0 – What's New

Résumé des nouvelles caractéristiques, changements et améliorations

Créé par :
Patrick Lobacher et Michael Schams

Traduction par :
Pierrick Caillon

07/June/2016

TYPO3

TYPO3 CMS 8.0 - What's New

Sommaire

Introduction

Faits

Introduction

TYPO3 CMS 8.0 – Faits

- Date de sortie : 22 Mars 2016
- Type de sortie : Sprint Release
- Slogan : Start your engines

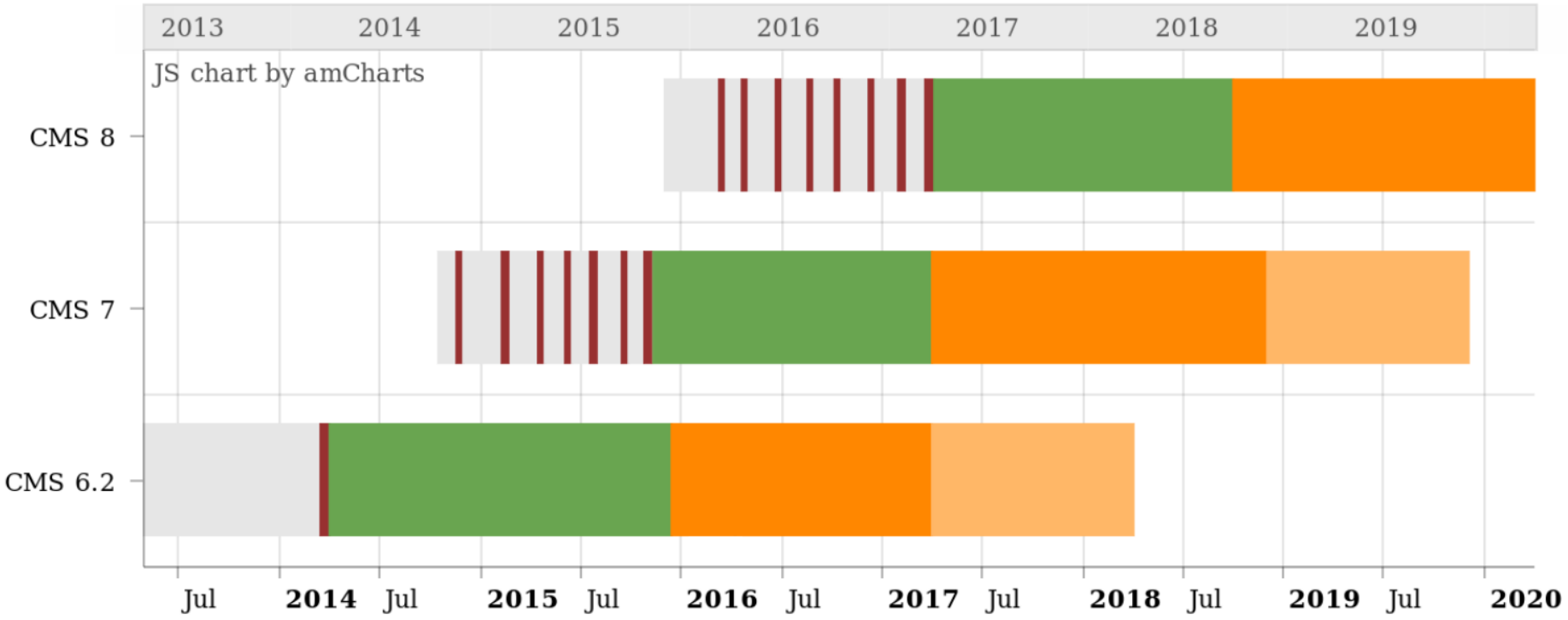
Introduction

Prérequis système

- PHP : version 7
- MySQL : version 5.5 à 5.7
- Espace disque : min. 200 Mo
- Configuration PHP :
  - `memory_limit` >= 128M
  - `max_execution_time` >= 240s
  - `max_input_vars` >= 1500
  - L'option de compilation `--disable-ipv6` NE doit PAS être utilisée
- Le backend nécessite Microsoft Internet Explorer 11 ou ultérieur, Microsoft Edge, Google Chrome, Firefox, Safari ou tout autre navigateur moderne compatible

Introduction

Chronologie des développements et sorties



Introduction

Feuille de route TYPO3 CMS

Dates de sortie et axes principaux :

- v8.0 22/Mars/2016 Adding last minute things
- v8.1 03/Mai /2016 Cloud Integration
- v8.2 05/Jui./2016 Rich Text Editor
- v8.3 30/Août/2016 Frontend Editing on Steroids
- v8.4 18/Oct./2016 *to be determined*
- v8.5 20/Déc./2016 Integrator Support
- v8.6 14/Fev./2017 *to be determined*
- v8.7 04/Avr./2017 LTS Preparation

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

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

Introduction

Installation

- Procédure officielle d'installation sous Linux/Mac OS X (DocumentRoot considéré `/var/www/site/htdocs`) :

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

- Liens symboliques sous Microsoft Windows :
  - Utiliser `junction` sous Windows XP/2000
  - Utiliser `mklink` sous Windows Vista et Windows 7

Introduction

Mise à jour vers TYPO3 CMS 8.x

- Les mises à jour sont possibles seulement depuis TYPO3 CMS 7.6 LTS
- TYPO3 CMS < 7.6 LTS doivent être mis à jour vers la 7.6 LTS en premier
- Instructions de mise à jour :

  http://wiki.typo3.org/Upgrade#Upgrading_to_8.0
- Guide TYPO3 officiel « TYPO3 Installation and Upgrading » :

  http://docs.typo3.org/typo3cms/InstallationGuide
- De manière générale :
  - Vérifier les prérequis système (PHP, MySQL, etc.)
  - Examiner **deprecation_*.log** de l'ancienne instance TYPO3
  - Mettre à jour toutes les extensions vers leurs dernières versions
  - Déployer les nouvelles sources et exécuter l'assistant de mise à jour de l'Install Tool
  - Examiner le module de démarrage des utilisateurs backend (optionnel)

Introduction

PHP Version 7

- PHP 7.0 est le prérequis minimum pour TYPO3 CMS 8.x
- TYPO3 supportera les sorties de PHP 7 au fur et à mesure
- Cette montée de version apporte une amélioration significative des performances de l'ensemble du système
- Non seulement les éditeurs backend remarquerons une interface plus fluide, mais le nouveau record de chargement d'une page entièrement en cache en frontend est sous les 7 millisecondes, approximativement 40% plus rapide que le même site avec PHP version 5.5
- Nous avons aussi commencé à utiliser les nouvelles fonctionnalités de cette version, par exemple les générateurs pseudo-aléatoires sécurisés cryptographiquement sont déjà utilisés. (Cryptographically secure pseudorandom number generator ; CSPRNG)

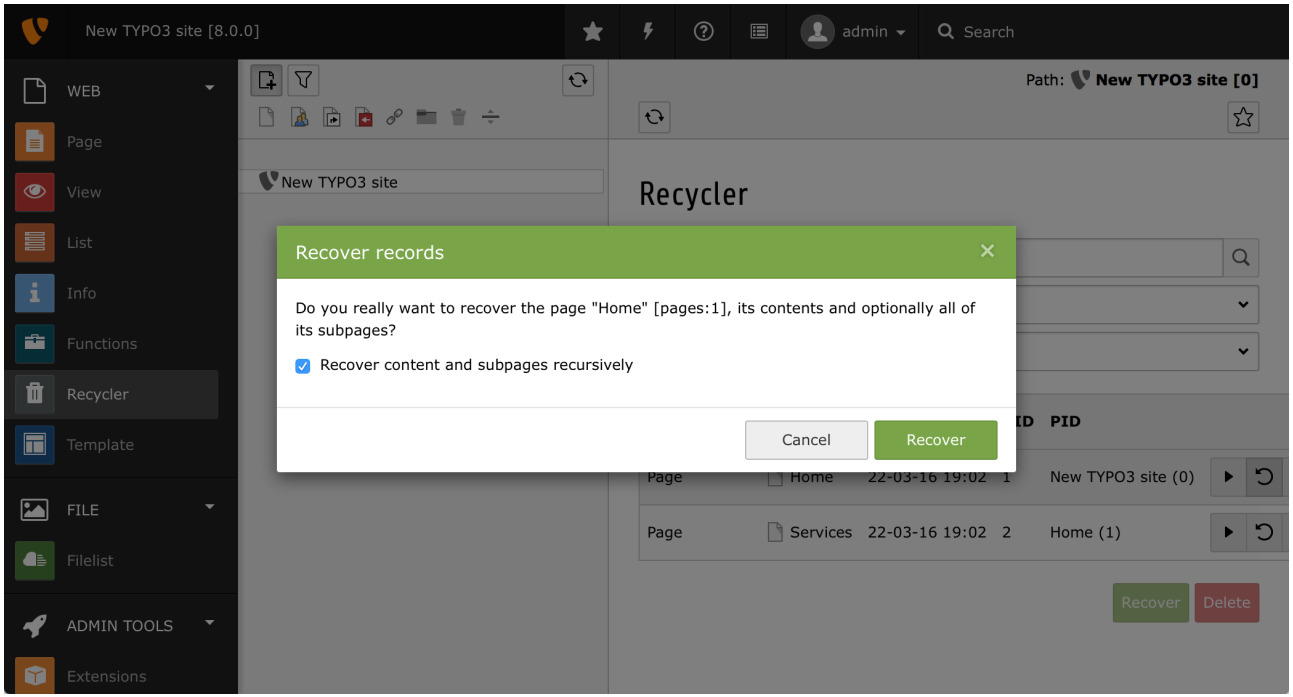
Chapitre 1 :

Interface Utilisateur Backend

Interface Utilisateur Backend

Restaurer les pages récursivement à la racine

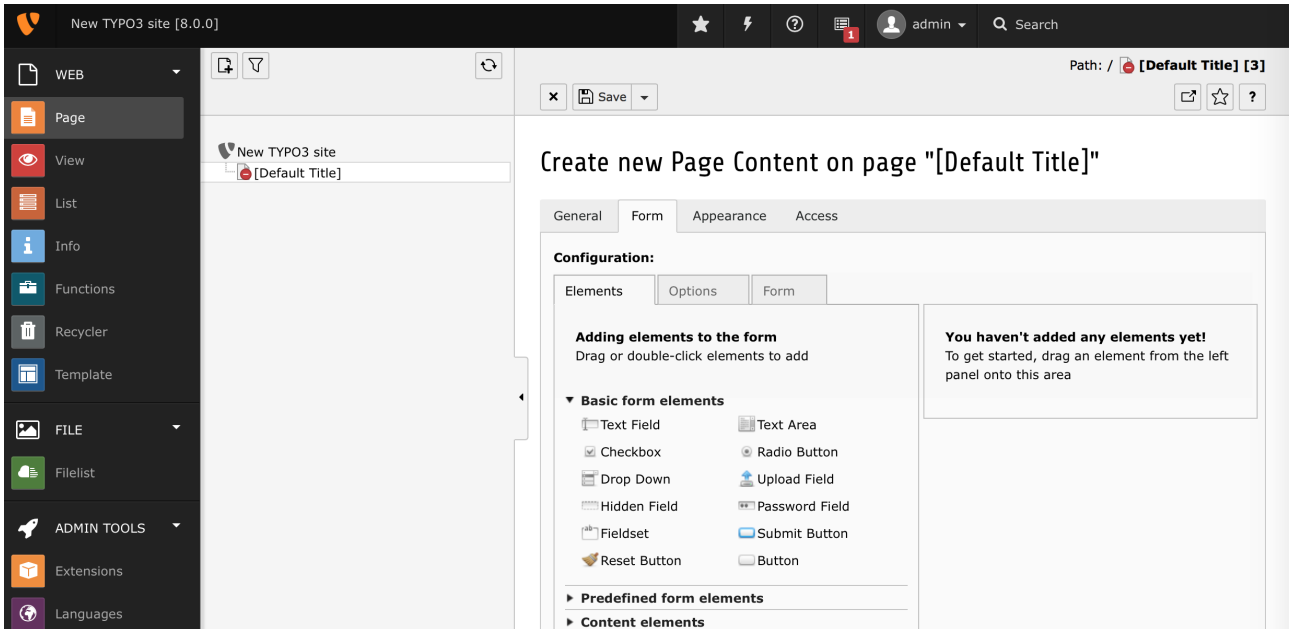
La Corbeille supporte la restauration récursive des pages supprimées à la racine. Cette fonctionnalité n'est disponible que pour les administrateurs en raison des permissions requises.



Interface Utilisateur Backend

Chargement sur place de l'assistant formulaire

L'assistant de EXT:form est chargé directement en tant qu'assistant sur place. Il n'y a plus besoin d'enregistrer et recharger le nouvel élément de contenu afin d'ouvrir l'assistant. C'est une importante amélioration de l'usabilité.



Interface Utilisateur Backend

Définir un logo Backend alternatif via le gestionnaire d'extensions

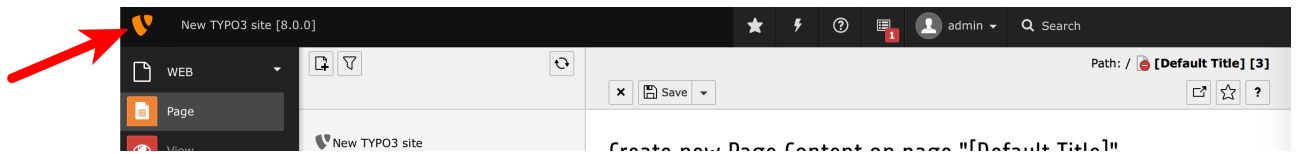
Le logo du Backend dans le coin haut gauche est configurable dans la configuration de l'extension backend dans le gestionnaire d'extensions.
Les options de configuration sont :

- chemin relatif à l'instance de TYPO3 vers une ressource

  ex. `"fileadmin/images/my-background.jpg"`
- chemin vers une ressource d'extension

  ex. `"EXT:my_theme/Resources/Public/Images/my-background.jpg"`
- adresse d'une ressource externe

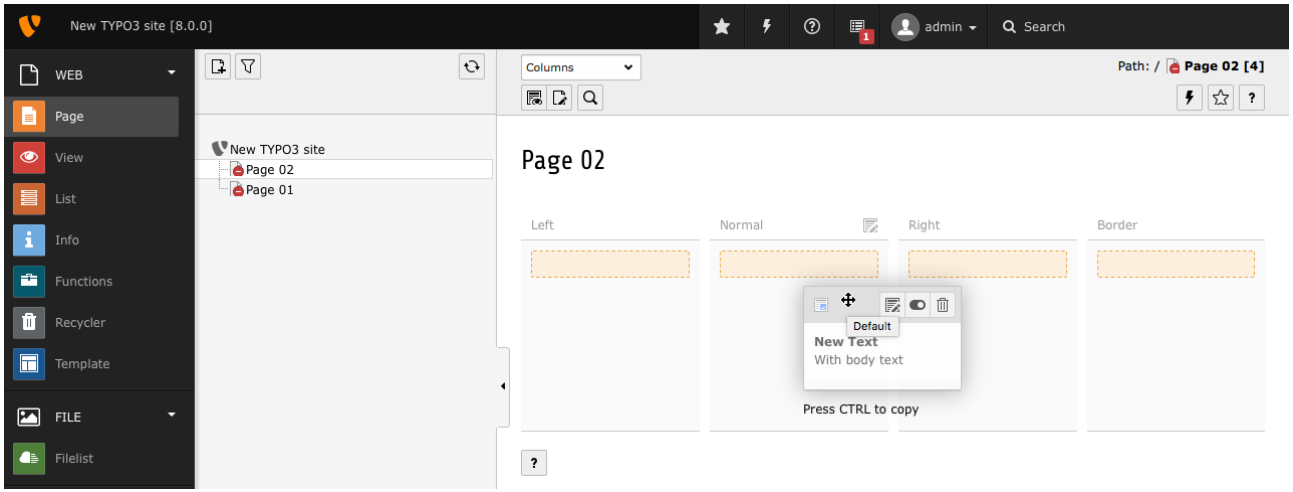
  ex. `"//example.com/my-background.png"`



Interface Utilisateur Backend

Pages : copie d'élément en glisser-déposer

En plus de la fonctionnalité habituelle de glisser-déposer dans le module page (qui *déplace* les éléments de contenu), des copies peuvent être créées : maintenir la touche CTRL lors du dépôt pour créer une copie de l'élément attrapé. À la fin de l'opération, le module page se recharge pour créer le nouvel élément avec toutes les informations nécessaires.



Chapitre 2 :

TSconfig & TypoScript

TSconfig & TypoScript

Ordre de tri des onglets de l'assistant nouveau contenu

- Il est possible de configurer l'ordre des onglets dans l'assistant nouvel élément de contenu avec les options `before` et `after` dans le TSconfig de page :

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

TSconfig & TypoScript

HTMLparser.stripEmptyTags.keepTags

- De nouvelles options pour HTMLparser.stripEmptyTags sont ajoutées, permettant de garder les balises configurées
- Avant le changement, seul une liste de balises devant être retirées pouvait être fournie
- L'exemple suivant retire toutes les balises vides **à l'exception** des balises tr et td :

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

Important : si l'option est utilisée, la configuration stripEmptyTags.tags n'a plus d'effet. Vous ne pouvez utiliser que l'une des deux options à la fois.

TSconfig & TypoScript

EXT:form - intégration de formulaires prédéfinis (1)

- L'élément de contenu de EXT:form permet l'intégration de formulaires prédéfinis.
- Un intégrateur peut définir des formulaires (par ex. dans une distribution) en utilisant plugin.tx_form.predefinedForms
- Un éditeur peut ajouter un nouvel élément de contenu mailform à une page et choisir un formulaire dans la liste des éléments prédéfinis
- Les intégrateurs peuvent construire leur formulaire en TypoScript, fournissant plus d'options qu'avec l'assistant formulaire (par ex. la fonctionnalité stdWrap, non disponible avec l'assistant (pour des raisons de sécurité))

TSconfig & TypoScript

EXT:form - intégration de formulaires prédéfinis (2)

- Les éditeurs n'ont plus besoin d'utiliser l'assistant formulaire. Ils peuvent utiliser les formulaires prédéfinis qui sont optimisés.
- Les formulaires peuvent être réutilisés dans toute l'instance
- Les formulaires peuvent être enregistrés en dehors de la base et versionnés
- Afin de pouvoir sélectionner un formulaire prédéfini en backend, le formulaire doit être inscrit en utilisant le PageTS :

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

TSconfig & TypoScript

EXT:form: intégration de formulaires prédéfinis (3)

- Formulaire d'exemple :

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


TYPO3

Chapitre 3 :

Changements en profondeur

Changements en profondeur

Support de PECL-memcached dans MemcachedBackend

- Le support du module PECL « memcached » est ajouté au MemcachedBackend du Caching Framework
- Si les deux extensions « memcache » et « memcached » sont installées, alors « memcached » est utilisé afin de ne pas provoquer d'anomalie.
- Un intégrateur peut définir l'option `peclModule` pour préférer utiliser le module PECL :

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

Changements en profondeur

Support natif de la Console Symfony (1)

- TYPO3 supporte directement le composant Console de Symfony en fournissant un nouveau script de ligne de commande placé dans `typo3/sysext/core/bin/typo3`. Pour les installations TYPO3 via Composer, un lien symbolique est créé dans le dossier `bin`, par ex. `bin/typo3`.
- Inscrire une commande pour qu'elle soit disponible via la commande `typo3` s'effectue par la création du fichier `Configuration/Commands.php` dans une extension installée. Il liste les classes Symfony/Console/Command à exécuter par `typo3` dans un tableau associatif. La clé est le nom de la commande à appeler, premier argument de `typo3`.

Changements en profondeur

Support natif de la Console Symfony (2)

- Le nouveau script supporte toujours les anciennes commandes lorsqu'aucune commande Symfony Console n'est trouvée pour la compatibilité
- Lors de l'inscription d'une commande, la propriété `class` est requise. La propriété optionnelle `user` est à utiliser pour qu'un utilisateur backend soit authentifié lors de l'appel.
- Un fichier `Configuration/Commands.php` peut ressembler à ceci :

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

Changements en profondeur

Support natif de la Console Symfony (3)

- En appel d'exemple peut être :

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

- Pour une installation non Composer :

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

Changements en profondeur

Cryptographically secure pseudorandom number generator

- Un nouveau « cryptographically secure pseudo-random number generator » (CSPRNG) est implémenté dans le noyau de TYPO3. Il utilise les nouvelles fonctions CSPRNG de PHP 7.
- L'API est dans la classe `\TYPO3\CMS\Core\Crypto\Random`
- Exemple :

```
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);
```

Changements en profondeur

Composant Assistant (1)

- Un nouveau composant d'assistant (wizard) est ajouté. Ce composant s'utilise pour des interactions guidées
- Le module RequireJS s'utilise en incluant `TYPO3/CMS/Backend/Wizard`
- Seul des actions simples sont supportées (pas de branchement pour le moment)
- Le composant assistant possède les méthodes publiques suivantes :

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

Changements en profondeur

Composant Assistant (2)

- L'événement `wizard-visible` est déclenché lorsque le rendu de l'assistant est terminé
- L'assistant se ferme lorsque l'événement `wizard-dismiss` est déclenché
- L'assistant déclenche l'événement `wizard-dismissed` s'il est fermé
- Vous pouvez intégrer vos propre gestionnaires en utilisant `Wizard.getComponent()`

Changements en profondeur

Déplacement des fichiers générés

- La structure des dossiers sous `typo3temp` est changée pour séparer les ressources accessibles aux clients et les fichiers créés temporairement (par ex. pour le cache ou le verrouillage et ne nécessitant que l'accès par le serveur).
- Ces éléments ont été déplacés des dossiers :
  `_processed_`, `compressor`, `GB`, `temp`, `Language`, `pics`
  et réorganisés sous :
  - `typo3temp/assets/js/`
  - `typo3temp/assets/css/`,
  - `typo3temp/assets/compressed/`
  - `typo3temp/assets/images/`

Changements en profondeur

Changements ImageMagick/GraphicsMagick (1)

- Les options des transformateurs graphique Image- ou GraphicsMagick sont renommées (fichier: `LocalConfiguration.php`).
  Ancien : `im_`
  Nouveau : `processor_`
- Les noms négatifs comme `noScaleUp` sont changés vers leur correspondance positive. Pendant la conversion, les valeurs des précédentes configurations sont inversées pour refléter le changement de sens des options.
- De plus, les références à des versions spécifiques de ImageMagick/GraphicsMagick ont été retirées des options.

Changements en profondeur

Changements ImageMagick/GraphicsMagick (2)

- L'option de configuration inutilisée `image_processing` est retirée sans remplacement
- L'option spécifique des transformateurs `colorspace` est mise en *espace de nom* sous la hiérarchie `processor_`

Changements en profondeur

Hooks et Signals (1)

- Un hook est ajouté à la méthode `BackendUtility::viewOnClick()` pour le post-traitement de l'URL de prévisualisation
- Inscription d'une classe de hook implémentant la méthode `postProcess` :

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

Changements en profondeur

Hooks et Signals (2)

- Avant TYPO3 CMS 7.6, il était possible de surcharger le recouvrement (overlay) dans `Web -> List`. Un nouveau hook dans TYPO3 CMS 8.0 fourni l'ancienne fonctionnalité.
- Le hook est appelé avec la signature suivante :

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

- Inscrire la classe de hook implémentant la méthode `postOverlayPriorityLookup` :

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

Changements en profondeur

Hooks et Signals (3)

- Un nouveau signal est implémenté avant l'initialisation d'un stockage de ressources.
- Inscrire la classe dans `ext_localconf.php` :

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

- La méthode est appelée avec les paramètres suivants :
  - int `$uid` the uid of the record
  - array `$recordData` all record data as array
  - string `$fileIdentifier` the file identifier

Changements en profondeur

Algorithme de hachage des mots de passe : PBKDF2

- L'algorithme de hachage des mots de passe « PBKDF2 » est ajouté à l'extension système « saltedpasswords »
- PBKDF2 est l'abréviation de : Password-Based Key Derivation Function 2
- L'algorithme est conçu pour être coûteux en temps machine pour résister aux tentatives de force brute

Chapitre 4 :

Extbase & Fluid

Extbase & Fluid

Fluid indépendant révisé

- Le moteur de rendu Fluid de TYPO3 CMS est remplacé par la version indépendante de Fluid inclue comme une dépendance Composer
- L'ancienne extension Fluid est convertie en *adaptateur Fluid* permettant à TYPO3 CMS d'utiliser le Fluid indépendant
- De nouvelles fonctionnalités sont ajoutés dans la plupart des domaines de Fluid
- Plus important : de nombreux composants Fluid qui étaient entièrement internes et impossible à remplacer par le passé sont facile à remplacer et sont intégrés dans une API publique

Extbase & Fluid

RenderingContext (1)

- La plus importante des nouvelles pièces de l'API publique est le RenderingContext (contexte de rendu)
- Le contexte de rendu précédemment interne seulement utilisé par Fluid est étendu pour être responsable de la nouvelle fonctionnalité Fluid : **provisionnement de l'implémentation**
- Permet aux développeurs de changer un ensemble de classes que Fluid utilise pour le traitement, la résolution, le cache, etc.
- L'utilisation s'effectue soit par l'implémentation d'un contexte de rendu personnalisé ou en manipulant celui par défaut avec les méthodes publiques.

Extbase & Fluid

RenderingContext (2)

- Les comportements suivants peuvent tous être contrôlés en manipulant le contexte de rendu. Par défaut, aucun n'est activé. Mais un seul appel (via l'instance de la vue) permet de les activer :

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

Extbase & Fluid

ExpressionNodes (1)

- Les ExpressionNodes (nœuds d'expression) sont un nouveau type de structure syntaxique de Fluid partageant une caractéristique : ils ne peuvent être utilisés qu'entre accolades

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

- Les développeurs peuvent ajouter leurs propres types d'expressions
- Chacun des types consiste en un motif à faire correspondre et des méthodes de l'interface pour traiter la correspondance
- Les nœuds d'expression existants servent de référence

Extbase & Fluid

ExpressionNodes (2)

Les nœuds d'expression permettent de nouvelles syntaxes comme :

- **CastingExpressionNode**
  Permet de convertir une variable à un type, pour par exemple garantir un entier ou un booléen. S'utilise à l'aide du mot clé `as` : `{myStringVariable as boolean}` ou `{myBooleanVariable as integer}` et ainsi de suite. Tenter de convertir une variable dans un type non compatible déclenche une erreur Fluid standard.
- **MathExpressionNode**
  Permet des opérations mathématiques basiques sur les variables, par exemple `{myNumber + 1}`, `{myPercent / 100}` ou `{myNumber * 100}` et ainsi de suite. Une expression impossible retourne une sortie vide.

Extbase & Fluid

ExpressionNodes (3)

Les nœuds d'expression permettent de nouvelles syntaxes comme :

- **TernaryExpressionNode**
  Permet une condition ternaire opérant uniquement sur les variables. Cas d'usage typique : "si cette variable alors utilise cette variable sinon une autre variable". Utilisé comme :
  `{myToggleVariable ? myThenVariable : myElseVariable}`
  Note : ne supporte pas les expressions imbriquées, les ViewHelper ou similaire à l'intérieur. S'utilise uniquement avec de simples variables.

Extbase & Fluid

Espaces de nom extensibles (1)

- Fluid permet aux alias d'espace de nom (par exemple `f:`) d'être étendu en y ajoutant un espace de nom PHP supplémentaire
- Les espaces de nom PHP sont aussi consultés pour la présence de classes ViewHelper
- Ceci implique que les développeurs peuvent surcharger un ViewHelper avec une version personnalisée et avoir leur ViewHelper appelé lorsque l'espace de nom `f:` est utilisé
- Ce changement implique que les espaces de nom ne sont plus monadiques. Lors de l'utilisation de `{namespace f=My\Extension\ViewHelpers\}` l'erreur « espace de nom déjà inscrit » n'est plus déclenchée. Fluid ajoutera cet espace de nom PHP à la place et y recherchera aussi des ViewHelper.

Extbase & Fluid

Espaces de nom extensibles (2)

- Les espaces additionnels sont consultés du bas vers le haut, permettant à ceux-ci de surcharger les classe ViewHelper en les plaçant dans la même portée
- Par exemple : `f:format.nl2br` peut être surchargé par `My\Extension\ViewHelpers\Format\Nl2brViewHelper`, avec l'inscription de l'espace de nom de la slide précédente

Extbase & Fluid

Rendu utilisant `f:render` (1)

Contenu par défaut d'un `f:render` optionnel :

- Lorsque `f:render` est utilisé et l'indicateur `optional = TRUE` est défini, le rendu d'une section manquante retourne vide.
- Au lieu de cette sortie vide, un nouvel attribut `default` (mixed) est ajouté et remplie avec une valeur par défaut de type repli.
- En alternative, le contenu de la balise s'utilise pour la valeur par défaut comme beaucoup d'autres ViewHelper avec contenu et attributs flexibles

Extbase & Fluid

Rendu utilisant `f:render` (2)

Passage du contenu de la balise `f:render` aux partial/section :

- Permet une nouvelle approche à la structuration du rendu du modèle Fluid
- Les partials et sections peuvent s'utiliser comme habillage de partie arbitraire de code de modèle
- Exemple :

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

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

Extbase & Fluid

Expressions conditionnelles complexes

- Fluid supporte tout degré de complexité d'expression conditionnelle avec imbrication et groupement :

```
<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>
```

- En plus, le comportement type « sinon si » est ajouté à `f:else` :

```
<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

Partie de nom de variable dynamique (1)

- Une nouvelle fonctionnalité forcée, aussi compatible arrière, est l'ajout de la capacité à utiliser des sous-références de variable lors de l'accès à une variable. Considérer l'initialisation de la vue suivante :

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

- Avec le modèle Fluid suivant :

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

Extbase & Fluid

Partie de nom de variable dynamique (2)

- La même approche s'utilise pour générer dynamiquement des parties d'un nom de variable :

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

- Avec le modèle Fluid suivant :

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

Extbase & Fluid

Nouveaux ViewHelper

- Quelques nouveaux ViewHelper sont ajoutés par l'utilisation du Fluid indépendant et sont donc disponibles pour TYPO3 :
  - `f:or`
    C'est une manière raccourcis d'écrire des conditions (chainées). Supporte la syntaxe suivante qui vérifie chaque variable et retourne le contenu de la première non vide :

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

  - `f:spaceless`
    S'utilise en balise autour du code de modèle pour éliminer les espaces redondants et lignes vides causés par l'usage indenté des ViewHelper

Extbase & Fluid

Extension des espaces de nom par PHP

- En accédant au ViewHelperResolver du contexte de rendu, les développeurs peuvent changer les espaces de nom des ViewHelper sur une base globale (lire : par instance de vue) :

```
$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

Arguments arbitraires de ViewHelper (1)

- Permet au ViewHelper de recevoir des arguments additionnels utilisant des noms personnalisés
- Fonctionne en séparant les arguments passés à chaque ViewHelper en deux groupes : ceux déclarés en utilisant `registerArgument` (ou les arguments de la méthode render) et ceux qui ne le sont pas
- Ces derniers sont passés à la méthode spéciale `handleAdditionalArguments` de la classe ViewHelper, qui dans l'implémentation par défaut déclenche une erreur si des arguments supplémentaires existent

Extbase & Fluid

Arguments arbitraires de ViewHelper (2)

- En surchargeant cette méthode dans un Viewhelper, il est possible de changer si et quand celui-ci devrait déclencher une erreur lors de la réception d'arguments non déclarés
- Cette fonctionnalité est celle permettant au TagBasedViewHelper d'accepter librement des arguments préfixés de `data-` sans échouer
- sur les TagBasedViewHelper, la méthode `handleAdditionalArguments` ajoute simplement les nouveaux attributs à la balise générée et déclenche une erreur si d'autres attributs ni déclarés ni préfixés de `data-` sont fournis

Extbase & Fluid

Argument « allowedTags » de f:format.stripTags

- L'argument `allowedTags` contenant la liste des balises HTML ne devant pas être retirés s'utilise sur `f:format.stripTags`
- La syntaxe de la liste est la même que pour le second paramètre de la fonction PHP `strip_tags` (voir : `http://php.net/strip_tags`)

Extbase & Fluid

Accès à ObjectStorage en tableau par Fluid

- L'alias `getArray()` de la méthode `toArray()` est créé
- Cet alias permet la récupération des éléments par la propriété virtuelle `array`
- Cette propriété est accédée par Fluid de manière transparente à l'aide de `ObjectAccess::getPropertyPath`
- Cette capacité n'est pas limité à Fluid
- Exemple : récupérer le 5ième élément (les indices commencent à zéro)

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

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

Chapitre 5 :

Fonctions dépréciées et retirées

Fonctions dépréciées et retirées

Divers

- Les options de configuration suivantes sont retirées :
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_utils']`
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_convMethod']`

  (la fonctionnalité est auto-détectée et `mbstring` est utilisé par défaut si disponible)
- La propriété TypoScript dépréciée `page.includeJSlibs` est retirée. Utiliser la propriété TypoScript `page.includeJSLibs` ("L" majuscule) à la place
- L'option TypoScript `config.renderCharset`, utilisée pour le codage de caractères pour les conversions internes d'une requête frontend, est retirée.

Chapitre 6 :

Sources et Auteurs

Sources et Auteurs

Sources

Actualités TYPO3 :

- http://typo3.org/news

Informations des sorties :

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

Suivi des anomalies TYPO3 :

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

Dépôts Git de TYPO3 et Fluid :

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

Sources et Auteurs

Équipe TYPO3 CMS What's New :

Andrey Aksenov, Pierrick Caillon, Sergio Catala, Jigal van Hemert,
Patrick Lobacher, Michel Mix, Sinisa Mitrovic, Angeliki Plati,
Nena Jelena Radovic, Michael Schams and Roberto Torresani

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