TYPO3 CMS 8.0 – Le novità

Riassunto delle funzionalità, modifiche e aggiornamenti

Creato da:
Patrick Lobacher e Michael Schams

Traduzione italiana di:
Roberto Torresani (roberto.torresani (at) typo3.org)

19/April/2016

TYPO3

TYPO3 CMS 8.0 - Le novità

Indice delle sezioni

Introduzione

I fatti in breve

Introduzione

TYPO3 CMS 8.0 – I fatti in breve

- Data di rilascio: 22 Marzo 2016
- Tipo di rilascio: Sprint Release
- Slogan: Start your engines

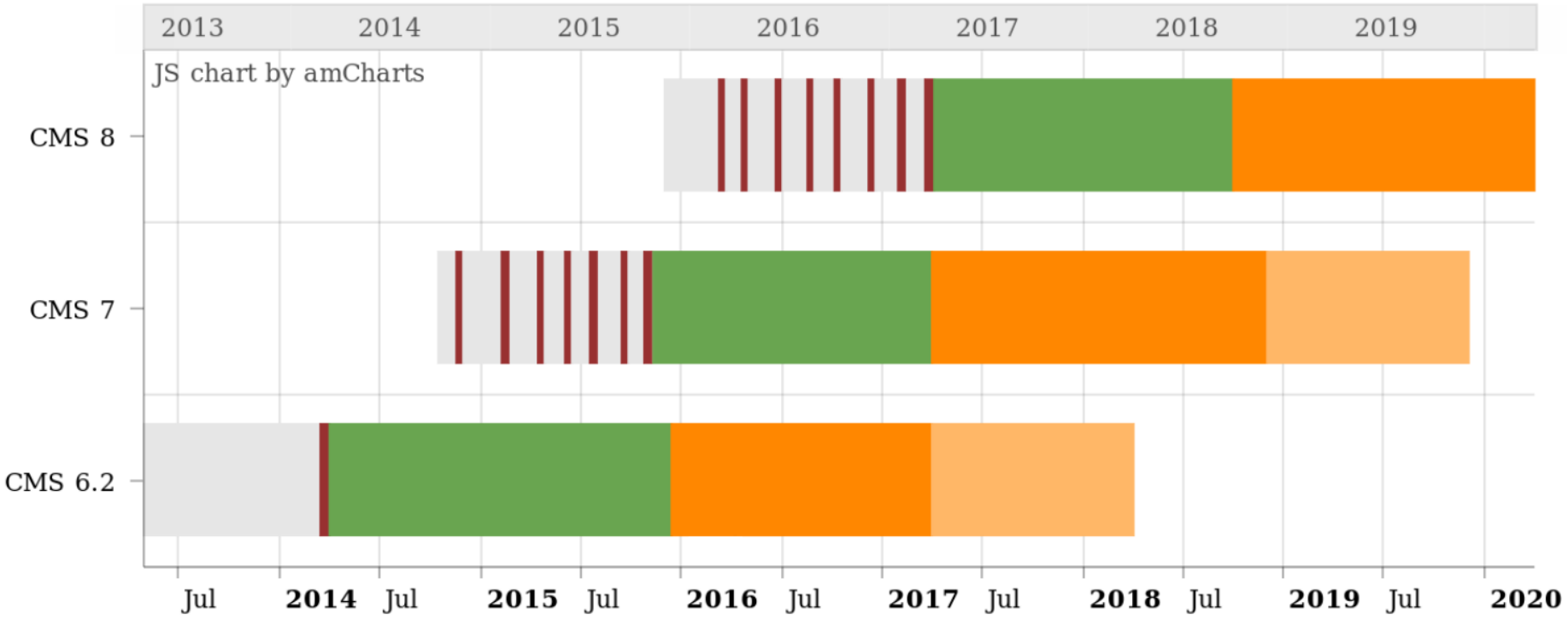
Introduzione

Requisiti di sistema

- PHP: versione 7
- MySQL: versione da 5.5 a 5.7
- Spazio disco: min 200 MB
- Impostazioni PHP:
  - `memory_limit` >= 128M
  - `max_execution_time` >= 240s
  - `max_input_vars` >= 1500
  - l'opzione di compilazione `--disable-ipv6` non deve essere usata
- Il Backend richiede Microsoft Internet Explorer 11 o superiore, Microsoft Edge, Google Chrome, Firefox, Safari o altro browser recente e compatibile

Introduzione

Sviluppo e tempi di rilascio



Introduzione

TYPO3 CMS Roadmap

Date di rilascio stimate e loro obiettivo principale:

- v8.0 22/Mar/2016 Aggiunta di parti dell'ultimo momento
- v8.1 03/Mag/2016 Integrazione cloud
- v8.2 05/Lug/2016 Rich Text Editor
- v8.3 30/Ago/2016 Miglioramento dell'Editing da Frontend
- v8.4 18/Ott/2016 *da determinare*
- v8.5 20/Dic/2016 Supporto Integrazione
- v8.6 14/Feb/2017 *da determinare*
- v8.7 04/Apr/2017 Preparazione LTS

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

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

Introduzione

Installazione

- Procedura ufficiale di installazione su Linux/Mac OS X (Directory Root ad esempio `/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
```

- Link simbolici in Microsoft Windows:
  - Usa `junction` in Windows XP/2000
  - Usa `mklink` in Windows Vista e Windows 7

Introduzione

Aggiornamento a TYPO3 CMS 8.x

- Aggiornamenti possibili solo da TYPO3 CMS 7.6 LTS
- TYPO3 CMS < 7.6 LTS deve essere prima aggiornato a TYPO3 CMS 7.6 LTS
- Istruzioni per l'aggiornamento:

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

  http://docs.typo3.org/typo3cms/InstallationGuide
- Approcio generale:
  - Verifica i requisiti minimi di sistema (PHP, MySQL, etc.)
  - Verifica **deprecation_*.log** nella vecchia istanza TYPO3
  - Aggiorna tutte le estensioni all'ultima versione
  - Imposta il nuovo sorgente ed esegui Install Tool -> Upgrade Wizard
  - Verifica il modulo di startup per gli utenti di backend (opzionale)

Introduzione

PHP Version 7

- PHP 7.0 è un requisito minimo per TYPO3 CMS 8.x
- TYPO3 supporterà i successivi rilasci di PHP 7 mano a mano che saranno pubblicati
- Questa versione fornisce un significativo incremento delle prestazioni del sistema
- Non solo gli editori di backend noteranno un interfaccia più veloce, ma il tempo di caricamento di un intera pagina di frontend in cache è inferiore a 7 millisecondi, che è circa il 40% più veloce paragonandolo allo stesso sito web con PHP versione 5.5
- Si sono iniziate ad utilizzare anche le nuove funzioni di questa versione di PHP, per esempio i generatori crittografici pseudo-casuali sono già in uso.

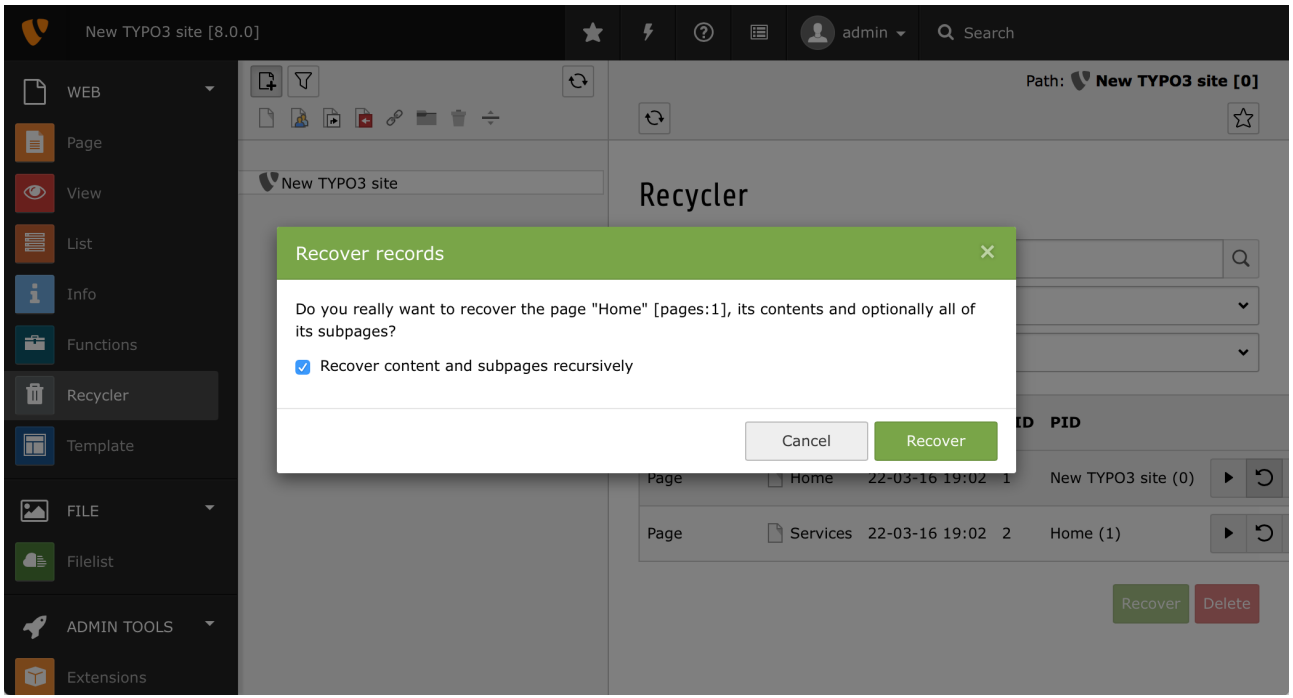
Capitolo 1:

Interfaccia utente Backend

Interfaccia utente Backend

Recupero ricorsivo delle pagine dalla rootline

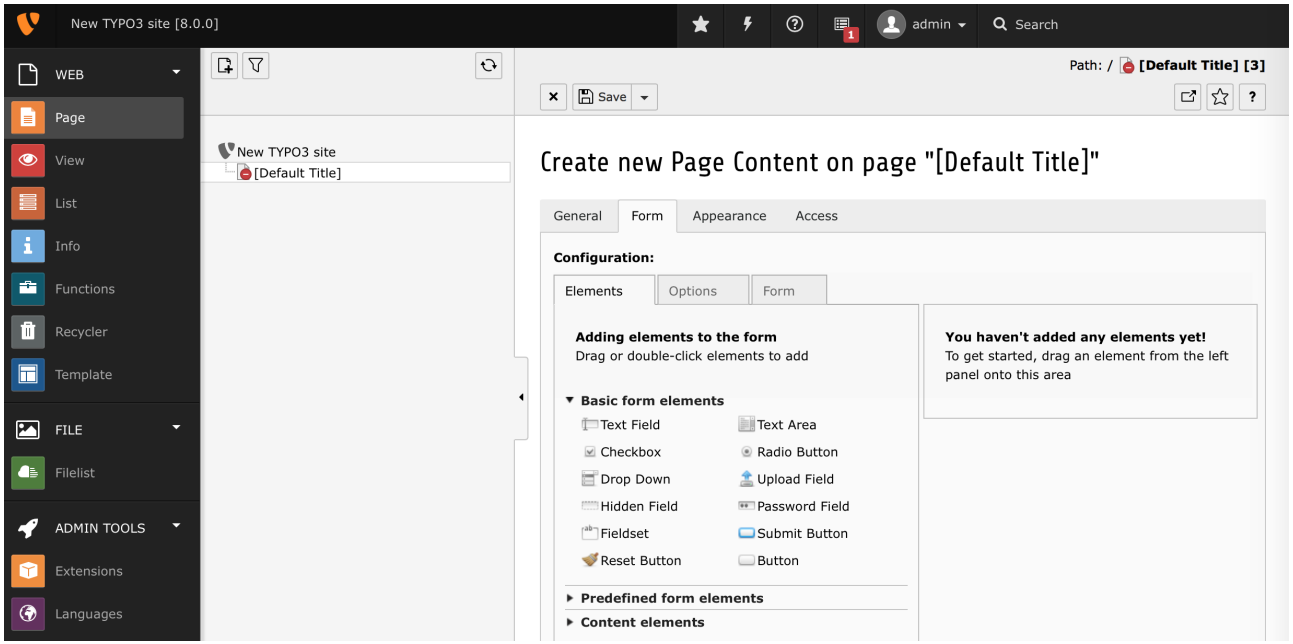
Il cestino supporta il ripristino ricorsivo delle pagine cancellate dalla rootline. Questa funzionalità è disponibile solo per gli utenti amministratori per delle restrizioni di permessi interne.



Interfaccia utente Backend

Caricamento diretto del wizard dei form come wizard in linea

Il wizard dell'EXT:form è caricato direttamente come wizard in linea. Non è più necessario salvare e ricaricare un nuovo elemento di contenuto con lo scopo di poter aprire il wizard. Si tratta di un enorme miglioramento di usabilità.



Interfaccia utente Backend

Impostare un logo di backend alternativo via Extension Manager

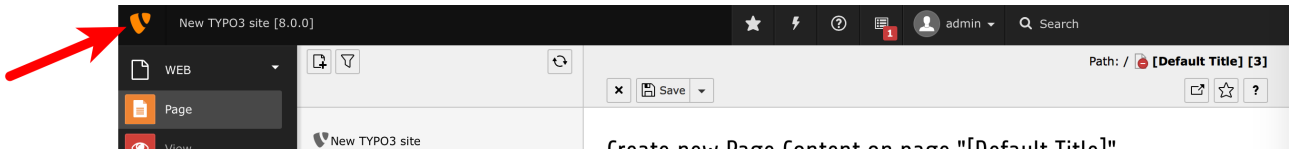
Il logo di backend, nell'angolo in alto a sinistra, può ora essere configurato nelle configurazioni dell'estensione EXT:backend nell'Extension Manager.
Le opzioni di configurazione sono:

- risorsa come percorso relativo dell'installazione TYPO3

  es. `"fileadmin/images/my-background.jpg"`
- percorso ad un estensione

  es. `"EXT:my_theme/Resources/Public/Images/my-background.jpg"`
- una risorsa esterna

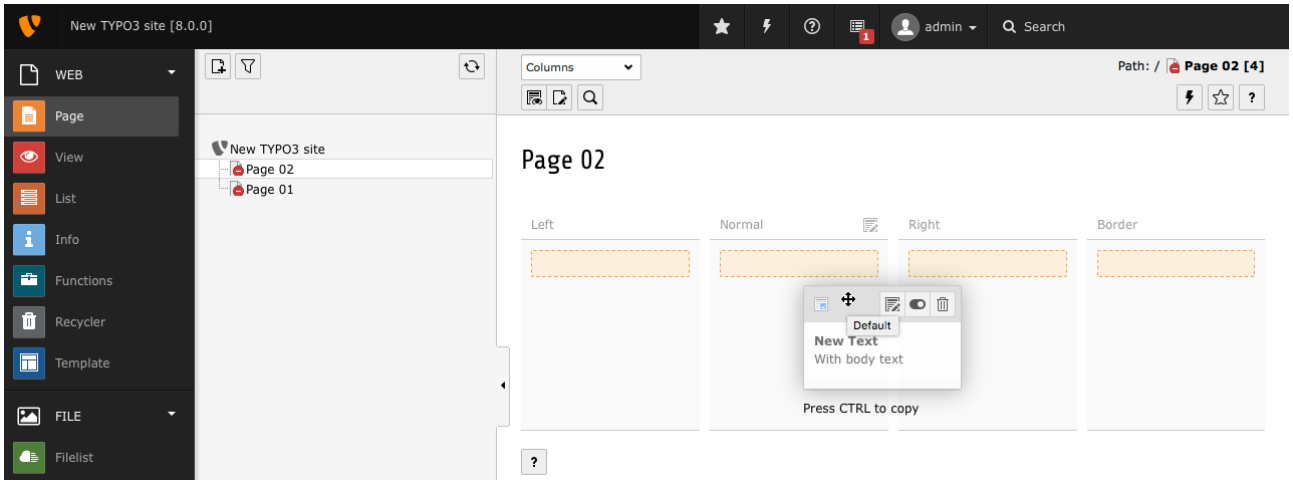
  es. `"//example.com/my-background.png"`



Interfaccia utente Backend

Copia delle pagine in modalità drag & drop

In aggiunta alla tradizionale funzionalità drag and drop nel modulo della pagina (che *muove* gli elementi di contenuto), è ora possibile creare delle copie: premere il tasto CTRL mentre si trascina un elemento per copiarlo. Dopo che il rilascio è completato, il modulo della pagina viene ricaricato per esser sicuri che il nuovo elemento sia stato generato con tutte le informazioni necessarie.



Capitolo 2:

TSconfig & TypoScript

TSconfig & TypoScript

Ordinamento di nuove schede di dati nei contenuti

- E' possibile configurare l'ordinamento delle schede di dati negli elementi di contenuto impostando i valori `before` e `after` nel TSconfig di pagina:

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

TSconfig & TypoScript

`HTMLparser.stripEmptyTags.keepTags`

- La nuova opzione per la configurazione di `HTMLparser.stripEmptyTags` è stata aggiunta, essa permette di mantenere liste di tag configurabili.
- Prima di questa modifica, solo la lista di tag forniti venivano rimossi.
- L'esempio seguente toglie tutti i tag vuoti **eccetto** i tag `tr` e `td`:

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

Importante: se è utilizzata questa configurazione, la configurazione `stripEmptyTags.tags` non ha più effetto. Si può usare una sola opzione alla volta.

TSconfig & TypoScript

`EXT:form` - integrazione di form predefiniti (1)

- L'elemento di contenuto della `EXT:form` permette l'integrazione di form predefiniti.
- Un integratore può definire dei form (es. all'interno di un pacchetto sito) usando `plugin.tx_form.predefinedForms`
- Un editore può aggiungere un nuovo elemento di contenuto `mailform` ad una pagina e scegliere un form da una lista di elementi predefiniti
- L'integratore può costruire i suoi form con TypoScript, che permette molte più opzioni che crearlo con il wizard dei form. Es. l'integratore può usare la funzionalità `stdWrap`, che non è disponibile quando si usa il wizard dei form (per ragioni di sicurezza)

TSconfig & TypoScript

EXT:form - integrazione di form predefiniti (2)

- Non è più necessario che un editore usi il wizard dei form. Gli editori possono predefinire dei form che sono ottimizzati nel layout.
- I form possono essere riutilizzati nell'intera installazione
- I form possono essere salvati fuori dal DB e versionati
- Per poter selezionare i form predefiniti nel backend, essi devono essere salvati utilizzando PageTS:

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

TSconfig & TypoScript

EXT:form: integrazione di form predefiniti (3)

- Esempio di form:

```
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
...
```

Capitolo 3:

Modifiche rilevanti

Modifiche rilevanti

Supporto a PECL-memcached in MemcachedBackend

- Il supporto al modulo PECL "memcached" è stato aggiunto a MemcachedBackend nel Framework di Caching
- Se entrambi, "memcache" e "memcached" sono installati, viene usato "memcache" per evitare blocchi a causa di cambiamenti sostanziali.
- Un integratore può impostare l'opzione `peclModule` per utilizzare il modulo PECL preferito:

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

Modifiche rilevanti

Supporto nativo per Symfony Console (1)

- TYPO3 supporta il componente Symfony Console out-of-the-box fornendo un nuovo script da riga di comando posizionato in `typo3/sysext/core/bin/typo3`. Sulle istanze TYPO3 installate via Composer, l'eseguibile è collegato nella directory `bin`, es. `bin/typo3`.
- Il nuovo eseguibile supporta ancora i parametri del comando esistenti quando nessun comando corretto di Symfony Console viene trovato.
- La registrazione di un comando, per essere disponibile nello strumento da riga di comando `typo3`, funziona inserendo un file `Configuration/Commands.php` dentro un estensione installata. Esso indica le classi Symfony/Console/Command da eseguire in `typo3` in un array associativo. La chiave è il nome del comando da chiamare come primo argomento in `typo3`.

Modifiche rilevanti

Supporto nativo per Symfony Console (2)

- Un parametro obbligatorio, quando si registra un comando, è la proprietà `class`. Opzionalmente il parametro `user` può essere impostato, così l'utente di backend deve essere collegato, quando esegue il comando.
- A `Configuration/Commands.php` può essere ad esempio:

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

Modifiche rilevanti

Supporto nativo per Symfony Console (3)

- Un esempio di chiamata può essere:

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

- Per un installazione non-Composer:

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

Modifiche rilevanti

Generatore di numeri crittograficamente sicuri pseudocasuali

- Un nuovo generatore di numeri crittograficamente sicuri pseudo casuali (CSPRNG) è stato implementato nel core TYPO3.
  Esso sfrutta le nuove funzioni CSPRNG di PHP 7.
- Le API risiedono nella classe `\TYPO3\CMS\Core\Crypto\Random`
- Esempio:

```
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);
```

Modifiche rilevanti

Wizard component (1)

- Un nuovo componente wizard è stato aggiunto. Questo componente può essere usato per interazioni guidate dell'utente
- Il modulo RequireJS può essere usato includendo `TYPO3/CMS/Backend/Wizard`
- Il wizard supporta solamente azioni semplici (azioni complesse non sono ancora possibili)
- Le API risiedono nella classe `\TYPO3\CMS\Core\Crypto\Random`
- Il componente guida ha i seguenti metodi pubblici:

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

Modifiche rilevanti

Wizard component (2)

- L'evento `wizard-visible` viene attivato quando il rendering del wizard è terminato
- I wizard possono essere chiusi chiamando l'evento `wizard-dismiss`
- I wizard attivano l'evento `wizard-dismissed` se esso viene chiuso
- E' possibile integrare il proprio ascoltatore utilizzando `Wizard.getComponent()`

Modifiche rilevanti

Spostati gli asset dei file generati

- La struttura delle cartelle dentro `typo3temp` è stata suddivisa tra un insieme di file che deve essere accessibile dai client e un insieme di file che vengono creati temporaneamente (per esempio la memorizzazione nella cache o per finalità di blocco) e richiedono l'accesso solo lato server.
- Questi file sono stati spostati dalle directory:
  `_processed_`, `compressor`, `GB`, `temp`, `Language`, `pics`
  e riorganizzati in :
  - `typo3temp/assets/js/`
  - `typo3temp/assets/css/`,
  - `typo3temp/assets/compressed/`
  - `typo3temp/assets/images/`

Modifiche rilevanti

Cambiamenti ImageMagick/GraphicsMagick (1)

- Le impostazioni grafiche del processore di immagini Image- o GraphicsMagick sono state rinominate (file: `LocalConfiguration.php`).
  VECCHIO: `im_`
  NUOVO: `processor_`
- Denominazioni negative come `noScaleUp` sono state rinominate in una controparte positiva. Durante la conversione, i valori precedenti della configurazione sono negati per riflettere il cambiamento di semantica in queste opzioni.
- Inoltre, i riferimenti a versioni specifiche di ImageMagick/GraphicsMagick sono stati rimossi dalle impostazioni di nomi e valori.

Modifiche rilevanti

Cambiamenti ImageMagick/GraphicsMagick (2)

- L'opzione di configurazione inutilizzata `image_processing` è stata rimossa senza sostituzione
- L'opzione di configurazione specifica `colorspace` è stata *spostata* sotto la gerarchia `processor_`

Modifiche rilevanti

Hooks e Signals (1)

- Un nuovo hook è stato aggiunto al metodo `BackendUtility::viewOnClick()` dopo la creazione dell'URL di anteprima
- Registrare una classe hook che implementa il metodo con il nome `postProcess`:

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

Modifiche rilevanti

Hooks e Signals (2)

- Prima di TYPO3 CMS 7.6, era possibile sovrascrive un record in `Web -> List`. Un nuovo hook in TYPO3 CMS 8.0 ripropone la vecchia funzionalità.
- L'hook è chiamato con la seguente sintassi:

```
/**
 * @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) { ... }
```

- Registrazione di una classe hook che implementa il metodo con il nome `postOverlayPriorityLookup`:

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

Modifiche rilevanti

Hooks e Signals (3)

- Un nuovo signal è stato implementato prima che una risorsa di archiviazione sia inizializzata.
- Registrazione di una classe che implementa la logica in `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'
);
```

- Il metodo è chiamato con i seguenti parametri:
  - int `$uid` l'uid del record
  - array `$recordData` tutti i record data come array
  - string `$fileIdentifier` l'identificatore del file

Modifiche rilevanti

Algoritmo di hash delle password: PBKDF2

- Il nuovo algoritmo di hashing delle password "PBKDF2" è stato aggiunto all'estensione di sistema "saltedpasswords"
- PBKDF2 sta per: Password-Based Key Derivation Function 2
- L'algoritmo è stato disegnato per essere resistente agli attacchi di "brute force" per la ricerca di password

Capitolo 4:

Extbase & Fluid

Extbase & Fluid

Standalone revised Fluid

- Il motore di rendering Fluid di TYPO3 CMS è sostituito dal pacchetto Fluid standalone che è incluso come una dipendenza di composer
- La vecchia estensione Fluid è convertita con il così chiamato *Fluid adapter* che permette a TYPO3 CMS di usare Fluid standalone
- Nuove caratteristiche/funzionalità sono state aggiunte in tutte le aree di Fluid
- Molto importante: molti dei componenti di Fluid che erano completamente interni ed impossibili da sostituire in passato, sono ora facilmente sostituibili e sono stati dotati di un API pubblica.

Extbase & Fluid

RenderingContext (1)

- La novità più importante delle API pubbliche è il RenderingContext
- Il precedente, solo interno, RenderingContext usato da Fluid è stato ampliato per essere responsabile di nuove funzionalità vitali di Fluid: **implementation provisioning**
- Questo permette agli sviluppatori di cambiare un range di classi, che Fluid usa per il parser, la conversione, il caching ecc.
- Questo può essere ottenuto sia includendo un RenderingContext personalizzato o intervenendo con i metodi pubblici sul RenderingContext di default.

Extbase & Fluid

Rendering Context (2)

- Le funzionalità delle seguenti slide possono essere controllate intervenendo sul RenderingContext. Di default, nessuna di esse è abilitata - ma chiamando un metodo semplice (attraverso un istanza View) permette di abilitarle:

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

Extbase & Fluid

ExpressionNodes (1)

- Le ExpressionNodes sono un nuovo tipo di sintassi della struttura Fluid che condividono tutte un comune tratto: funzionano solo all'interno delle parentesi grafe

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

- Gli sviluppatori possono aggiungere le loro ExpressionNode
- Ciascuna di esse consiste in un modello da abbinare e dei metodi dettati da un interfaccia di processo
- Ogni tipo esistente di ExpressionNode può essere usato come riferimento

Extbase & Fluid

ExpressionNodes (2)

ExpressionNodeTypes permettono nuove sintassi come ad esempio:

- **CastingExpressionNode**
  permette il casting di una variabile ad un certo tipo, per esempio per garantire un integer o un boolean. E' usato semplicemente con la chiave `as`: `{myStringVariable as boolean}` o `{myBooleanVariable as integer}` e così via. Il tentativo di cast di una variabile ad un tipo incompatibile, causa un errore Fluid standard.
- **MathExpressionNode**
  permette operazioni matematiche di base sulle variabili, per esempio `{myNumber + 1}`, `{myPercent / 100}` o `{myNumber * 100}` e così via. Un espressione impossibile ritorna un output vuoto.

Extbase & Fluid

ExpressionNodes (3)

ExpressionNodeTypes permettono nuove sintassi come ad esempio:

- **TernaryExpressionNode**
  permette una condizione ternaria che opera solo su variabili. Un caso di uso tipico è: "se questa variabile allora usa quella variabile altrimenti usa un'altra variabile". E' usato come:
  `{myToggleVariable ? myThenVariable : myElseVariable}`
  Nota: non supporta espressioni nidificate, sintassi di inline ViewHelper o simili al suo interno. Deve essere utilizzata solo con variabili standard come input.

Extbase & Fluid

I Namespace sono estensibili (1)

- Fluid permette che ogni alias di namespace (ad esempio `f:`) possa essere esteso con l'aggiunta di un ulteriore namespaces
- Anche i namespace PHP sono verificati per la presenza di classi ViewHelper
- Questo significa che gli sviluppatori possono sovrascrivere singoli ViewHelper con versioni personalizzate ed avere i loro ViewHelper chiamati quando si usa il namespace `f:`
- Questo cambiamento implica anche che i namespace non siano più monadic. Usando `{namespace f=My\Extension\ViewHelpers\}` non si riceverà più un errore "namespace already registered". Fluid aggiungerà invece questo namespace PHP e cercherà i ViewHelper anche lì.

Extbase & Fluid

I Namespaces sono estensibili (2)

- I namespace aggiunti sono verificati dal basso verso l'alto, permettendo ai namespace aggiunti di sostituire le classi ViewHelper mettendole alla stessa portata
- Per esempio: `f:format.nl2br` può essere sostituito in `My\Extension\ViewHelpers\Format\Nl2brViewHelper`, con la registrazione del namespace alla precedente slide

Extbase & Fluid

Uso del rendering `f:render` (1)

Consente contenuto predefinito come opzionale `f:render`:

- Ogniqualvolta `f:render` è usato con flag `optional = TRUE` impostato, la visualizzazione di una sezione mancante da come risultato un output vuoto.
- Invece di visualizzare un output vuoto, un nuovo attributo di `default` (mixed) è aggiunto e può essere compilato con un valore di default di tipo fallback.
- In alternativa, il contenuto tag può essere usato per definire questo valore predefinito come tanti altri contenuti/attributi flessibili dei ViewHelper

Extbase & Fluid

Uso del rendering `f:render` (2)

Passaggio di contenuti con tag dal `f:render` al partial/section:

- Permette un nuovo approcio per strutturare la composizione dei template Fluid
- Partial e section possono essere utilizzati come "wrapper" per parti arbitrarie di codice.
- Esempio:

```
<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

Istruzioni condizionali complesse

- Ora Fluid supporta qualsiasi tipo di istruzione complessa di condizioni con nidificazioni e raggruppamenti:

```
<f:if condition="({variableOne} && {variableTwo}) || {variableThree} || {variableFour}">
  // Esegue se le variabili One e Two valgono true ,
  // o se lo sono le variabili Three o Four.
</f:if>
```

- In più, `f:else` è stato dotato del comportamento tipo "elseif":

```
<f:if condition="{variableOne}">
  <f:then>Fa questo se la variabile One vale true</f:then>
  <f:else if="{variableTwo}">
    Oppure fa questo se invece la variabile Two vale true
  </f:else>
  <f:else if="{variableThree}">
    Oppure fa questo se la variabile Three vale true
  </f:else>
  <f:else>
    Oppure fa questo se nessuno dei precedenti vale true
  </f:else>
</f:if>
```

Extbase & Fluid

Parti dei nomi di variabili dinamiche (1)

- Una nuova caratteristica è la possibilità di utilizzare dei riferimenti dinamici con variabili quando si accede ad una variabile. Si consideri il seguente template Fluid con array di variabili:

```
$mykey = 'foo'; // o 'bar', impostato in un codice sorgente
$view->assign('data', ['foo' => 1, 'bar' => 2]);
$view->assign('key', $mykey);
```

- Con il seguente template Fluid:

```
Scegliendo: {data.{key}}.
(output: "1" con la chiave "foo" o "2" con "bar")
```

Extbase & Fluid

Parti dei nomi di variabili dinamiche (2)

- Lo stesso approcio può essere utilizzato per generare parti dinamiche del nome di variabile:

```
$mydynamicpart = 'First'; // o 'Second', impostato in un codice sorgente
$view->assign('myFirstVariable', 1);
$view->assign('mySecondVariable', 2);
$view->assign('which', $mydynamicpart);
```

- Con il seguente template Fluid:

```
Scegliendo: {my{which}Variable}.
(output: "1" se which vale "First" o "2" se which vale "Second")
```

Extbase & Fluid

Nuovi ViewHelper

- Alcuni nuovi ViewHelper sono stati aggiunti come parti standalone di Fluid e come tali sono disponibili da ora in TYPO3:
  - `f:or`
    Questo è un modo più breve di scrivere condizioni concatenate. Esso supporta la seguente sintassi, che verifica ogni variabile restituendo la prima che non è vuota:

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

  - `f:spaceless`
    Può essere utilizzato nei tag nel codice dei template per eliminare gli spazi e righe vuote per esempio creati dall'uso di indentazione dei ViewHelper

Extbase & Fluid

I namespaces dei ViewHelper possono essere estesi da PHP

- Accedendo al ViewHelperResolver del RenderingContext, gli sviluppatori possono cambiare le inclusioni del namespace di base del ViewHelper a livello globale (leggi: istanza View):

```
$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

I ViewHelper possono accettare argomenti opzionali (1)

- Questa funzionalità permette alla classe ViewHelper un numero qualsiasi di parametri aggiuntivi utilizzando qualsiasi variabile si desideri
- Funziona separando i parametri che vengono passati ad ogni ViewHelper in due gruppi: quelli che vengono dichiarati utilizzando `registerArgument` (o come parametri del metodo) e quelli che non lo sono
- I parametri non dichiarati, sono passati ad una funziona speciale `handleAdditionalArguments` nella classe del ViewHelper, che nell'implementazione predefinita genera un errore se esistono parametri aggiuntivi.

Extbase & Fluid

I ViewHelper possono accettare argomenti opzionali (2)

- Con l'ovveride di questo metodo nel ViewHelper, si può modificare se e quando il ViewHelper dovrebbe registrare un errore alla ricezione di parametri non registrati
- Questa funzionalità è quella che permette ai TagBasedViewHelpers di accettare liberamente parametri con prefisso `data-` senza andare in errore
- In TagBasedViewHelpers, il metodo `handleAdditionalArguments` aggiunge semplicemente nuovi attributi al tag generato e crea un errore se sono presenti parametri aggiuntivi che non sono registrati o con il prefisso `data-`.

Extbase & Fluid

Parametri "allowedTags" per f:format.stripTags

- Il parametro `allowedTags` contiene una lista di tag HTML che non devono essere tolti usando `f:format.stripTags`
- La sintassi della lista di tag è identica al secondo parametro del comando PHP `strip_tags` (vedi: http://php.net/strip_tags)

Extbase & Fluid

Consentire l'accesso a ObjectStorage come array in Fluid

- Creando un alias di `toArray()` si permette al metodo di essere chiamato come `getArray()` che a sua volta permette al metodo di essere chiamato da `ObjectAccess::getPropertyPath`, che consente l'accesso a Fluid e altre parti
- Con la creazione di un semplice alias di `toArray()` su ObjectStorage, permette di essere chiamato come `getArray()`
- Esempio: ottenere il quarto elemento

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

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

Capito 5:

Funzionalità deprecate/rimosse

Funzionalità deprecate/rimosse

Varie

- Le seguenti opzioni di configurazione sono state rimosse:
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_utils']`
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_convMethod']`

  (la funzionalità è ora rilevata automaticamente e `mbstring` è usata di default se disponibile)
- La proprietà TypoScript deprecata `page.includeJSlibs` è stata rimossa. Utilizzare la proprietà TypoScript `page.includeJSLibs` ("L" maiuscola) al suo posto
- L'opzione TypoScript `config.renderCharset`, che era usata come set di caratteri di conversione interna ad una richiesta di frontend, è stata rimossa

Capitolo 6:

Fonti e autori

Fonti e autori

Fonti

TYPO3 News:

- http://typo3.org/news

Note sulla release:

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

TYPO3 Bug-/Segnalazioni:

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

Repositori Git di TYPO3 e Fluid:

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

Fonti e autori

Team 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 e **Roberto Torresani**

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