TYPO3 CMS 8.0 – Qué hay Nuevo

Resumen de las nuevas características, cambios y mejoras

Creado por:
Patrick Lobacher y Michael Schams

Traducción en Español por:
Michel Mix y Sergio Catalá

16/April/2016

TYPO3

TYPO3 CMS 8.0 - What's New

Resumen de Capítulos

Introducción

Los Hechos

Introducción

TYPO3 CMS 8.0 – Los Hechos

- Fecha de lanzamiento: 22 Marzo 2016
- Tipo de lanzamiento: Lanzamiento Sprint
- Eslogan: Enciendan sus motores

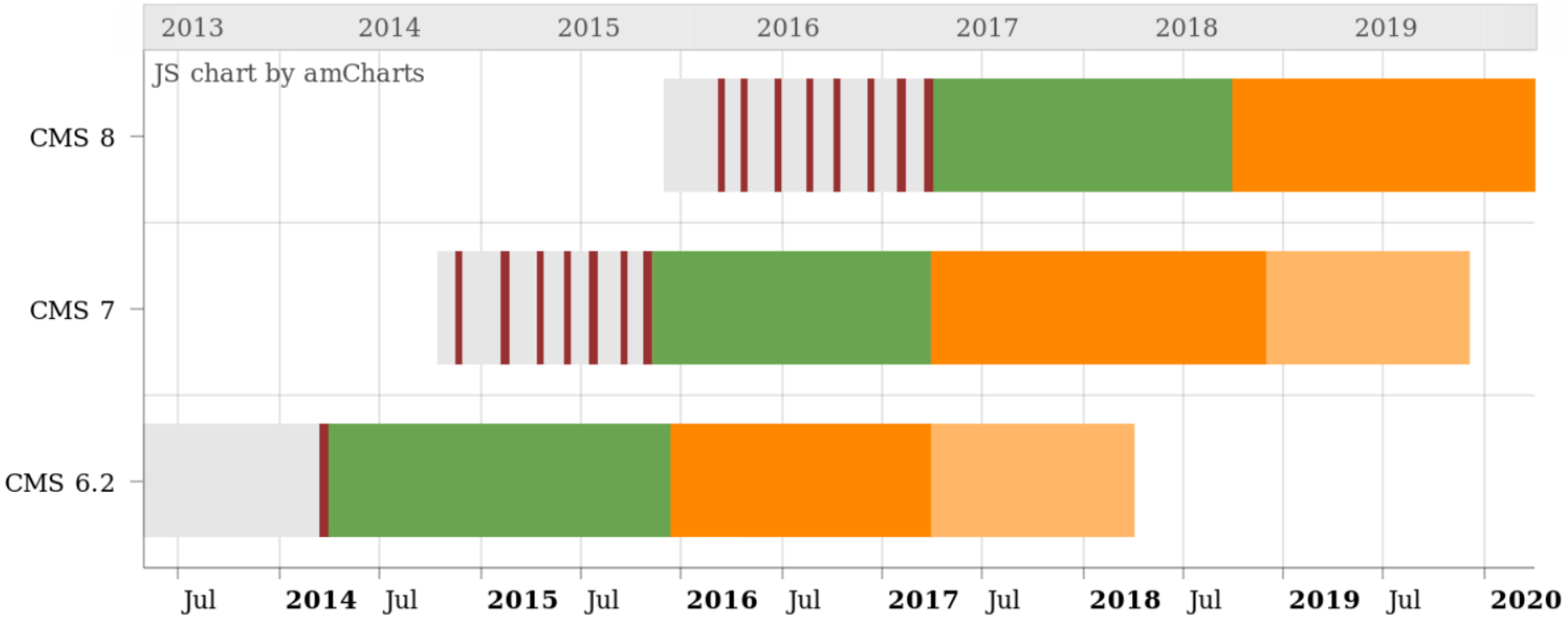
Introducción

Requisitos del Sistema

- PHP: versión 7
- MySQL: versión 5.5 to 5.7
- Disk space: mín 200 MB
- Ajustes de PHP:
  - `memory_limit` >= 128M
  - `max_execution_time` >= 240s
  - `max_input_vars` >= 1500
  - opción de compilación `--disable-ipv6` NO debe ser usada
- El backend requiere Microsoft Internet Explorer 11 o posterior, Microsoft Edge, Google Chrome, Firefox, Safari o cualquier otro navegador moderno y compatible

Introducción

Línea de tiempo de Desarrollo y Lanzamiento



Introducción

Línea de lanzamiento de TYPO3 CMS

Fechas de lanzamiento y sus enfoques principales:

- v8.0 22/Mar/2016 Añadiendo cosas de última hora
- v8.1 03/May/2016 Integración con la Nube
- v8.2 05/Jul/2016 Editor de Texto Enriquecido
- v8.3 30/Ago/2016 Edición del Frontend con Steroids
- v8.4 18/Oct/2016 *por determinar*
- v8.5 20/Dic/2016 Soporte de Integrador
- v8.6 14/Feb/2017 *por determinar*
- v8.7 04/Abr/2017 Preparación LTS

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

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

Introducción

Instalación

- Procedimiento de instalación oficial bajo Linux/Mac OS X (DocumentRoot por ejemplo `/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
```

- Enlaces simbólicos bajo Microsoft Windows:
  - Use `junction` en Windows XP/2000
  - Use `mklink` en Windows Vista y Windows 7

Introducción

Actualización a TYPO3 CMS 8.x

- Actualizaciones sólo posibles desde TYPO3 CMS 7.6 LTS
- TYPO3 CMS < 7.6 LTS debe ser actualizado a TYPO3 CMS 7.6 LTS primero
- Instrucciones de actualización:

  http://wiki.typo3.org/Upgrade#Upgrading_to_8.0
- Guía oficial de TYPO3 "Instalación y Actualización de TYPO3":

  http://docs.typo3.org/typo3cms/InstallationGuide
- Enfoque general:
  - Comprobar requisitos mínimos del sistema (PHP, MySQL, etc.)
  - Revisar **deprecation_*.log** en instancia antigua de TYPO3
  - Actualizar todas las extensiones a la última versión
  - Desplegar fuentes nuevas y ejecutar Herramienta de Instalación -> Asistente de Actualización
  - Revisar el módulo de inicio para usuarios backend (opcionalmente)

Introducción

PHP Versión 7

- PHP 7.0 es el requisito mínimo para TYPO3 CMS 8.x
- TYPO3 soportará posteriores lanzamientos de PHP 7 cuando aparezcan
- Este aumento de versión proporciona un significativo aumento de rendimiento de todo el sistema
- No sólo los editores del backend notarán una interfaz más fluida, sino que el tiempo al completo para una llamada de página cacheada en el frontend no supera los 7 milisegundos ahora, que es aproximadamente el 40% más rápido si lo comparamos a ejecutar la misma página web con PHP versión 5.5
- También comenzamos a usar nuevas características de esta versión de PHP, por ejemplo los generadores seguros criptográficamente pseudo-aleatorios están ya en uso activo

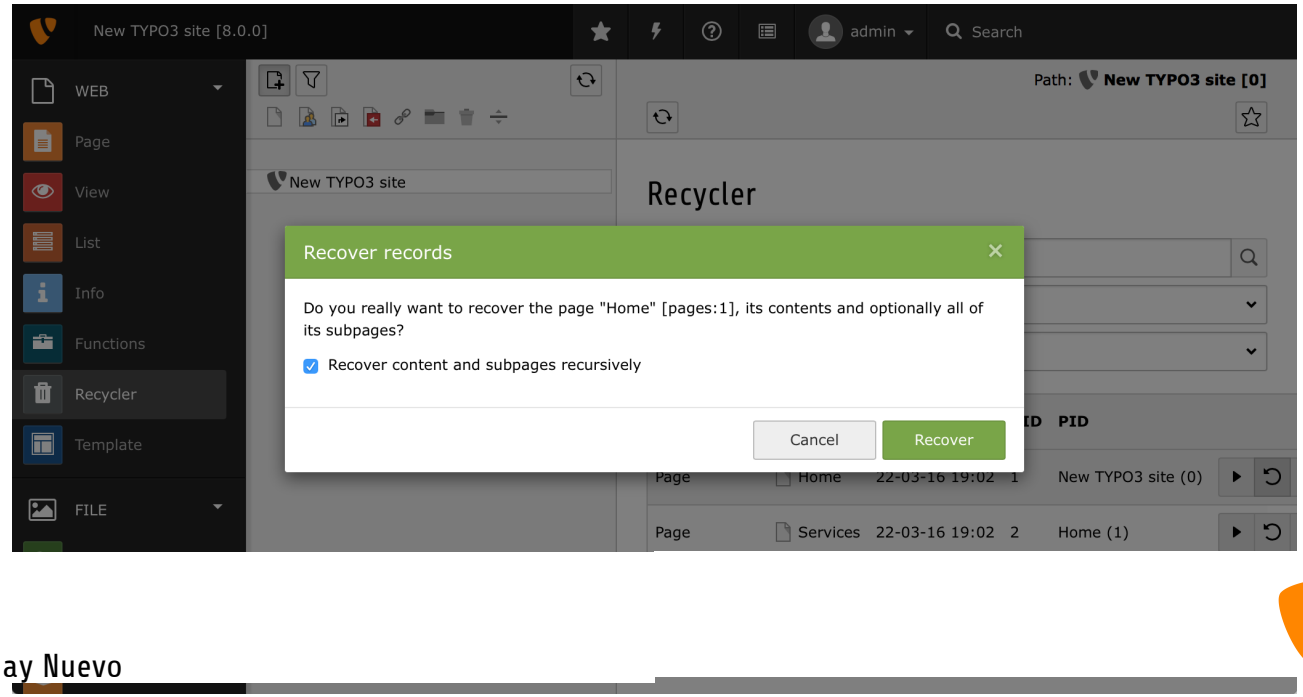
Capítulo 1:

Interfaz de Usuario de Backend

Interfaz de Usuario de Backend

Recupera páginas recursivamente hasta la parte superior de la raíz

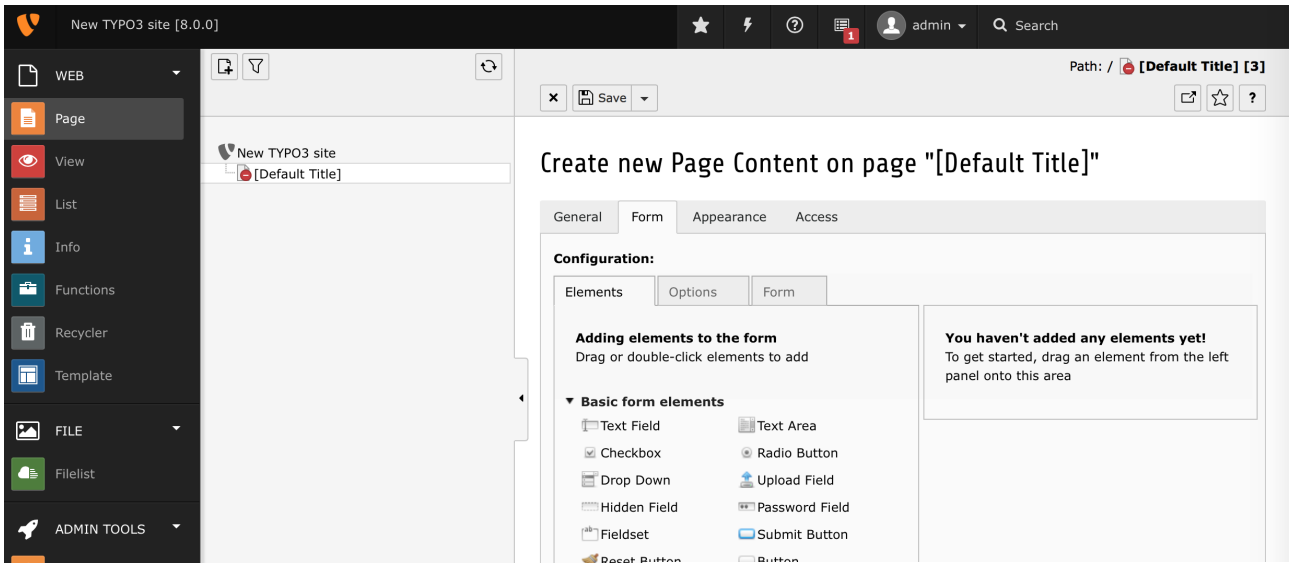
La Papelera de reciclaje soporta la recuperación recursiva de páginas borradas hasta la parte superior de la raíz ahora. Esta característica está sólo disponible para usuarios admin debido a restricciones internas de permisos.



Interfaz de Usuario de Backend

Cargue directamente el asistente del formulario como asistente inline

El asistente de EXT:form es cargado directamente como asistente inline. No es necesario ya guardar y recargar el elemento de contenido recién creado para ser capaces de abrir el asistente. Esto es una gran mejora de usabilidad.



Interfaz de Usuario de Backend

Configure un logo alternativo para el backend vía Manejador de Extensiones (1)

El logo del backend en la esquina superior izquierda puede ahora configurarse en la configuración de la extensión EXT:backend en el Manejador de Extensiones.
Las opciones de configuración son:

- recurso como una ruta relativa de la instalación de TYPO3

  p.e. "`fileadmin/images/my-background.jpg`"
- ruta a una extensión

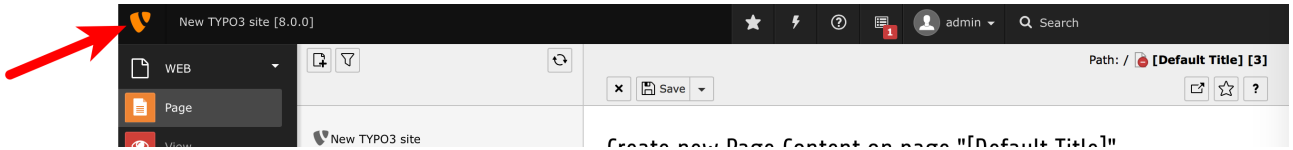
  p.e. "`EXT:my_theme/Resources/Public/Images/my-background.jpg`"

Interfaz de Usuario de Backend

Configure un logo alternativo para el backend vía Manejador de Extensiones (2)

- un recurso externo

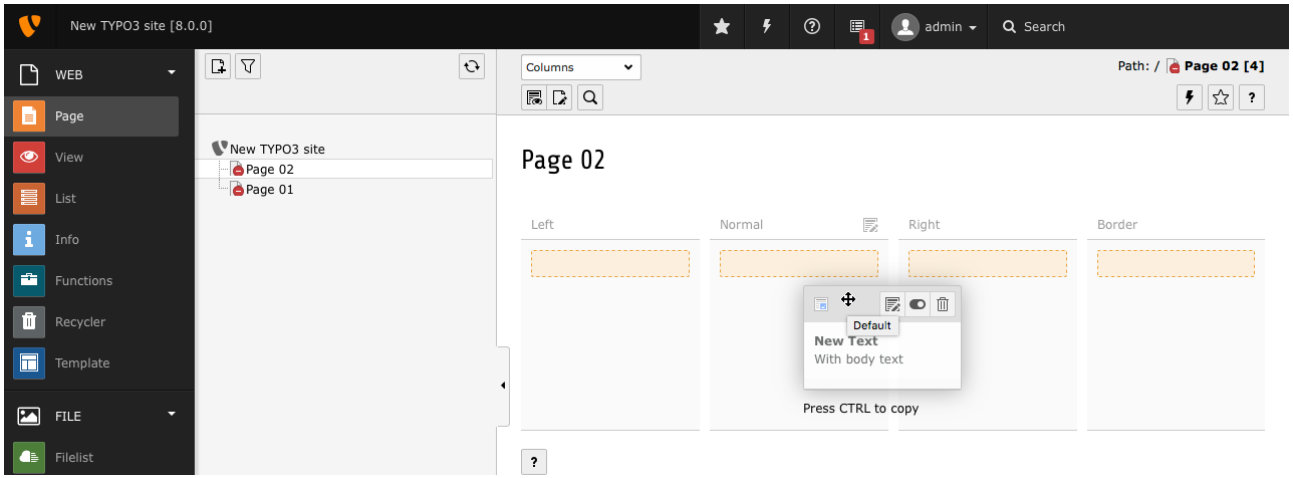
  p.e. `"//example.com/my-background.png"`



Interfaz de Usuario de Backend

Copie páginas en modo arrastrar & soltar

Además de la característica usual de arrastrar y soltar en el módulo de página (que *movía* elementos de contenido), ahora es posible crear copias: pulse la tecla CTRL mientras arrastra para crear una copia del elemento arrastrado. Después de que el arrastre se ha completado, el módulo de página se recargará para asegurar que el nuevo elemento será generado con toda la información necesaria.



Capítulo 2:

TSconfig & TypoScript

TSconfig & TypoScript

Clasifique orden de tabs del asistente de nuevo elemento de contenido

- Es posible configurar el orden de los tabs en el asistente de nuevo elemento de contenido configurando los valores `before` y `after` en Page TSconfig:

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

TSconfig & TypoScript

HTMLparser.stripEmptyTags.keepTags

- Ha sido añadida una nueva opción para la configuración de HTMLparser.stripEmptyTags, que permite mantener etiquetas configuradas
- Antes de este cambio, sólo una lista de etiquetas podía ser proporcionada para ser eliminada
- El siguiente ejemplo elimina todas las etiquetas vacías **excepto** las etiquetas tr y td:

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

Importante: si se usa este parámetro, la configuración stripEmptyTags.tags no tiene más efecto. Sólo puede usar una opción a la vez.

TSconfig & TypoScript

`EXT:form` - integración de formularios predefinidos (1)

- El elemento de contenido de `EXT:form` ahora permite la integración de formularios predefinidos.
- Un integrador puede definir formularios (p.e. dentro de un paquete de sitio web) usando `plugin.tx_form.predefinedForms`
- Un editor puede añadir un nuevo elemento de contenido `mailform` a una página y elegir un formulario de una lista de elementos predefinidos
- Los integradores pueden construir sus formularios con TypoScript, que proporciona muchas más opciones que al hacerlo dentro del asistente de formulario (p.e. integradores pueden usar la funcionalidad `stdWrap`, que no está disponible al usar el asistente de formulario (por razones de seguridad)

TSconfig & TypoScript

EXT:form - integración de formularios predefinidos (2)

- No hay necesidad para los editores de usar el asistente de formulario nunca más. Los editores pueden elegir los formularios predefinidos que están optimizados en base al diseño.
- Los formularios pueden ser reutilizados a través de la instalación al completo
- Los formularios pueden almacenarse fuera de la BBDD y versionarse
- Para ser capaces de seleccionar el formulario predefinido en el backend, el formulario tiene que registrarse usando PageTS:

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

TSconfig & TypoScript

EXT:form: integración de formularios predefinidos (3)

- Formulario de ejemplo:

```
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

Capítulo 3:

Cambios en Profundidad

Cambios en Profundidad

Soporte de PECL-memcached en MemcachedBackend

- Se ha añadido soporte para el módulo PECL "memcached" al MemcachedBackend del Framework de Cacheo
- Si ambos, "memcache" y "memcached" están instalados, se usa "memcache" para evitar que sea un cambio en profundidad.
- Un integrador puede configurar la opción `peclModule` para usar el módulo PECL preferido:

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

Cambios en Profundidad

Soporte nativo para la Consola de Symfony (1)

- TYPO3 soporta ahora el componente de Consola de Symfony listo para usar proporcionando un nuevo script de línea de comandos ubicado en `typo3/sysext/core/bin/typo3`. En instancias TYPO3 instaladas a través de Composer, el binario está enlazado dentro del directorio `bin`, p.e. `bin/typo3`.
- El nuevo binario todavía soporta los argumentos existentes de la línea de comandos al si no se encuentra por defecto un comando adecuado de Consola de Symfony.

Cambios en Profundidad

Soporte nativo para la Consola de Symfony (2)

- Registrar un comando para estar disponible a través de la herramienta de línea de comandos `typo3` funciona poniendo un fichero `Configuration/Commands.php` en una extensión instalada. Esto lista las clases Symfony/Console/Command classes para ser ejecutadas por `typo3` es un vector asociativo. La clave es el nombre del comando a ser llamado como el primer argumento para `typo3`.

Cambios en Profundidad

Soporte nativo para la Consola de Symfony (3)

- Un parámetro requerido al registrar un comando es la propiedad `class`. Opcionalmente el parámetro `user` puede ser configurado para que un usuario del backend esté autenticado al llamar al comando.
- Un `Configuration/Commands.php` podría parecerse a:

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

Cambios en Profundidad

Soporte nativo para la Consola de Symfony (3)

- Una llamada de ejemplo podría parecerse a:

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

- Para una instalación no-Composer:

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

Cambios en Profundidad

Generador de números pseudoaleatorio criptográficamente seguro

- Un nuevo generador de números pseudo-aleatorios criptográficamente seguros (CSPRNG) ha sido implementado en el núcleo de TYPO3.
  Se beneficia de las nuevas funciones CSPRNG en PHP 7.
- El API reside en la clase `\TYPO3\CMS\Core\Crypto\Random`
- Ejemplo:

```
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);
```

Cambios en Profundidad

Componente asistente (1)

- Un nuevo componente asistente ha sido añadido. Este componente puede ser usar para interacciones guiadas de usuario
- El módulo RequireJS puede ser usado incluyendo `TYPO3/CMS/Backend/Wizard`
- El asistente soporta acciones sencillas sólo (junctions no son posibles todavía)
- El API reside en la clase `\TYPO3\CMS\Core\Crypto\Random`
- El componente asistente tiene los siguientes métodos públicos:

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

Cambios en Profundidad

Componente asistente (2)

- El evento `wizard-visible` es lanzado cuando el renderizado del asistente ha finalizado
- Los asistentes pueden ser cerrados al lanzar el evento `wizard-dismiss`
- Los asistentes lanzan el evento `wizard-dismissed` si el asistente es cerrado
- Puedes integrar tu propio manejador usando `Wizard.getComponent()`

Cambios en Profundidad

Movidos ficheros asset generados

- La estructura de carpeta dentro de `typo3temp` cambió a assets separados que necesitan ser accedidos por el cliente desde los ficheros para los cuales son temporalmente creados (p.e. para propósitos de cacheo o cierre y requiere sólo acceso al lado del servidor).
- Estos assets fueron movidos desde las carpetas:
  `_processed_`, `compressor`, `GB`, `temp`, `Language`, `pics`
  y reorganizados en:
  - `typo3temp/assets/js/`
  - `typo3temp/assets/css/`,
  - `typo3temp/assets/compressed/`
  - `typo3temp/assets/images/`

Cambios en Profundidad

Cambios en ImageMagick/GraphicsMagick (1)

- Los ajustes del procesador de gráficos para Image- y GraphicsMagick han sido renombrados (fichero: `LocalConfiguration.php`).
  OLD: `im_`
  NEW: `processor_`
- Nombramientos negativos tales como `noScaleUp` han sido cambiados a equivalentes positivos. Durante la conversión, los valores de configuración previos son negados para reflejar los cambios en semántica de estas opciones.
- Adicionalmente, las referencias a versiones de ImageMagick/GraphicsMagick han sido eliminadas de los nombres y valores de configuración.

Cambios en Profundidad

Cambios ImageMagick/GraphicsMagick (2)

- La opción de configuración no empleada `image_processing` ha sido eliminada sin reemplazo
- La opción de configuración de procesador específico `colorspace` se ha transformado a *espacio de nombres* bajo la jerarquía `processor_`

Cambios en Profundidad

Hooks y Señales (1)

- Un hook adicional ha sido añadido al método `BackendUtility::viewOnClick()` para post-procesar la URL de previsualización
- Registre una clase hook que implementa el método con el nombre `postProcess`:

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

Cambios en Profundidad

Hooks y Señales (2)

- Previamente a TYPO3 CMS 7.6, era posible reemplazar una superposición de registro en `Web -> List`. Un nuevo hook en TYPO3 CMS 8.0 proporciona la antigua funcionalidad.
- El hook es llamado con la siguiente signatura:

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

- Registre la clase hook que implementa el método con el nombre `postOverlayPriorityLookup`:

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

Cambios en Profundidad

Hooks y Señales (3)

- Una nueva señal ha sido implementada antes de que sea inicializado un almacenamiento de recursos.
- Registre la clase que implementa su lógica en `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'
);
```

- El método es llamado con los siguientes argumentos:
  - int `$uid` el uid del registro
  - array `$recordData` todos los datos de registro como vector
  - string `$fileIdentifier` el identificador de fichero

Cambios en Profundidad

Algoritmo de hashing para contraseñas: PBKDF2

- Un nuevo algoritmo "PBKDF2" de hashing para contraseñas ha sido añadido a la extensión del sistema "saltedpasswords"
- PBKDF2 se refiere a: Password-Based Key Derivation Function 2
- El algoritmo está diseñado para ser computacionalmente caro al resistir el crackeo de fuerza bruta

Capítulo 4:

Extbase & Fluid

Extbase & Fluid

Fluid independiente revisado

- El motor de renderizado Fluid de TYPO3 CMS es reemplazado por Fluid independiente que es ahora incluido como una dependencia de composer
- La antigua extensión Fluid es convertida a un denominado *adaptador Fluid* que permite a TYPO3 CMS usar Fluid independiente
- Nuevas características/prestaciones han sido añadidas a casi todas las áreas de Fluid
- Más importante: varios de los componentes Fluid que fueron completamente internos e imposibles de reemplazar en el pasado, son ahora fáciles de reemplazar y han sido ajustados con una API pública

Extbase & Fluid

RenderingContext (1)

- La nueva pieza más importante de la API pública es el RenderingContext
- El RenderingContext previamente sólo usado a nivel interno por Fluid ha sido expandido para ser responsable de una nueva característica Fluid vital: **aprovisionamiento de implementación**
- Esto permite a los desarrolladores el cambiar un rango de clases, que Fluid usa para parsear, resolver, cachear etc.
- Esto puede ser alcanzado o bien incluyendo un RenderingContext personalizado o manipulando el RenderingContext por defecto a través de métodos públicos.

Extbase & Fluid

Rendering Context (2)

- Los siguientes comportamientos pueden todos ser controlados manipulando el RenderingContext. Por defecto, ninguno de ellas están habilitados - pero llamando un simple método (vía su instancia de Vista) le permite habilitarlas:

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

Extbase & Fluid

ExpressionNodes (1)

- ExpressionNodes son un nuevo tipo de estructuras de sintaxis Fluid las cuales todas comparten un rasgo común: sólo trabajan dentro de llaves

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

- Los desarrolladores pueden añadir sus propios tipos ExpressionNode adicionales
- Cada uno consiste en un patrón para ser emparejado y métodos dictados por una interfaz para procesar los emparejados
- Cualquier tipo de ExpressionNode existente puede ser usado como referencia

Extbase & Fluid

ExpressionNodes (2)

ExpressionNodeTypes permiten nuevas sintaxis tales como:

- **CastingExpressionNode**
  permite transformar una variable a ciertos tipos, por ejemplo para garantizar un entero o un booleano. Es usado simplemente con una clave `as`: `{myStringVariable as boolean}` o `{myBooleanVariable as integer}` y etcétera. Intentar transformar una variable a un tipo incompatible causa un error Fluid estándar.
- **MathExpressionNode**
  permite operaciones matemáticas básicas en variables, por ejemplo `{myNumber + 1}`, `{myPercent / 100}` o `{myNumber * 100}` y etcétera. Una expresión imposible devuelve una salida vacía.

Extbase & Fluid

ExpressionNodes (3)

ExpressionNodeTypes permiten nuevas sintaxis tales como:

- **TernaryExpressionNode**
  permite una condición ternaria inline que sólo opera sobre variables. Un típico caso de uso es: "si esta variable entonces usa esta variable sino usa otra variable". Es usado como:
  `{myToggleVariable ? myThenVariable : myElseVariable}`
  Nota: no soporta expresiones anidadas, sintaxis inline ViewHelper o similares dentro. Debe ser usado sólo con variables estándar como entrada.

Extbase & Fluid

Espacios de nombres son extensibles (1)

- Fluid permite que cada alias de espacio de nombres (por ejemplo `f:`) sea extendido añadiendo un espacio de nombres PHP adicional a él
- Espacios de nombres PHP son también chequeados para la presencia de clases ViewHelper
- Esto también significa que los desarrolladores pueden reemplazar ViewHelpers individuales con versiones personalizadas y llamar a sus ViewHelpers cuando el espacio de nombres `f:` es usado

Extbase & Fluid

Espacios de nombres son extensibles (2)

- Este cambio también implica que los espacios de nombres no son más monádicos.
  Al usar `{namespace f=My\Extension\ViewHelpers\}`
  no recibirá más un error de "espacio de nombres ya registrados". Fluid añadirá este espacio de nombres PHP en cambio y buscará ViewHelpers allí también.
- Espacios de nombres adicionales son chequeados de abajo a arriba, permitiendo que los espacios de nombres adicionales reemplacen clases ViewHelper al colocarlos en el mismo alcance
- Por ejemplo: `f:format.nl2br` puede ser sobreescrito por `My\Extension\ViewHelpers\Format\Nl2brViewHelper`, dado el registro de espacio de nombres en la diapositiva previa

Extbase & Fluid

Renderizado usando `f:render` (1)

Permita contenido por defecto en `f:render` opcional:

- Al usarse `f:render` y el flag `optional = TRUE` es configurado, renderizar una sección que falta produce una salida vacía.
- En lugar de renderizar una salida vacía, un nuevo atributo `default` (mixed) es añadido y puede ser llenado con un valor por defecto fallback-type.
- Alternativamente, el contenido de etiqueta puede ser usado para definir este valor por defecto como muchos otros ViewHelpers flexibles de contenido/atributo

Extbase & Fluid

Renderizado usando `f:render` (2)

El paso de contenido de etiqueta desde `f:render` a parcial/sección:

- Permite una nueva aproximación para estructurar el renderizado de template Fluid
- Parciales y secciones pueden ser usados como "envolturas" para una pieza arbitraria de código de template.
- Ejemplo:

```
<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

Sentencias condicionales complejas

- Fluid ahora soporta cualquier grado de sentencias condicionales complejas con anidamiento y agrupamiento:

```
<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>
```


TYPO3

Extbase & Fluid

Sentencias condicionales complejas (2)

- Adicionalmente, `f:else` ha sido ajustado con un comportamiento de tipo "elseif":

```
<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

Partes de nombres de variables dinámicas (1)

- Otra nueva característica forzada, compatible para atrás, es la añadida habilidad para usar referencias de sub-variable al acceder a sus variables. Considere el siguiente array de variables template Fluid:

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

- Con el siguiente template Fluid:

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

Extbase & Fluid

Partes de nombres de variables dinámicas (2)

- La misma aproximación puede ser también empleada para generar partes dinámicas de un nombre de variables de cadena:

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

- Con el siguiente template Fluid:

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

Extbase & Fluid

Nuevos ViewHelpers

- Un par de nuevos ViewHelpers han sido añadidos como parte de Fluid independiente y como tales están también disponibles en TYPO3 desde ahora:
  - `f:or`
    Esto es una forma más corta de escribir condiciones (encadenadas). Soporta la siguiente sintaxis, que chequea cada variable y saca por salida la primera que no esté vacía:

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

  - `f:spaceless`
    Esto puede ser usado en modo de etiqueta en un código de template para eliminar espacios en blanco y líneas vacías redundantes por ejemplo causados por usos de ViewHelper indentados

Extbase & Fluid

Nombres de espacios de ViewHelper pueden ser también extendidos desde PHP (1)

- Al acceder al ViewHelperResolver del RenderingContext, los desarrolladores pueden cambiar las inclusiones de espacio de nombres de ViewHelper en una base (léase: por instancia Vista) global:

Extbase & Fluid

Nombres de espacios de ViewHelper pueden ser también extendidos desde PHP (2)

```
$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

ViewHelpers pueden aceptar argumentos arbitrarios (1)

- Esta característica permite a su clase ViewHelper recibir cualquier número de argumentos adicionales usando cualquier nombre que usted desee
- Funciona separando los argumentos que son pasados a cada ViewHelper en dos grupos: aquellos que están declarados usando `registerArgument` (o renderiza argumentos de métodos), y aquellos que no lo están
- Aquellos que no están declarados, son pasados a una función especial `handleAdditionalArguments` en la clase ViewHelper, que en la implementación por defecto lanza un error si existen argumentos adicionales

Extbase & Fluid

ViewHelpers pueden aceptar argumentos arbitrarios (2)

- Sobreescribiendo este método en su ViewHelper, usted puede cambiar si y cuándo el ViewHelper debería lanzar un error al recibir argumentos no registrados
- Esta característica es también la que permite a los TagBasedViewHelpers libremente aceptar argumentos prefijados `data-` arbitrarios sin fallar
- En TagBasedViewHelpers, el método `handleAdditionalArguments` simplemente añade nuevos atributos a la etiqueta que se genera y lanza un error si se proporciona cualquier argumento adicional que no esté ni registrado ni prefijado con `data-`.

Extbase & Fluid

Argumento "allowedTags" para f:format.stripTags

- El argumento `allowedTags` conteniendo una lista de etiquetas HTML que no serán eliminadas pueden ser ahora usadas en `f:format.stripTags`
- La sintaxis de la lista de etiquetas es idéntica al segundo parámetro de la función PHP `strip_tags` (ver: http://php.net/strip_tags)

Extbase & Fluid

Permitir acceder a ObjectStorage como un array en Fluid

- Crea un alias de `toArray()` permitiendo que el método sea llamado como `getArray()` lo que a cambio permite que el método sea llamado transparentemente desde `ObjectAccess::getPropertyPath`, habilitando el acceso en Fluid y otros lugares
- Creando un aliasing my simple de `toArray()` sobre ObjectStorage, permitiendo que sea llamado como `getArray()`
- Ejemplo: conseguir el 4º elemento

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

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

Capítulo 5:

Funciones Obsoletas/Eliminadas

Funciones Obsoletas/Eliminadas

Miscelánea

- Las siguientes opciones de configuración han sido eliminadas:
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_utils']`
  - `$TYPO3_CONF_VARS['SYS']['t3lib_cs_convMethod']`

  (la funcionalidad es ahora auto-detectada y se usa `mbstring` por defecto si está disponible)
- La propiedad obsoleta TypoScript `page.includeJSlibs` ha sido eliminada. Use en su lugar la propiedad TypoScript `page.includeJSLibs` ("L" mayúscula)
- La opción de TypoScript `config.renderCharset`, que era usada como conjunto de caracteres para conversión interna dentro de una petición frontend, ha sido eliminada

Capítulo 6:

Fuentes y Autores

Fuentes y Autores

Fuentes

Noticias TYPO3:

- http://typo3.org/news

Informaciones de Lanzamiento:

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

Sistema de seguimiento de errores de TYPO3:

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

Repositorios TYPO3 y Fluid:

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

Fuentes y Autores

Equipo Qué hay Nuevo TYPO3 CMS:

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

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