Apéndices

Costasur

Apéndice A
Usuarios

AL hablar de usuarios nos referimos a todo lo relacionado con:

- Login
- Permisos
- Control de Acceso
- Seguridad
- Roles

Para trabajar con estos conceptos se ha generado el UserBundle, y en él, centrarenos todo el codigo que tenga relación con esta entidad.

Para el nuevo Costasur hemos intentado partir desde cero, y desacernos de las limitaciones y del "lastre" que arrastramos de la versión antigua de Costasur. Esto a veces no es del todo posible. En este caso nos econtramos con el caso de la encriptacion de las contraseñas. Para el antiguo costasur se encriptaba usando el sistema md5 y poseriormente sha1. Actualmente existen mejores sistemas de encriptacion, pero nos es imposible utilizarlos sin obligar a los usuarios a que cambien su contraseña.

Hemos solventado este problema creando un servicio que permite codificar utilizando este sistema:

```
/src/Costasur/UserBundle/Resources/Service/Sha1Md5.php
```

A.1 FOSUserBundle

A.2 Colección User

La colección está mapeada de forma similar al resto, si bien tiene una característica especial. Esta colección extiende de otra primaria que nos aporta el FOSUserBundle, de esta forma aportamos algumos métodos que son necesarios para este bundle.

De este Vendor obtenemos varios beneficios:

- Login.
- Ahorro en código.
- Formularios básicos.
- Validacion de formularios.
- Plantillas básicas.
- Gestión de usuarios desde ordenes de consola.

- Controladores: Con las acciones básicas.
- Routing básico: Con URLS asociadas a los controladores y acciones básicas. (monoidioma)
- Internacionalizacion: Con textos básicos traducidos en varios idiomas.

A.3 Symfony

El propio Symfony implementa una gran parte de la seguridad se encarga de gestionar los roles y la herencia dentre ellos. En el se realiza la configuración de todo el sistema de seguridad.

A.4 ACL

Lo considero el sistema mas potente de gestión de persmisos que se puede implementar en Symfony. Para adaptarlo a nuestra base de datos se ha instalado:

```
"iampersistent/mongodb-acl-bundle": "dev-master",
```

Este sistema consiste básicamente en poder asignar persmisos de nivel atomico a:

- Clases
- Metodos
- Objetos
- Instancias de Objetos
- Campos

Cierto que es muy potente, pero tambien es cierto, que no hay mucha gente que lo utilice y por lo pronto tampoco nosotros lo usaremos, las razones son:

- Es complejo y emborra nuestro código las tareas de comprobación de permisos: Una gran parte del código que escribiriamos se la llevaría esta parte.
- Los procesos de eliminación y alta de permisos son tambien complejos y tediosos: En relaciones N a N (p.e.: Cada usuario es propietario), se traduce a un nuevo documento en una colección.
- A todo esto se le une que utiliando *JMSSecurityExtraBundle* (lo explicamos en el siguiente punto) y realizando algunos cambios en el diseño de las colecciones contemplamos un buen sistema de permisos que cumple con nuestros requisitos actuales de forma mas sencilla y eficiente.

A.5 JMSSecurityExtraBundle

Esta extensión viene por defecto en la versión actual de Synfony. Consite básicamente en aplicar seguridad por medio de anotaciones, En las los docbox de cada clase, metodo, decidimos que rol tendra acceso al mismo.

Este bundle está en pleno desarrollo (v 1.2 Febrero de 2013), y para conocerlo lo mejor será visitar su web.

A.5.1 ROLES

Los roles están definidos en el fichero *secutity.yml* del app/config.

Se implementan en forma de herencia. Podemos decir el ROLE "A" hereda del ROLE "B" y por lo tanto pertenecer al ROLE "A" implica tener tambien todos los permismos del ROLE "B".

El usuario anónimo en Sinfomy "está autentificando" y su role es "IS_AUTHENTICATED_ANONYMOUSLY".

En este mismos fichero *security.yml* podemos configurar la protección de acceso por url.

A.6 Proceso de login en barra de navegación

Hemos modificado el formulario de login para utilizar temporalmente el que se encuentra en la barra superior de navegación.

Usando los CSS de bootstrap, se han adaptado las plantillas. Se ha trasladado la lógica de la barra de navegación desde la plantilla del html5bundle hasta la base.twig del WebBundle.

Allí hemos dejado un pequeño ejemplo de control de acceso en el que se muestra un pequeño menú si eres un usuario "ADMINISTRADOR".

A.7 Codigos de Ejemplo

Aqui dejamos algunos ejemplos básicos del uso y la comprobación de los permisos:

```
```

Recoger el nombre del usuario que actualmente está logueado:

```
$usuario = $this->get('security.context')->getToken()->getUser();
```

A.7.1 Trabajo con ROLES

Recoger los ROLES alos que pertenece el usuario logeado NOTA IMPORTANTE: La asignación de roles no se hace efectivo hasta que nos delogeamos y nos volvemos a logear.

```
$roles = $securityContext->getToken()->getRoles();
```

Comprobar si un usuario pertenece a al rol "ROLE_ADMIN" (Explicitamente o de forma heredada)

```
if ($this->get('security.context')->isGranted('ROLE_ADMIN')) { ... }
```

La forma correcta de denegar el acceso por ROLE es la siguiente:

```
if (false === $this->get('security.context')->isGranted('ROLE_ADMIN')) {
    throw new AccessDeniedException();
}
```

Denegar acceso por ROLE desde Twig:

```
{% if is_granted('ROLE_ADMIN') %}
    Texto protegido solo pueden verlos los usuarios con ROLE_ADMIN
{% endif %}
```

Tambien podemos limitar el acceso por ip (dentro del security.yml), añadiendola como si fuera

un rol, ej:

```
- { path: ^/esi, roles: IS_AUTHENTICATED_ANONYMOUSLY, ip: 127.0.0.1 }
```

A.7.2 JMS SecurityExtraBundle

Permite aplicar seguridad a los médotodos usando anotaciones Es importante que no se nos olvide el "use"

```
use JMS\SecurityExtraBundle\Annotation\Secure;
```

```
/**
 * @Secure(roles='ROLE_ADMIN, ROLE_USER')
 */
public function helloAction($name)
{
    // ...
}
```

Con esta opción hacemos una preautorización [php] /** @PreAuthorize("hasRole('ROLE_USER')") */

Apéndice B
Comandos en Symfony

Los comandos en symfony nos permiten ejecutar tareas desde la línea de comandos.

Se ejecutan desde la carpeta raiz del repo (app/console comando) y nos serán muy utiles en un futuro cuando necesitemos hacer tareas periódicas.

Para realizar un comando en Symfony debemos seguir estos pasos:

- Crearnos el comando detro de la carpeta */Command* de algún bundle
- El fichero se puede llamar como queramos pero debe terminar en *Command.php*
- La clase extenderá de *ContainerAwareCommand*
- Diseñamos dos metodos:
    - Configure: Con el nombre del comando, los argumentos, help etc.
    - Execute: Con el código que se ejecutará al lanzar el comando.

Lo mas recomendable será copiar algun comando de los que ya tenemos creados en el sistema

B.1 Comandos Costasur

Cuando creamos un comando para nuestro proyecto debe serguir la nomencladura costasur:Objeto:Acccion. Lo hacemos así para que al ejecutar app/console nos agrupe todos los comandos que hemos creado.

Actualmente están disponibles los comandos:

```
costasur:generate:documentation
```

Se encuentra dentro del DocumentationBundle y sirve para regenerar la documentacion de Costasur.

Si deseamos crear un comando que use argumentos, podemos utilizar este como base.

```
costasur:assets:install
```

Se aloja dentro de ServiceBundle. Y su funcion principal es la de regenerar los assets.

```
costasur:cache:clear
```

Se encuentra en ServiceBundle. Su función principal es borrar la cache en los entornos de producción y dev.

```
costasur:psr:fix
```

Se encuentra en ServiceBundle. Este comando se encarga de "fixear" nuestros ficheros para que cumplan ciertas especificaciones PSR.

Este comando no es mas que un parcheo de otro comando existente en /vendor/bin y podemos tomarlo de ejemplo cuando necesitemos hacer algo parecido.

B.2 Clases de HELP

Dentro de nuestro ServiceBundle, hemos creado dos clases de Help, para usarlas en nuestros comandos:

- PrintFuntions.php: Aqui encontraremos todo lo relacionado con la parte estética de los comandos. En esta clase están los métodos que "pintan" por pantalla.
- HelpFuntions.php: Aqui guardaremos métodos que sean suceptibles de reutilización en otros comandos.

Estas clases se encuentran en:

```
/src/Costasur/ServiceBundle/Command/Helpers
```

El proposito de generar estas clases es el de evitar la duplicidad de código y el de disminuir el número de lineas de nuestras funciones.

B.3 Atajos

Todos los comandos de consola pueden ser acortados, basta con escribir algo como :

```
app/console cos:g:d
```
