Symfony 3

¿Qué hay de nuevo viejo?

PHPSevilla

24 Febrero 2016

Juan Luis García Borrego

Sobre mí

Juan Luis García Borrego

Desarrollador Symfony en Emergya

@JuanluGarciaB

blog.juanluisgarciaborrego.com

Índice

Aprendiendo Symfony

en 10 minutos

Novedades

de Symfony 3

Aprendiendo Symfony
en 10 minutos

Novedades
de Symfony 3

Aprendiendo Symfony

en 10 minutos

Novedades

de Symfony 3

Composer.json

"require": {

“PHP Orientado a Objetos”,

“Inyección de dependencias”,

"Composer",

“Git",

“Ganas de Aprender”

},

Aprendiendo Symfony
en 10 minutos

¿Que es Symfony?

Es un framework PHP para crear aplicaciones.

Está formado por componentes desacoplados reutilizabas que permiten construir mejores aplicaciones PHP, como Drupal, Laravel, Joomla, Magento, Prestashop, etc.

HTTP

Flujo de trabajo

HTTP

HTTP

HTTP

HTTP

HTTP

Request - Response

Symfony

Petición

Respuesta

Petición

Respuesta

Petición

Request

Symfony\Component\HttpFoundation\Request

Respuesta

Petición

Request

Symfony\Component\HttpFoundation\Response

Response

Respuesta

Petición

Request

LOGICA DE TU APP

Response

Respuesta

Symfony
Flujo de trabajo

Request

Routing

Response

#AppBundle/AppBundle/Resources/config/routing.yml

```yaml
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Routing

#AppBundle/AppBundle/Resources/config/routing.yml

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Nombre único de una ruta

Routing

#AppBundle/AppBundle/Resources/config/routing.yml

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Ruta

Routing

#AppBundle/AppBundle/Resources/config/routing.yml

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Nombre del Bundle

Routing

#AppBundle/AppBundle/Resources/config/routing.yml

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Nombre del Controlador

Routing

#AppBundle/AppBundle/Resources/config/routing.yml

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Nombre del método

Routing

Request

Routing

Controller

//tu lógica de negocio

Response

#AppBundle/Controller/PhpSevillaController.php

```
namespace AppBundle\Controller;

use Symfony\Bundle\Framework\Controller\Controller;

class PhpSevillaController extends Controller
{
    public function homeAction($city)
    {
        return $this->render('phpsevilla/homepage.html.twig', ['city' => $city])
    }
}
```

Controller

#AppBundle/Controller/PhpSevillaController.php

```
namespace AppBundle\Controller:

use Symfony\Bundle\Framework\Controller\Controller;

class PhpSevillaController extends Controller
{
    public function homeAction($city)
    {
        return $this->render('phpsevilla/homepage.html.twig', ['city' => $city])
    }
}
```

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Controller

#AppBundle/Controller/PhpSevillaController.php

```
namespace AppBundle\Controller:

use Symfony\Bundle\Framework\Controller\Controller;

class PhpSevillaController extends Controller
{
    public function homeAction($city)
    {
        return $this->render('phpsevilla/homepage.html.twig', ['city' => $city])
    }
}
```

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Controller

#AppBundle/Controller/PhpSevillaController.php

```
namespace AppBundle\Controller:

use Symfony\Bundle\Framework\Controller\Controller;

class PhpSevillaController extends Controller
{
    public function homeAction($city)
    {
        return $this->render('phpsevilla/homepage.html.twig', ['city' => $city])
    }
}
```

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Controller

Controller

Algunas acciones típicas

```php
public function homeAction(Request $request)
{
    #phpsevilla.com/?page=2
    $page = $request->query->get('page');
}
```

Obtener parámetros

```
public function homeAction()
{
    return $this->redirectToRoute('nombre ruta');
}
```

Redireccionar

```
public function homeAction()
{
    return $this->redirectToRoute('php_sevilla');
}
```

```
php_sevilla:
    path:     /hola/{city}
    defaults: { _controller: AppBundle:PhpSevilla:home }
```

Nombre único de una ruta

Redireccionar

```
public function homeAction()
{

    $this->get('mailer');


}
```

Accediendo a servicios

```
public function homeAction()
{
    $this->get('nombreServicio');

    $this->container->get('nombreServicio');
}
```

Accediendo a servicios

```
public function homeAction()
{

    $post = …;


     if (!$post) {
        throw $this->createNotFoundException('message');
     }}

}
```

Manejando errores

```
public function homeAction()
{

    //creando la sesión "bar"
    $this->get('session')->set('bar', 'foo');


    //obteniendo sesión "bar"
    $this->get('session')->get('bar');
}
```

Variables de session

Otras funcionalidades

Request

Routing

Controller

//tu lógica de negocio

Entities

Response

#AppBundle/Entity/Post.php

```php
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Post
 *
 * @ORM\Table(name="post")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\PostRepository")
 */
class Post
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="title", type="string", length=255)
     */
    private $title;
```

Entity

```
#AppBundle\Repository\PostRepository;

namespace AppBundle\Repository;

class ContentRepository extends \Doctrine\ORM\EntityRepository
{
    public function findBySlugIfPostIsPublished($slug)
        {
            return $this->createQueryBuilder('post')
                ->where('post.status = :status and post.slug = :slug')
                ->setParameter('status', true)
                ->setParameter('slug', $slug)
                ->getQuery()
                ->getOneOrNullResult();
        }
}
```

Repository

```
#AppBundle\Repository\PostRepository;

namespace AppBundle\Repository;

class ContentRepository extends \Doctrine\ORM\EntityRepository
{
    public function findBySlugIfPostIsPublished($slug)
        {
            return $this->createQueryBuilder('post')
                ->where('post.status = :status and post.slug = :slug')
                ->setParameter('status', true)
                ->setParameter('slug', $slug)
                ->getQuery()
                ->getOneOrNullResult();
        }
}
```

```
$this->getDoctrine()->getRepository('AppBundle:Post')->findBySlugIfPostIsPublished($slug)
```

Repository

Request

Routing

Controller

//tu lógica de negocio

Entities

Response

Consejo

Antonio Garcia Marin
@antoniogarcia78

Siguiendo

A veces es mejor leerse la ayuda del comando que buscar a lo loco en Stackoverflow. La solucion esta mas cerca de lo que crees.

9:56 - 26 sept. 2014

Symfony documentación

Novedades
Symfony 3

Symfony 3.0 fue lanzado junto a **Symfony 2.8**

Symfony 3.0.0 released

Read release notes

November 30, 2015  Fabien Potencier

Symfony 2.8.0 released

Read release notes

November 30, 2015  Fabien Potencier

Todas las **features** de symfony **3.0**

se **añadieron** en versiones **2.X**

¿Qué hay de **nuevo** en Symfony3 **?**

Symfony 3

elimina la capa de **retrocompatibilidad**

con versiones 2.X.

Toda **funcionalidad marcada** como **obsoleta** en Symfony2 **dejará de funcionar** en la nueva versión.

La **versión PHP mínima**

para Symfony3 es **5.5**

Version	Votes	%
5.3.3	31	2%
5.3.x	56	3%
5.4.x	260	16%
5.5.x	568	35%
5.6.x	495	30%
7.x	225	14%

Symfony 3.4 will be a **long term support** version published in November 2017.

Roadmap

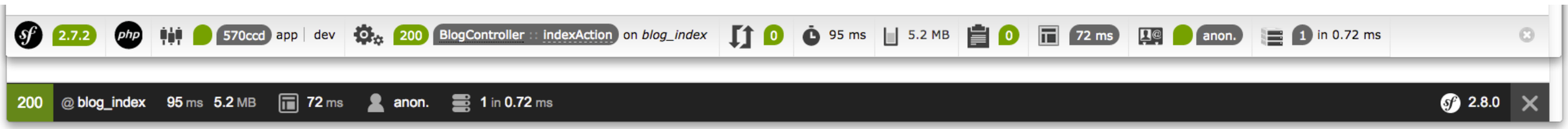
Symfony busca un
entorno agradable para
el **desarrollador**

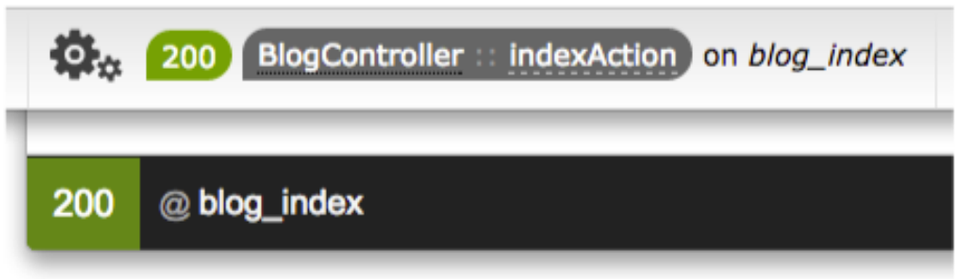
Symfony 2.8

Rediseño del web debug toolbar

OLD light, textured, curved

NEW dark, flat, straight





OLD

NEW

OLD

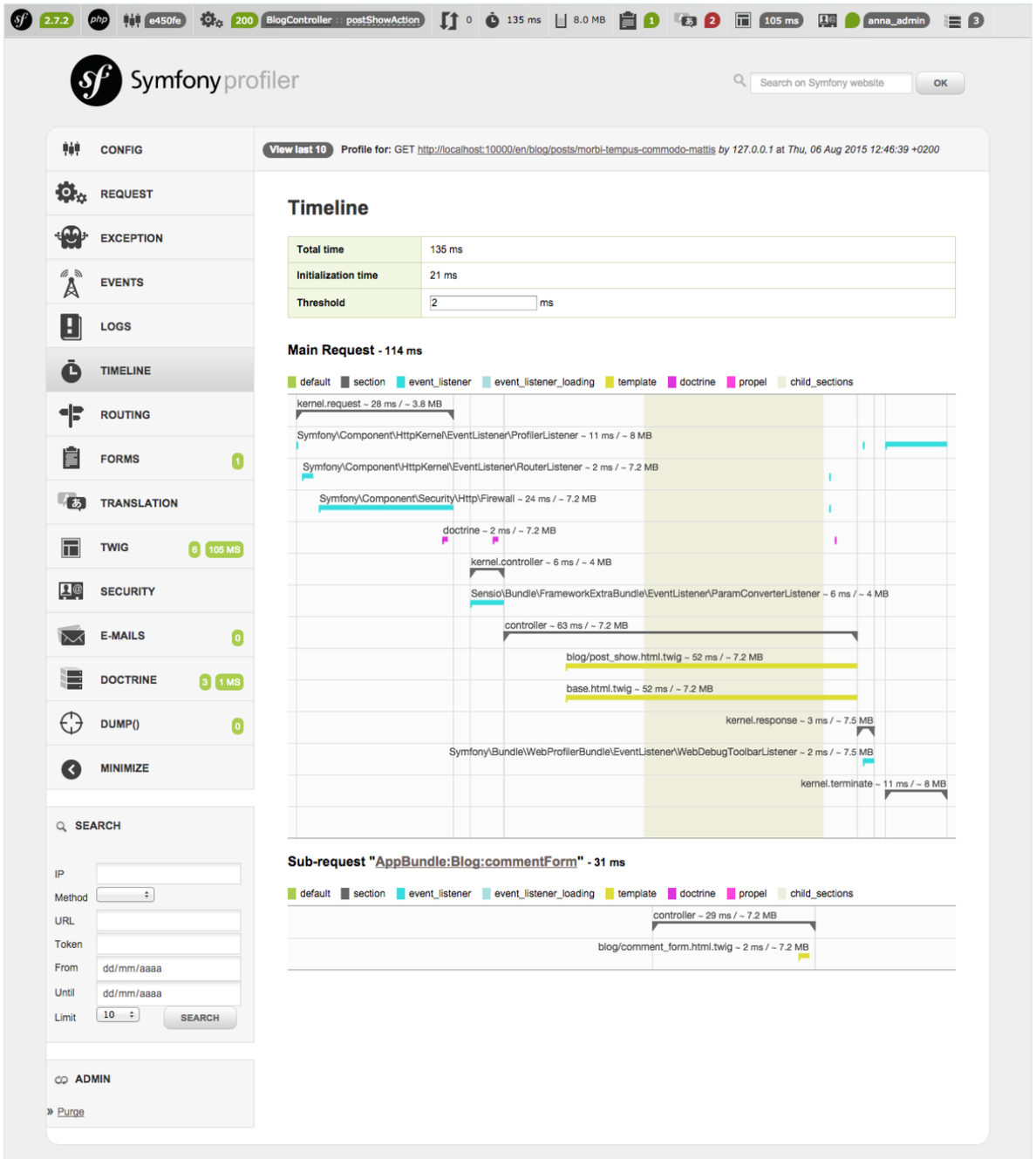
NEW

http://symfony.com/blog/new-in-symfony-2-8-redesigned-web-debug-toolbar

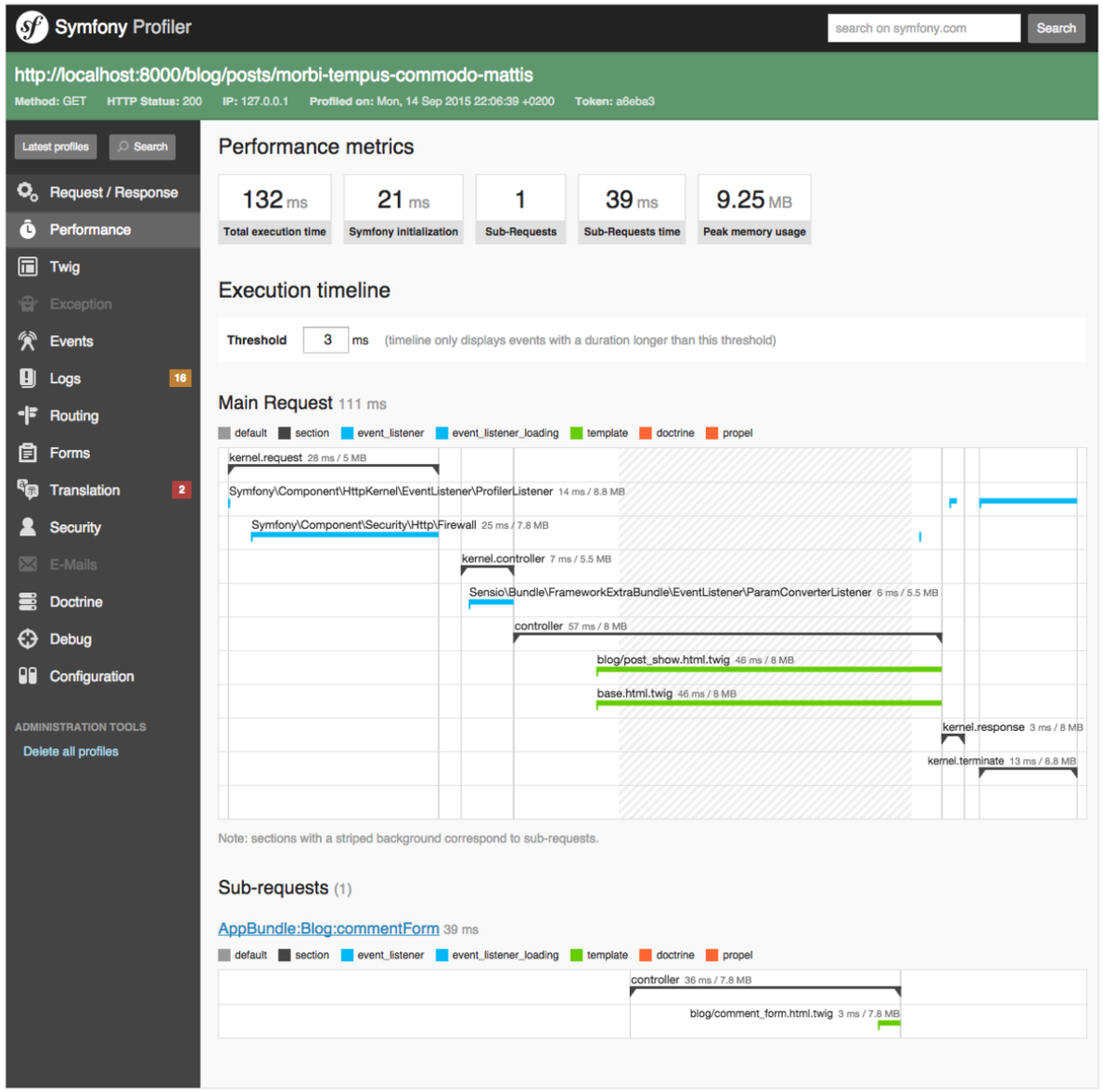
Symfony 2.8

Rediseño del profiler

OLD



NEW



http://symfony.com/blog/new-in-symfony-2-8-redesigned-profiler

Symfony Profiler

Profile Search

Last 10 | Latest | Search

IP

Method: Any

URL

Token

From: dd/mm/aaa

Until: dd/mm/aaa

Results: 10

Search

10 results found

Status	IP	Method	URL
301	127.0.0.1	GET	http://127.0.0.1:8000/_profiler
200	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/categories/
302	127.0.0.1	POST	http://127.0.0.1:8000/es/admin/category/new
200	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/category/new
200	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/categories/
200	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/contents/1/pc
200	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/
401	127.0.0.1	GET	http://127.0.0.1:8000/es/admin/

Symfony 2.8

Mejoras en Console Component

```
Bundle name [AcmeAppBundle]:
> <Enter>

Configuration format (yml, xml, php, annotation) [annotation]:
> <Enter>

Do you want to enable the bundle? (yes/no) [yes]:
> <Enter>

Configuration format (select one) [annotation]:
> yml
> xml
> php
> annotation

! [NOTE] Duis aute irure dolor in reprehenderit in voluptate velit esse
! cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat.

[OK] Lorem ipsum dolor sit amet, consectetur adipisicing elit

[ERROR] Duis aute irure dolor in reprehenderit in voluptate velit esse.

[WARNING] Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi.
```

http://symfony.com/blog/new-in-symfony-2-8-console-improvements
http://symfony.com/blog/new-in-symfony-2-8-console-style-guide

Symfony 2.8

Estructura de directorios

ANTES (<2.7)

```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)

```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)

```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

Toda la carpeta var/ tiene que tener permisos de escritura para el servidor

AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)

```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

Todos los archivos ejecutables binarios estarán en la carpeta bin/

AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)

```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

PHPUnit se puede ejecutar desde la raíz
#Antes: phpunit -c app/
#**Ahora**: phpunit

AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)

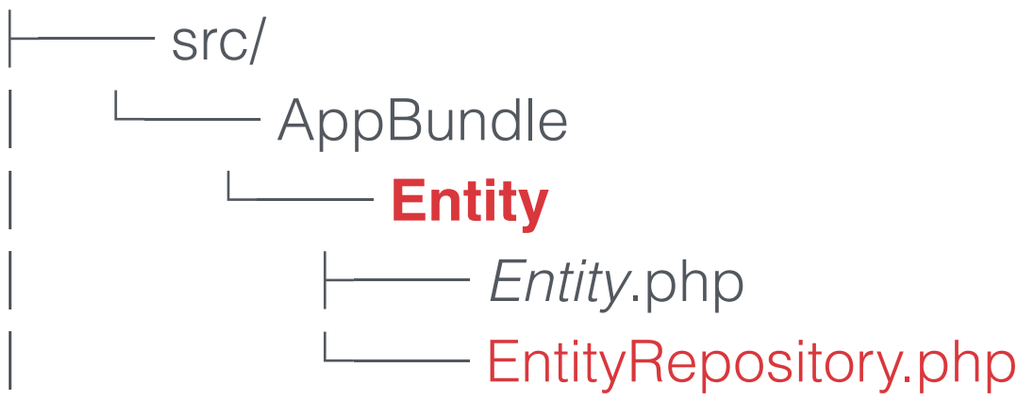
```
├── app/
│   ├── cache/
│   ├── console
│   ├── logs/
│   ├── phpunit.xml.dist
│   └── ...
├── bin/
│   ├── ...
├── src/
│   └── ...
├── vendor/
│   ├── ...
└── web/
    └── ...
```

Para proyectos “acoplados”(AppBundle)

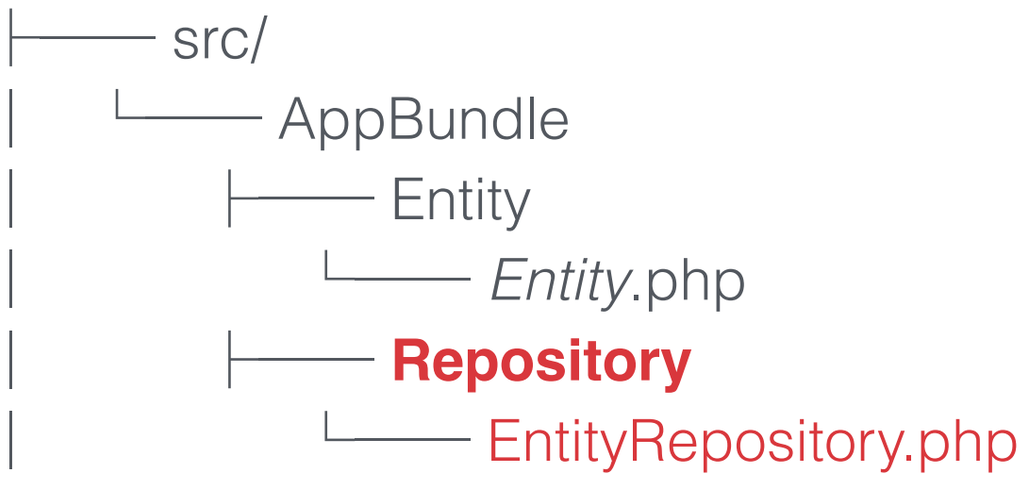
AHORA (>2.8)

```
├── app/
│   └── ...
├── bin/
│   ├── ..
│   └── console
├── src/
│   └── ...
├── phpunit.xml.dist
├── tests/
│   └── ...
├── var/
│   ├── ...
│   ├── cache/
│   └── logs/
├── vendor
│   └── ...
└── web
    └── ...
```

ANTES (<2.7)



AHORA (>2.8)



Los Repositorios de entidades en una carpeta diferente a Entity

Symfony 2.8

Service Auto Wiring

En Symfony podemos configurar nuestros servicios en un archivo y definirle los argumento que le pasamos por el constructor.

http://symfony.com/blog/new-in-symfony-2-8-service-auto-wiring

En Symfony podemos configurar nuestros servicios en un archivo y definirle los argumento que le pasamos por el constructor.

```
#app/config/services.yml
services:
    mi_servicio:
    class: AppBundle/Util/MiServicio
    arguments: ["@session"]
```

http://symfony.com/blog/new-in-symfony-2-8-service-auto-wiring

En Symfony podemos configurar nuestros servicios en un archivo y definirle los argumento que le pasamos por el constructor.

```
#app/config/services.yml
services:
        mi_servicio:
            class: AppBundle/Util/MiServicio
            arguments: [“@session”]


    #AppBundle/Util/MiServicio.php
    class MiServicio {
            private $session;
            public function __construct(Session $session)
            {
              $this->session = $session;
            }
```

http://symfony.com/blog/new-in-symfony-2-8-service-auto-wiring

Gracias a **auto wiring**, podemos omitir en la definición de nuestro servicio los argumentos y el se encargará por nosotros.

#app/config/services.yml

```
services:
    mi_servicio:
        class: AppBundle/Util/MiServicio
        autowire: true
```

http://symfony.com/blog/new-in-symfony-2-8-service-auto-wiring

Symfony 2.8

Symfony as a Microframework

La versión 2.8 incluye la funcionalidad de utilizar Symfony como un micro-framework.

Incluye un trait Symfony\Bundle\FrameworkBundle\Kernel\MicroKernelTrait para crear MicroKernel y poder crear un “mini symfony” en un solo archivo php.

https://github.com/CawaKharkov/symfony-micro

http://symfony.com/blog/new-in-symfony-2-8-symfony-as-a-microframework

http://symfony.es/noticias/2015/11/19/nuevo-en-symfony-28-usando-symfony-como-un-microframework/

Deprecation-Detector

Analiza tu proyecto Symfony2 y busca usos de métodos obsoletos, clases e interfaces.

```
$ deprecation-detector check src/
```

https://github.com/sensiolabs-de/deprecation-detector

~/Sites/proyects/miniSimpleCart

```
+---+-------------------------------------------------------------------------------+------+--
-----------------------------------------------------------------------+
|   | /Users/juanluis/Sites/proyects/miniSimpleCart/src/AppBundle/Form/ProductType.php
                                                                       |
+---+-------------------------------------------------------------------------------+------+--
-----------------------------------------------------------------------+
| 1 | Using deprecated interface Symfony\Component\OptionsResolver\OptionsResolverInterface | 31   | s
ince version 2.6, to be removed in 3.0. Use {@link OptionsResolver} instead. |
| 2 | Overriding deprecated method AppBundle\Form\ProductType->setDefaultOptions()          | 31   | s
ince version 2.7, to be renamed in 3.0.                                |
|   |                                                                               |      |
Use the method configureOptions instead. This method will be           |
|   |                                                                               |      |
added to the FormTypeInterface with Symfony 3.0.                       |
+---+-------------------------------------------------------------------------------+------+--
-----------------------------------------------------------------------+
Finished rendering output.

2 deprecations found.
Checked 5 source files in 16.239 seconds, 23 MB memory used
juanluis@MacBook-Pro-de-Juan-Luis miniSimpleCart (master)*$
```

deprecation-detector

PHPStorm

Symfony Plugin

illa

-sevilla (~/Sites/lab/intro-

Preferences

Other Settings › Symfony Plugin For current project Reset

Enable Plugin for this Project (change need restart) Clean Index Open Help Page

Path to urlGenerator.php: var/cache/dev/appDevUrlGenerator.php ... Default

Translation Root Path var/cache/dev/translations ... Default

App Directory app ... Default

Web Directory web ... Default

- Appearance & Behavior
- Keymap
- Editor
- Plugins
- Version Control
- Project: intro-symfony-php-sev...
- Build, Execution, Deployment
- Languages & Frameworks
- Tools
- Other Settings
  - Git ToolBox
  - GitLab Integration
  - Markdown
  - MultiMarkdown
  - Silex Plugin
  - Symfony Plugin

PhpTypes Resolver

- Container Service
- Repository Entity
- ObjectRepository (find*)
- ObjectManager::find

Code Folding

- Route (PHP)
- Repository Entity (PHP)
- Template (PHP)
- Route (Twig)
- Template (Twig)
- Constant (Twig)

Twig Annotator

- Route
- Template

Php Annotator

- Route
- Template

Cancel Apply OK

Por defecto está en app/

ates: A new version of PhpStorm is available! (2 minutes ago) n/a n/a Git: master Git Stat: no remote

Symfony Installer

Actualiza el instalador de symfony antes de instalar la nueva versión

```
# Linux, Mac OS X
$ symfony self-update

# Windows
c:\> php symfony self-update
```

https://github.com/symfony/symfony-installer

Symfony 2.8

Formularios

Al crear un campo de formulario utilizábamos:

$builder->add('propiedad', '**tipoPropiedad**');

$builder->add('title', '**text**');

form.type.text

http://symfony.com/blog/new-in-symfony-2-8-form-improvements

Al crear un campo de formulario utilizábamos:

$builder->add('propiedad', '**tipoPropiedad**');

$builder->add('title', '**text**');

form.type.text

Al hacer uso en un controlador de un formulario tipo:

$form = $this->createForm(**new PostType()**, ..);

http://symfony.com/blog/new-in-symfony-2-8-form-improvements

Resolución de nombres de clases mediante ::class

Permite usar NombreClase::class para obtener el nombre de la clase .

TIPOS DE CAMPOS

```
use Symfony\Component\Form\Extension\Core\Type\TextType;

$builder->add('title', 'TextType::class');
```

PHP 5.5

Resolución de nombres de clases mediante ::class

Permite usar NombreClase::class para obtener el nombre de la clase .

FORMULARIO TIPO

```
use AppBundle\Type\PostType;

$form = $this->createForm(PostType::class, ..)
```

```
class PostType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('username', TextType)
    }


    public function configureOptions(OptionsResolver $resolver) { . . . }

    public function getName()
    {
        return 'app_form_post';
    }
}
```

¿ ?

PostType

```
class PostType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('username', TextType)
    }


    public function configureOptions(OptionsResolver $resolver) { . . . }
}
```

PostType

Pregunta

¿Cómo se denomina al operador de ámbito ::?

¿Cómo se denomina al operador de ámbito ::?

Paamayim Nekudotayim

En realidad, significa doble dos-puntos - en Hebreo.

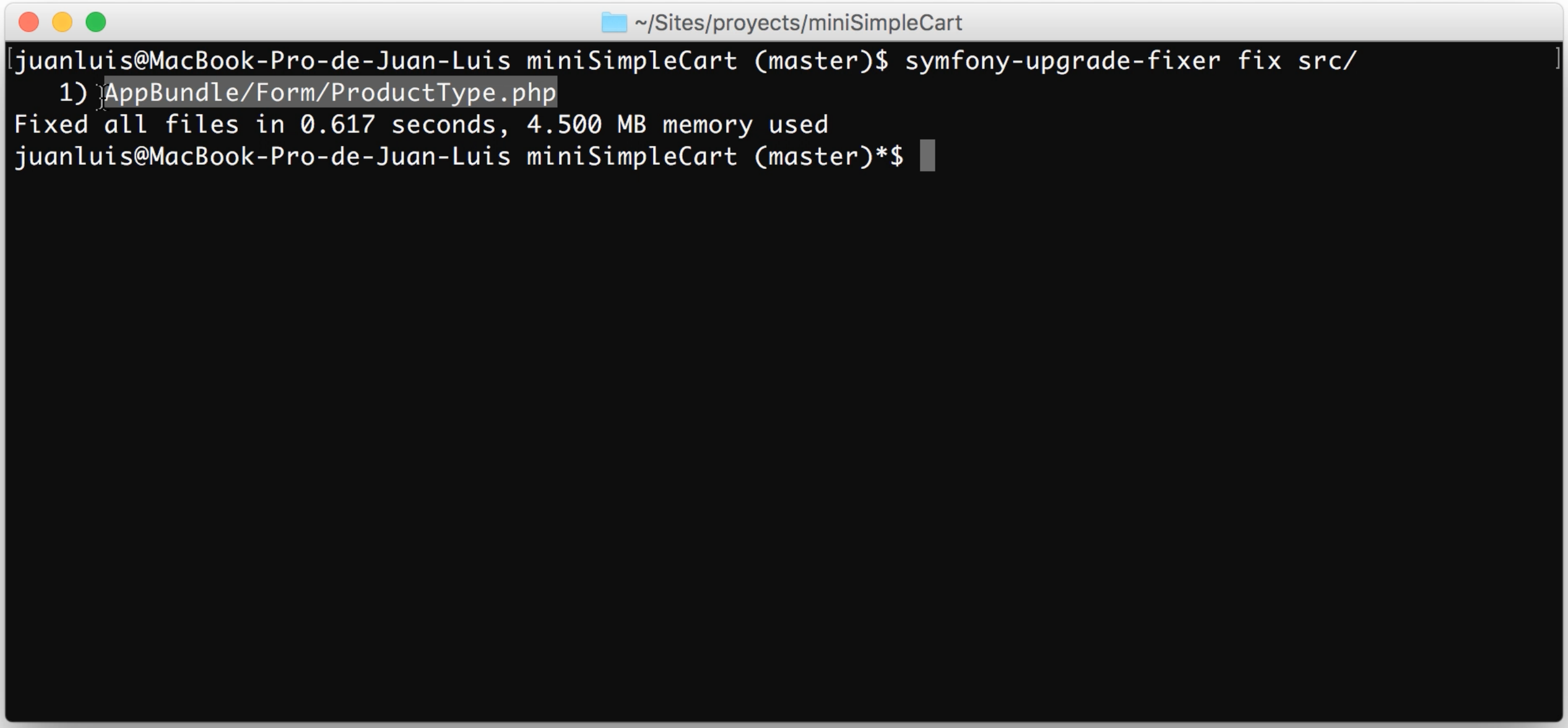
http://php.net/manual/es/language.oop5.paamayim-nekudotayim.php

Symfony-Upgrade-Fixer

Analiza tu proyecto Symfony e intenta hacerlo compatible con la nueva versión.

```
$ symfony-upgrade-fixer fix /path/to/dir
$ symfony-upgrade-fixer fix /path/to/file
```

https://github.com/umpirsky/Symfony-Upgrade-Fixer



symfony-upgrade-fixer

Symfony 2.8

Guard authentication component

Un nuevo componente de seguridad que tiene como objetivo simplificar la tarea de autenticación.

Se basa en la creación de una sola clase que implementa GuardAuthenticatorInterface.

Permite personalizar y utilizar diferentes maneras de autenticación como necesitemos (Form login, token API, Facebook, Twitter u otro tipo de autenticación OAuth).

http://symfony.com/blog/new-in-symfony-2-8-guard-authentication-component

Autenticación

Sin Guard Component

```yaml
security:

    firewalls:
        mi_area_segura:
            anonymous: ~
            provider: tuProvider
            form_login:
                login_path: /login #AppBundle\Controller\SecurityControlller::login
                check_path: /login-check #AppBundle\Controller\SecurityControlller::logout
    logout:
        path: /logout
        target: /
```

App/config/security.yml

Autenticación

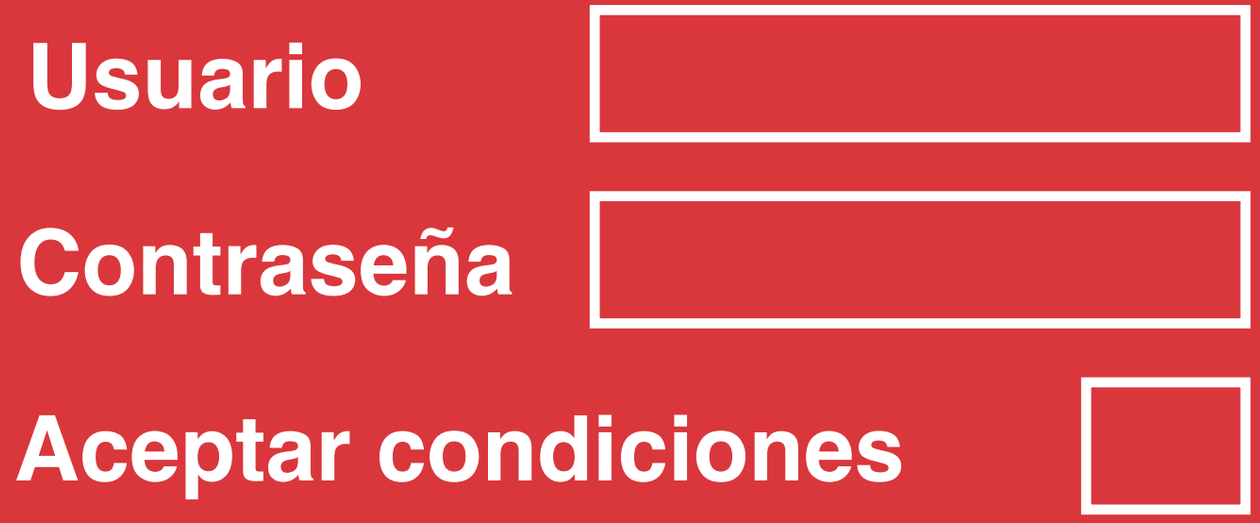
Con Guard Component

PASOS A SEGUIR

1. Creamos una clase que implemente GuardAuthenticatorInterface.
   *[**AppBundle/Security**/MiFormulario.php]*
2. Creamos un servicio de nuestra clase.
3. En *security.yml* sustituimos form_login por nuestro servicio

Formulario Login

Con Guard Component



PASOS A SEGUIR

1. Creamos una clase que implemente GuardAuthenticatorInterface.
   *[**AppBundle/Security**/MiFormulario.php]*
2. Creamos un servicio de nuestra clase.
3. En *security.yml* sustituimos form_login por nuestro servicio

```
class MiFormulario extends AbstractFormLoginAuthenticator
{
```

Implementa **GuardAuthenticatorInterface**

```
    public function getCredentials(Request $request)
    {}

    public function getUser($credentials, UserProviderInterface $userProvider)
    {}

    public function checkCredentials($credentials, UserInterface $user)
    {}

    protected function getLoginUrl()
    {}

    protected function getDefaultSuccessRedirectUrl()
    {}
}
```

src/AppBundle/Security/MiFormulario.php

Devolvemos la ruta del formulario de login.

```
protected function getLoginUrl()
{
    return $this->router->generate('login_route');
}
```

src/AppBundle/Security/MiFormulario.php

```php
// Devolvemos la ruta del formulario de login.
protected function getLoginUrl()
{
    return $this->router->generate('login_route');
}

// Devolvemos el destino después de autenticarse
protected function getDefaultSuccessRedirectUrl()
{
    return $this->router->generate('login_admin');
}
```

src/AppBundle/Security/MiFormulario.php

getCredentials es llamado en cada petición y su trabajo consiste en recibir las credenciales y devolverlas a getUser.

```
public function getCredentials(Request $request)
{
    if ($request->getPathInfo() != '/login-check')) {
        return;
    }

    return [
        'username' => $request->request->get('_username'),
        'password' => $request->request->get('_password'),
        'condiciones' => $request->request->get('_condiciones'),
    ];
}
```

src/AppBundle/Security/MiFormulario.php

Su objetivo es devolver un objeto que implemente UserInterface.
En caso afirmativo checkCredentials será llamado, si no lanza

```
public function getUser($credentials, UserProviderInterface
$userProvider)
{
    $username = $credentials['username'];

    return $userProvider->loadUserByUsername($username);

}
```

src/AppBundle/Security/MiFormulario.php

Si getUser devuelve un User entonces verificamos las credenciales.

```
public function checkCredentials ($credentials, UserInterface $user)
{

if (!$this->encoder->isPasswordValid($user, $credentials['password']))
{
    throw new BadCredentialsException();
}
```

```
if (!$credentials['condiciones']) {
     throw new CustomUserMessageAuthenticationException(
       'Eh ‘.$credentials['username'].'Debes aceptar para entrar’
   );
 }
    return true;
}
```

src/AppBundle/Security/MiFormulario.php

PASOS A SEGUIR

1. Creamos una clase que implemente GuardAuthenticatorInterface.
   *[**AppBundle/Security**/MiFormulario.php]*
2. Creamos un servicio de nuestra clase.
3. En *security.yml* sustituimos form_login por nuestro servicio

```yaml
services:

    mi_formulario_de_login:
        class: AppBundle\Security\MiFormulario
        arguments: ["@security.password_encoder", "@router"]
```

App/config/services.yml

PASOS A SEGUIR

1. Creamos una clase que implemente GuardAuthenticatorInterface.
   *[**AppBundle/Security**/MiFormulario.php]*
2. Creamos un servicio de nuestra clase.
3. En *security.yml* sustituimos form_login por nuestro servicio

```yaml
security:

    firewalls:
            ...
            guard:
                authenticators:
                    - mi_formulario_de_login
```

App/config/security.yml

Puedes ver el ejemplo en

GitHub

/JuanLuisGarciaBorrego/SecurityGuardLab

Un pequeño CMS multiidioma creado en Symfony3

Jedy

/JuanLuisGarciaBorrego/jedy

Y eso es todo amigos!

Gracias

blog.juanluisgarciaborrego.com

@JuanluGarciaB