



symfony

symfony as a platform

Fabien Potencier
<http://www.symfony-project.com/>
<http://www.sensiolabs.com/>

symfony 1.1

- A lot of internal refactoring... really a lot
- Some new features... but not a lot
 - Tasks are now classes
 - More cache classes: APC, Eaccelerator, File, Memcache, SQLite, XCache
 - Configurable Routing: Pattern, PathInfo
 - ... and some more
- Yes, we will have a new form/validation system... but we still need some internal refactoring

Singletons

1.0

sfConfigCache
sfContext
sfRouting
sfWebDebug
sfI18N
sfLogger

1.1

sfConfigCache
sfContext

Remove
Singletons

Why?

```
// somewhere in the backend application
$frontendRouting = sfContext::getInstance('frontend')->getRouting();

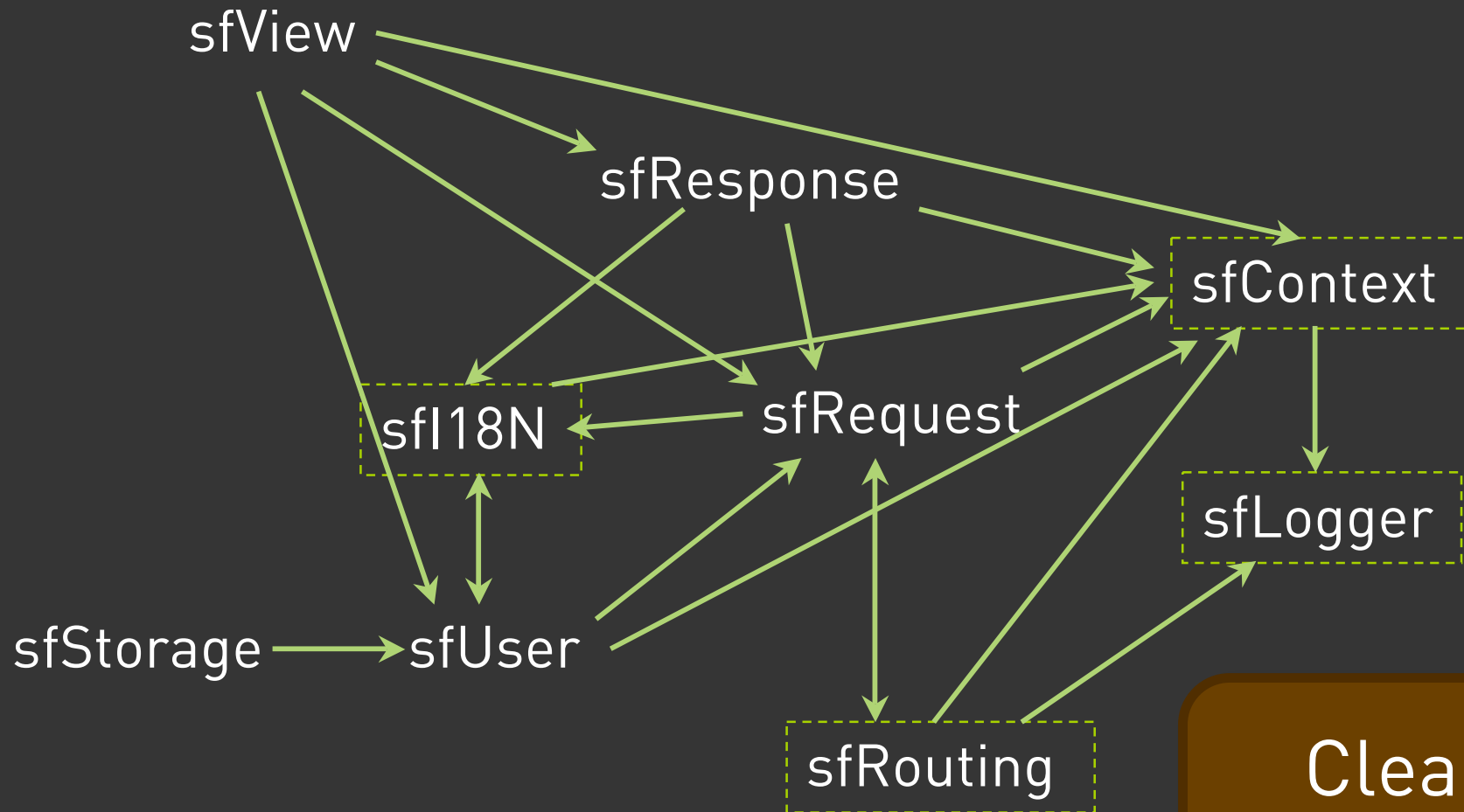
$url = $frontendRouting->generate('article', array('id' => 1));
```

```
// somewhere in a functional test
$fabien = new sfTestBrowser('frontend');
$thomas = new sfTestBrowser('frontend');

$fabien->get('/talk_to/thomas/say/Hello');
$thomas->get('/last_messages');
$thomas->checkResponseElement('li', '/Hello/i');
```

Don't try this at home... it won't work... yet

symfony 1.0 Dependencies

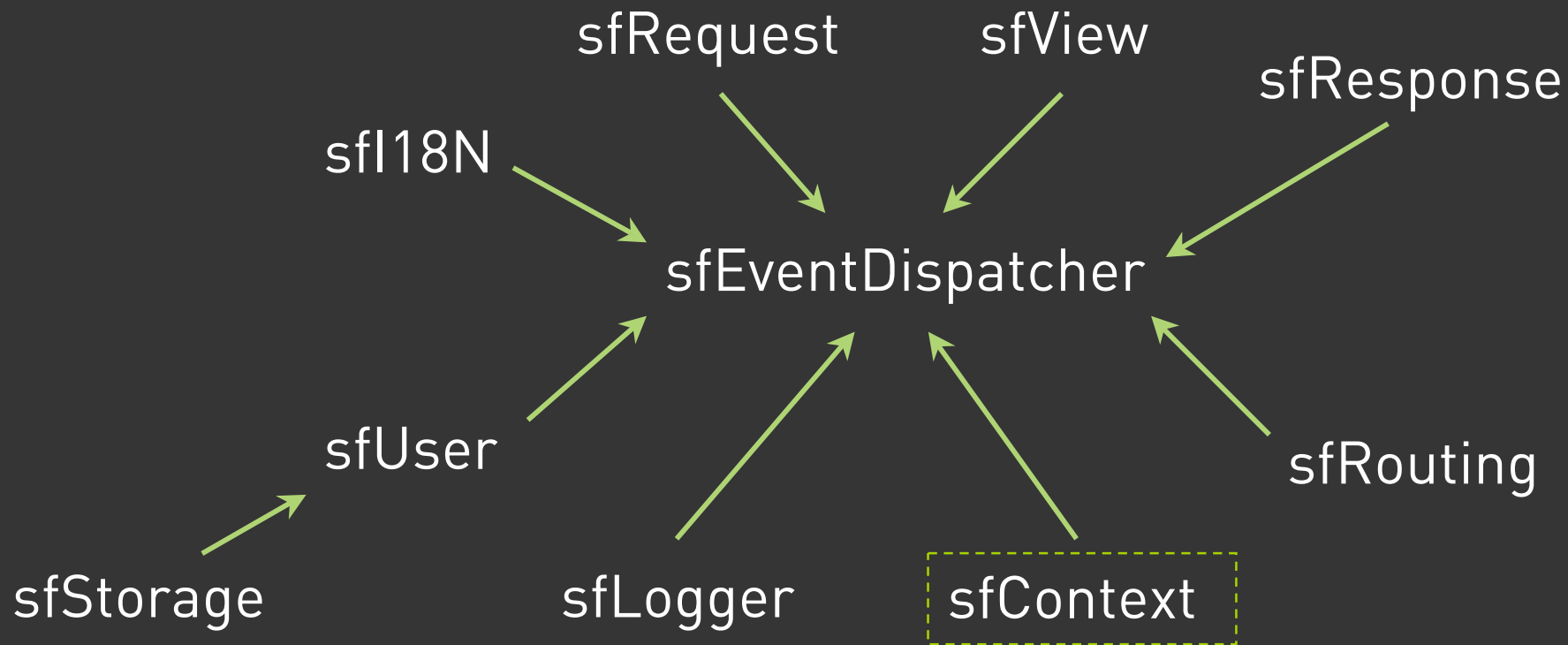


Cleanup
Dependencies

singleton

dependency

symfony 1.1 Dependencies



Cleanup
Dependencies

singleton

dependency

sfEventDispatcher

- Based on Cocoa Notification Center

```
// sfUser  
$event = new sfEvent($this, 'user.change_culture', array('culture' => $culture));  
$dispatcher->notify($event);
```

```
// sfI18N  
$callback = array($this, 'listenToChangeCultureEvent');  
$dispatcher->connect('user.change_culture', $callback);
```

- sfI18N and sfUser are decoupled
- « Anybody » can listen to any event
- You can notify existing events or create new ones

Why?

Let's talk about symfony 2.0...

The Web Workflow

The **User** asks a **Resource** in a Browser

The Browser sends a **Request** to the Server

The Server sends back a **Response**

The Browser displays the **Resource** to the **User**

```
session_start();
```

```
if (isset($_GET['name']))
```

```
{
```

```
    $name = $_GET['name'];
```

```
}
```

```
else if (isset($_SESSION['name']))
```

```
{
```

```
    $name = $_SESSION['name'];
```

```
}
```

```
else
```

```
{
```

```
    $name = 'World';
```

```
}
```

```
$_SESSION['name'] = $name;
```

```
echo 'Hello ' . $name;
```

User



Resource



Request



Response

sfWebRequest

PHP

```
$_SERVER['REQUEST_METHOD']  
$_GET['name']
```

```
get_magic_quotes_gpc() ?  
stripslashes($_COOKIE[$name]) : $_COOKIE[$name];
```

```
{  
  isset($_SERVER['HTTPS'])  
  && {  
    strtolower($_SERVER ['HTTPS']) == 'on'  
    ||  
    strtolower($_SERVER ['HTTPS']) == 1  
  }  
  || {  
    isset($_SERVER ['HTTP_X_FORWARDED_PROTO'])  
    &&  
    strtolower($_SERVER ['HTTP_X_FORWARDED_PROTO']) == 'https'  
  }  
}
```

Abstract
Parameters
Headers

Object

```
→getMethod()  
→getParameter('name')
```

```
→getCookie('name')
```

```
→isSecure()
```

Mock sfWebRequest
to test
without a HTTP layer

```
require dirname(__FILE__).'/../symfony_platform.php';

$dispatcher = new sfEventDispatcher();
$request = new sfWebRequest($dispatcher);

session_start();

if ($request->hasParameter('name'))
{
    $name = $request->getParameter('name');
}
else if (isset($_SESSION['name']))
{
    $name = $_SESSION['name'];
}
else
{
    $name = 'World';
}

$_SESSION['name'] = $name;

echo 'Hello ' . $name;
```

User



Resource



Request



Response

sfWebResponse

PHP

```
echo 'Hello World!'
```

```
header('HTTP/1.0 404 Not Found')
```

```
setcookie('name', 'value')
```

Abstract
Headers
Cookies

Object

```
→setContent('Hello World')
```

```
→setStatusCode(404)
```

```
→setCookie('name', 'value')
```

Mock sfWebResponse
to test
without a HTTP layer

```
require dirname(__FILE__).'/../symfony_platform.php';
```

```
$dispatcher = new sfEventDispatcher();  
$request = new sfWebRequest($dispatcher);  
session_start();
```

```
if (!$name = $request->getParameter('name'))  
{  
    if (isset($_SESSION['name']))  
    {  
        $name = $_SESSION['name'];  
    }  
    else  
    {  
        $name = 'World';  
    }  
}
```

```
$_SESSION['name'] = $name;
```

```
$response = new sfWebResponse($dispatcher);  
$response->setContent('Hello '.$name);  
$response->send();
```

User



Resource



Request



Response

sfUser / sfStorage

PHP

```
$_SESSION['name'] = 'value'
```

Object

```
→setAttribute('name', 'value')
```

```
→setCulture('fr')
```

```
→setAuthenticated(true)
```

Native session storage +
MySQL, PostgreSQL, PDO, ...

Abstract
\$_SESSION
Add features

Mock sfUser
to test
without a HTTP layer

```
require dirname(__FILE__).'/../symfony_platform.php';
```

```
$dispatcher = new sfEventDispatcher();
```

```
$storage = new sfSessionStorage();
```

```
$user = new sfUser($dispatcher, $storage);
```

```
$request = new sfWebRequest($dispatcher);
```

```
if (!$name = $request->getParameter('name'))
```

```
{
```

```
    if (!$name = $user->getAttribute('name'))
```

```
    {
```

```
        $name = 'World';
```

```
    }
```

```
}
```

```
$user->setAttribute('name', $name);
```

```
$response = new sfWebResponse($dispatcher);
```

```
$response->setContent('Hello '.$name);
```

```
$response->send();
```

User



Resource



Request



Response

```
require dirname(__FILE__). '/../symfony_platform.php';
```

```
$dispatcher = new sfEventDispatcher();
```

```
$storage = new sfSessionStorage();
```

```
$user = new sfUser($dispatcher, $storage);
```

```
$request = new sfWebRequest($dispatcher);
```

```
$name = $request->getParameter('name', $user->getAttribute('name', 'World'));
```

```
$user->setAttribute('name', $name);
```

```
$response = new sfWebResponse($dispatcher);
```

```
$response->setContent('Hello '.$name);
```

```
$response->send();
```

User



Resource



Request



Response

sfRouting

- Clean URLs \leftrightarrow Resources
- Parses `PATH_INFO` to inject parameters in the `sfRequest` object
- Several strategies: `PathInfo`, `Pattern`
- Decouples Request and Controller

```
require dirname(__FILE__). '/../symfony_platform.php';

$dispatcher = new sfEventDispatcher();
$storage = new sfSessionStorage();
$user = new sfUser($dispatcher, $storage);
$routing = new sfPatternRouting($dispatcher);
$routing->connect('hello', '/hello/:name');
$request = new sfWebRequest($dispatcher);

$name = $request->getParameter('name', $user->getAttribute('name', 'World'));

$user->setAttribute('name', $name);

$response = new sfWebResponse($dispatcher);
$response->setContent('Hello '.$name);
$response->send();
```

User



Resource



Request



Response

The Web Workflow

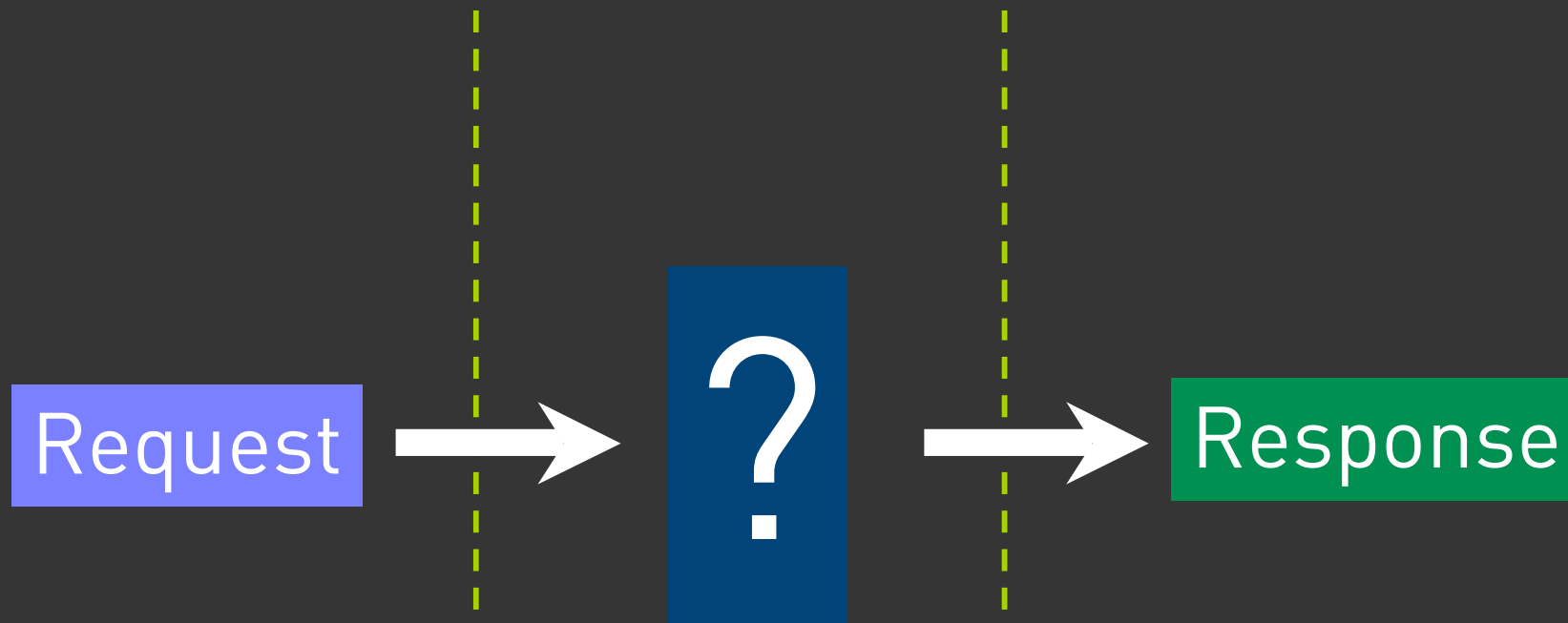
The **User** asks a **Resource** in a Browser

The Browser sends a **Request** to the Server

The Server sends back a **Response**

The Browser displays the **Resource** to the **User**

symfony 2.0 philosophy



symfony gives
you a `sfRequest`

symfony wants
a `sfResponse`

Let's create symfony 2.0...

The Controller

- The dispatching logic is the same for every resource
- The business logic depends on the resource and is managed by the controller
- The controller responsibility is to « convert » the request to the response

```
require dirname(__FILE__). '/../symfony_platform.php';
```

```
$dispatcher = new sfEventDispatcher();
```

```
$storage = new sfSessionStorage();
```

```
$user = new sfUser($dispatcher, $storage);
```

```
$routing = new sfPatternRouting($dispatcher);
```

```
$routing->connect('hello', '/hello/:name',  
    array('controller' => 'hello', 'action' => 'index'));
```

```
$request = new sfWebRequest($dispatcher);
```

```
$class = $request->getParameter('controller').'Controller';
```

```
$method = $request->getParameter('action').'Action';
```

```
$controller = new $class();
```

```
$response = $controller->$method($dispatcher, $request, $user);
```

```
$response->send();
```

```
class helloController
{
    function indexAction($dispatcher, $request, $user)
    {
        $name = $request->getParameter('name', $user->getAttribute('name', 'World'));

        $user->setAttribute('name', $name);

        $response = new sfWebResponse($dispatcher);
        $response->setContent('Hello '.$name);

        return $response;
    }
}
```

The RequestHandler

- Handles the dispatching of the request
- Calls the Controller
- Has the responsibility to return a `sfResponse`

```
require dirname(__FILE__).'/../symfony_platform.php';
require dirname(__FILE__).'/step_07_framework.php';

$dispatcher = new sfEventDispatcher();
$storage = new sfSessionStorage();
$user = new sfUser($dispatcher, $storage);
$routing = new sfPatternRouting($dispatcher);
$routing->connect('hello', '/hello/:name',
    array('controller' => 'hello', 'action' => 'index'));
$request = new sfWebRequest($dispatcher);

$response = RequestHandler::handle($dispatcher, $request, $user);

$response->send();
```

```
class RequestHandler
{
    function handle($dispatcher, $request, $user)
    {
        $class = $request->getParameter('controller').'Controller';
        $method = $request->getParameter('action').'Action';

        $controller = new $class();
        try
        {
            $response = $controller->$method($dispatcher, $request, $user);
        }
        catch (Exception $e)
        {
            $response = new sfWebResponse($dispatcher);
            $response->setStatusCode(500);
            $response->setContent(sprintf('An error occurred: %s', $e->getMessage()));
        }

        return $response;
    }
}
```

I WANT a sfResponse

```
try
{
    if (!class_exists($class))
    {
        throw new Exception('Controller class does not exist');
    }

    $controller = new $class();

    if (!method_exists($class, $method))
    {
        throw new Exception('Controller method does not exist');
    }

    $response = $controller->$method($dispatcher, $request, $user);

    if (!is_object($response) || !$response instanceof sfResponse)
    {
        throw new Exception('Controller must return a sfResponse object');
    }
}
catch (Exception $e)
{
    $response = new sfWebResponse($dispatcher);
    $response->setStatusCode(500);
    $response->setContent(sprintf('An error occurred: %s', $e->getMessage()));
}
```

The Application

```
$response = $controller->$method($dispatcher, $request, $user);
```

- I need a container for my application objects
 - The dispatcher
 - The user
 - The routing
 - The I18N
 - The Database
 - ...
- These objects are specific to my Application and does not depend on the request

```
abstract class Application
{
    public $dispatcher, $user, $routing;

    function __construct($dispatcher)
    {
        $this->dispatcher = $dispatcher;

        $this->configure();
    }

    abstract function configure();

    function handle($request)
    {
        return RequestHandler::handle($this, $request);
    }
}
```

```
class HelloApplication extends Application
{
    function configure()
    {
        $storage = new sfSessionStorage();
        $this->user = new sfUser($this->dispatcher, $storage);

        $this->routing = new sfPatternRouting($this->dispatcher);
        $this->routing->connect('hello', '/hello/:name',
            array('controller' => 'hello', 'action' => 'index'));
    }
}

$dispatcher = new sfEventDispatcher();
$request = new sfWebRequest($dispatcher);

$application = new HelloApplication($dispatcher);
$response = $application->handle($request);

$response->send();
```

Sensible defaults

- Most of the time
 - The dispatcher is a `sfEventDispatcher`
 - The request is a `sfWebRequest` object
 - The controller must create a `sfWebResponse` object
- Let's introduce a new `Controller` abstract class
- Modify the `Application` to takes defaults

```
class Controller
{
    public $application;

    function __construct($application)
    {
        $this->application = $application;
    }

    function renderToResponse($content)
    {
        $response = new sfWebResponse($this->application->dispatcher);
        $response->setContent($content);

        return $response;
    }
}

class helloController extends Controller
{
    function indexAction($request)
    {
        $name = $request->getParameter('name', $this->application->user->getAttribute('name', 'World'));

        $this->application->user->setAttribute('name', $name);

        return $this->renderToResponse('Hello '.$name);
    }
}
```

```
abstract class Application
{
    function __construct($dispatcher = null)
    {
        $this->dispatcher = is_null($dispatcher) ? new sfEventDispatcher() : $dispatcher;

        $this->configure();
    }

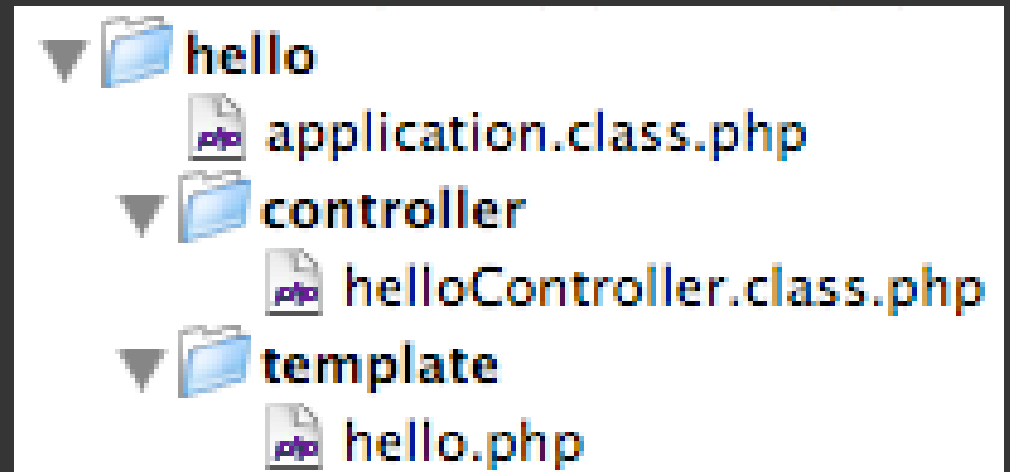
    function handle($request = null)
    {
        $request = is_null($request) ? new sfWebRequest($this->dispatcher) : $request;

        return RequestHandler::handle($this, $request);
    }
}
```

Let's have a look at the code

The directory structure

- A directory for each application
 - An application class
 - A controller directory
 - A template directory



```
class HelloApplication extends Application
{
    function configure()
    {
        $storage = new sfSessionStorage();
        $this->user = new sfUser($this->dispatcher, $storage);

        $this->routing = new sfPatternRouting($this->dispatcher);
        $this->routing->connect('hello', '/hello/:name',
            array('controller' => 'hello', 'action' => 'index'));

        $this->template = new Template();

        $this->root = dirname(__FILE__);
    }
}
```

```
class Template
{
    function render($template, $parameters = array())
    {
        extract($parameters);

        ob_start();
        require $template;

        return ob_get_clean();
    }
}

class Controller
{
    ...
    function renderToResponse($templateName, $parameters = array())
    {
        $content = $this->application->template->render($templateName, $parameters);

        $response = new sfWebResponse($this->application->dispatcher);
        $response->setContent($content);

        return $response;
    }
}
```

Customization

- Add some events in the RequestHandler
 - application.request
 - application.controller
 - application.response
 - application.exception
- They can return a sfResponse and stop the RequestHandler flow

```
$parameters = array('request' => $request);  
  
$event = new sfEvent($application, 'application.request', $parameters);  
  
$application->dispatcher->notifyUntil($event);  
  
if ($event->isProcessed())  
{  
    return $event->getReturnValue();  
}
```

What for?

- Page caching
 - `application.request`: If I have the page in cache, unserialize the response from the cache and returns it
 - `application.response` : Serialize the response to the cache
- Security
 - `application.controller`: Check security and change the controller if needed

```
class HelloApplication extends Application
{
    function configure()
    {
        $storage = new sfSessionStorage();
        $this->user = new sfUser($this->dispatcher, $storage);

        $this->routing = new sfPatternRouting($this->dispatcher);
        $this->routing->connect('hello', '/hello/:name',
            array('controller' => 'hello', 'action' => 'index'));

        $this->template = new Template();

        $this->root = dirname(__FILE__);

        CacheListener::register($dispatcher, $this->root.'/cache/cache.db');
    }
}
```

```

class CacheListener
{
    public $cache, $key;

    static function register($dispatcher, $database)
    {
        $listener = new __CLASS__();
        $listener->cache = new sfSQLiteCache(array('database' => $database));
        $dispatcher->connect('application.request', array($listener, 'listenToRequest'));
        $dispatcher->connect('application.response', array($listener, 'listenToResponse'));
    }

    function listenToRequest($event)
    {
        $this->key = md5($event->getParameter('request')->getPathInfo());

        if ($cache = $this->cache->get($this->key))
        {
            $this->setProcessed(true);
            $this->setReturnValue(unserialize($cache));
        }
    }

    function listenToResponse($event, $response)
    {
        $this->cache->set($this->key, serialize($response));

        return $response;
    }
}

```

Next symfony Workshops

En français : Paris, France - Dec 05, 2007

In English : Paris, France - Feb 13, 2008

More info on www.sensiolabs.com

Join Us

- Sensio Labs is recruiting in France
 - project managers
 - web developers
- You have a passion for the web?
 - **Web Developer** : You have a minimum of 3 years experience in web development with Open-Source projects and you wish to participate to development of Web 2.0 sites using the best frameworks available.
 - **Project Manager** : You have more than 5 years experience as a developer and/or a project manager and you want to manage complex Web projects for prestigious clients.

SENSIO S.A.

26, rue Salomon de Rothschild
92 286 Suresnes Cedex
FRANCE

Tél. : +33 1 40 99 80 80

Fax : +33 1 40 99 83 34

Contact

Fabien Potencier
fabien.potencier@sensio.com

<http://www.sensiolabs.com/>

<http://www.symfony-project.com/>