Controllers

Bien qu'un contrôleur puisse être une classe normale, il est conseillé d'étendre la classe Controller ou AbstractController, pour bénéficier des méthodes utilitaires définies dans ces classes.

Classe de base

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class LuckyController extends Controller
{
    // ...
}
```

Opérations courantes

Génération d'urls

```
$url = $this->generateUrl('app_lucky_number', array('max' => 10));
```

Redirections

```
use Symfony\Component\HttpFoundation\RedirectResponse;

// ...
public function index()
{
    // redirect to the "homepage" route
    return $this->redirectToRoute('homepage');

    // redirectToRoute is a shortcut for:
    // return new RedirectResponse($this->generateUrl('homepage'));

    // do a permanent - 301 redirect
    return $this->redirectToRoute('homepage', array(), 301);

    // redirect to a route with parameters
    return $this->redirectToRoute('app_lucky_number', array('max' => 10));

    // redirect externally
    return $this->redirect('http://symfony.com/doc');
```

```
}
```

Appel d'un template

```
// renders templates/lucky/number.html.twig
return $this->render('lucky/number.html.twig', array('name' => $name));
```

Appel de services

L'utilisation d'un service peut se faire par injection de dépendances :

```
use Psr\Log\LoggerInterface
// ...

/**
 * @Route("/lucky/number/{max}")
 */
public function number($max, LoggerInterface $logger)
{
    $logger->info('We are logging!');
    // ...
}
```

La commande suivante liste les services directement injectables :

```
php bin/console debug:autowiring
```

L'objet Request

L'objet **request** est utilisable par injection de dépendance :

```
use Symfony\Component\HttpFoundation\Request;

public function index(Request $request, $firstName, $lastName)
{
    $page = $request->query->get('page', 1);

    // ...
}
```

ou en utilisant la méthode statique :

```

```

```
$request = Request::createFromGlobals();
```

L'objet Session

L'utilisation de la session se fait par injection de dépendance d'une instance de **SessionInterface** :

```
use Symfony\Component\HttpFoundation\Session\SessionInterface;

public function index(SessionInterface $session){
    // store an attribute for reuse during a later user request
    $session->set('foo', 'bar');

    // get the attribute set by another controller in another request
    $foobar = $session->get('foobar');

    // use a default value if the attribute doesn't exist
    $filters = $session->get('filters', array());
}
```

Flash messages

Les Flash messages permettent de communiquer des notifications à l'utilisateur. Ils sont stockés en session, et supprimés automatiquement après leur utilisation (get) :

Exemple d'utilisation de création de flash message lors de l'envoi d'un formulaire :

```
use Symfony\Component\HttpFoundation\Request;

public function update(Request $request)
{
    // ...

    if ($form->isSubmitted() && $form->isValid()) {
        // do some sort of processing

        $this->addFlash(
            'notice',
            'Your changes were saved!'
        );
        // $this->addFlash() is equivalent to
$request->getSession()->getFlashBag()->add()

        return $this->redirectToRoute(...);
    }

    return $this->render(...);
}
```

Affichage des flash messages dans un template :

```
{# templates/base.html.twig #}

{# you can read and display just one flash message type... #}
{% for message in app.flashes('notice') %}
    <div class="flash-notice">
        {{ message }}
    </div>
{% endfor %}

{# ...or you can read and display every flash message available #}
{% for label, messages in app.flashes %}
    {% for message in messages %}
        <div class="flash-{{ label }}">
            {{ message }}
        </div>
    {% endfor %}
{% endfor %}
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/framework-web/symfony/controllers

Last update: **2019/08/31 14:21**