





Symfony 2 : Les contrôleurs

Benoît Roche
@rocheb83





LES CONTRÔLEURS

Benoît Roche / @rocheb832



LES CONTRÔLEURS







Ne pas confondre contrôleur et contrôleur frontal.

Nous traitons ici les **contrôleurs**.

Le contrôleur frontal

- ✓ est le fichier app.php (mode prod) ou app_dev.php (mode dev)
- ✓ Il est situé le répertoire app/web
- ✓ toutes les requêtes sont dirigées vers lui.
- ✓ On n'aura en principe jamais besoin de le modifier .






Exemple


```
class TutorielCtrlController extends Controller
{
    public function helloAction($nom){
        $page="<html><body> Bonjour ". $nom . "</body></html>";
    }
}
```

Benoît Roche / @rocheb83

3




LES CONTRÔLEURS



1 bundle = 1 contrôleur au moins

- ✓ Le contrôleur de bundle est le chef d'orchestre de l'architecture MVC.
- ✓ Il contient la logique applicative de l'application web
- ✓ On peut spécialiser les contrôleurs à l'intérieur d'un bundle.
- ✓ Exemples :
 - ✓ Contrôleur pour gérer les employés
 - ✓ Contrôleur pour gérer les inscriptions



Voir Doctrine

Benoît Roche / @rocheb83

4



LES CONTRÔLEURS



1 bundle = 1 contrôleur au moins

- ✓ C'est une classe qui peut hériter de la classe **controller** située dans le dossier **vendor**
- ✓ Un contrôleur contient des méthodes publiques qui constituent des **Actions**
- ✓ Une action d'un contrôleur est en général appelée par une route (annotation ou yml)

```
bdl5_ctrl_redirect:
  pattern: /redirection
  defaults: { _controller: Bdl5CtrlBundle:TutorialCtrl:redirect }
```

```
class TutorialCtrlController extends Controller {
    public function redirectAction() {
        $param = 'BienRedirige';
        $url = $this->generateUrl('bdls_ctrl_exercise
```

En routing yml

Benoît Roche / @rocheb83

5



LES CONTRÔLEURS




- ✓ Une méthode **Action** reçoit une requête (objet de la **classe Request**)
- ✓ Elle traite la requête
- ✓ Elle renvoie TOUJOURS une réponse sous la forme d'un objet de la classe **Response**

```
public function faireTest0Action() {
    $nb=12345;
    $response = new Response('indice = '. $nb,200);
    return $response;
}
```


localhost/TutoSymfonyBR_Ctrl/web/app_dev.php/ctrl/test0
indice = 12345

Benoît Roche / @rocheb83

6



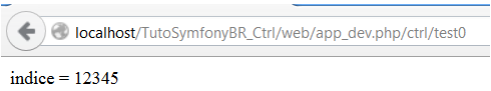
LES CONTRÔLEURS




- ✓ Une méthode **Action** reçoit une requête (objet de la **classe Request**)
- ✓ Elle traite la requête
- ✓ Elle renvoie **TOUJOURS** une réponse sous la forme d'un objet de la classe **Response**

```


public function faireTest0Action() {
    $nb=12345;
    $response = new Response('indice = '. $nb,200);
    return $response;
}
    
```



Benoît Roche / @rocheb83
7



LES CONTRÔLEURS



Exemples d'utilisation de l'objet **Response** :

- ✓ une page html ou un template

```

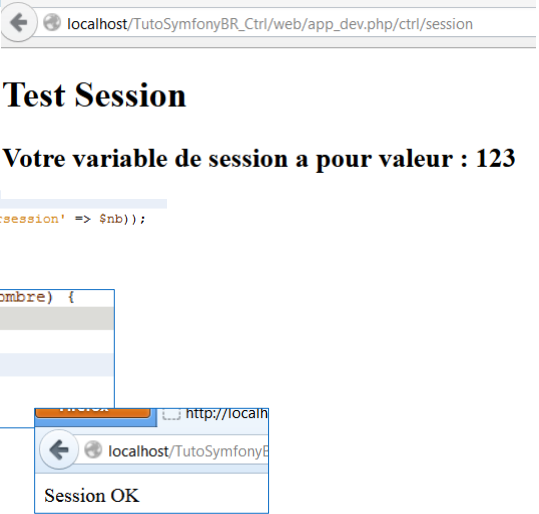
public function getSessionAction(Request $request) {
    $session = $request->getSession();
    $nb = $session->get('indice');
    return $this->render('BdlCtrlBundle:TutorielCtrl:getSession.html.twig', array('varsession' => $nb));
}
    
```

```

public function setSessionV2Action(Request $request, $nombre) {
    $session = $request->getSession();
    $session->set('indice', $nombre);
    $page = "<html><body> Session OK</body></html>";
    return new Response($page);
}
    
```

Test Session

Votre variable de session a pour valeur : 123



La plupart du temps on utilisera un objet render qui générera lui-même l'objet Response

Benoît Roche / @rocheb83
8



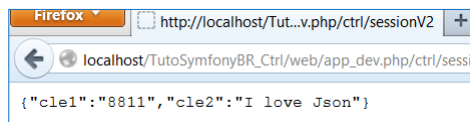
LES CONTRÔLEURS



Exemples d'utilisation de l'objet **Response** :

- ✓ un tableau json sérialisé

```
public function getSessionV2Action() {
    $session = $this->get('session');
    $nb = $session->get('indice');
    $response = new Response(json_encode(array('cle1' => $nb, 'cle2' => 'I love Json'), 0));
    $response->headers->set('Content-Type', 'application/json');
    return $response;
}
```



La plupart du temps on utilisera un objet render qui générera lui-même l'objet Response

Benoît Roche / @rocheb83

9



LES CONTRÔLEURS



On peut faire hériter le contrôleur de la classe controller,

C'est fortement déconseillé si on veut publier notre bundle

- ✓ Le contrôleur fonctionne aussi bien avec que sans l'héritage
- ✓ La classe mère Controller fournit des raccourcis à notre contrôleur.
- ✓ Elle fournit des méthodes qui sont des raccourcis pour utiliser certaines fonctionnalités de Symfony.
- ✓ **elle simplifie donc l'utilisation**

Exemples de méthodes de la classe Controller :

- generateUrl(...)
- render(...)
- getRequest(...)

[description de la classe](#)

Benoît Roche / @rocheb83

10



LES CONTRÔLEURS



Pour l'apprentissage, on va utiliser les méthodes de la classe **Controller**

On utilisera très souvent ces méthodes, car elles correspondent généralement à ce que l'on attend de notre contrôleur :



- ✓ Rediriger

`controller::redirect()`

- ✓ Faire suivre vers un autre contrôleur

`controller::forward()`

- ✓ Appeler une vue (template)

`controller::render()`

`controller::renderView()`

- ✓ Accéder à d'autres services

`controller::get()`

Benoît Roche / @rocheb83

11



LES CONTRÔLEURS



La méthode

`controller::forward($controller,[$tabParams[,tabQueries]])`

Cette méthode. ***permet de changer l'action en cours pour une autre.***
La nouvelle action peut être ou non dans le même contrôleur.



On reste dans la même requête http. L'URL client reste inchangée

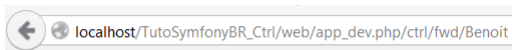
Paramètres :

`$controller` : suit la même syntaxe que dans les paramètres de routage

`$tabparams` : tableau des paramètres à fournir au contrôleur

`$tabQueries` : tableau des autres paramètres à passer au contrôleur

```
public function forwarderAction($nom) {
    $param = $nom . " ( après Forward )";
    $response=$this->forward('BdlsCtrlBundle:TutorielCtrl:Hello',array('nom'=>$param));
    return $response;
}
```



Bonjour Benoit (après Forward)

Benoît Roche / @rocheb83

12



LES CONTRÔLEURS



Les méthodes

controller::render()
controller::renderView()

Ce sont ses méthodes que l'on va le plus souvent utiliser !!

Les méthodes **render** permettent au contrôleur d'appeler un template qui sera chargé de générer l'affichage d'une page HTML.

- ✓ La méthode **renderView** renvoie un template qui sera passé en paramètre du constructeur de l'objet response.
- ✓ La méthode **render**instanciera directement l'objet response. Elle est donc plus rapide à écrire



Benoît Roche / @rocheb83

13



LES CONTRÔLEURS

**La méthode controller::renderView()**

Elle renvoie un template qui sera passé en paramètre du constructeur de l'objet response.

Elle accepte 2 paramètres :

- ✓ Le nom de la vue,
- ✓ Un tableau contenant les paramètres à envoyer à la vue

```
public function rendreVueAction($nom) {  
    $unTemplate = $this->renderView('BdlsCtrlBundle:TutorielCtrl:premiereVue.html.twig', array('nom' => $nom));  
    return new response($unTemplate);  
}
```



On pourra intercepter ici l'inexistence de la vue appelée

Benoît Roche / @rocheb83

14



LES CONTRÔLEURS



La méthode

controller::renderView()

[Démonstration](#)

(http://localhost/TutoSymfonyBR_Ctrl/web/app_dev.php/ctrl/rendrevue/Benoit)

```
$unTemplate = $this->renderView('BdlsCtrlBundle:TutorielCtrl:premiereVue.html.twig', array('nom' => $nom));
return new response($unTemplate);
```

Voici le template :

```
<body>
  <h1>Premier Template</h1>
  <h2>Vous êtes dans le template de : {{ nom }} </h2>
</body>
```

Et la variable template renvoyée :

C'est cette chaîne de caractères qui sera renvoyée au navigateur via l'objet response.

```
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta charset="utf-8" >
6     <title>Première vue</title>
7   </head>
8   <body>
9     <h1>Premier Template</h1>
10    <h2>Vous êtes dans le template de : Benoit </h2>
```

Benoît Roche / @rocheb83

15



LES CONTRÔLEURS



La méthode

controller::render()

Elle renvoie un objet **response** qui appelle donc directement la vue à afficher .

Elle accepte 2 paramètres :

- ✓ Le nom de la vue,
- ✓ Un tableau contenant les paramètres à envoyer à la vue

[Démonstration](#)

(http://localhost/TutoSymfonyBR_Ctrl/web/app_dev.php/ctrl/rendre/Benoit)

```
// Afficher des templates
public function rendreAction($nom) {
    return $this->render('BdlsCtrlBundle:TutorielCtrl:premiereVue.html.twig', array('nom' => $nom));
}
```

La vue et le résultat sont les mêmes que pour la méthode renderView

Benoît Roche / @rocheb83

16



NOTION DE SERVICE

Benoît Roche / @rocheb83

17



LES CONTRÔLEURS



Notion de service :

Un service est un objet PHP dont l'objectif est de remplir une fonction.

- ✓ Il doit être accessible depuis n'importe où dans l'application
- ✓ Il peut gérer des services simples (envoi de mails)
- ✓ Il peut gérer des services plus compliqués (doctrine)



Pourquoi cette notion ?

- ✓ Séparer chaque fonctionnalité de l'application
- ✓ Réutiliser les services dans plusieurs applications

Benoît Roche / @rocheb83

18



LES CONTRÔLEURS



Notion de service :

La technologie sous-jacente est la technologie **SOA** (Service Oriented Architecture)

Le service, contrairement à une simple classe, a besoin d'un conteneur de service pour pouvoir fonctionner.

Pour utiliser un service, on doit donc passer par son conteneur.

C'est lui qui instancie le service

Pour chaque service, on définira ainsi :

- ✓ Son nom qui permettra de l'identifier au sein de son conteneur,
- ✓ Sa classe qui permettra au conteneur d'instancier l'objet correspondant au service,
- ✓ Les paramètres dont il a besoin pour fonctionner



LES CONTRÔLEURS



Accéder à d'autres services : **`controller::get()`**

On peut utiliser de nombreux **services** fournis par symfony ... sans avoir à écrire du code compliqué.


On utilisera pour ces services la méthode

`Controller::get('nomduservice')`


Exemple :

Envoyer un mail : `$mailer=$this->get('mailer');`

[voir cookbook](#)



LES CONTRÔLEURS



Liste des services utilisables : commande `symfony : container:debug`

```
jms_di_extra.metadata_driver      container JMS\DiExtraBundle\Metadata\Driver\AnnotationDriver
kernel                            container Symfony\Component\HttpKernel\EventListener\LocaleListener
locale_listener                   container Symfony\Bridge\Monolog\Logger
logger                            container Swift_Mailer
mailer                            container Symfony\Bridge\Monolog\Handler\DebugHandler
monolog.handler.debug             container Symfony\Bridge\Monolog\Handler\FirePHPHandler
monolog.handler.firephp           container Monolog\Handler\StreamHandler
monolog.handler.main
```

Benoît Roche / @rocheb8321



LES SESSIONS



Benoît Roche / @rocheb8322



LES CONTRÔLEURS



Les Sessions

On peut, tout comme en PHP classique, gérer des sessions.
Pour cela on utilise le service session.

Principe :

1- Récupérer l'objet session

Par le conteneur	Par l'objet Request
<code>\$session= \$this->get('session');</code> (<code>\$this</code> : objet controller)	<code>\$session=\$request->getSession()</code> (<code>\$request</code> : objet de la classe Request)

```
public function getSessionAction(Request $request) {
    $session = $request->getSession();

    public function getSessionV2Action(Request $request) {
        $session= $this->get('session');
```

Benoît Roche / @rocheb83

23



LES CONTRÔLEURS



Les Sessions

2- Créer une variable de session :

```
$session->set('uncle', 'unevaleur');
```

```
$session->set('indice', $nombre);
```

3- Récupérer une variable session :

```
$maVar=$session->get('uncle');
```

```
$nb = $session->get('indice');
```

Benoît Roche / @rocheb83

24



L'OBJET REQUEST

Benoît Roche / @rocheb83

25



LES CONTRÔLEURS



L'objet Request

On l'a aperçu, l'objet Controller du bundle a accès à l'objet Request, **à condition qu'il hérite de la classe *Controller de base***

On le passe en paramètre de la méthode du controller

```
public function testRequest(Request $request) {  
    //  
}
```

Il fournit des méthodes intéressantes :

- ✓ isXMLHttpRequest() // indique si c'est une requête ajax
- ✓ query->get('XX') // retourne le paramètre \$_GET
- ✓ request->get('yy') // retourne le paramètre \$_POST

Benoît Roche / @rocheb83

26



LA GESTION DES ERREURS

Benoît Roche / @rocheb83

27



LES CONTRÔLEURS



Gestion des erreurs

On peut, dans le contrôleur déclencher des exceptions avec l'instruction **throw**.

Ainsi, on peut :

- ✓ Déclencher une exception qui provoquera une erreur 404
- ✓ Déclencher une exception not found qui affichera une page 500

```
public function erreurAction($nombre) {  
    if ($nombre > 100) {  
        throw new \Exception('Pas de nombre supérieur à 100');  
    } else {  
        throw $this->createNotFoundException('Pas de nombre inférieur ou égal à 100');  
    }  
}
```

“



Pas de nombre supérieur à 100
500 Internal Server Error - [Exception](#)

“

Pas de nombre inférieur ou égal à 100
404 Not Found - [NotFoundHttpException](#)

Benoît Roche / @rocheb83

28



LES CONTRÔLEURS

Gestion des erreurs

Les mêmes erreur en mode prod ;

```
public function erreurAction($nombre) {  
    if ($nombre > 100) {  
        throw new \Exception('Pas de nombre supérieur à 100');  
    } else {  
        throw $this->createNotFoundException('Pas de nombre inférieur ou égal à 100');  
    }  
}
```

Oops! An Error Occurred

The server returned a "404 Not Found".

Something is broken. Please e-mail us at [email] and let us know what you were doing when this error occurred. We will fix it as soon as possible. Sorry for any inconvenience caused.

Oops! An Error Occurred

The server returned a "500 Internal Server Error".

Something is broken. Please e-mail us at [email] and let us know what you were doing when this error occurred. We will fix it as soon as possible. Sorry for any inconvenience caused.

Benoît Roche / @rocheb83

29



EXERCICE

Benoît Roche / @rocheb83

30

**EXERCICE**

Il est temps de mettre en pratique tout ceci....



Exercice:

Il est temps de mettre en pratique tout ceci....

[Voir : exercices sur les controleurs](#)

Benoît Roche / @rocheb83

31



Fin du cours, merci

Questions/Réponses

Benoît Roche
@rocheb83

Benoît Roche / @rocheb83

32