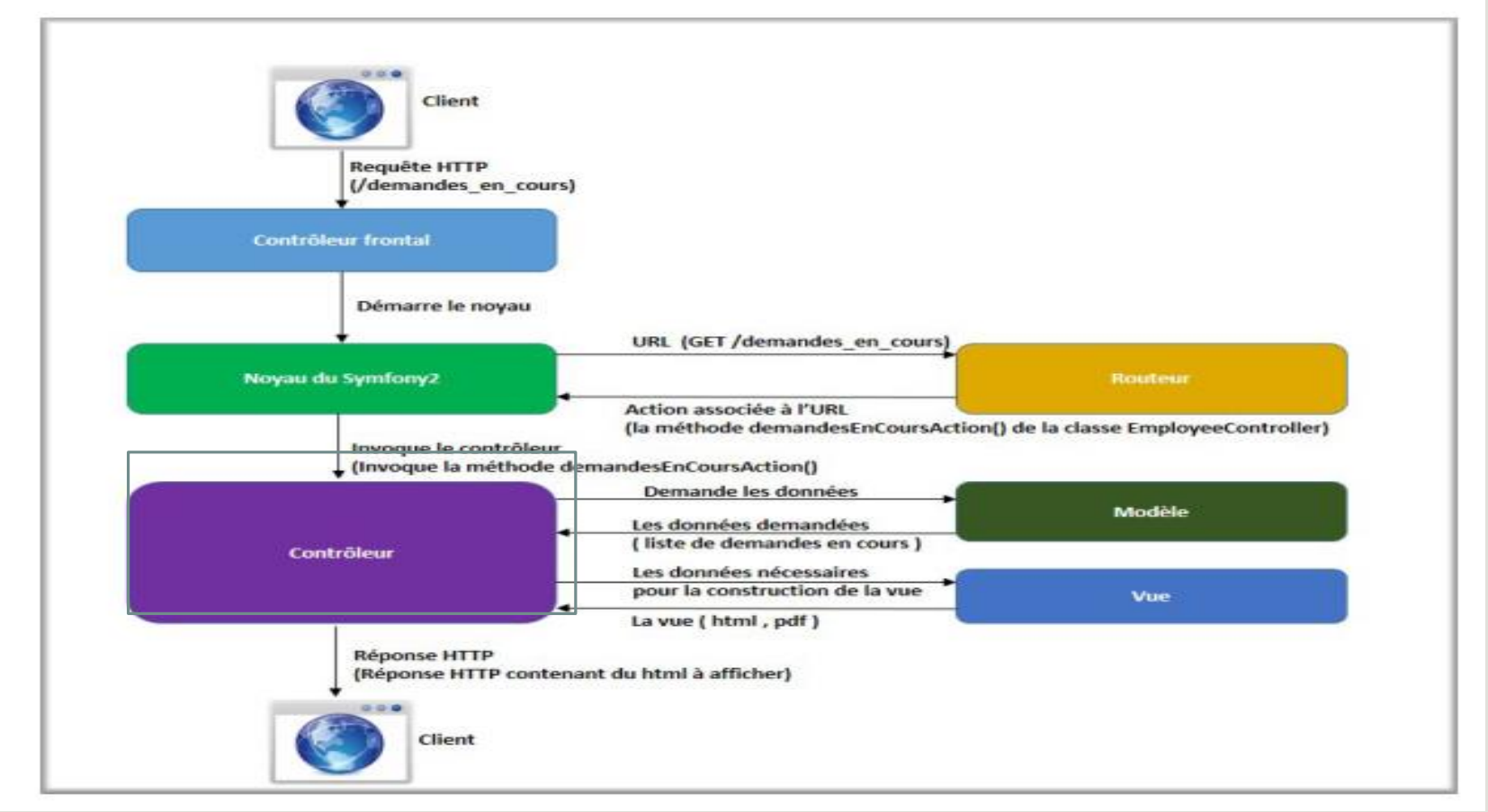
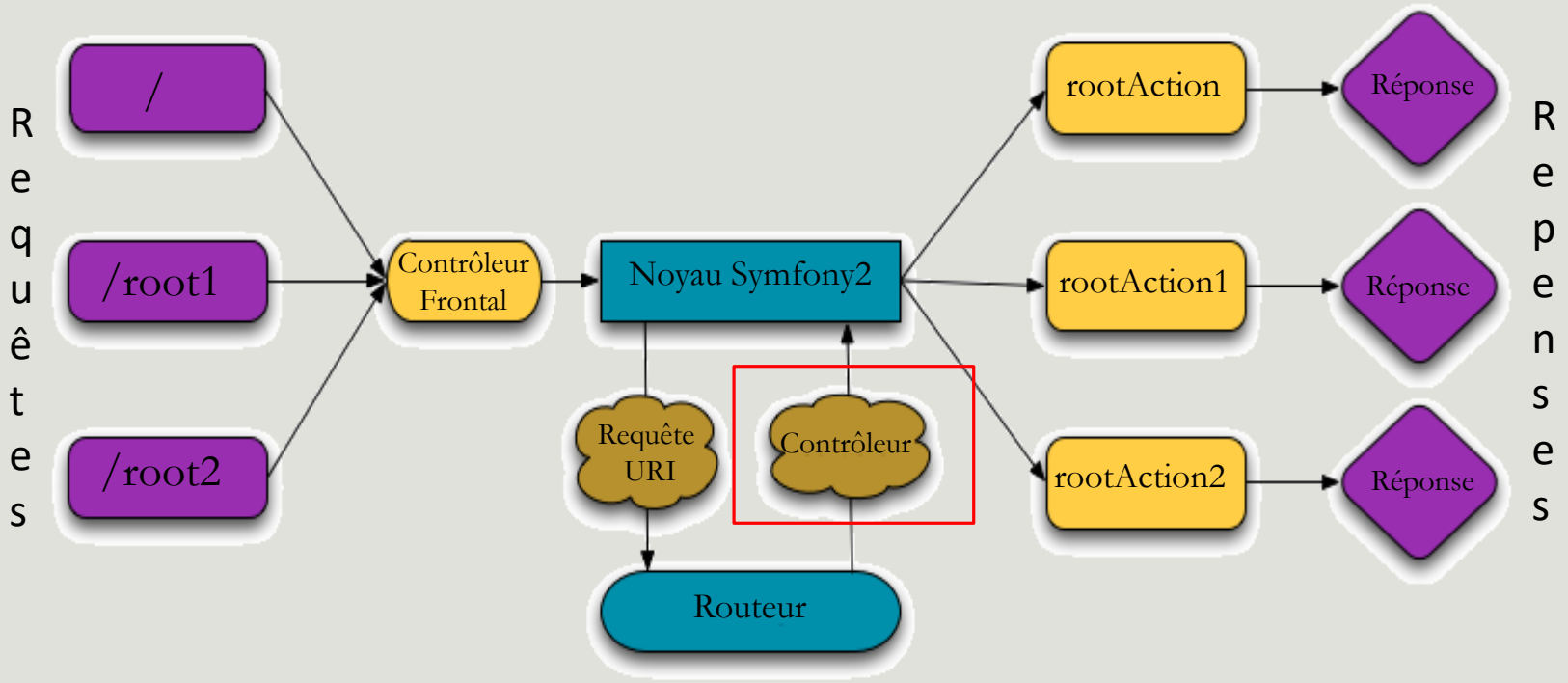
Symfony2
Les contrôleurs

AYMEN SELLAOUTI

Introduction (1)



Introduction (2)



Introduction (3)

Fonction PHP (action)

Rôle : Répondre aux requêtes des clients.

Exemple d'un contrôleur

```
<?php

namespace Forma\TestBundle\Controller; //espace de nom

use Symfony\Bundle\FrameworkBundle\Controller\Controller; // on importe la classe Controller

use Symfony\Component\HttpFoundation\Response;

class HelloController extends Controller // Si on veut que notre contrôleur hérite de la classe afin de pouvoir utiliser ces méthodes helper (http://api.symfony.com/2.8/Symfony/Bundle/FrameworkBundle/Controller/Controller.html)

{   // Par convention chaque contrôleur (action) se termine par le mot clé Action

    public function indexAction() // le nom index est l'identifiant de l'action c'est avec ce nom que nous l'appelons dans la root
//

    { // On crée un objet Response puis on la retourne c'est le rôle du contrôleur

        $resp = new Response('<html><body>Bonjour le monde !</body></html>');

        return $resp;

 }

}
```

Fonctions de base de la classe Contoller

Méthode	fonctionnalité	Valeur de retour
generateUrl(string $route, array() $parameters)	Génère une URL à partir de la route	String
forward(String Controller, array () $parameters)	Forward la requête vers un autre contrôleur	Response
Redirect(string $url int $statut)	Redirige vers une url	RedirectResponse
Render(string $view array $parameters)	Affiche une vue	Response
getRequest()	Raccourci pour récupérer le service request	Request
Get(string $id)	Retourne un service à travers son id	object
createNotFoundException(String $messag)	Retourne une NotFoundException	NotFoundException

http://api.symfony.com/2.7/Symfony/Bundle/FrameworkBundle/Controller/Controller.html

Lien entre la root et le contrôleur
Création de la root

Voici un exemple de correspondance entre une route et le contrôleur qui lui est associé :

Nous prenons l'exemple d'une route écrite en YAML.

_controller:NomBundle:NomClasseController:NomAction

```
root_bjr:
    path:    /bonjour
    defaults: { _controller: FormaTestBundle:Exemple:index}
```

Lien entre la root et le contrôleur
Passage de paramétré : root vers contrôleur

Afin de récupérer les paramètres de la root dans le contrôleur nous utilisons les noms des paramètres.

Exemple

```
root_bjr:
    path:    /bonjour/{section}
    defaults: { _controller: FormaBundle:Exemple:index}
public function indexAction($section)
{
      $resp = new Response('<html><body>Bonjour'. $section.'!</body></html>');
       return $resp;
 }
```

Récupérer les paramètres de la requête (1)

Afin de récupérer l'objet **Request** dans le contrôleur il suffit de le déclarer dans l'entête du contrôleur en question En spécifiant qu'il s'agit d'un objet de type Request.

Exemple

```
public function indexAction(Request $req)
{ … }
```

Récupérer les paramètres de la requête (2)

L'objet **Request** permet de récupérer l'ensemble des attributs passées dans la requête

Type de paramètres	Méthode Symfony2	Méthode traditionnelle	Exemple
Variables d'URL	$request->query	$_GET	$request->query->get('var')
Variables de formulaire	$request->request	$_POST	$request->request->get('var')
Variables de cookie	$request->cookies	$_COOKIE	$request->cookies->get('var')

Récupérer les paramètres de la requête (3)

Exemple : pour l'url suivante :
http://127.0.0.1/symfoRT4/web/app_dev.php/test/bonjour/RT4?groupe=1

Pour récupérer le groupe passé dans l'url (donc du Get) on devra récupérer le request puis utiliser $request->query->get('tag')

Exemple

```
public function indexAction($section,Request $req)
{
$groupe = $request->query->get('groupe');
    return new Response(« Bonjour ».$section.« groupe  ».$groupe);
 }
```

Récupérer les paramètres de la requête (4)

La classe Request offre plusieurs informations concernant la requête HTTP à travers un ensemble de méthodes (http://api.symfony.com/2.8/Symfony/Component/HttpFoundation/Request.html)

getMethod() : retourne la méthode de ma requête

isMethod() : vérifie la méthode

getLocale : retourne la locale de la requête (langue)

isXmlHttpRequest : retourne vrai si la requête est de type XmlHttpRequest

…

Réponse aux requêtes
Renvoi (1)

Le traitement se fait à travers le service templating qui se charge de créer et d'irriguer un objet de type Response.

Rôle : créer la réponse et la retourner

Méthode :

- Sans passer par les helpers de la la classe Controller :
  - $this->get('templating')->render('l'url', 'les paramétres à transferer');
- En utilisant les helpers :
  - $this->render ('l'url', 'les paramétres à transferer');

L'url est selon le format suivant : @NomBundle/Dossier_dans_ressources/page
Le nom du bundle est sans le mot clé Bundle

Réponse aux requêtes
Renvoi (2)

Exemple :

```
 public function indexAction($section,Request $req)
{  $groupe = $request->query->get('groupe');
   return $this->get('templating')->render('@Forma/Default/index.html.twig',array('section'=>$section,
                                                                                   'goupe'=>$groupe));
}
 public function indexAction($section,Request $req)
{  $groupe = $request->query->get('groupe');
   return $this->render('@Forma/Default/index.html.twig',array('section'=>$section,
                                                               'goupe'=>$groupe));
}
```

Réponse aux requêtes
Redirection

Rôle : Redirection vers une deuxième page (en cas d'erreur ou de paramètres erronées ou de user non identifié par exemple)

Méthode :

- Sans passer par les helpers de la la classe Controller :
  - $url=$this->get('router')->generate('notre_route');
  - return new RedirectResponse($url);
- En utilisant les helpers :
  - $url=$this->get('router')->generate('notre_route');
  - return this->redirect($url);

  - Ou la méthode la plus rapide qui se charge de tout :
  - return this->redirectToRoute('nomDeMaRoute');

Réponse aux requêtes Forwarding

Rôle : Forwarder vers un autre contrôleur

Méthode :

- Sans passer par les helpers de la la classe Controller en utilisant le httpKernel :
  - $httpKernel = $this->container->get('http_kernel');Return new RedirectResponse($url);
  - $response = $httpKernel ->forward('NomBundle:NomController:NomAction', array('label1' => $param1,
    'label2' => $label2,
    ));
    return($response);
- En utilisant les helpers :
  - $response = $this->forward('NomBundle:NomController:NomAction', array('label1' => $param1,
    'label2' => $label2,
    ));
    return($response);

Gestion des sessions

- Une des fonctionnalités de base d'un contrôleur est la manipulation des **sessions**.
- Un objet **session** est fournit avec l'objet **Request.**
- La méthode **getSession()** permet de récupérer la session.
- L'objet Session fournit deux méthodes : **get( )** pour récupérer une variable de session et **set( )** pour la modifier ou l'ajouter.
- **get** prend en paramètre la variable de session.
- **set** prend en entrée deux paramètres la clef et la valeur.
- Dans la TWIG on récupère les paramètres de la session avec la méthode

```
app.session.get('nomParamétre')
```

Gestion des sessions : les méthodes

- **all()**

Retourne tous les attributs de la session dans un tableau de la forme clef valeur

- **has()**

Permet de vérifier si un attribut existe dans la session. Retourne Vrai s'il existe Faux sinon

- **replace()**

Définit plusieurs attributs à la fois: prend un tableau et définit chaque paire clé => valeur

- **remove()**

Efface un attribut d'une clé donnée.

- **clear()**

Efface tous les attributs.

Gestion des sessions : les FlashMessages

- Les variables de sessions qui ne dure que le temps d'une seule page sont appelées **message Flash.**

- Utilisées généralement pour afficher un message après un traitement particulier (Ajout d'un enregistrement, connexion, …).

- La méthode getFlashBag() permet de récupérer l'objet FlashBag à partir de l'objet session.

- La méthode add de cet objet permet d'ajouter une variable à cet objet.

- Pour récupérer le Flash message de la TWIG on utilise app.session.flashbag.get `('nomParamétre')`.