Développement WEB en JAVA avec le Framework

Antoine CHAUVIN
IR3
22 janvier 2013

- I) Historique
- II) Qu'est ce que PLAY
- III) Les grands concepts
- IV)Développer en java avec PLAY

Historique

- Play créé par Guillaume Bort, alors qu'il travaillait chez Zenexity.
- Le projet est démarré en 2007 avec pour objectif de simplifier grandement le développement WEB en JAVA
- La première version publique est disponible en Mai 2008
- Play 1.1 a été publié en Novembre 2010, et est considéré comme la première version vraiment utilisable du framework
- Play 1.2 a été publié en Avril 2011
- Sadek Drobi rejoint Guillaume Bort fin 2011 pour créer Play 2.0 qui a été publié le 13 Mars, 2012

Qu'est ce que PLAY 2.0

Play 2.0 "*a new web framework for a new era*"

Play 2.0 **un Framework d'application Web pour Java et Scala**

- Un modèle complètement non bloquant (réactif) construit pour la programmation asynchrone
- Une maitrise totale de la consommation des ressources (mémoire, CPU)
- Une architecture sans état permettant de très forte montée en charge

Qu'est ce que PLAY 2.0

Mais aussi un **serveur indépendant** se passant donc de la brique serveur d'application, des Servlets et de son écosystème.

Les principaux concepts

1- Convention plutôt que configuration

2- Modèle MVC

3- La programmation asynchrone

4- Java et Scala

5- Capacité de mise à l'échelle

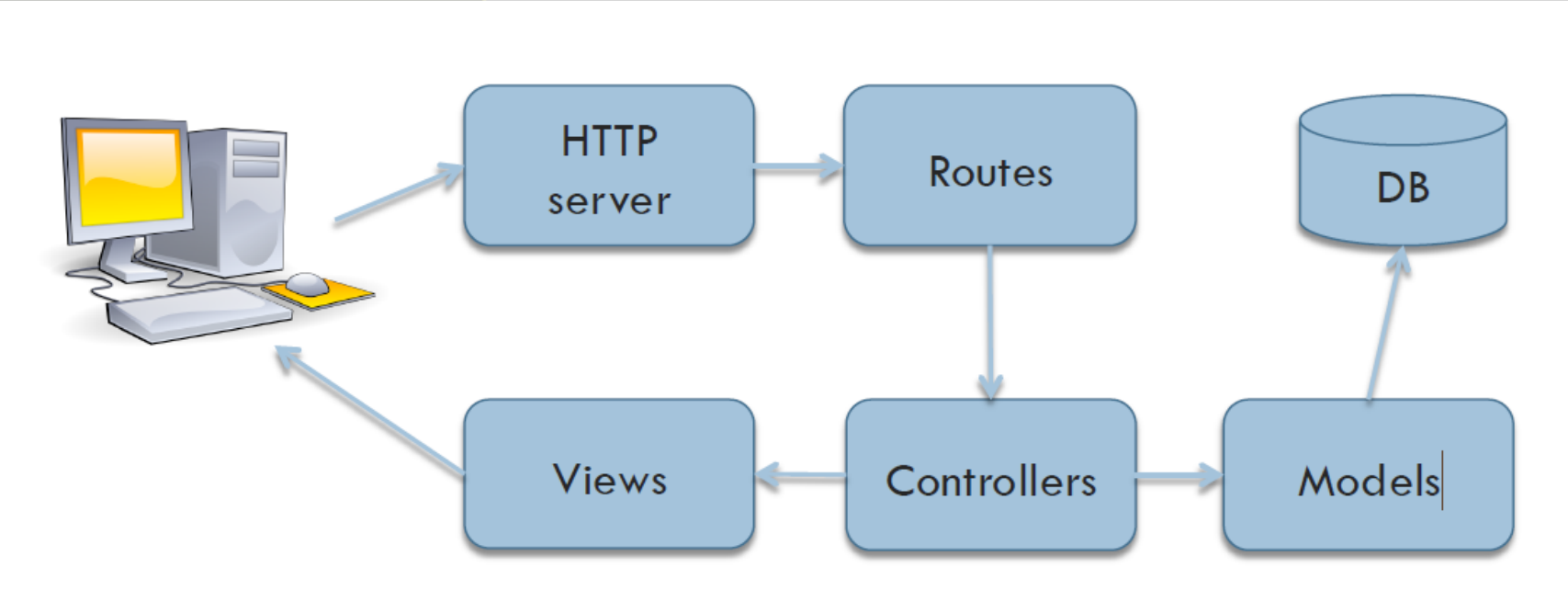
1)Convention plutôt que configuration

Très peu de configuration

Pas besoin de déployer un serveur d'application

Optimiser la productivité des développeurs

2)Modèle MVC



3)Programmation Asynchrone

Evolution du WEB vers de plus en plus de traitements de données temps réel concurrentes

- Besoin d'un modèle de programmation asynchrone:
  Play utilise la possibilité du Java à gérer les I/O de façon asynchrone.

3)Programmation Asynchrone

- Concrètement le résultat renvoyé au client peut dépendre d'un calcul coûteux ou un appel de service Web longtemps.

  Ce qui entraine dans un comportement commun un blocage.

  Solution: *Promise<Result>*

  Blocage client, mais non serveur.

3)Programmation Asynchrone

Evolution du WEB vers de plus en plus de traitements de données temps réel concurrentes

- Utilisation native du Framework Akka
  Un framework disponible en Scala et en Java permettant de gérer très efficacement des applications multithread et concurrentes.

4) Java et Scala

Globalement en Java, mais avec un moteur de Template en Scala

Ce qui a permis le typage des mes templates. En clair, chaque template attend des paramètres d'entrée typés. C'est à la compilation que tout cela va être vérifié, ce qui rend les templates très robustes.

5)Capacité de mise à l'échelle

Stateless RESTful

Pas de java EE session

Mise à l'échelle horizontale

Développer en Java avec PLAY

1) Installation

Pré requis: JDK 6 ou supérieur

Télécharger le paquet binaire

… C'est tout !

Développer en Java avec PLAY

2) Création d'un projet

```
antoine@ubuntu:~/Bureau/exemple$ play new applicationTest

play! 2.0.4, http://www.playframework.org

The new application will be created in /home/antoine/Bureau/exemple/applicationT
est

What is the application name?
> applicationTest

Which template do you want to use for this new application?

  1 - Create a simple Scala application
  2 - Create a simple Java application
  3 - Create an empty project

> 2

OK, application applicationTest is created.

Have fun!

antoine@ubuntu:~/Bureau/exemple$
```

Développer en Java avec PLAY

```
app                     → Application sources
 └ assets               → Compiled asset sources
    └ stylesheets       → Typically LESS CSS sources
    └ javascripts       → Typically CoffeeScript sources
 └ controllers          → Application controllers
 └ models               → Application business layer
 └ views                → Templates
conf                    → Configurations files and other non-compiled
 └ application.conf     → Main configuration file
 └ routes               → Routes definition
public                  → Public assets
 └ stylesheets          → CSS files
 └ javascripts          → Javascript files
 └ images               → Image files
```

Développer en Java avec PLAY

```
project                    → sbt configuration files
 └ build.properties        → Marker for sbt project
 └ Build.scala             → Application build script
 └ plugins.sbt             → sbt plugins
lib                        → Unmanaged libraries dependencies
logs                       → Standard logs folder
 └ application.log         → Default log file
target                     → Generated stuff
 └ scala-2.9.1
    └ cache
    └ classes              → Compiled class files
    └ classes_managed      → Managed class files (templates, ...)
    └ resource_managed     → Managed resources (less, ...)
    └ src_managed          → Generated sources (templates, ...)
test                       → source folder for unit or functional tests
```

Développer en Java avec PLAY

3) La console play

```
antoine@ubuntu:~/Bureau/exemple/applicationTest$ play
[info] Loading project definition from /home/antoine/Bureau/exemple/applicationTest/project
[info] Set current project to applicationTest (in build file:/home/antoine/Bureau/exemple/applicationTest/)
      _            _
 _ __ | | __ _ _  _| |
| '_ \| |/ _' | || |_|
|  __/|_|\____|\__ (_)
|_|             |__/

play! 2.0.4, http://www.playframework.org

> Type "help play" or "license" for more information.
> Type "exit" or use Ctrl+D to leave this console.

[applicationTest] $
```

Développer en Java avec PLAY

3) La console play

Run : le serveur sera lancé avec la fonction d'auto-reload activé

Compile : compile l'application mais ne lance pas le serveur

Debug : lance le serveur avec un port JPDA (Java Platform Debugger Architecture)

Développer en Java avec PLAY

3) La console play

~Run et ~Compile: compilation sera déclenché à chaque fois que vous changez un fichier source.

Développer en Java avec PLAY

4) Intégration avec Eclipse

« Play eclipsify »

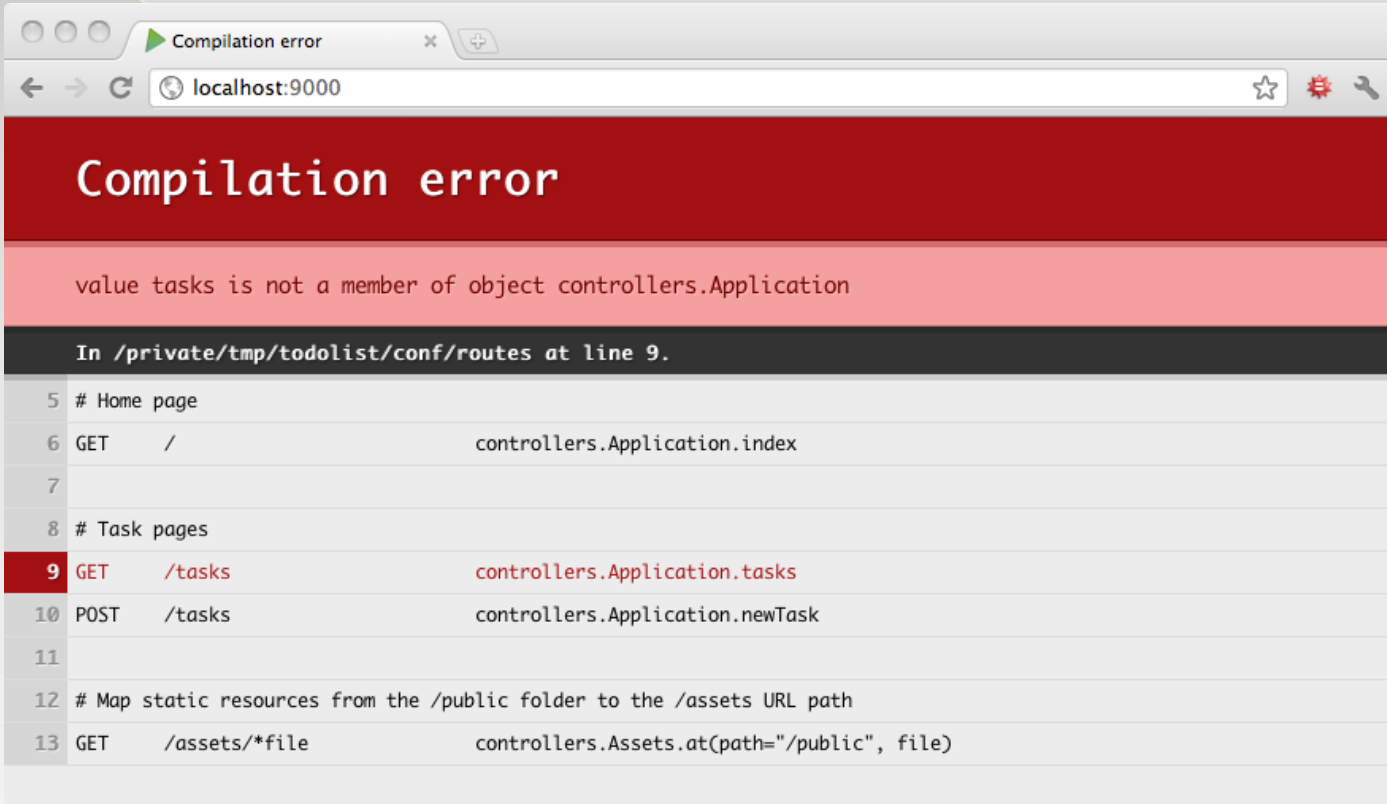
Import existing project

Module scala pour Eclipse.

Développer en Java avec PLAY

5) Les facilités de développement
Erreurs directement dans le navigateur

-Java
-Javascript
-Templates
-CSS (Less framework)



Développer en Java avec PLAY

5) Les facilités de développement

Pas besoin de redéployer ni de configurer un serveur d'application.

Rechargement de code à chaud:

- Je modifie une classe
- Elle est automatiquement recompilée
- J'actualise mon navigateur pour obtenir le résultat

Développer en Java avec PLAY

6) Développement HTTP

Trois grands objets:

- Actions
- Controllers
- Results

Développer en Java avec PLAY

6) Développement HTTP

- Actions

  La plupart des demandes reçues par une application Play sont traitées par une action.

  Une action est une méthode Java qui traite les paramètres de la demande, et produit un résultat renvoyé au client.

Développer en Java avec PLAY

6) Développement HTTP

Une action renvoie une valeur play.mvc.Result, représentant la réponse HTTP à envoyer au client Web. Dans cet exemple on construit une réponse ok 200

```
public static Result index() {
  return ok("Got request " + request() + "!");
}
```

Développer en Java avec PLAY

6) Développement HTTP

- Contrôleurs

  Un contrôleur n'est rien de plus qu'une classe étendant *play.mvc .Controller* et qui regroupe plusieurs Actions.

Développer en Java avec PLAY

6) Développement HTTP

- Results

  Un *result* est une réponse basique faite au navigateur. Elle contient un code d'état, un ensemble d'en-têtes HTTP et un corps.

  play.mvc.Result

  Voici quelques exemples qui créent des *Results* différents:

```
Result ok = ok("Hello world!");
Result notFound = notFound();
Result pageNotFound = notFound("<h1>Page not found</h1>").as("text/html");
Result badRequest = badRequest(views.html.form.render(formWithErrors));
Result oops = internalServerError("Oops");
Result anyStatus = status(488, "Strange response type");
```

Développer en Java avec PLAY

6) Développement HTTP

- Results

  Une redirection est aussi considéré comme un *result*

```
public static Result index() {
  return redirect("/user/home");
}
```

Développer en Java avec PLAY

6) Développement HTTP

Les templates: en Scala

```
@(customer: Customer, orders: List[Order])

<h1>Welcome @customer.name!</h1>

<ul>
@for(order <- orders) {
  <li>@order.getTitle()</li>
}
</ul>
```

```
Content html = views.html.Application.index.render(customer, orders);
```

Développer en Java avec PLAY

7) Déploiement d'application

Prévu pour être déployé sur le serveur JBOSS Netty:

```
$ play clean compile stage
```

Ce qui crée un export du projet avec un script permettant de lancer le serveur Web directement

Développer en Java avec PLAY

7) Déploiement d'application

Il existe un plugin pour Play2 permettant de créer un WAR directement à partir des sources afin de pouvoir l'utiliser sur un serveur d'application tel JBOSS ou Tomcat.

Questions ?