Play Framework 2

Julia Leven, Jérôme Garcia, Romain Philippon

08 / 12 /2014

Master IAGL

Sommaire

Introduction

Un peu d’histoire :

- Chez Zenity en 2007 (Guillaume Bort)
- En 2009 - Play est Open Source
- 16 Novembre 2011 - Passage à Play 2

Introduction

- Framework fullsack - *What you code is what you need*
- Stateless
- Scalable
- High-Productivy

JAVA ou Scala ?

JAVA ou Scala ?

Pas d'importance

JAVA	Scala
`play.mvc`	`play.api.mvc`

JAVA ou Scala ?

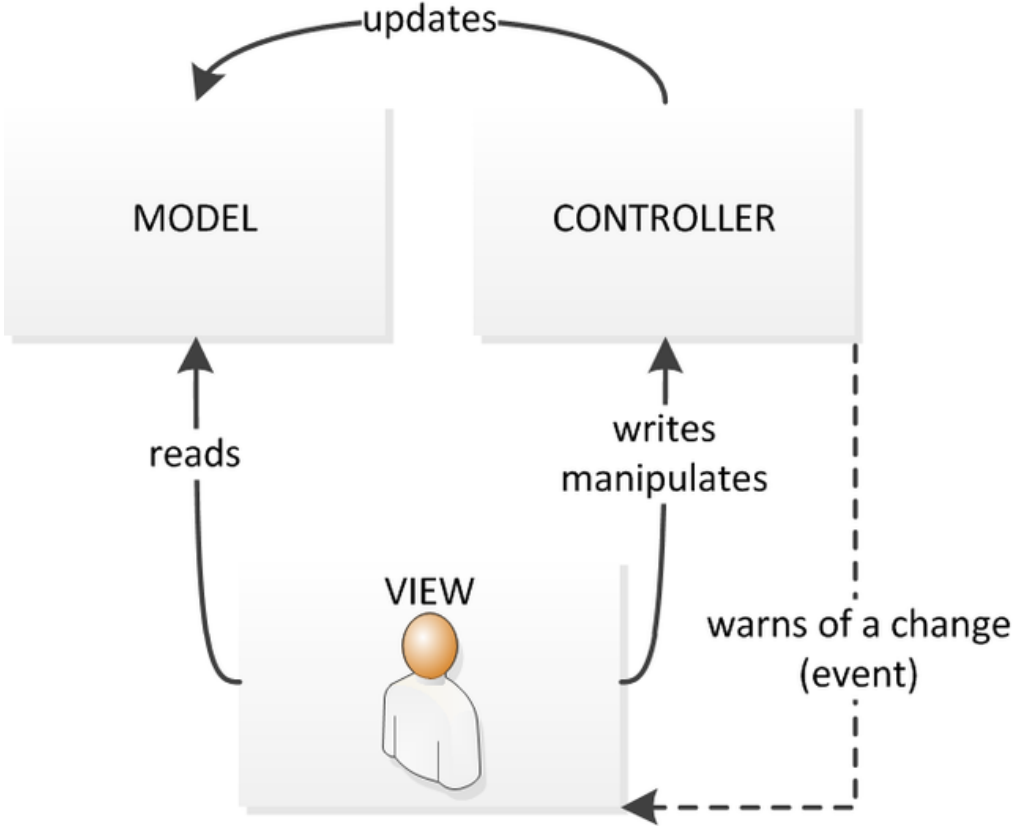
Mais pourquoi Scala ?

- Un web de plus en dynamique
- Faciliter le développement
- Un build puissant
- C’est une vieille histoire

Architecture MVC

Architecture MVC

- Modèle
- Vue
- Controleur



Architecture MVC - Structure

app → Sources compilées de l'application
 └ controllers → Vos controleurs
 └ models → Vos modèles
 └ views → Vos templates
build.sbt → build script
conf → Fichiers de configuration & ressources non-compilées (chargées dans le classpath)
 └ application.conf → Fichier de configuration
 └ routes → Définition des routes
public → Assets - ressources statiques
 └ stylesheets → Fichiers CSS
 └ javascripts → Fichiers Javascript
 └ images → Images
project → Fichiers de configuration pour SBT
 └ build.properties → Fichier de configuration du build
 └ plugins.sbt → Plugins SBT pour Play
lib → Dépendences non-gérées
logs → Répertoire des logs
 └ application.log → Fichier de log par défaut
test → Répertoire sources pour les tests unitaires et fonctionnels

Architecture MVC - Controller

Un controller en Play,
c'est rien que
des Actions

Architecture MVC - Controller

Une action = Une méthode

```
import play.mvc.*;

public class AwesomeController extends Controller {
    public static Result iAmAction() {
        return ok(“Let’s do it”);
    }

    public static Result iAmActionParameter(String name) {
        return ok(“Let’s do it ”+ foo); // let’s do it Billy
    }
}
```

Une action sans paramètre

Une action avec paramètre

Architecture MVC - Model

Un modèle = Un objet métier = Bean Java

```
import play.mvc.*;

public class User extends Model {
     private String name;
     private int age;

     public User() { super(); }

     public void setName(String newName) { this.name = newName; }
     public String getName() { return this.name; }
     public void setAge(int newAge) { this.age = newAge ; }
     public int getAge() { return this.age; }
}
```

Architecture MVC - Model

Avec aussi des annotations JPA

```
import play.mvc.*;

@Entity
public class User extends Model {
    @Id
    private String name;
    private int age;

    public User() { super(); }

    public void setName(String newName) { this.name = newName; }
    public String getName() { return this.name; }
    public void setAge(int newAge) { this.age = newAge ; }
    public int getAge() { return this.age; }
}
```

Architecture MVC - Model

Avec aussi des annotations Play

```
import play.mvc.*;

@Entity
public class User extends Model {
     @Id
     private String name;
     @Required
     @Max(99)
     private int age;

     public User() { super(); }

     public void setName(String newName) { this.name = newName; }
     public String getName() { return this.name; }
     public void setAge(int newAge) { this.age = newAge ; }
     public int getAge() { return this.age; }
}
```

Architecture MVC - Model

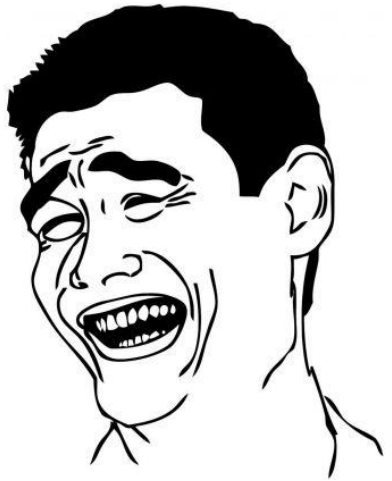
Et pour finir des modèles qui n'étendent pas de Model

```
import play.mvc.*;
import import play.data.validation.Constrai

@Entity
public class User {
    @Id
    private String name;
    @Required
    @Max(99)
    private int age;

    public User() { super(); }

    public void setName(String newName) { this.name = newName; }
    public String getName() { return this.name; }
    public void setAge(int newAge) { this.age = newAge ; }
    public int getAge() { return this.age; }
}
```



Architecture MVC - View

Une vue en Play est un template Twirl

Twirl est un moteur de template Scala

Architecture MVC - View

```
@(images : List[Image])

@main(Html("Page d'accueil"), nav="index") {
<div class="container">
      <div class="row center">
            <h1>Welcome</h1>

            <div id="listImage">
                  @images.grouped(3).toList.map { list =>
                        <div class="row">
                              @for(img <- list) {
                                    <a href="/preview/@img.getId">
                                          <img
                                                src="@routes.Assets.at(img.getPath)"
                                                alt="preview image"
                                                class="img-rounded"
                                          />
                                    </a>
                              }
                        </div>
                  }
            </div>
      </div>
</div>
}
```

Configuration : Tout est dans le conf

Comme pour tout les autres frameworks, tout est une question de routing

Dans *conf/routes* :

```
GET     /                           controllers.Application.index()
GET     /article/:id                controllers.Application.getArticle(Long : id)
POST    /article/:id/add-comment    controllers.Application.addComment(Long : id)
```

Mot HTTP

URL

Action appelée

Configuration : Tout est dans le conf

La plupart des configurations d'un projet Play se situe dans le fichier *conf/application.conf*

- Vous pouvez ajouter, éditer et supprimer des configurations
- Pour commenter une configuration, comme en shell : #
- Pour récupérer une configuration via l'API :

```
Play.application().configuration().get(Type)(String nomConfig)
```

Sources

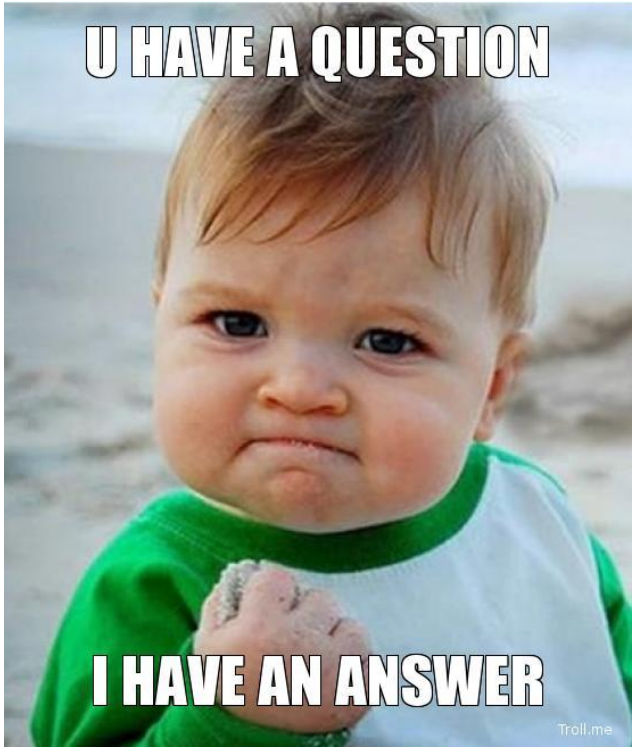
Philosophie - Play 2 Documentation

Article sur la sortie de Play 2

Wikipédia - Image MVC

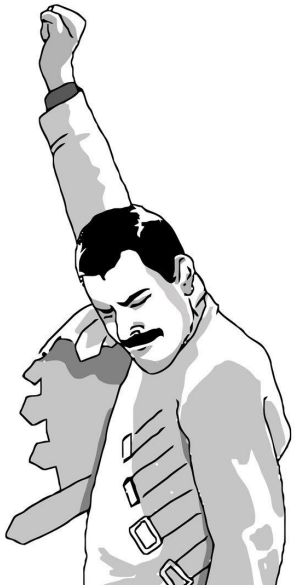
Diaporama Play 2 - SlideShare

Questions



Travaux Pratiques

https://github.com/JimiPepper/tuto-play-framework-iagl



OU

http://bit.ly/1zg6WuQ