


Play! framework

Mathieu ANCELIN

- Ingénieur d'étude @SERLI
- Java & OSS
 - JOnAS, GlassFish, Weld, etc ...
 - Poitou-Charentes JUG
- Membre de l'expert group CDI 1.1 (JSR-346)
- @TrevorReznik



Historique

- 
- 2007 : créé par Guillaume Bort (besoin interne)
 - 2008 : framework open-source
 - 2009 : version 1.0
 - 2010 : module Scala (expérimentations)
 - 2011 : version 1.2
 - 2012 : version 2.0 (réécriture, Scala, Akka)

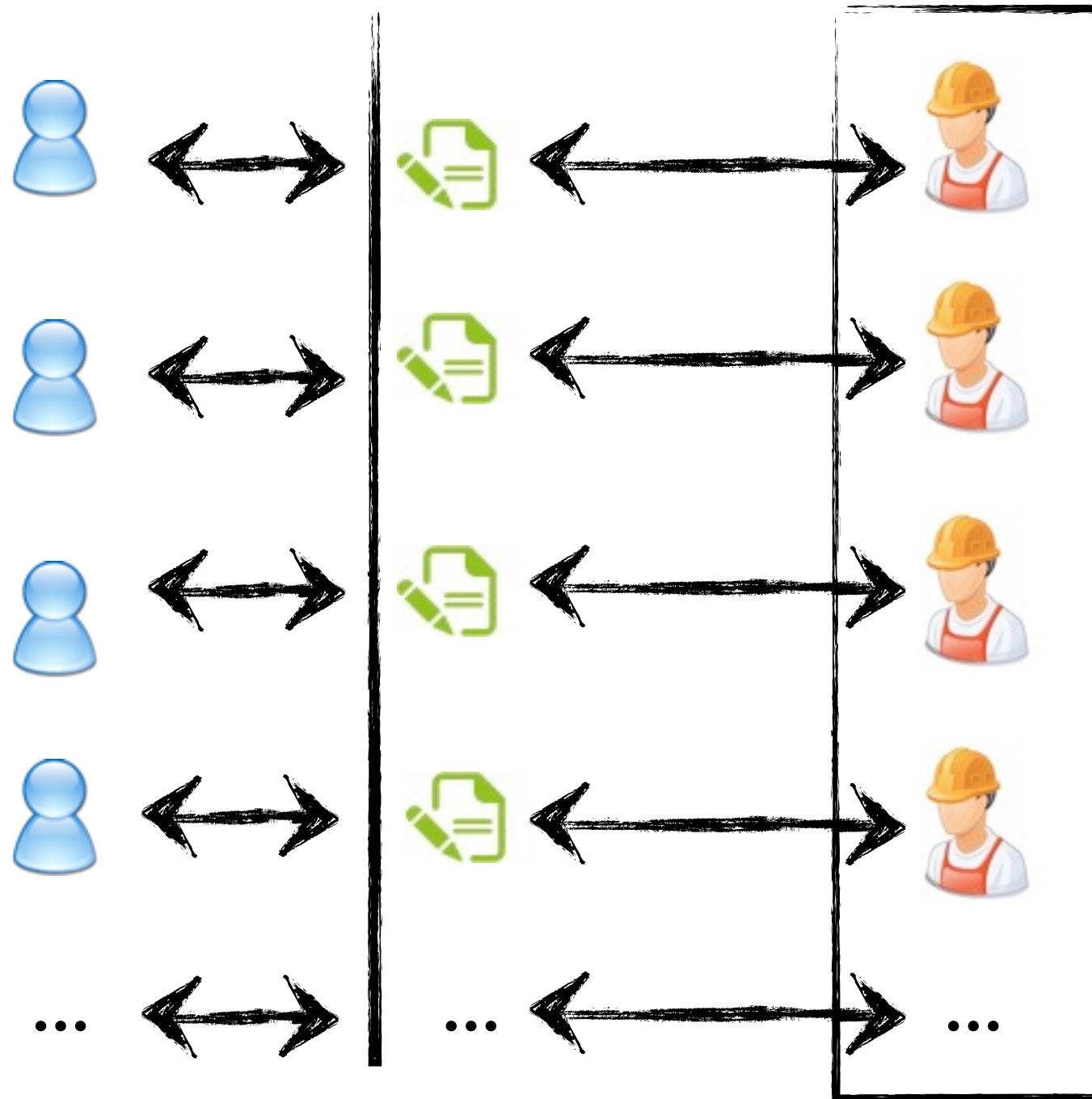
Play framework

- framework web server-side pour la JVM
 - conçu pour rendre la création d'applications web simple
 - pas de techno. client
- basé sur une architecture légère, stateless et web-friendly
- framework fullstack (compilation => deploy)
- API utilisable en Java et en Scala

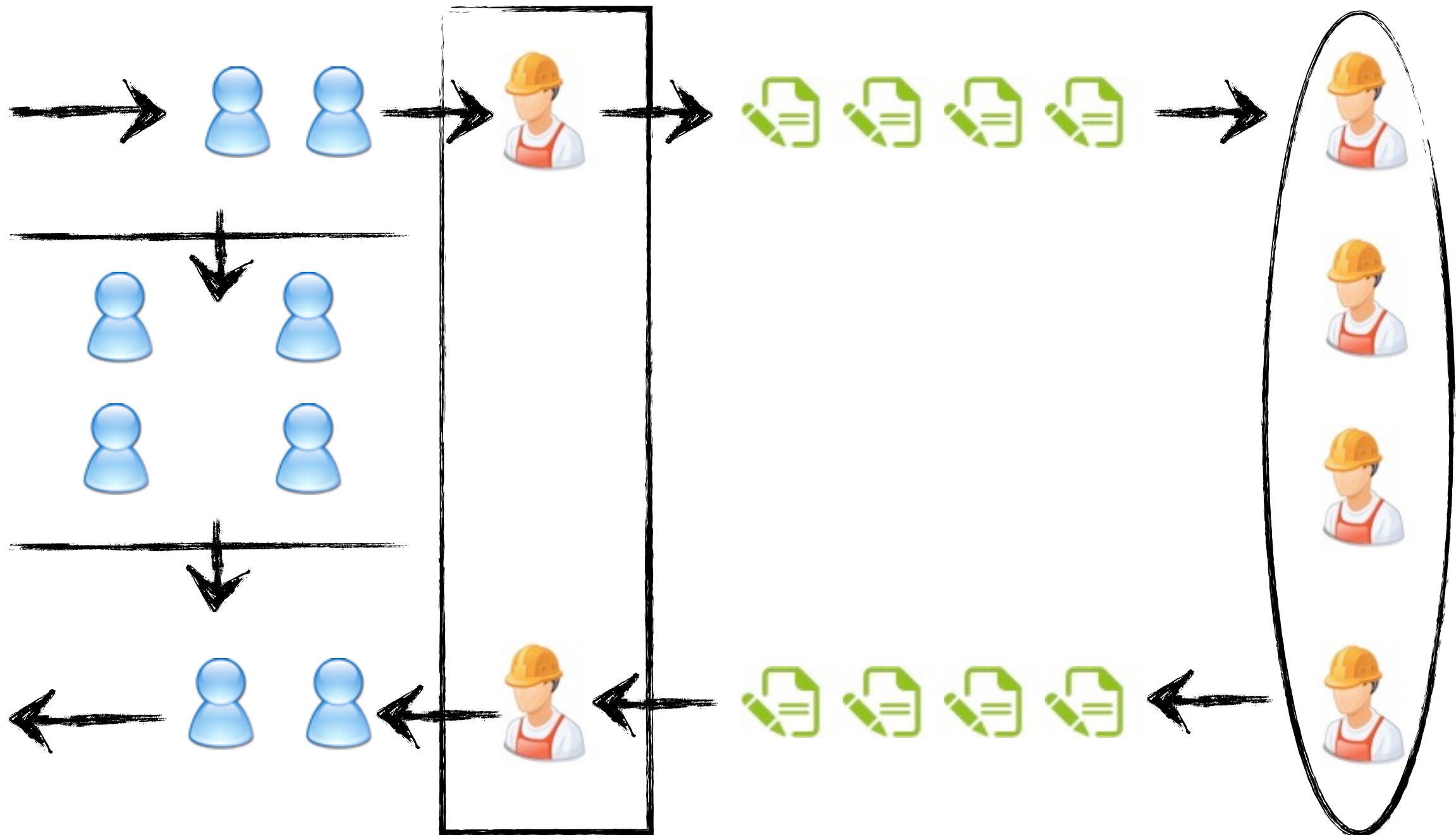
Play framework

- Simple et efficace
- philosophie basée sur Django, Rails, etc ...
 - véritable framework WEB !!!
 - Stateless, RESTful
- « Une grande partie de la complexité en Java est culturelle et non imposée » - Guillaume Bort

Pas de servlet ?



Pas de servlet ?



Play framework

- Basé sur JBoss Netty
 - Pile réseau => HTTP
 - Java NIO
- Entièrement asynchrone
 - slide précédant
 - ressources prédictibles

Un framework web

- Any Java Framework
- [http://www.myapp.com/cont/servlet/buyService?
sessionId=3gh3haaa2gg2&action=nextPage
&idExpanse=false&springWebFlow=suck
s&entityId=23bS&role=p&date=04%2F12
%2F2010&returnPage=homeS02&out=true](http://www.myapp.com/cont/servlet/buyService?sessionId=3gh3haaa2gg2&action=nextPage&idExpanse=false&springWebFlow=sucks&entityId=23bS&role=p&date=04%2F12%2F2010&returnPage=homeS02&out=true)
- Pas très web

Un framework web

- Les frameworks web Java sont créé par des développeurs Java et non des développeurs web
- Play est un framework web pour les développeurs web et non un framework pour les développeurs Java
- Java est simplement le langage choisit, cela aurait pu être autre chose

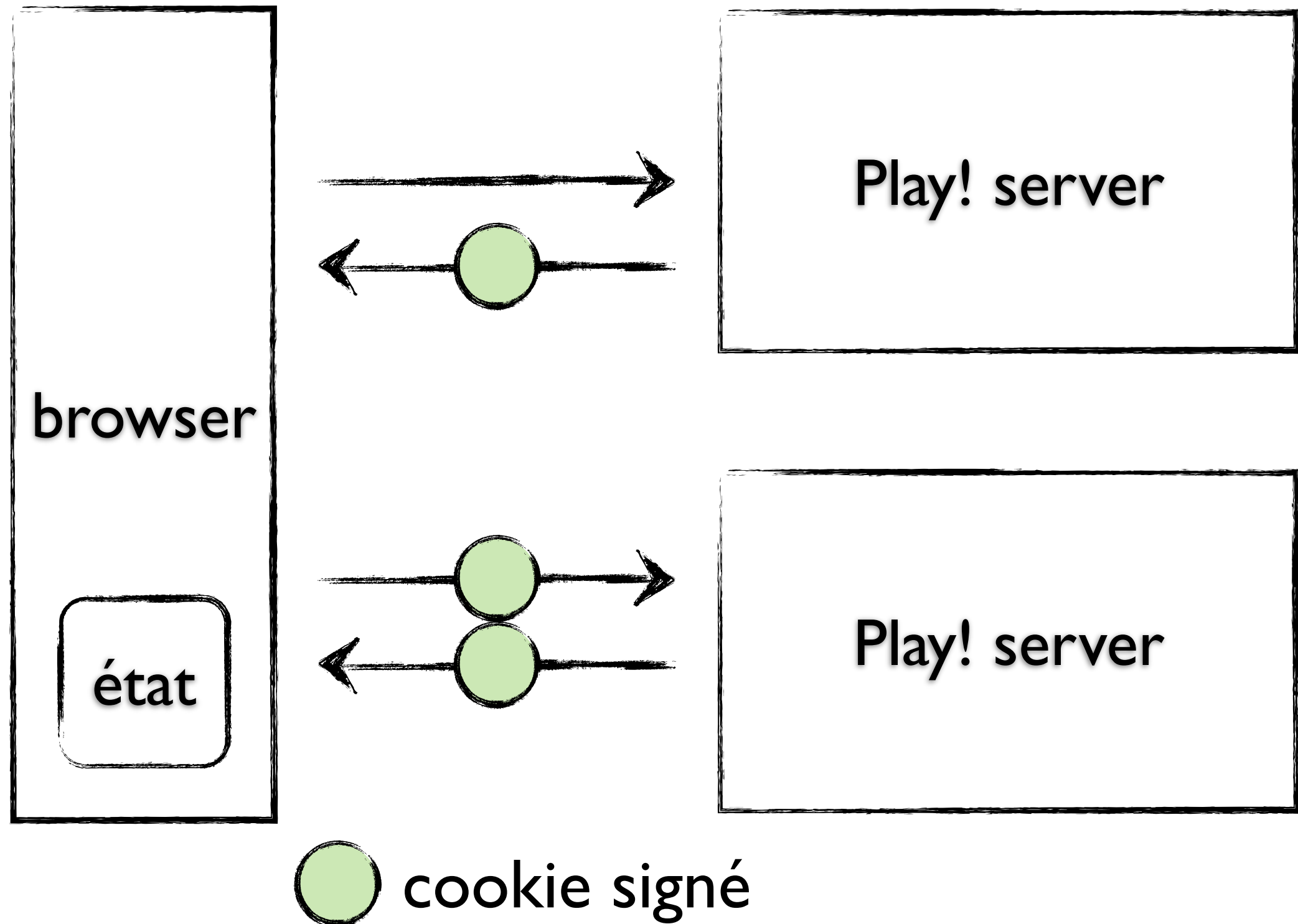
Un framework web

- Les URLs
 - <http://www.myapp.com/mycompany/user/1>
 - <http://www.myapp.com/blog/post/123>
 - [http://www.myapp.com/blog/post/123/
comment/12](http://www.myapp.com/blog/post/123/comment/12)
- lisible, bookmarkable, échangeable, etc ...

Un framework web

- Les application peuvent être RESTful
- Les URLs sont importantes
- HTTP réellement exploité dans tout son sens
- Attention au effets de bord et à l'idempotence
- Le navigateur retient l'état de l'application et non le serveur

« framework web sans état côté serveur »



Et le côté client ?

- HTML / CSS / JavaScript
 - JQuery, JQuery UI
 - ext-JS
 - GWT
 - etc ...

Projet Java classique

- Apache Tomcat
- Spring Webflow
- Spring faces
- Hibernate
- JUnit
- JSF et librairies de composants
- Selenium
- Maven

Super Hero Pattern

Projet Play

- Framework fullstack
 - les librairies utilisées fonctionnent ensemble
 - tout fonctionne ‘out of the box’
 - prend en charge la compilation, le déploiement, la mise en production

Projet Play

- Entièrement pluginifiable
- Développement incrémental
 - fast turnaround
- Moteur de persistance (JPA/Hibernate)
- Tests runners
- Email
- Async jobs
-

Expérience du développeur

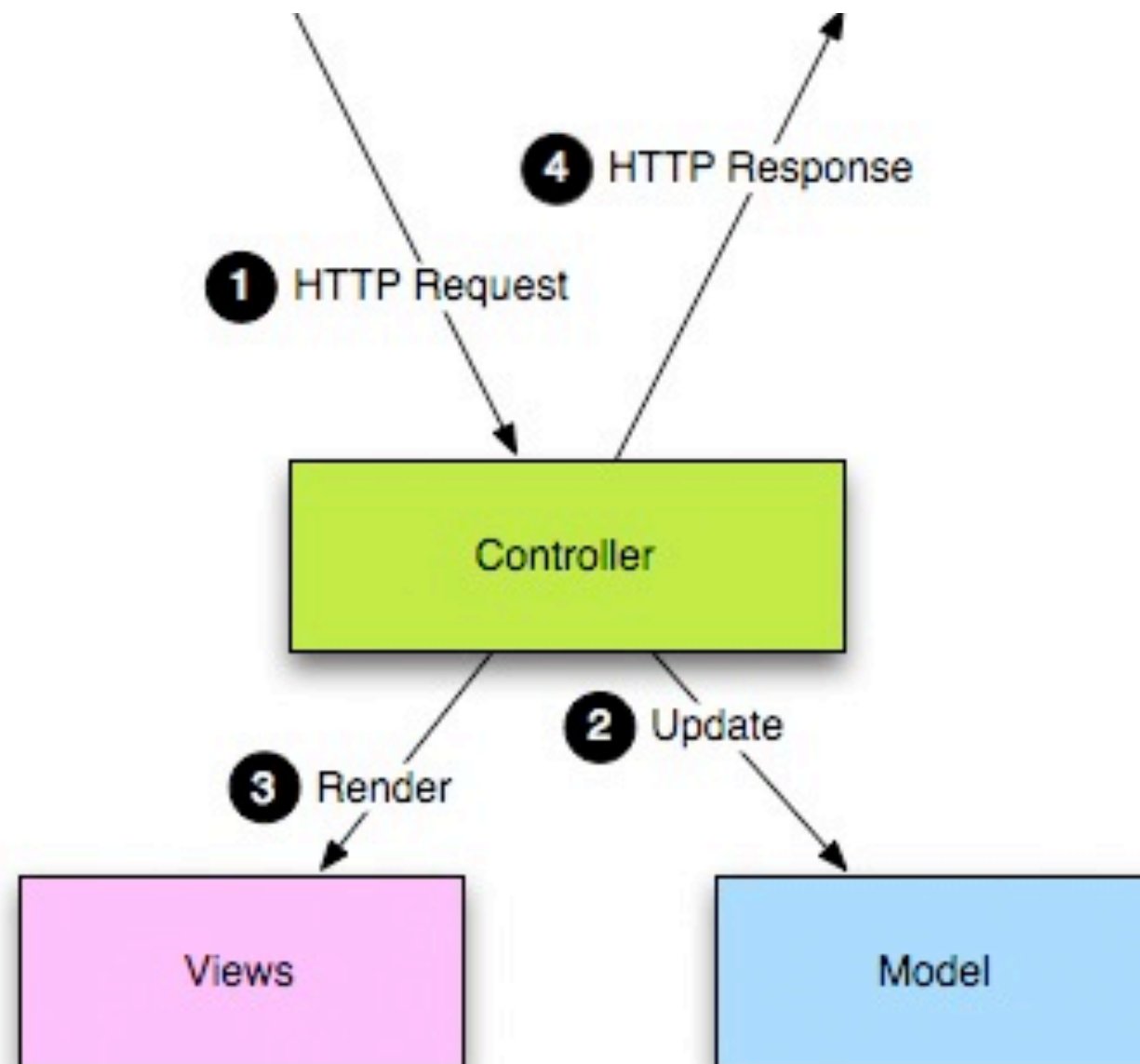
- Cycle de développement ultra rapide
 - écrire, recharger, tester
- Pas de session == redéploiement sans casse
 - test en temps réel de l'application
- Affichage des erreurs de compilation dans le browser
- Tests fonctionnels dans le browser

Play 2.0

- Sortie le 14 mars 2012
- Réécriture complète du framework
 - majoritairement en Scala
- Nouvelle API Java accessible
- Tout est typesafe, validé, compilé
- Orienté sur la programmation réactive
 - WebSockets, Comet, Data Streams, etc ...
- Plus robuste et moins de magie noire

Utilisation du framework

MVC



Cycle de développement

Application error

◀ ▶

+

http://localhost:9000/

↻

Q Google

Java compilation error

The file `/app/controllers/Application.java` could not be compiled. Error raised is : **Syntax error, insert ";" to complete BlockStatements**

In `/app/controllers/Application.java` (around line 12)

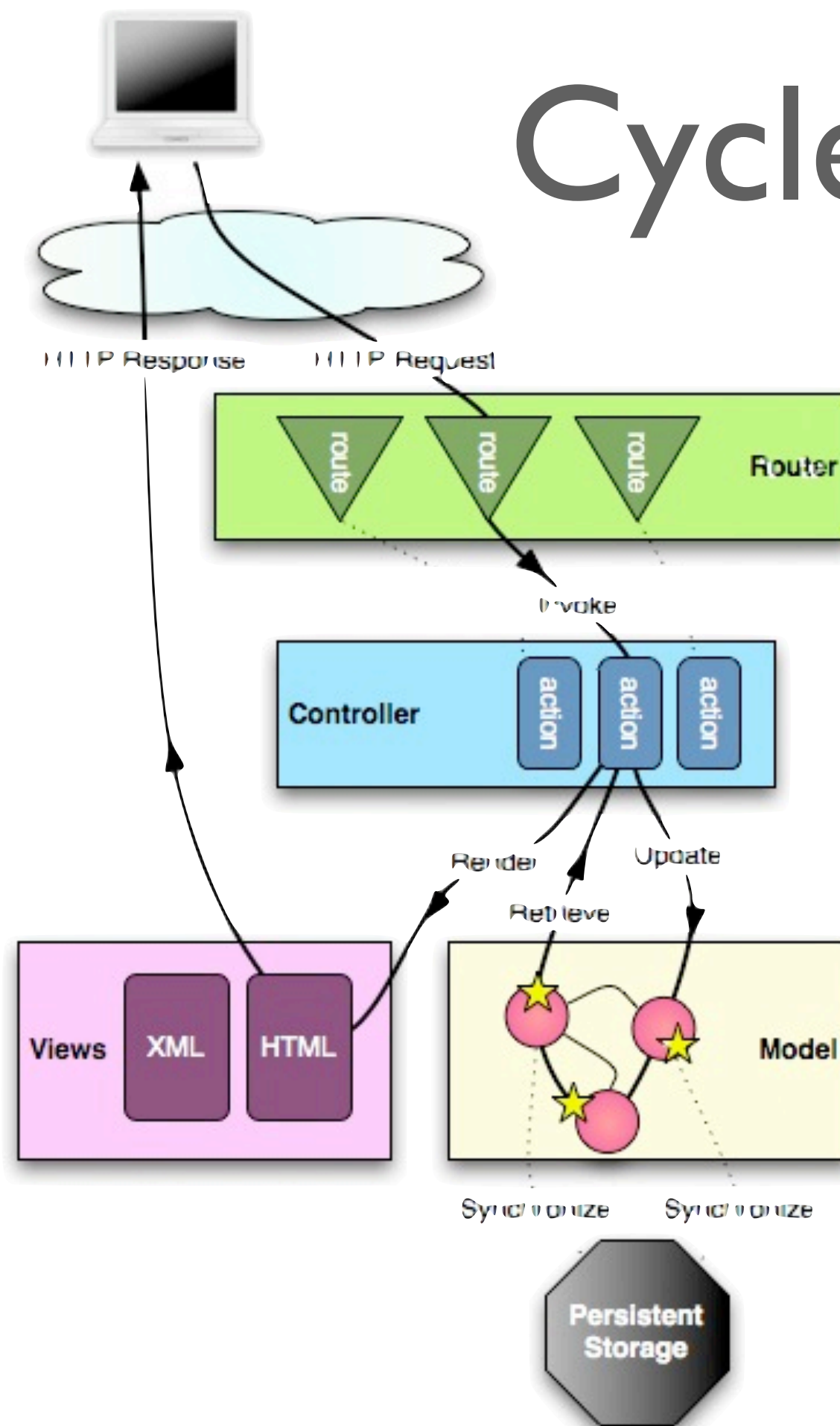
```
8:
9: public class Application extends Controller {
10:
11:     public static void index() {
12:         render()
13:     }
14:
15: }
```

This exception has been logged with id 5pmmm6494

Anatomie d'une application

- play new maSuperApplication
- maSuperApplication
 - ➔ conf
 - ➔ application.conf
 - ➔ routes
 - ➔ app
 - ➔ controllers
 - ➔ models
 - ➔ views
 - ➔ tests
 - ➔ lib
 - ➔ public

Cycle de vie



Routeage

GET	/	Application.index
GET	/hotels	Hotels.index
GET	/hotels/list	Hotels.list
GET	/hotels/{id}	Hotels.show
GET	/hotels/{id}/booking	Hotels.book
POST	/hotels/{id}/booking	Hotels.confirmBooking
DELETE	/bookings/{id}	Hotels.cancelBooking
GET	/register	Application.register
POST	/register	Application.saveUser
GET	/settings	Hotels.settings
POST	/settings	Hotels.saveSettings
POST	/login	Application.login
GET	/logout	Application.logout

Contrôleurs

- Classes se trouvant dans le package 'controllers' devant étendre 'play.mvc.Controller'
- Les méthodes pouvant être atteinte via HTTP ont pour signature 'public static void'
- Les méthodes sont appelées par rapport aux routes
- Chaque action à accès à plusieurs types de rendu
 - template, texte, xml, json, image, fichier
- Il est possible de chaîner les actions
 - redirect, invocation d'une action

Contrôleurs

```
public class Clients extends Controller {  
  
    public static void show(Long id) {  
        Client client = Client.findById(id);  
        render(client);  
    }  
  
    public static void delete(Long id) {  
        Client client = Client.findById(id);  
        client.delete();  
    }  
  
}
```

Contrôleurs

- Les paramètres d'entrées des contrôleurs
 - Ce sont les variables en GET / POST / DELETE / PUT
 - Les variables peuvent être défini dans le fichier de routes au niveau du pattern d'url (/user/{id})
 - Il n'y pas que des types simples, on peut y mettre des objets de notre modèle. Ces objets seront créée via les paramètres de la request. Exemple :
 - user.name, user.mail, ...
 - Il est également possible de passer des valeurs dans le corps de la méthode HTTP
 - via formulaire par exemple

Contrôleurs

- Les paramètres de sorties des contrôleurs
- Il suffit d'ajouter les objets à la méthode render exemple :
 - `render(resto, randomID, arg1, arg2)`
 - `render('Users/all.html', arg1, arg2)`
- Ou d'appeler la méthode renderArgs
 - `renderArgs.put('resto', resto);`

Contrôleurs

- Scopes
 - Application
 - Session
 - Stocké dans un cookie sécurisé et accessible durant toute la session de l'utilisateur (4Kb).
- Flash
 - Stocké dans un cookie sécurisé et uniquement accessible par la prochaine requête (4Kb).
- Request
- Pour tout le reste, il y a le cache !

Contrôleurs

- Possibilité de filtrer les contrôleurs
 - `@With(MonFiltre.class)`
 - `@Before`
 - `@After`
 - `@Finally`

Vues et templates

- Fichiers HTML rangés par défaut dans
 - app
 - views
 - nomDuControleur
 - nomDeLaMethode.html
 - Possible de spécifier quel template rendre
- Tags Play! pour la génération vers le client
 - Utilise Groovy en tant qu'expression language
- Support inclusion et tags customisés

Vue

- Trio gagnant
 - HTML5 / CSS3 / JQuery
- Des centaines de frameworks JavaScript
- Penser au templating JavaScript
- Cependant essayer de ne pas trop faire dans l'exotisme
 - pensez à celui qui devra maintenir l'application

Vue

- Animation
 - Quelques frameworks JavaScript
 - Transitions CSS3
 - ajout de classes via manipulation DOM
 - HTML5 Canvas :-)
- N'importe quelle IHM capable de consommer des services REST HTTP
- GWT, etc ...

Vue

- Stateful => état maintenu sur le client
 - Cookies
 - Session Storage
 - Local Storage

Templates

- `${...}` : expression
 - `${user.name}`
- `&{...}` : message il 8n
- `@{...}` : routes
 - `@{MonControleur.MaMethode()}`
 - `@@{MonControleur.MaMethode()}`
- `*{...}* :` commentaires
- `%{}%` : scripts groovy

Décorateurs

```
{% extends 'simpladesign.html' %}
```

```
{% set title: 'A decorated page' %}
```

This content will be decorated.

```
<html>
  <head>
    <title>{% get 'title' %}</title>
  </head>
  <body>
    <h1>{% get 'title' %}</h1>
    {% doLayout %}
    <div class="footer">play! framework</div>
  </body>
</html>
```

Tags

- Fragments HTML que l'on peut appeler avec des paramètres
- Fichiers html situés dans app/views/tags
 - pour app/views/tags/hello.html
 - `Hello ${_name} !`
 - `<h1>#{hello name: 'Bob' }</h1>`

Ajax

- Très simple de faire des applications avec IHM réactive
 - Action retournant du texte, du json ou de simples codes HTTP
- Utilisation massive de techniques dites AJAX
 - GET ou POST asynchrones réalisés via JavaScript
 - Préférer l'utilisation d'un framework tel que jquery

Modèles

- Classe représentant une entité JPA (annotée avec `@Entity`) et étendant la classe `play.db.jpa.Model`, située dans le package `models`
- Définition des relations via les annotations JPA
 - `@OneToOne`, `@OneToMany`, `@ManyToOne`, `@ManyToMany`
- Contrairement à du JPA classique, il est possible de déclarer tous les membres en public
 - Play génère les getters/setters au runtime
- La classe `Model` est un helper pour le requêtage

Modèles

```
@Entity
```

```
public class Post extends Model {  
    public String title;  
    public String author;  
}
```

```
Post aPost = Post.findById(5L);
```

```
List<Post> posts = Post.findAll();
```

```
List<Post> posts = Post.all().from(50).fetch(100);
```

```
Post.find("byTitle", "My first post").fetch();
```

```
Post.find("byTitleLike", "%hello%").fetch();
```

```
Post.find("byAuthorIsNull").fetch();
```

```
Post.find("byTitleLikeAndAuthor", "%hello%", connectedUser).fetch();
```

```
Post.find("title", "My first post").fetch();
```

```
Post.find("title like ?", "%hello%").fetch();
```

```
Post.find("author is null").fetch();
```

```
Post.find("title like ? and author is null", "%hello%").fetch();
```

```
Post aPost = Post.findById(5L);
```

```
aPost.author = "John Doe";
```

```
aPost.save();
```

```
aPost.delete();
```

Validation

- Les modèles et les paramètres d'actions sont soumis à validation.
- Pour l'activer, il suffit d'ajouter les annotations de validation de Play!
- Tags `#{ifErrors}` `#{/ifErrors}`

```
@Entity
public class Post extends Model {
    @Required @Size(20)
    public String title;
    public String author;
}

    public static void show(@Required Long id) {
        Client client = Client.findById(id);
        render(client);
    }
```

Jobs

```
package jobs;
```

```
import play.jobs.*;
```

```
public class VideoEncoder extends Job {
```

```
    public void doJob() {  
        // execute logic here ...  
    }
```

```
}
```

```
public static void encode(Long id) {
```

```
    new VideoEncoder(id).now();  
    renderText("Encoding started");
```

```
}
```

Jobs

```
package jobs;
```

```
import play.jobs.*;
```

```
@OnApplicationStart
```

```
public class VideoEncoder extends Job {
```

```
    public void doJob() {
```

```
        // execute logic here ...
```

```
    }
```

```
}
```

Jobs

```
package jobs;
```

```
import play.jobs.*;
```

```
@OnApplicationStop
```

```
public class VideoEncoder extends Job {
```

```
    public void doJob() {
```

```
        // execute logic here ...
```

```
    }
```

```
}
```

Jobs

```
package jobs;
```

```
import play.jobs.*;
```

```
@Every("1h")
```

```
public class VideoEncoder extends Job {
```

```
    public void doJob() {
```

```
        // execute logic here ...
```

```
    }
```

```
}
```

Jobs

```
package jobs;

import play.jobs.*;

/** Fire at 12pm (noon) every day */
@On("0 0 12 * * ?")
public class VideoEncoder extends Job {

    public void doJob() {
        // execute logic here ...
    }
}
```

Cache

```
public static void allProducts() {  
    List<Product> products =  
        Cache.get("products", List.class);  
    if(products == null) {  
        products = Product.findAll();  
        Cache.set("products",  
                  products, "30mn");  
    }  
    render(products);  
}
```


Cache

```
public static void showProduct(String id) {  
    Product product = Cache.get("product_" + id, Product.class);  
    if(product == null) {  
        product = Product.findById(id);  
        Cache.set("product_" + id, product, "30mn");  
    }  
    render(product);  
}  
  
public static void addProduct(String name, int price) {  
    Product product = new Product(name, price);  
    product.save();  
    showProduct(product.id);  
}
```

Cache

```
public static void editProduct(String id, String name, int price) {  
    Product product = Product.findById(id);  
    product.name = name;  
    product.price = price;  
    Cache.set("product_" + id, product, "30mn");  
    showProduct(id);  
}
```

```
public static void deleteProduct(String id) {  
    Product product = Product.findById(id);  
    product.delete();  
    Cache.delete("product_" + id);  
    allProducts();  
}
```

Tests

- Tests unitaires basés sur JUnit
- Tests fonctionnels basés sur Selenium
- Joués dans le navigateur
 - Peuvent être joué en mode offline
 - navigateur en mémoire basé sur HTMLUnit

Tests

```
public class ApplicationTest extends FunctionalTest {

    @Test
    public void testTheHomePage() {
        Response response = GET("/");
        assertStatus(200, response);
    }

    @Test
    public void testUserIsFoundAndPassedToView() {
        Response response =
            POST("/user/find?name=mark&dob=18011977");
        assertThat(renderArgs("user"), is(notNullValue()));
        User user = (User) renderArgs("user");
        assertThat(user.name, is("mark"));
    }
}
```

Tests

```
# {selenium 'Test security'}

// Try to log in the administration area
clearSession()
open( '/admin' )
assertTextPresent( 'Login' )
type( 'login', 'admin' )
type( 'password', 'secret' )
clickAndWait( 'signin' )
// Verify that the user is correctly logged in
assertText( 'success', 'Welcom admin!' )

# {/selenium}
```

Tests

- Fixtures
- Fichiers de description des données au format YAML

```
Company( google ) :  
    name:      Google
```

```
Company( zen ) :  
    name:      Zenexity
```

```
User( guillaume ) :  
    name:      guillaume  
    company:   zen
```

```
public void setUp() {  
    Fixtures.deleteAll();  
    Fixtures.loadModels( "data.yml" );  
}
```

Conclusion

Go Play!

<http://www.playframework.org/documentation/1.2.4/home>