

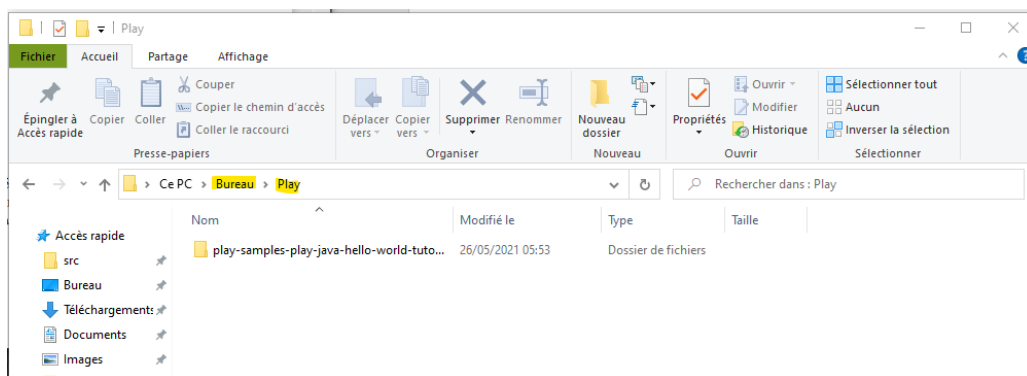
TD1 : WEB DYNAMIQUE

- Projet 1 : Premiers pas

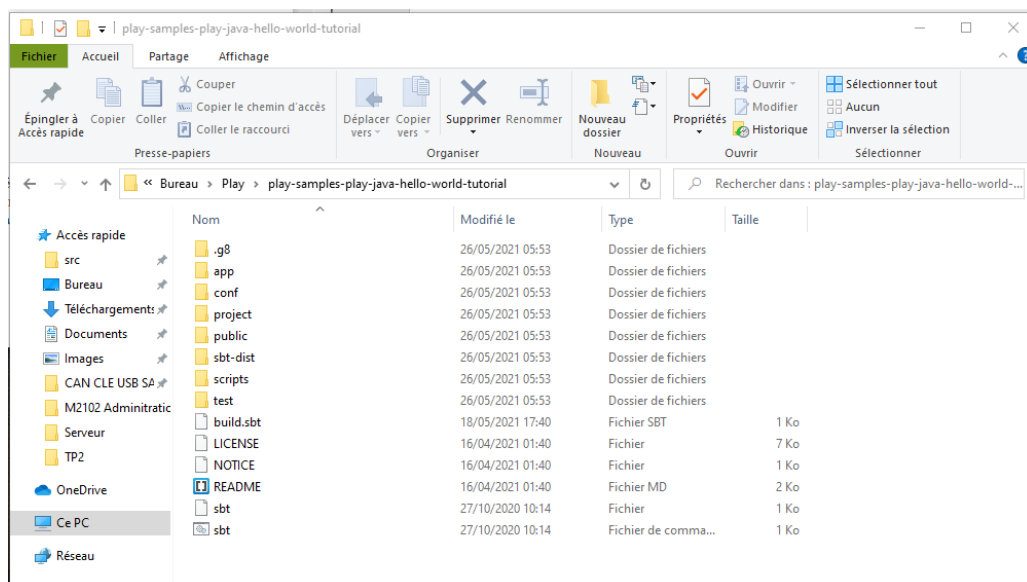
1.1) Installation de l'environnement

On a bien installé « Play » de moodle sur le bureau qu'on a décompressé par la suite.

(Captures d'écran faite sur ordinateur personnelle)



Archive a bien été décompressé dans le dossier « Play » :



- *Memento* :

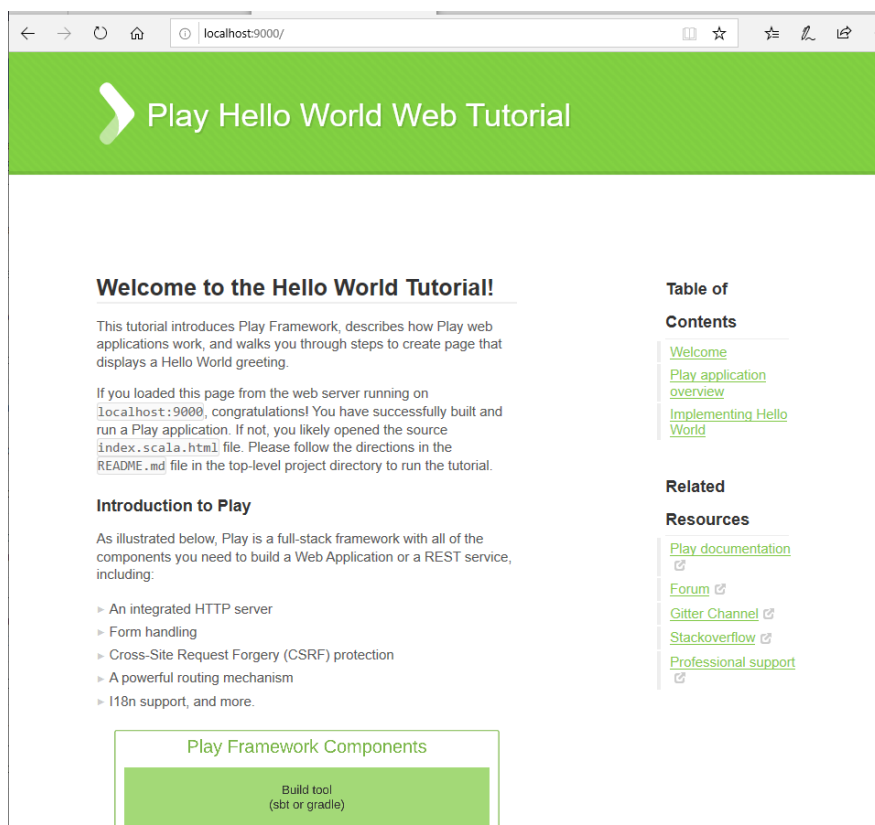
App	Dossier où se concentre toute la partie html/java
Conf	Dossier où se trouve les routes (définies les routes des applications), logback et application.conf
lib	
public	Dossier où l'on retrouve les images de l'application, les javascripts.
Test	Dossier où l'on peut effectuer des modifications sur un exemple de script.

1.2) Lancer l'application

```
C:\Users\Utilisateur>cd play-samples-play-java-hello-world-tutorial
C:\Users\Utilisateur\play-samples-play-java-hello-world-tutorial>
```

Pour vérifier que toutes les commandes tapées sont justes, on tape sur le moteur de recherche l'adresse : <https://localhost:9000>

Et on trouve bien ce résultat !



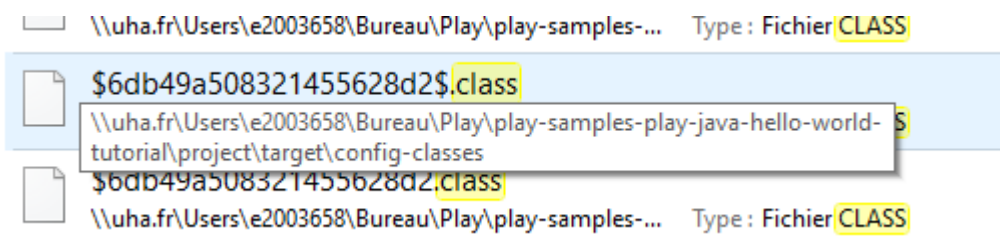
- *Quoi ?*

A partir du terminal de commandes d'une machine physique, on a pimerai alors lancer une application.

- *Comment ?*

Dans un terminal de commande windows, on a donc parcouru le répertoire (cd .playHelloWorldWebTutorial) du dossier Play dans lequel on a pu exécuter des commandes tel que « run » pour accéder à l'application sur un site web.

- *Où sont stockés les fichiers .class ?*



Les fichiers classe sont stockés sur <\\uha.fr\Users\e2003658\Bureau\Play\play-samples-play-java-hello-world-tutorial\project\target\config-classes>

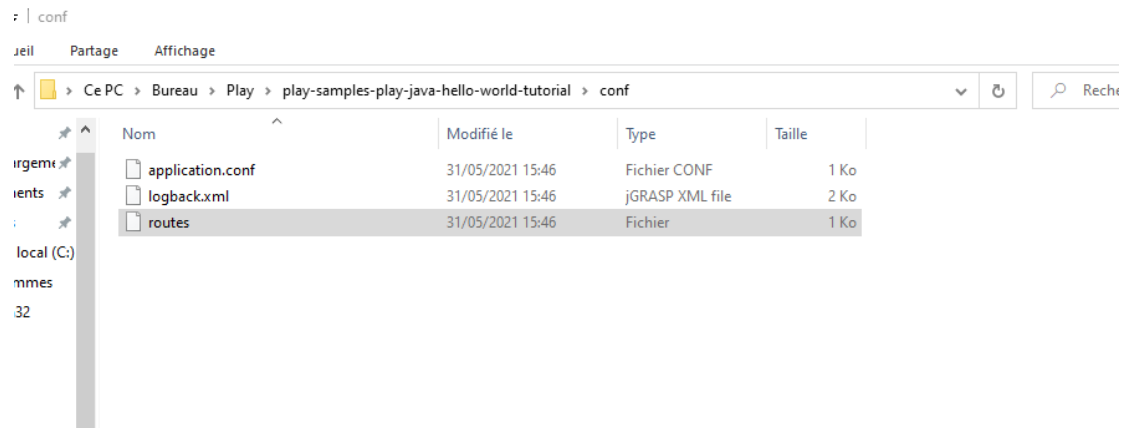
- *3 caractères non-reconnus par l'encodage UTF8 utilisé par Play :*
 - Accent grave
 - Accent aigüe
 - Accent circonflexe

```
--- (Running the application, auto-reloading is enabled) ---
[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:9000
(Server started, use Enter to stop and go back to the console...)
[info] Compiling 9 Scala sources and 2 Java sources to U:\Bureau\Play\play-samples-play-java-hello-world-tutorial\target\scala-2.13\classes ...
[info] Non-compiled module 'compiler-bridge_2.13' for Scala 2.13.6. Compiling...
[info] Compilation completed in 7.554s.
[info] p.a.h.EnabledFilters - Enabled Filters (see <https://www.playframework.com/documentation/latest/Filters>):
  play.filters.csrf.CSRFFilter
  play.filters.headers.SecurityHeadersFilter
  play.filters.hosts.AllowedHostsFilter
[info] play.api.Play - Application started (Dev) (no global state)
```

- *Quelle est cette route choisie ?*

Pour savoir quelle est la route choisie, on doit aller dans ce **PC → Bureau → Play → play-samples-play-java-hello-world-tutorial → conf → routes**.

GET /RouteChoisie



On ouvre le fichier « routes » avec le mode « bloc-notes » et on obtient donc le résultat suivant :

```
# Routes
# This file defines all application routes (Higher priority routes first)
# ~~~~

# An example controller showing a sample home page
GET    /                controllers.HomeController.index
GET    /explore         controllers.HomeController.explore
GET    /tutorial        controllers.HomeController.tutorial

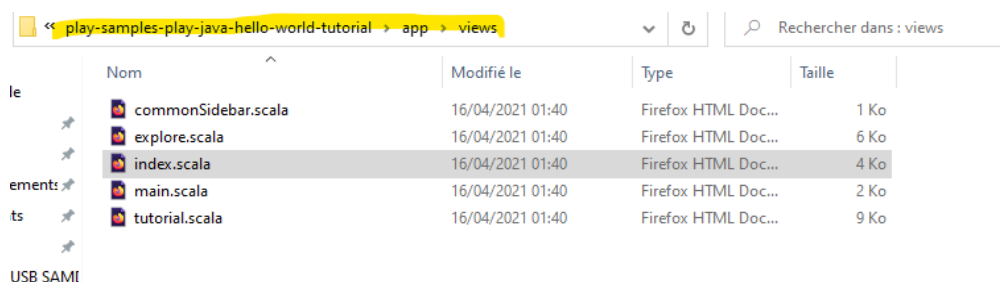
# Map static resources from the /public folder to the /assets URL path
GET    /assets/*file    controllers.Assets.versioned(path="/public", file: Asset)
```

On peut donc remarquer 3 « GET /RouteChoisie » :

- GET / : route permettant d'accéder à l'accueil (index)
- GET /explore : route permettant d'accéder au fichier explore
- GET /tutorial : route permettant d'accéder au fichier tutorial

- On veut lire le viewer (fichier *index.scala.html* dans *app/view/*) :

Pour lire le viewer, on doit se rendre dans le dossier « **Play** » → **play-samples-play-java-hello-world-tutorial → app → views**



Puis, on ouvre le fichier « index.scala » sur Brackets (par exemple) où on va modifier la ligne 10, soit « Welcome to Hello World Tutorial » que l'on remplace par « Je suis le viewer et j'obéis ».

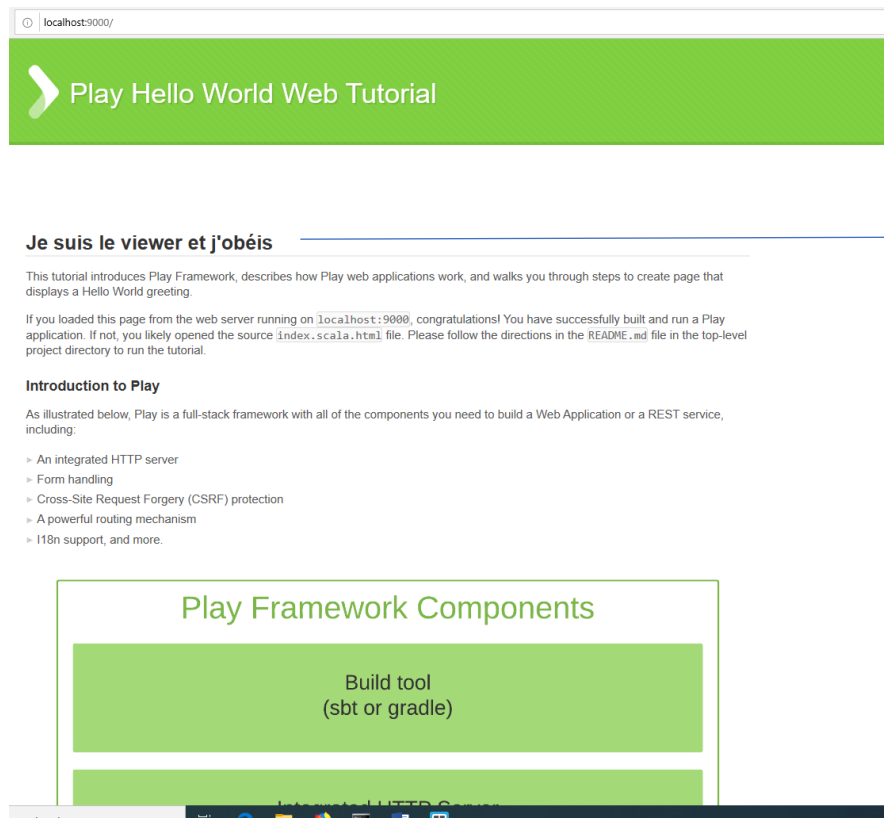
```

1 @()
2
3
4 @main("Welcome") {
5   @defining(play.core.PlayVersion.current) { version =>
6
7     <section id="content">
8       <div class="wrapper doc">
9         <article>
10           <h2>Je suis le viewer et j'obéis</h2>
11           <p>This tutorial introduces Play Framework, describes how Play web applications work, and walks you through steps
12             to create page that displays a Hello World greeting.</p>
13           <p>If you loaded this page from the web server running on <code>localhost:9000</code>, congratulations! You have
14             successfully built and run a Play application. If not, you likely opened the source <code>index.scala.html</code>
15             file. Please follow the directions in the <code>README.md</code> file in the top-level project directory to run
16             the tutorial.</p>
17
18           <h3 id="introduction">Introduction to Play</h3>
19           <p>As illustrated below, Play is a full-stack framework with all of the components you need to build a Web
20             Application or a REST service, including:</p>
21           <ul>
22             <li>An integrated HTTP server</li>
23             <li>Form handling</li>
24             <li>Cross-Site Request Forgery (CSRF) protection</li>
25             <li>A powerful routing mechanism</li>
26             <li>Libraries support, and more.</li>
27           </ul>
28
29           
30
31           <p>Play integrates with many object relational mapping (ORM) layers. It has out-of-the-box support for <a
32             href="https://www.playframework.com/documentation/@version/Anorm"
33             target="_blank">Anorm</a>, <a href="https://www.playframework.com/documentation/@version/JavaEbean" target=

```

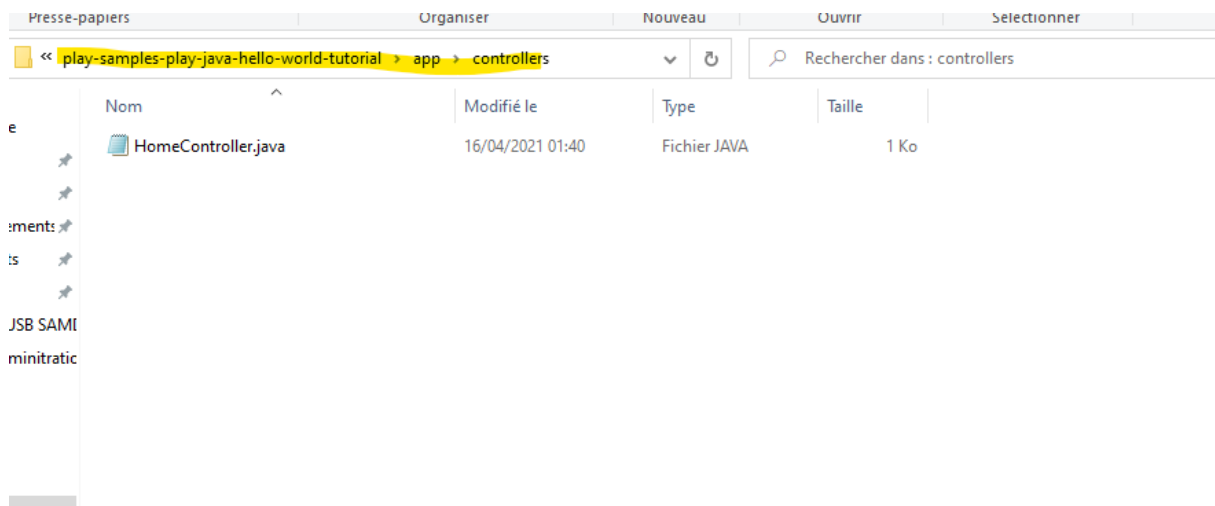
On enregistre les commandes, on ferme et on rafraîchit le site web : localhost :9000

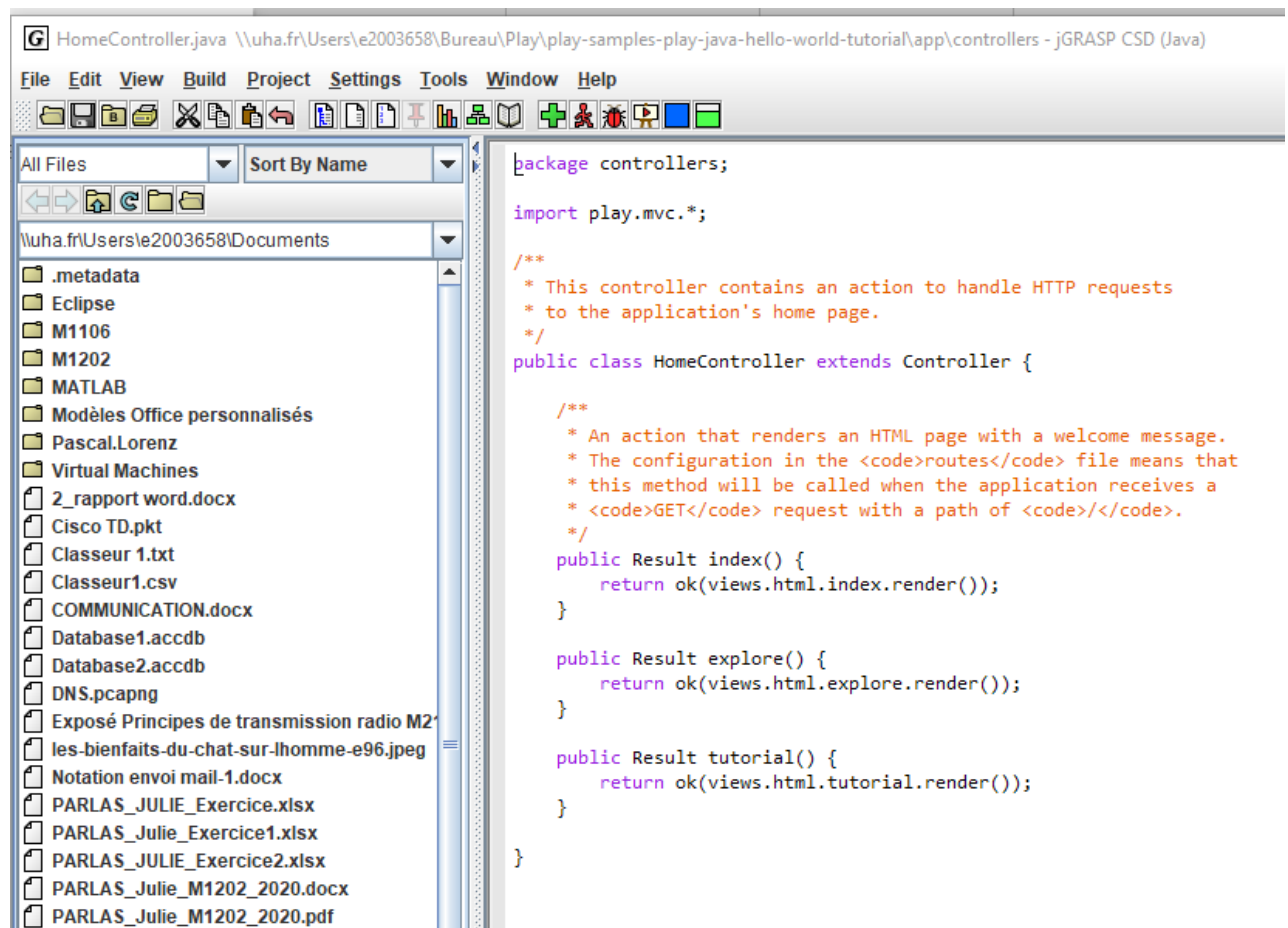
On obtient le résultat suivant :



Fichier *HomeController.java* dans *app/controllers*

➔ Dans le dossier « Play » ➔ play-samples-play-java-hello-world-tutorial ➔ app ➔ controllers





- *Que signifie l'aspect statique ?*

On appelle une méthode statique toute méthode utilisée par tout utilisateur de la classe.

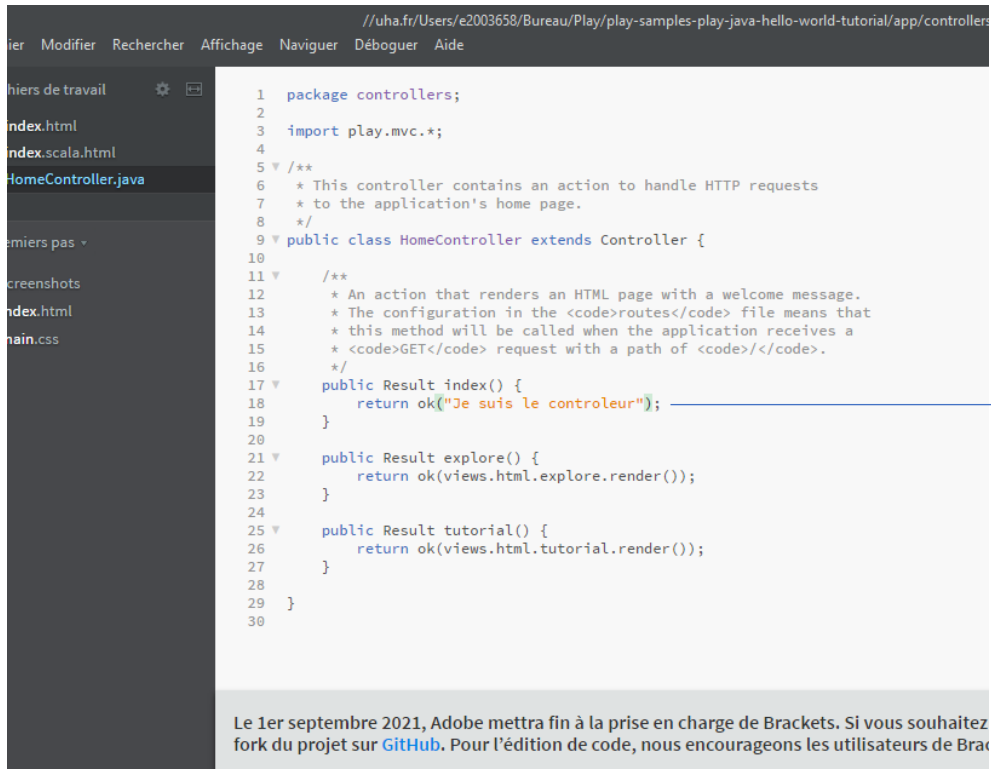
- *Que fait la méthode render() ?*

La méthode render() définit le rendu (affichage) des valeurs.

- Afficher le contrôleur (fichier `app/HomeControllers.java`) :

Donc, on va dans le dossier Play → app → Controllers → HomeController.java

On modifie ensuite la ligne concernée.

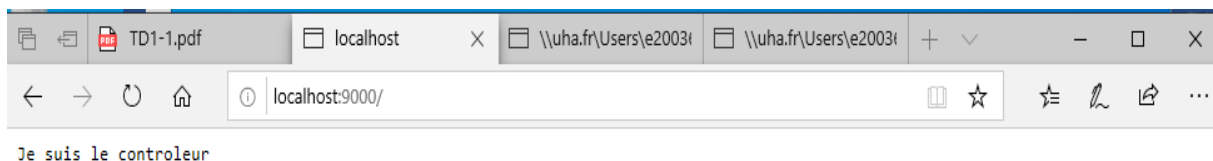


```
1 package controllers;
2
3 import play.mvc.*;
4
5 /**
6  * This controller contains an action to handle HTTP requests
7  * to the application's home page.
8  */
9 public class HomeController extends Controller {
10
11     /**
12      * An action that renders an HTML page with a welcome message.
13      * The configuration in the <code>routes</code> file means that
14      * this method will be called when the application receives a
15      * <code>GET</code> request with a path of <code>/</code>.
16      */
17     public Result index() {
18         return ok("Je suis le controleur");
19     }
20
21     public Result explore() {
22         return ok(views.html.explore.render());
23     }
24
25     public Result tutorial() {
26         return ok(views.html.tutorial.render());
27     }
28 }
29
30
```

Le 1er septembre 2021, Adobe mettra fin à la prise en charge de Brackets. Si vous souhaitez fork du projet sur [GitHub](#). Pour l'édition de code, nous encourageons les utilisateurs de Brackets de passer à VS Code.

Par exemple,
on modifie
cette ligne-là
par « *Je suis le
contrôleur* »

On rafraichit la page localhost :9000 et on obtient donc le résultat suivant :



La page a donc bien été pris en compte le message et uniquement le message !

Auto-évaluation :

Travail réalisé	Noté sur	Ma note	Note du professeur
Réponses écrites	5	3	
Correspondance controller/vue	3	2	
Partie html (pratique)	3	2.5	
Partie css (pratique)	3	2.5	
Hyperliens proposés (pratique)	3	2	
Code bien écrit (pratique)	3	2	
Travail sur le Git (point bonus)	2	2	
Auto-évaluation correcte (point bonus)	1	0	