Création de sa première application Web avec Spring Boot

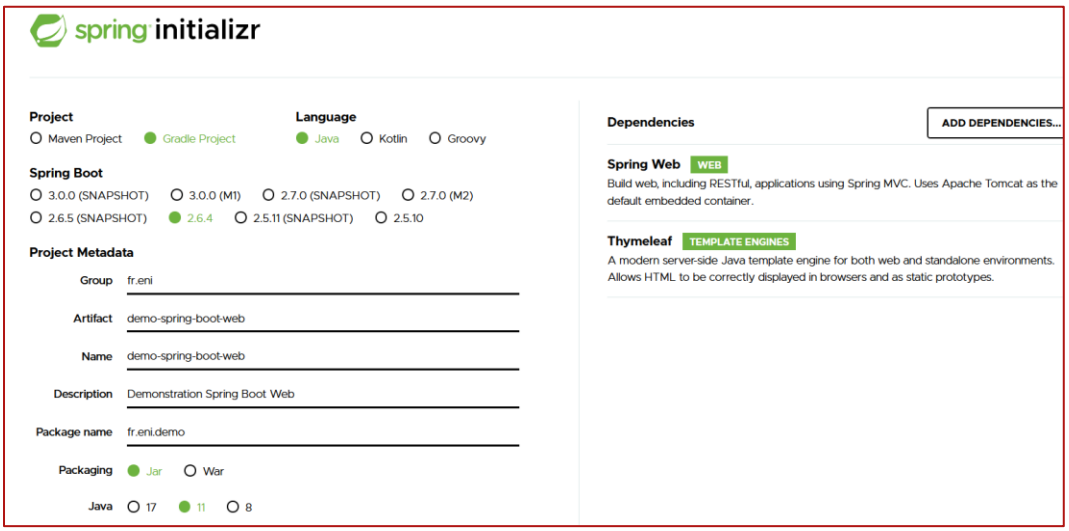
Démonstration 3 du module 2

L'objectif de cette démonstration est de créer une première application avec Spring Boot

Déroulement

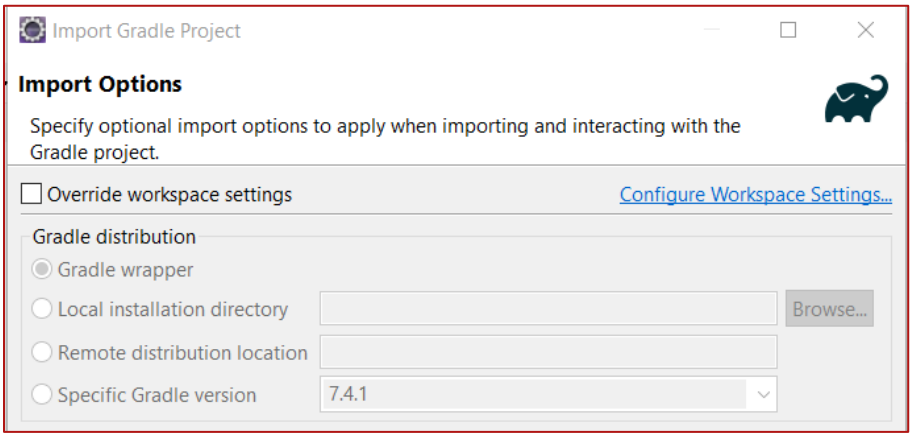
Création d'un projet Spring Boot

- Aller sur le site : https://start.spring.io/
- Sélectionner :
  - Gradle Project (Gradle est le gestionnaire de dépendances que nous allons utiliser)
  - Java 11
  - Pour ce cours, nous utilisons la version la plus récente de Spring Boot GA : 2.6.4
  - Group : fr.eni
  - Artifact : demo-spring-boot-web
  - Package name : fr.eni.demo
- Le livrable sera créé sous la forme d'un .jar et non d'un .war ce qui facilite le déploiement
- Ajouter les librairies : Spring Web et Thymeleaf

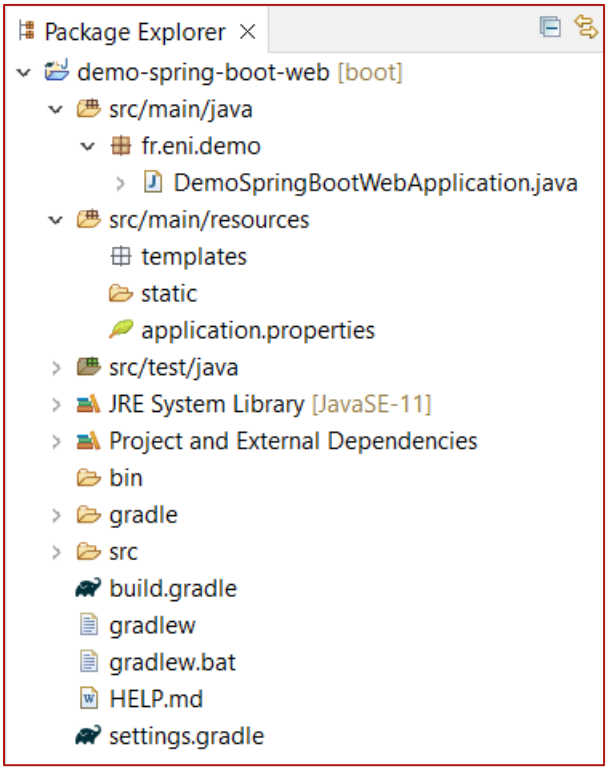


Intégration du projet dans Eclipse

- Copier le zip dans le répertoire du Workspace d'Eclipse et dézipper le.
- Faire « Import », sélectionner « Existing Gradle Project » dans Eclipse
    - Sélectionner le projet dans le Workspace
    - Vérifier la configuration de Gradle en 7.4.1 :



    - Et valider



Création d'un premier contrôleur

- Créer la classe WelcomeController dans le package : fr.eni.demo.controller
- Voici le code de la classe :

```
package fr.eni.demo.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class WelcomeController {

	@GetMapping("/")
	public String welcome() {
		return "welcome";
	}
}
```

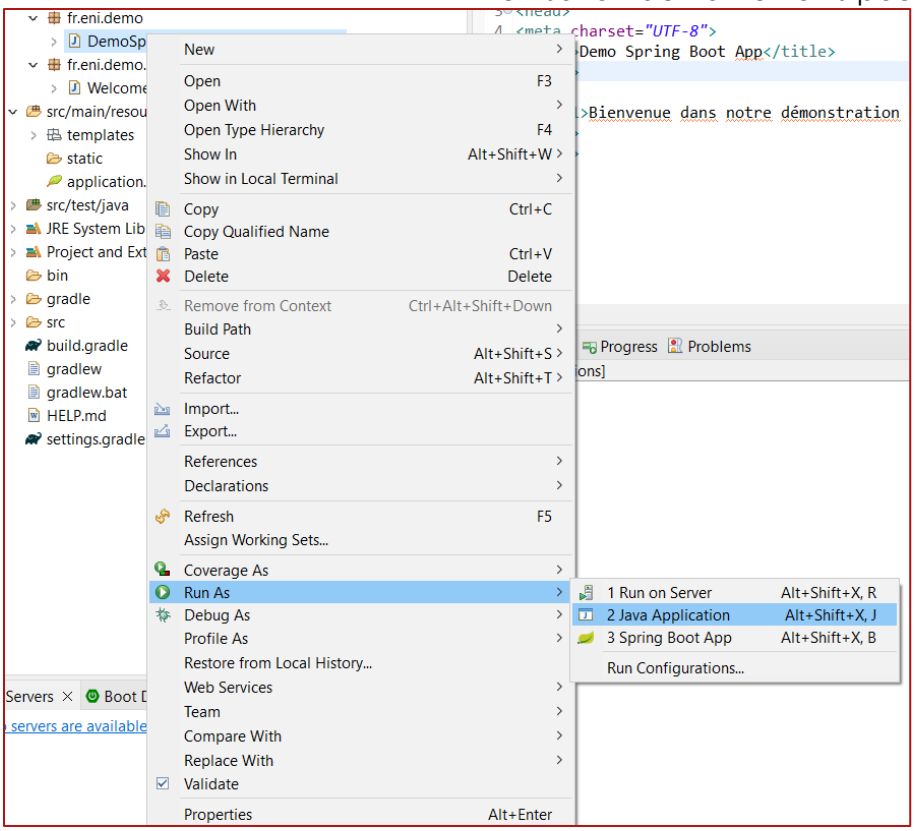
Création d'une première vue

- Dans le dossier templates, créer la page welcome.html :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Boot App</title>
</head>
<body>
	<h1>Bienvenue dans notre démonstration Spring Boot</h1>
</body>
</html>
```

Exécution

- Lancer l'exécution de l'application en exécutant la classe : DemoSpringBootWebApplication
  - Soit avec « Run Java Application »
  - Soit avec « Run d »



- Il n'y a pas de différence, Spring Boot va lancer un serveur Tomcat embarqué dans les 2 cas.
- Voici le type de traces de démarrage :

- Tester l'application avec l'URL : http://localhost:8080

Ajout du starter « Spring Boot Dev Tools »

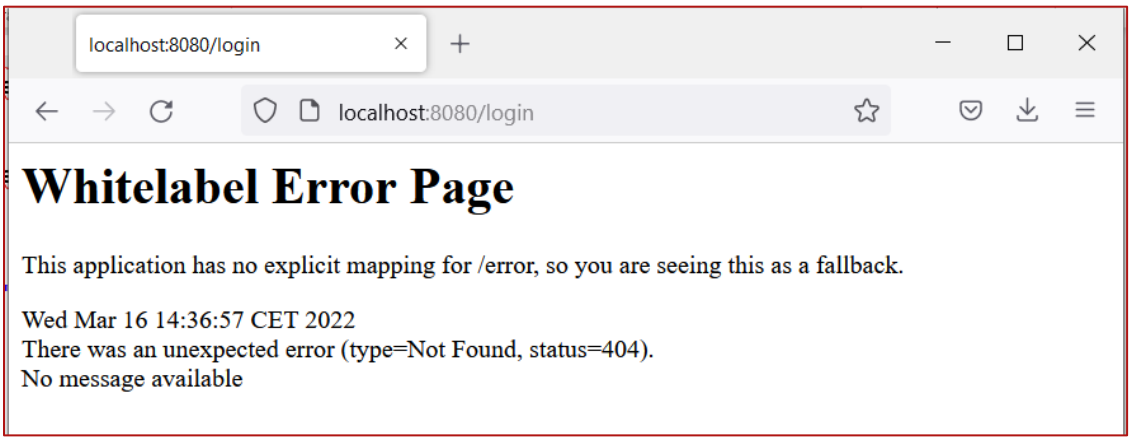
- Dupliquer le template welcome.html en login.html
    - Modifier le contenu de la page :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Boot App</title>
</head>
<body>
	<h1>Connection à l'application de démonstration</h1>
</body>
</html>
```

- Dans le contrôleur, ajouter une méthode pour appeler ce template :

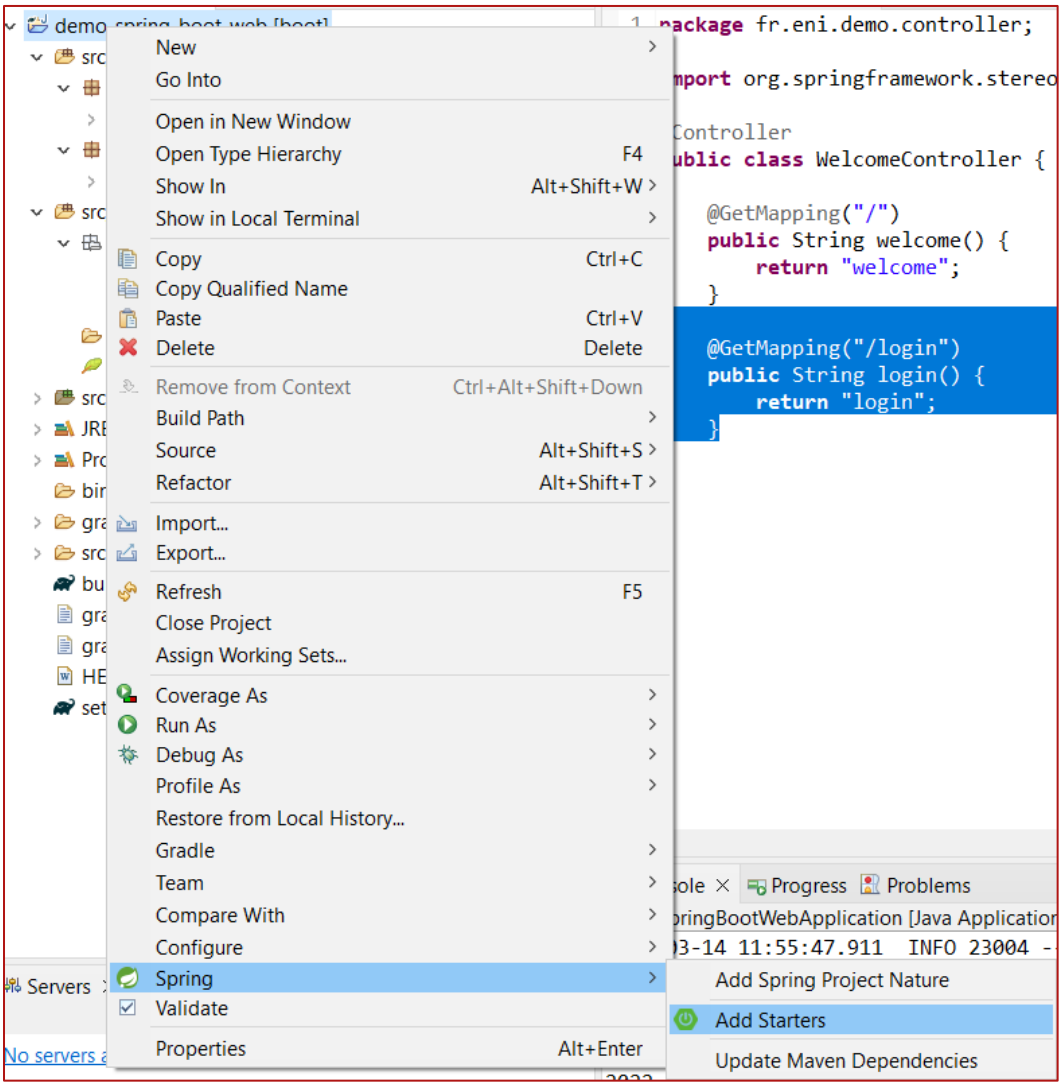
```
package fr.eni.demo.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class WelcomeController {

	@GetMapping("/")
	public String welcome() {
		return "welcome";
	}


	@GetMapping("/login")
	public String login() {
		return "login";
	}
}
```

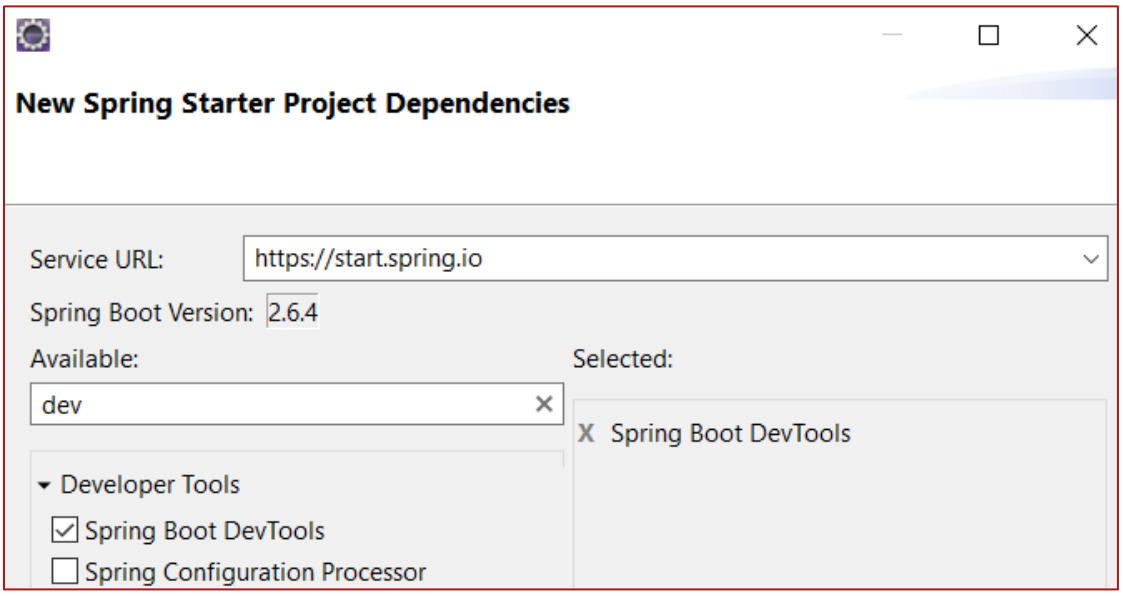
- Appeler l'URL http://localhost:8080/login



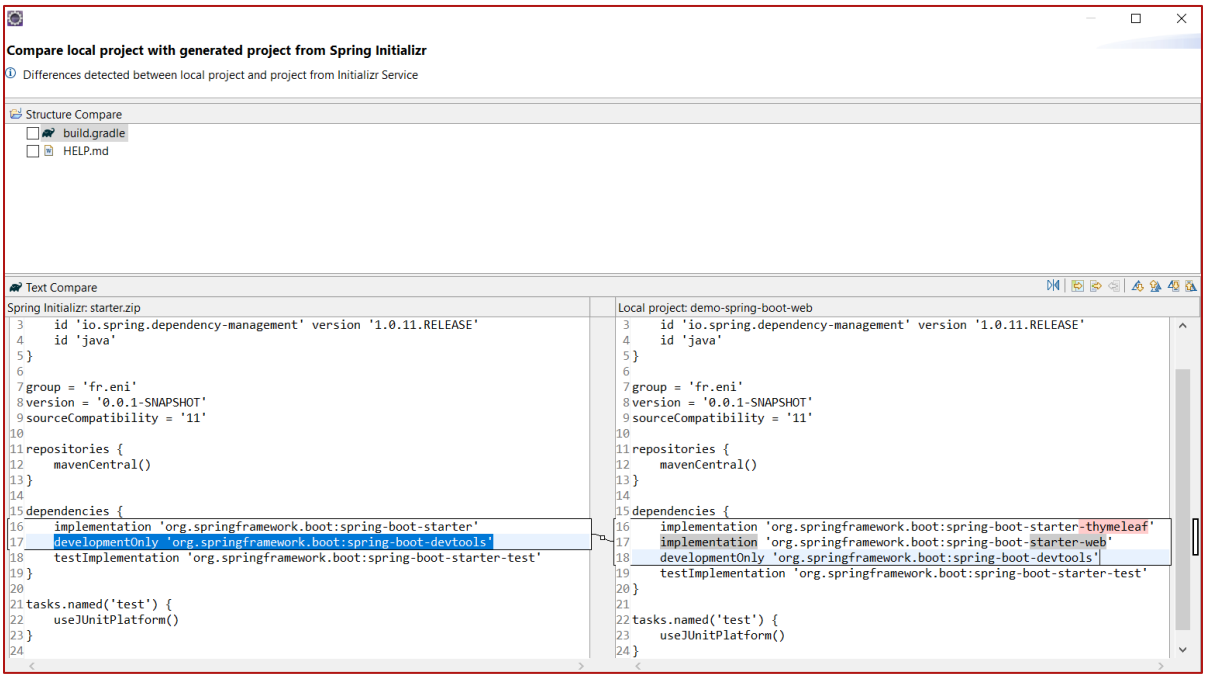
- Le serveur ne se redéploye pas automatiquement

- Ajouter le starter « Spring Boot Dev Tools » pour permettre une mise à jour automatique du déploiement
    - Clic droit sur le projet → Spring → Add Starters





- Attention, à l'étape suivante, l'outil va vouloir remplacer les dépendances précédentes par la nouvelle
    - A la place, il faut merger les dépendances



- Le fichier build.gradle :

```
plugins {
	id 'org.springframework.boot' version '2.6.4'
	id 'io.spring.dependency-management' version '1.0.11.RELEASE'
	id 'java'
}

group = 'fr.eni'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
	mavenCentral()
}

dependencies {
	/*compileOnly - put the dependency on the compile classpath only
	  runtimeOnly - put the dependency on the runtime classpath only
	 implementation - put the dependency on both classpaths*/

	implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
	implementation 'org.springframework.boot:spring-boot-starter-web'
	developmentOnly 'org.springframework.boot:spring-boot-devtools'
	testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test') {
	useJUnitPlatform()
}
```

- Relancer l'application Spring une fois le starter rajouter
    - Sans surprise, le redémarrage fait fonctionner maintenant l'URL http://localhost:8080/login

    - Modifier à nouveau le contrôleur, avec comme URL

```
@GetMapping("/connexion")
```

    - Enregistrer le contrôleur, constater le rechargement automatique du contexte dans la console :

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.4)

2022-03-14 12:35:21.099  INFO 22372 --- [  restartedMain] f.eni.demo.DemoSpringBootWebApplication
2022-03-14 12:35:21.099  INFO 22372 --- [  restartedMain] f.eni.demo.DemoSpringBootWebApplication
2022-03-14 12:35:21.306  INFO 22372 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2022-03-14 12:35:21.306  INFO 22372 --- [  restartedMain] o.apache.catalina.core.StandardService
2022-03-14 12:35:21.307  INFO 22372 --- [  restartedMain] org.apache.catalina.core.StandardEngine
2022-03-14 12:35:21.332  INFO 22372 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]
2022-03-14 12:35:21.332  INFO 22372 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext
2022-03-14 12:35:21.417  INFO 22372 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2022-03-14 12:35:21.430  INFO 22372 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2022-03-14 12:35:21.434  INFO 22372 --- [  restartedMain] f.eni.demo.DemoSpringBootWebApplication
2022-03-14 12:35:21.436  INFO 22372 --- [  restartedMain] .ConditionEvaluationDeltaLoggingListener
```

    - Vérifier que l'URL http://localhost:8080/connexion fonctionne