

Labo Spring 1

Première application Web

Etape 1 - Installation d'Eclipse et de Maven

Eclipse

Installez Eclipse ⇒ Eclipse Downloads ⇒ Eclipse IDE for Java **EE** Developers

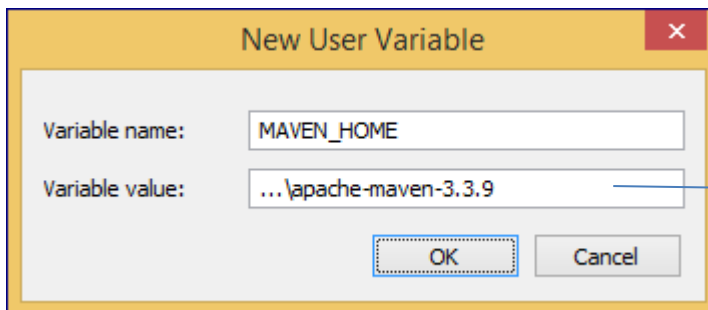
Maven

Installez Maven ⇒ Download Apache Maven 3.3.X (*retenez le chemin d'installation*)

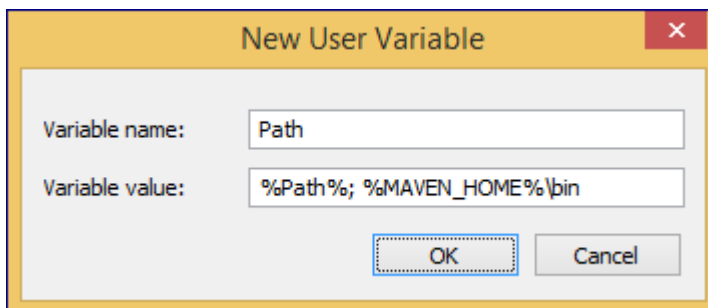
Ajoutez 2 variables d'environnement utilisateur (MAVEN_HOME et Path) :

Control Panel ⇒ All Control Panel Items ⇒ System ⇒ Advanced system settings

⇒ Advanced ⇒ Environment Variables ⇒ User variables for ... ⇒ New



Y copier le chemin
d'installation de Maven



Etape 2 – Création d'un nouveau projet

File ⇒ New ⇒ Other ⇒ Maven ⇒ Maven project

☐ Create a simple project (skip archetype selection) ⇐ *Ne pas cocher*

⇒ Next

Filter : org.apache.maven.archetypes maven-archetype-webapp

⇒ Next

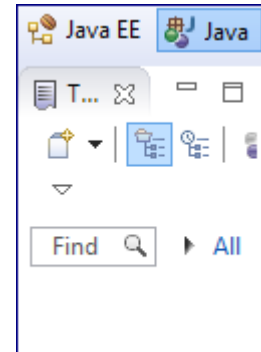
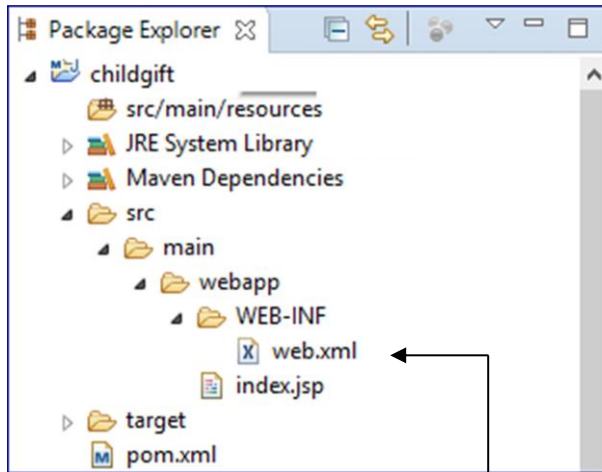
Group Id : *com.spring.henallux*

Artifact Id : *childgift* (Nom du projet)

Etape 3 – Vues du projet

Choisissez la vue Package Explorer (Java) plutôt que Project Explorer (Java EE)

Window ⇒ Show View ⇒ Package Explorer



Etape 4 – Fichier web.xml

Supprimer le fichier web.xml

Etape 5 - Fichier pom.xml

Ajoutez au bon emplacement dans le fichier pom.xml (*inspirez-vous du fichier exemple_pom.xml à votre disposition sur claroline*) :

Parent : groupId : org.springframework.boot

artifactId : spring-boot-starter-parent

Properties : project.build.sourceEncoding : UTF-8

java.version : 1.8

Dependencies :

1. groupId : org.springframework.boot
artifactId : **spring-boot-starter-web**
Pour utiliser Spring Boot
2. groupId : javax.**servlet**
artifactId : **jstl**
Pour utiliser JSTL
3. groupId : org.springframework.boot
artifactId : spring-boot-starter-**tomcat**
Pour serveur Tomcat
4. groupId : org.apache.tomcat.embed
artifactId : **tomcat**-embed-jasper
5. groupId : org.springframework.boot
artifactId : spring-boot-starter-test

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.3.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <groupId>com.spring.henallux</groupId>
  <artifactId>childgift</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>childgift Maven Webapp</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
  </properties>

```

```

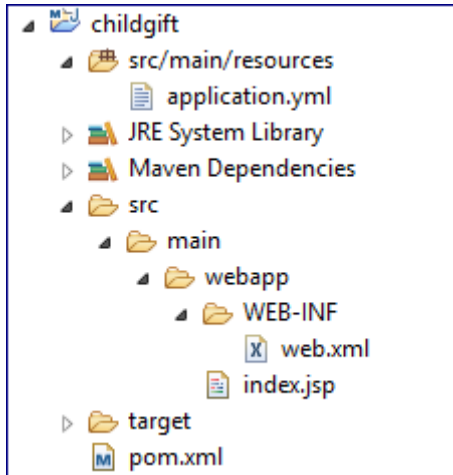
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
  </dependency>
  <!-- Added to allow configuration as a web MVC, built as a WAR file (still
    executable) -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
  </dependency>
</dependencies>

<build>
  <finalName>childgift</finalName>
</build>
</project>

```

Etape 6 – Fichier application.yml

Créez le fichier *application.yml* dans *src/main/resources*.



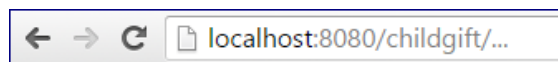
Contenu du fichier (*inspirez-vous du fichier exemple_application.yml disponible sur claroline*) :

ContextPath : path du projet

```
application.yml
1 # Local server
2 server:
3   # port is used by spring-boot-admin
4   port: 8080
5   contextPath: /childgift
```

Attention : Ce type de fichier est très sensible aux indentations et espaces : exactement 2 espaces avant chaque niveau d'indentation et aucun espace à la fin d'une ligne, ni ligne blanche !

Utilisation des informations de ce fichier lors de l'appel des pages dans un navigateur (*quand l'application sera déployée - voir plus loin*) :



Etape 7 – Maven Build

Créez une "launch configuration".

Sélectionnez le projet ⇒ Run ⇒ Run Configurations ⇒ Maven Build ⇒ New

Name : *clean install childgift*

⇒ Main (onglet)

Base directory ⇒ Browse Workspace : *choisir le projet childgift*

Goals : **clean install**

⇒ Apply

⇒ Run

Etape 8 - Répertoire des classes Java

Créez le répertoire `src/main/java` qui contiendra les classes java.

Clic droit sur le projet ⇒ New ⇒ Folder

Folder Name : `src/main/java`

Placez-y le package `com.spring.henallux`

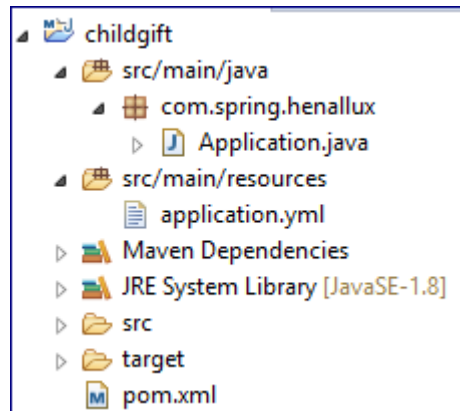
New ⇒ Package

Name : `com.spring.henallux`

Etape 9 – La classe Application

Créez la classe `Application.java` dans le package `com.spring.henallux`.

Attention, cette classe doit impérativement se trouver au même niveau que les futurs packages que vous créerez dans `com.spring.henallux` !



Cette classe contient la méthode `main` qui appelle la méthode `run` de `SpringApplication` :

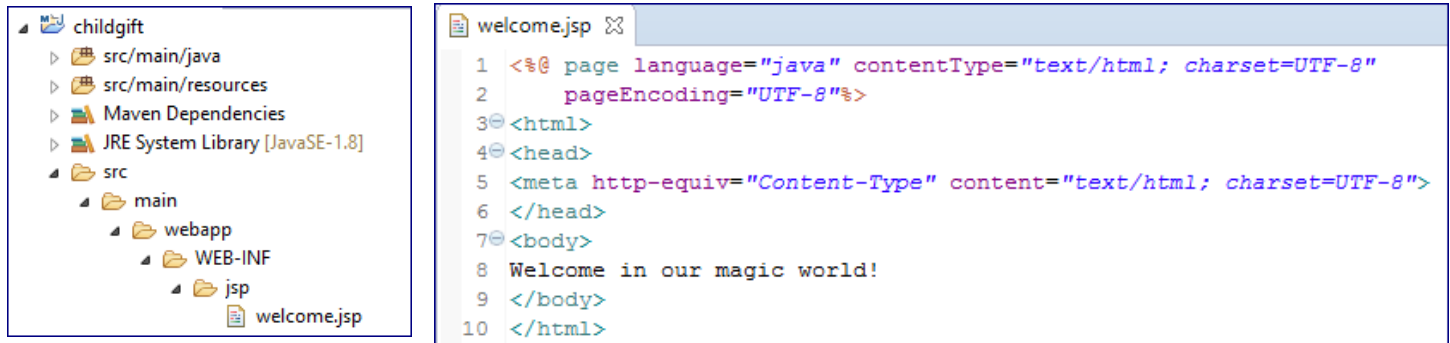
```
Application.java
1 package com.spring.henallux;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
5 import org.springframework.context.annotation.ComponentScan;
6 import org.springframework.context.annotation.Configuration;
7
8 @Configuration
9 @EnableAutoConfiguration
10 @ComponentScan
11 public class Application {
12
13     public static void main(String[] args) {
14         SpringApplication.run(Application.class, args);
15     }
16 }
```

Etape 10 – Page welcome.jsp

Supprimez la page *index.jsp* du répertoire *WEB-INF*.

Créez un répertoire *jsp* dans *WEB-INF*.

Créez la page *welcome.jsp* dans *WEB-INF/jsp*. Cette page doit contenir un message de bienvenue sur le site.



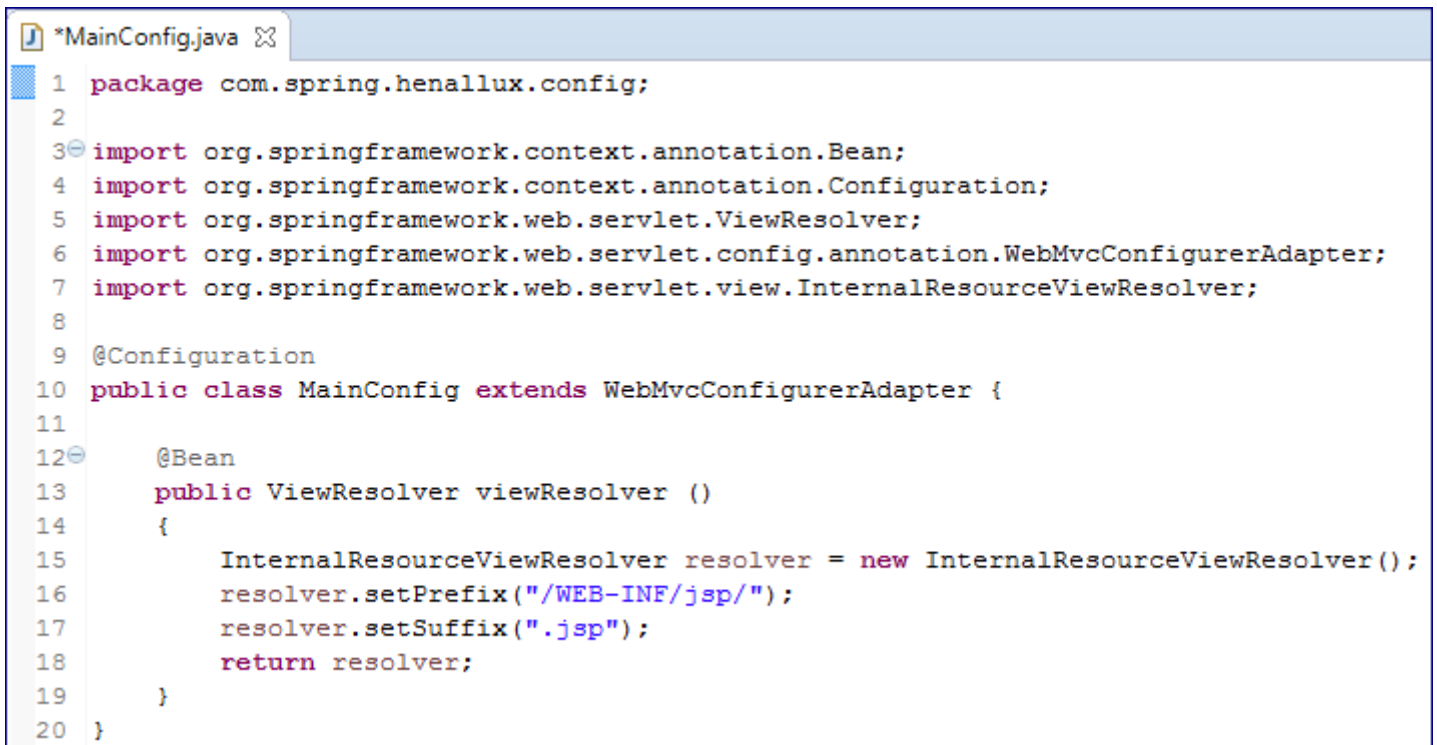
Etape 11 – ViewResolver

Créez le package *config* dans *src/main/java/com.spring.henallux*.

Créez-y la classe *MainConfig* contenant un bean (*ViewResolver*) permettant de localiser les pages jsp à partir du répertoire où elles sont stockées et de leur suffix jsp.

La classe doit être annotée *@Configuration* et être une sous-classe de *WebMvcConfigurerAdapter*.

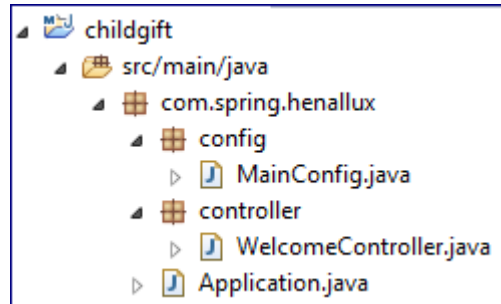
Le *ViewResolver* doit être annoté *@Bean* (injection de dépendance).



Etape 12 – Welcome Controlleur

Créez le package *controller* dans *src/main/java/com.spring.henallux*.

Créez-y la classe *WelcomeController* qui doit être annotée *@Controller*. Ce controller contient une méthode *get* qui retourne la page *welcome.jsp*. Précisez le *path* correspondant à l'appel de ce controller via l'annotation *@RequestMapping*.



```
1 package com.spring.henallux.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RequestMethod;
7
8 @Controller
9 @RequestMapping(value="/welcome")
10 public class WelcomeController {
11
12     @RequestMapping(method=RequestMethod.GET)
13     public String home(Model model) {
14         return "welcome";
15     }
16 }
```

Etape 13 – Déploiement de l'application Web

Build de l'application (si nécessaire)

Sélectionnez le projet ⇒ Run ⇒ Run Configurations ⇒ Maven Build
⇒ choisir *clean install childgift* ⇒ Run

Déploiement de l'application

Clic droit sur la classe Application.java ⇒ Run As ⇒ Java Application

Attention, ne peut se faire que si aucune autre application déployée utilisant la même adresse.

Si erreur du type `java.net.BindException: Address already in use: bind`

⇒ Stoppez l'autre application qui utilise la même adresse :



1. Clic sur remove launch
2. Puis clic sur teminate

Test de l'application web

Dans navigateur

