Mémento

Spring (et Spring Boot)

Version 0.1 (créé le 10/10/2024, modifié le 10/10/2024)

Spring est un Framework conçu pour réaliser des applications web en Java.

Spring Boot est une modalité pour faire du Spring avec une capacité de s'autoconfigurer.

Loric Informatique

Table des matières

1. Prise en main

1.1. Outils nécessaires

- Un logiciel de codage (Visual Studio Code ou Notepad++)
- Java JDK (version 8 ou supérieure)
- Ou un logiciel tout en un (NetBeans (fortement recommandé))
- Java Maven (version 3.9.6 ou supérieure)
- Spring Boot, installée automatiquement (version 2.7.15 ou supérieure)
- Apache Tomcat (serveur d'application Java) (version 9.0.93 ou supérieure)
- Connaissances en HTML et CSS

1.2. Différences entre Spring et Spring Boot

Le Spring est un Framework en Java avec beaucoup de configurations à faire dans les fichiers XML (ou Java). En Spring, le chemin pour accéder à un contrôleur est le suivant :

web.xml -> Dispatcher -> Recherche du contenu par annotation @Controller -> Contrôleur

Le Spring Boot quant à lui est une modalité pour faire du Spring avec une capacité de s'autoconfigurer (il y a beaucoup moins de configurations dans les fichiers XML). Le contrôleur doit être dans le même package que la classe suivie d'un @SpringBootApplication (signifie que les classes et sous-packages doivent être dans le même package pour être visible).

2. Bases du Spring

2.1. Initialiser un projet Spring

Avant de commencer, sur NetBeans, faites clic-droit sur Server (dans Services), puis « Add Server ». Sélectionner Apache Tomcat or TomEE, ensuite choisissez le dossier d'Apache Tomcat téléchargé précédemment, et choisissez un nom d'utilisateur et un mot de passe.

Sur NetBeans, créer un projet Maven et web application, ensuite quand demandé, choisir le serveur « Apache Tomcat or TomEE » et Java EE 7 Web.

Une fois le projet créé, entrer dans les propriétés du projet (clic-droit, puis Properties). Ensuite dans Frameworks, cliquer sur Add..., puis choisir Spring Web MVC (il est également possible de changer le « dispatcher-name » et le « dispatcher-mapping » dans configuration).

Enfin, dans onglet Services, cliquez sur Servers, clic droit sur Apache Tomcat or TomEE, et sur Start pour démarrer le serveur.

Note : Si le projet n'arrive pas à se compiler, remplacer la version de maven-war-plugin par 3.4.0 dans le fichier pom.xml.

2.2. Le fichier applicationContext.xml (dans le dossier WEB-INF)

Bientôt disponible

2.3. Le fichier dispatcher-servlet.xml (dans le dossier WEB-INF)

Ce fichier permet de définir des beans (pour configurer le projet Spring) pour accéder aux contrôleurs.

Il est possible de remplacer le nom dispatcher par un autre nom, mais nécessite de modifier son nom dans le fichier web.xml et nb-configurations.xml

Syntaxe d'un bean de base :

```
<bean id="monId"
      class="monPackage.maClasse"
      parametres />
```

2.3.1. Le bean « viewResolver »

Ce bean permet de spécifier le chemin par défaut des fichiers .jsp.

Syntaxe de base :

```
<bean id="viewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewReso
lver"
      p:prefix="/WEB-INF/jsp/"
      p:suffix=".jsp" />
```

2.3.2. Le bean pour lier à un contrôleur

Ce bean permet d'ajouter une classe en Java précédée d'un @Controller dans le projet Spring.

Syntaxe :

```
<bean name="maClasseController"
      class="monPackage.maClasseController " />
```

2.3.3. La balise pour lier à un package

Cette permet de récupérer directement tous les fichiers Java d'un package dans le projet Spring contenant @Controller, sans avoir à créer un bean pour renseigner chaque classe.

Syntaxe :

```
<context:component-scan base-package="monPackage"/>
```

2.3.4. Autres balises possibles

Balise	Description
`<mvc:annotation-driven/>`	Indiquer l'utilisation du @ comme annotation en Java (facultatif).
`<mvc:default-servlet-handler/>`	Indiquer qu'il s'agit de la servlet par défaut et que tous les contenus seront redirigées vers elle

2.4. Le fichier web.xml (dans le dossier WEB-INF)

Ce fichier permet de définir les dispatchers selon le chemin spécifié dans l'URL.

Une page de bienvenue est paramétrée dans ce fichier, pour la retirer, supprimer les balises `<welcome-file-list>` et son contenu. Le fichier redirect.jsp peut aussi être supprimé.

`<url-pattern>*.htm</url-pattern>` permet de configurer les types de liens qui utilisent tel dispatcher. Pour prendre en compte tous les liens, remplacer `*.htm` par `/`

Syntaxe de base :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param>
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
```

```
    </listener>
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>*.htm</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
        <context-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/applicationContext.xml</param-
value>
        </context-param>
    </session-config>
</web-app>
```

3. Bases du Spring Boot

3.1. Initialiser un projet Spring Boot

Sur le site https://start.spring.io/, choisir un projet Maven, le langage Java et le packaging War (le reste est entièrement personnalisable).

Ensuite, ajouter les dépendances Spring Boot Dev Tools et Spring Web.

Enfin, sélectionner « Générer » pour télécharger le projet initialisé.

Note : Dans le fichier pom.xml, il est possible de personnaliser la version de Spring Boot dans les balises project > parent > version (dernière version utilisée : 2.7.15), et la version de Java dans les balises project > properties > java.version (dernière version utilisée : 1.8 (pour la version 8 du JDK)).

Sur NetBeans, si des problèmes sont rencontrées, assurez-vous d'être allé dans les propriétés du projet (clic-droit, puis « Properties ») et d'avoir résolu les problèmes de dépendances.

3.2. Le fichier SpringBootApplication.java (dans le package par défaut dans le dossier java)

Ce fichier contient une classe qui permet de démarrer l'application web. Il n'y a pas besoin de modifier ce fichier.

3.3. Le fichier ServletInitialiser.java (dans le package par défaut dans le dossier java)

Bientôt disponible

3.4. Le fichier application.properties (dans le dossier resources)

Ce fichier permet la configuration de l'application.

Balise	Description
`spring.mvc.view.prefix: /WEB-INF/jsp/` `spring.mvc.view.suffix: .jsp`	Spécifier le chemin par défaut des fichiers .jsp.
`server.port=8081`	Changer le port de l'application (par défaut 8080)

3.5. Le fichier nbactions.xml (à la racine)

Bientôt disponible

4. Le fichier context.xml (dans le dossier META-INF)

Ce fichier permet de configurer le chemin vers le projet à ajouter après le nom du domaine.

Contenu de base :

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path=""/>
```

5. Le fichier nb-configurations.xml (à la racine)

Bientôt disponible

6. Les contrôleurs

2 contrôleurs existent : @Controller gère que les vues et @RestController est un sur-ensemble de @Controller avec la gestion des fichiers et des vues.

6.1. Les contrôleurs de base (@Controller)

6.1.1. Créer un contrôleur (à mettre dans un package Java)

```
import …;

@Controller
@RequestMapping("chemin")
public class maClasseController {
    …
}
```

`@RequestMapping` n'est pas obligatoire à mettre devant le nom de la classe.

6.1.2. Créer une méthode dans le contrôleur (appelée selon le chemin)

```
@XXXMapping("chemin")
public ModelAndView maMethode(@RequestParam("parametre1") type1
parametre1, @RequestParam("parametre2") type2 parametre2…) {
    return new ModelAndView("maPageJsp");
}
```

Note : `@RequestParam("parametre1") type1 parametre1` permet de passer un paramètre du formulaire (provenant du jsp) directement dans le contrôleur. Si le formulaire n'est pas utilisé, alors les paramètres ne sont pas à spécifier.

6.1.3. Les différents mappings existant

Mapping	Description
`@RequestMapping`	Chemin de base
`@Postmapping`	Chemin après une soumission du formulaire via la méthode POST
`@Getmapping`	Chemin après une soumission du formulaire via la méthode GET
`@PutMapping`	Chemin après une soumission du formulaire via la méthode PUT
`@DeleteMapping`	Chemin après une soumission du formulaire via la méthode DETETE
`@PatchMapping`	Chemin après une soumission du formulaire via la méthode PATCH

6.1.4. Les différentes méthodes du ModelAndView

Instruction	Description
`ModelAndView modelAndView = new ModelAndView("maPageJsp");`	Créer un ModelAndView et indiquer la page jsp à rediriger
`modelAndView.addObject("variable", variable);`	Donner un accès à une variable depuis un fichier jsp

6.2. Les contrôleurs REST (@RestController)

Bientôt disponible

7. Les fichiers .jsp (dans le dossier WEB-INF)

Les fichiers .jsp permettent d'ajouter du code HTML et d'interagir avec le code Spring.

Il est conseillé de mettre les fichiers .jsp dans un dossier appelé « jsp ».

Le mémento HTML est disponible en ligne sur le site : https://loricaudin.github.io/loric-informatique/langages/html/langage_html.html

Le mémento CSS est disponible en ligne sur le site : https://loricaudin.github.io/loric-informatique/langages/css/langage_css.html

7.1. Première page « Hello World »

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Titre du projet</title>
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html>
```

Note : Cette page est souvent générée automatiquement par l'IDE.

7.2. La SpEL

SpEL permet d'accéder directement aux propriétés des beans.

Instruction	Description
`${monBean}`	Accéder à un bean

`${monBean.maPropriete}`
ou
`${monBean ["maPropriete"]}`
	Accéder à une propriété d'un bean

7.3. La JSTL

JSTL est un composant de JEE permettant de remplacer les scriptlets dans les pages JSP.

Toutes les importations peuvent être mises dans un fichier jsp commun à tous les autres jsp, mais dans web.xml, il faut ajouter :

```
<jsp-config>
    <jsp-property-group>
        <url-pattern>*.jsp</url-pattern>
        <include-prelude>/WEB-INF/parametresJstl.jsp</include-
prelude>
    </jsp-property-group>
</jsp-config>
```

7.3.1. La librairie Core

La librairie Core est utile pour les principaux de l'algorithmique (déclaration et gestion de variables, les structures conditionnelles et itératives…).

Importation : `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`

7.3.1.1. Conditions

Une condition renvoie `true` si elle est respectée et `false` sinon.

7.3.1.1.1. Opérateurs de comparaison

Condition	Description de ce que vérifie la condition
`a == b`	*a* égal à *b*
`a < b`	*a* strictement inférieur à *b*
`a > b`	*a* strictement supérieur à *b*
`a <= b`	*a* supérieur ou égal à *b*
`a != b`	*a* n'est pas égal à *b*
`a == null`	Tester si une variable est nulle
`||`	À mettre entre deux conditions, permet d'avoir une des deux conditions qui doit être vraie
`&&`	À mettre entre deux conditions, permet d'avoir deux conditions qui doivent être vraie
`!condition`	Ne doit pas respecter la condition

7.3.1.1.2. Tests de conditions

Instruction	Description

`<c:if test={condition1}>`
`  <balise1>contenu1</balise1>`
`</c:if>`
	Si *condition1* est vraie, alors on affiche *contenu1*

`<c:choose>`
`  <c:when test={condition1}>`
`  <balise1>contenu1</balise1>`
`  </c:when>`
`  <c:when test={condition2}>`
`  <balise2>contenu2</balise2>`
`  </c:when>`
`  <c:otherwise>`
`  <balise3>contenu3</balise3>`
`  </c:otherwise>`
`</c:choose>`
	Si *condition1* est vraie, alors on affiche *contenu1*, sinon, si *condition2* est vraie, on affiche *contenu2*, sinon, on affiche *contenu3*

7.3.1.2. Boucles

Instruction	Description

`<c:forEach var="i" begin="d" end="f" step="p">`
`  <balise1>contenu1</balise1>`
`</c:forEach>`
	On répète $(f-d)/p$ fois la balise pour i allant de *d* compris à *f* compris avec pour pas égal à *p*

`<c:forEach var="elt" items="${liste}">`
`  <balise1>contenu1</balise1>`
`</c:forEach>`
	On parcourt la liste pour *elt* prenant toutes les valeurs de la liste

`<c:forEach var="elt" items="${liste}" varStatus="statut">`
`  <balise1>contenu1</balise1>`
`</c:forEach>`
	On parcourt la liste pour *elt* prenant toutes les valeurs de la liste avec une récupération des informations avec les propriétés de *statut* (`first`, `last`, `count`…)

`<c:forTokens var="sousChaine" items="${chaine}" delims="delimitateur">`
`  <balise1>contenu1</balise1>`
`</c:forEach>`
	On parcourt la chaîne de caractère pour *elt* prenant toutes les valeurs de la chaîne selon le délimitateur (on récupère un caractère à la fois avec le délimitateur vide)

7.3.1.3. Autres balises

Instruction	**Description**
`<c:out value="texte" />`	Affiche un texte à l'écran
`<c:set var="variable" value="valeur"/>`	Déclarer une variable et lui affecter une valeur
`<c:out value="Valeur : ${variable}" />`	Afficher une variable à l'écran