Rapport Projet LO54 :

Gestion de l'offre de formation d'une école privée

Technologie : Spring BOOT

Enseignant :

M. Olivier Richard

- FANTA DILI Nathalie
- LO Aminata

I-CONTEXTE DU PROJET :

Notre travail porte sur la création d'un site web qui permet de gérer l'offre de formation d'une école privée. Pour cela un catalogue est édité en ligne pour lister les formations disponibles et les dates de sessions prévues pour chacune de ces formations. L'application permet à un utilisateur de pouvoir consulter ces sessions de s'y inscrire s'il le souhaite. Il est aussi possible de faire une recherche en fonction de lettres contenu dans le titre de la formation, ou d'une session à une date donnée, ou en fonction du lieu de la session.

II-Présentation de la technologie :

Qu'est-ce que Spring Boot ?

Spring Boot est un Framework ayant pour but de faciliter la configuration (on a des configurations de Spring atténuées) d'un projet Spring et de réduire le temps alloué au démarrage d'un projet. Spring Boot soutient les conteneurs embarqués (embedded containers) et permet à l'application web de s'exécuter indépendamment sans déploiement sur Web Server.

Pour arriver à remplir cet objectif, Spring boot se base sur plusieurs éléments :

- Un site web (https://start.spring.io/) qui permet de générer rapidement la structure de votre projet en y incluant toutes les dépendances Maven nécessaires à une nouvelle application. Cette génération est aussi disponible via le logiciel Eclipse STS.
- L'utilisation de « Starters » pour gérer les dépendances. Spring a regroupé les dépendances Maven de Spring dans des « méga dépendances » afin de faciliter la gestion de celles-ci. Par exemple si vous voulez ajouter toutes les dépendances pour gérer la sécurité il suffit d'ajouter le starter « spring-boot-starter-security ».

Habituellement, le déploiement d'une application Spring nécessite la génération d'un fichier .war qui doit être déployé sur un serveur comme un Apache Tomcat. Spring Boot simplifie ce mécanisme en offrant la possibilité d'intégrer directement un serveur Tomcat dans votre exécutable. Au lancement de celui-ci, un Tomcat embarqué sera démarré afin de faire tourner votre application.

Comment ça marche ?

Spring Boot configure automatiquement les applications en fonction des dépendances que vous avez ajoutées au projet à l'aide de l'annotation @EnableAutoConfiguration. Par exemple, si la base de données MySQL se trouve sur votre chemin d'accès aux classes, mais que vous n'avez configuré aucune connexion de base de données, Spring Boot configure automatiquement une base de données en mémoire.

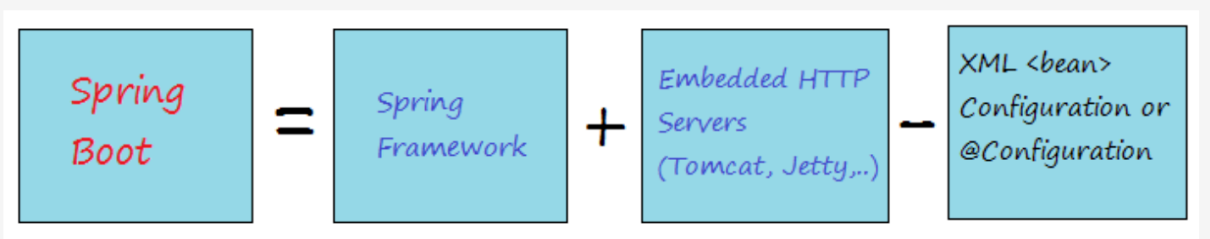
Pourquoi Spring Boot ?

Spring Boot propose les avantages suivants :

- Il fournit un moyen flexible de configurer Java Beans, des configurations XML et des transactions de base de données.
- Il fournit un traitement par lots puissant et gère les points de terminaison REST.
- Dans Spring Boot, tout est configuré automatiquement. Aucune configuration manuelle n'est nécessaire.
- Il offre une application de ressort basée sur des annotations
- Facilite la gestion des dépendances
- Il comprend un conteneur de servlets intégré

Buts

Spring Boot peut s'expliquer simplement par l'illustration ci-dessous :



Pour dire que Spring boot a été conçu avec objectifs suivants :

- Pour éviter une configuration XML complexe pour Spring
- Développer plus facilement des applications Spring prêtes pour la production
- Pour réduire le temps de développement et exécuter l'application indépendamment
- Offrir un moyen plus simple de démarrer avec l'application

III-Tutoriel sur la technologie :

1) Installation

Pour obtenir un projet en Spring boot il faut rajouter le Spring-boot-starter-parent :

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.5.RELEASE</version>
</parent>
```

Tous les projets Spring Boot utilisent le Spring-boot-starter-parent comme parent dans leur pom.xml car cela permet au projet d'hériter des caractéristiques ou éléments propre à Spring-boot c'est aussi le cas pour les projets et modules enfants.
Les éléments pris en compte lors de l'héritage sont :

-Configuration : configuration de la version de java etc…

-Gestion des dépendances - Version des dépendances

-Configuration de plugin par défaut

2)Configuration du projet :

La configuration dans notre projet s'est faite par annotation**.** Pour configurer le projet on a dû rajouter des classes de configurations CoreConfig et ApplicationConfig pour annotation **@Configuration** (qui précise que la classe est un fichier de configuration) et **@ComponentScan (**permet de scanner des packages à la recherche de beans (controller, repository, service) pour une utilisation ultérieure par le conteneur léger). L'assemblage de ces 2 annotations permet d'indiquer à Spring les packages à analyser pour les composants annotés.

```
1  package fr.utbm.gestion.ecole.config;
2
3  import org.springframework.context.annotation.ComponentScan;
5
6  @Configuration
7  @ComponentScan({"fr.utbm.gestion.ecole.service", "fr.utbm.gestion.ecole.service.impl"})
8  @ComponentScan({"fr.utbm.gestion.ecole.service","fr.utbm.gestion.ecoleservice.impl","fr.utbm.gestion.ecole.repository"})
9  public class CoreConfig {
10
11 }
```

```
1  package fr.utbm.gestion.ecole.controller;
2
3  import org.springframework.context.annotation.ComponentScan;
8
9  @Configuration
10 @Import({CoreConfig.class})
11 @ComponentScan({"fr.utbm.gestion.ecole.controller"})
12 public class ApplicationConfig {
13
14 }
15
```

Le point d'entrée de l'application de démarrage est la classe qui contient l'annotation @SpringBootApplication et la méthode principale.

```
1  package fr.utbm.gestion.ecole.controller;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class Application {
8      public static void main(String[] args) {
9          SpringApplication.run(Application.class, args);
0      }
1
```

Toutes les classes qui devront être utilisées dans le contexte de Spring Boot sont annotés avec **@Controller**, **@Service** ou **@Repository**.

Méthode de résolution des vues et des fichiers de ressources

Mise en plus d'un chemin d'accès des vues & d'une résolution des fichiers de ressources avec prise en charge de versionning.

```java
package fr.utbm.gestion.ecole.config;

import org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;


public class MvcConfiguration implements WebMvcConfigurer
{
    public void configureViewResolvers(ViewResolverRegistry registry) {
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/jsp/");
        resolver.setSuffix(".jsp");
        resolver.setViewClass(JstlView.class);
        registry.viewResolver(resolver);
    }

    public void configureDefaultServletHandling(
      DefaultServletHandlerConfigurer configurer) {
        configurer.enable();
    }


}
```

Ensuite dans le fichier application.properties on rajoute les éléments ci-dessous pour indiquer le lieu où on met les fichier JSP pour Spring Boot.

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

Injection de dépendances dans un contrôleur avec @Autowired

```java
package fr.utbm.gestion.ecole.controller;

import java.text.ParseException;

@Controller
public class IndexController {

    @Autowired
    private CourseServiceImpl courseServiceImpl;

    @Autowired
    private LocationServiceImpl locationServiceImpl;

    @Autowired
    private CourseSessionImpl courseSessionImpl;


    @RequestMapping(value = "/", method = RequestMethod.GET)
    public ModelAndView getHome() {

        ModelAndView modelAndView = new ModelAndView("Acceuil");
        List<Course> courses = new ArrayList<>(courseServiceImpl.getAllCourses());
        modelAndView.addObject("courses", courses);
        modelAndView.addObject("locations", locationServiceImpl.getAllLocations());
        return modelAndView;
    }

    @RequestMapping(value = "/search", method = RequestMethod.POST)
    public ModelAndView getSearchTitre(@RequestParam(value = "titre", defaultValue = "") String titre) {
        ModelAndView modelAndView = new ModelAndView("Acceuil");

```

Conclusion :

Au terme de la réalisation de notre projet, on constate qu'on a beaucoup d'avantages à utiliser Spring Boot car il facilite le développement d'application basées sur Spring avec Java(simplicité), il diminue énormément du temps et augmente la productivité. Il est très facile d'intégrer des applications Spring Boot avec ses écosystème de Spring comme Spring JDBC, Spring ORM, Spring Data, Spring Security etc. Il suit l'approche « Configuration par défaut » afin de diminuer le temps et l'effort de développement. Il fournit des serveurs intégrés (Embedded HTTP servers) comme Tomcat, Jetty etc… Afin de développer et de tester des applications web à la manière la plus facile. Il fournit beaucoup de plugins afin de travailler avec des serveurs intégrés et la base de données stockées dans la mémoire Databases) facilement. Utiliser cette technologie a été une vraie découverte et un challenge pour nous et avec tous les avantages qu'elle nous propose cela nous a un peu faciliter la tâche notamment dans le déploiement de l'application sur le serveur, ainsi que la configuration automatique. Il est ainsi possible d'amender n'importe quel composant (@Component) ou configuration (@Configuration) afin de les associer à un ou plusieurs profils et ainsi de les "activer" ou non selon le profil actif.