

## Partie 1 – Structure d'un projet Spring Boot / Écriture de classes du domaine / Premier contrôleur et première vue basiques

**Important !** Nous allons étudier les différentes étapes du développement d'une application web dynamique avec Spring Boot et IntelliJ. Ces étapes étant dépendantes, **il est indispensable que chaque partie soit entièrement réalisée avant de passer à la suivante**. Une partie pourra occuper plusieurs séances de TP. Vous êtes invité à établir un dépôt **git** personnel pour gérer votre projet.

### 1. Création du projet *FriendsOfMine*

1. **Si vous travaillez sur votre machine personnelle** : lancez IntelliJ. Si c'est la première fois que vous utilisez l'IDE, vous devez vous procurer la licence « student pack » (<https://www.jetbrains.com/student/>).  
Créez un nouveau projet de type « Spring Initializr » dans votre workspace. Assurez vous que le SDK du « Projet SDK » référence un JDK 1.8.  
Le nom du projet est « FriendsOfMine ». Complétez les propriétés Maven du projet en spécifiant comme nom d'artifact et nom de group « friendsofmine ».  
Assurez vous ensuite que la version de Spring Boot est bien positionné à 1.5.1.  
Sélectionnez les dépendances suivantes :
  - Core > Validation
  - Web > Web ; Rest Repositories
  - Template Engines > Thymeleaf
  - SQL > JPA ; H2

**Si vous travaillez sur une machine du bâtiment U3** : démarrez sous la partition Windows et lancez IntelliJ 13. Connectez vous ensuite au site <http://start.spring.io/> qui vous assistera dans la génération d'un projet Maven. Sélectionnez des valeurs dans les menus déroulants de telle façon que s'affiche « Generate a Maven Project with Spring Boot 1.5.1 ». Dans la partie « Projet Metadata », spécifiez comme nom d'artifact et nom de group « friendsofmine ». Cliquez sur « Switch to the full version » et sélectionnez les dépendances suivantes :

- Core > Validation
- Web > Web ; Rest Repositories
- Template Engines > Thymeleaf
- SQL > JPA ; H2

Cliquez enfin sur le bouton « Generate Project ». Dézippez ensuite cette archive dans le répertoire sur Z : hébergeant vos projets. Vous devez maintenant importer ce projet dans IntelliJ : choisissez « Import project » puis, après avoir indiqué le répertoire comportant le projet dézippé, choisissez « Import project from external module » avec Maven, cochez « Import Maven projects automatically ». Le téléchargement des dépendances peut prendre un peu de temps.

2. Créez un dépôt git pour héberger votre projet : « VCS > Import into Version Control

- > Create Git Repository ».
- Examinez le fichier « pom.xml » et l'arborescence générés. En tenant compte des numéros de version indiqués dans le fichier « pom.xml » ou figurant dans le nom des librairies référencées dans « External Libraries », trouvez les documentations en ligne de Spring MVC, Spring Boot et Thymeleaf qui vous seront indispensables tout au long des différents TP.
  - Examinez l'arborescence générée. Dans quelle classe se trouve la méthode main du projet ? Quelle annotation est associée à ce fichier ? Consultez la documentation de l'API de Spring Boot pour identifier son rôle vis à vis de l'injection de dépendances.
  - L'IDE a créé de manière automatique la configuration de lancement permettant de lancer l'application FriendsOfMine.  
Observez la configuration créée par l'IDE en ouvrant la fenêtre d'édition de configuration de lancement (Menu « Run » > « Edit Configurations... »).  
Lancez l'application FriendsOfMine en utilisant la configuration créée par l'IDE et tentez d'accéder à l'URL suivante à l'aide de votre navigateur :  
<http://localhost:8080/>  
Cherchez un message « WRN » (warning) dans la console Spring Boot ; ce message devrait vous permettre de comprendre pourquoi aucune page web n'est produite.

## 2. Création d'une vue et d'un contrôleur basique

- On souhaite mettre en place une page d'accueil pour notre application. Créez dans le répertoire « resources/templates » un fichier « index.html » dont le contenu est le suivant :  
<https://gist.github.com/raclet/3e9e8e9a8766deacb24f6470418f5482>  
Remarquez que cette page est de type HTML5 et inclut l'espace de nom XML de Thymeleaf.
- On souhaite mettre en place un contrôleur permettant d'associer la page d'accueil précédente à l'URL racine de votre application. Créez une classe Java « friendsofmine.controllers.IndexController » dans le répertoire « src/main/java » ainsi qu'une classe « friendsofmine.IndexControllerTest » dans le répertoire « test/java ». Modifiez le contenu de la classe de test précédente pour qu'il soit identique à ceci :  
<https://gist.github.com/raclet/09518fc0ae52dddebb17476805acd4b5>  
Modifiez le code de IndexController de telle façon que ce test s'exécute avec un verdict « succès ».  
Remarque : l'instruction « `andDo(print())` » permet d'obtenir dans la console un diagnostic de la requête soumise au contrôleur dans la méthode de test.
- Relancez l'exécution de votre application et constatez que l'URL  
<http://localhost:8080/>  
mène bien maintenant à l'affichage de la page web précédente.

### 3. Création de classes du domaine avec validation

L'objet de notre application est la gestion d'une communauté d'utilisateurs fédérée par des activités. Dans cette partie, nous allons créer les classes du domaine Utilisateur et Activite. La description des relations entre ces entités ne fait pas l'objet de cet exercice, elle sera étudiée dans la 2ème partie.

1. Depuis votre IDE, créez la classe du domaine « friendsofmine.domain.Activite » (il s'agit de créer une classe Java standard dans le répertoire « src/main/java »).
2. Dans le dossier « test/java », ajoutez une classe friendsofmine.ActiviteTest dont le contenu est celui-ci :  
<https://gist.github.com/raclet/1a7b5fed5ba8ae464cc6c194893a338c>
3. Lancez les tests de la classe ActiviteTest depuis IntelliJ. Constatez que les tests échouent.
4. Modifiez la classe Activite de telle sorte que les tests passent. Vous vous aiderez de la documentation en ligne suivante :  
<http://docs.oracle.com/javaee/6/tutorial/doc/gircz.html>
5. De manière similaire, créez la classe du domaine « friendsofmine.domain.Utilisateur » et la classe de test « friendsofmine.UtilisateurTest ». Adaptez le contenu de la classe de test conformément au lien suivant :  
<https://gist.github.com/raclet/ee88bb2f2fc296f061a6e135d640ce66>  
Modifiez la classe Utilisateur pour que les tests précédents produisent tous un verdict « succès ».

### 4. Mise en place d'un jeu de données

Afin de disposer de quelques données au lancement de l'application nous allons mettre en place la création de quelques utilisateur et activités au démarrage de l'application. Comme nous n'avons pas encore configuré de base de données, nous allons conserver les objets Utilisateur et Activite dans des ArrayList distinctes.

1. Créez la classe « service.InitialisationService » qui sera annotée @Service, qui remplit dans une méthode initDonnees() une ArrayList d'Utilisateur et une ArrayList de Activite.
2. Dans le répertoire « src/main/java », écrire une classe friendsofmine.Bootstrap qui est annotée @Component et qui a un attribut de type InitialisationService obtenu par injection de dépendances. La classe Bootstrap définira une méthode init() appelant initDonnees() de InitialisationService. Pour assurer que l'initialisation n'a lieu qu'une seule fois et au bon moment, on utilisera l'annotation @PostConstruct. Pourquoi avoir utilisé @PostConstruct plutôt que d'avoir directement mis un appel à initDonnees dans un constructeur de Bootstrap ?  
D'après la documentation, quelle différence y-a-t'il entre l'annotation @Component et @Service ?
3. Créez un contrôleur « UtilisateurController » comportant une méthode « list » transmettant à une vue Thymeleaf utilisateurs.html la liste des utilisateurs

enregistrés. Le contrôleur obtiendra un Bootstrap par injection de dépendances. La vue affichera un tableau des utilisateurs avec toutes leurs données (nom, prénom, sexe, etc).

4. Réalisez la même modification qu'à la question précédente pour l'autre classe du domaine Activite.

Accédez aux URL suivantes :

<http://localhost:8080/utilisateurs>


<http://localhost:8080/activites>

Remarque : le découplage entre les classes Bootstrap et InitialisationService permet de masquer la façon dont les données sont persistées. Plus tard quand on aura configuré une base de données, on pourra modifier InitialisationService pour interroger la base de données ; la modification sera transparent pour les composants qui n'interagissent qu'avec la classe Bootstrap.

## 5. Tests de contrôleurs

1. On souhaite écrire une classe de test pour le contrôleur UtilisateurController. Créez dans le package de test une classe UtilisateurControllerTest comportant un code similaire à celui de IndexControllerTest que vous adaptez pour tester l'envoi d'une requête GET sur l'URL associée au contrôleur UtilisateurController. Constatez que le test s'exécute en erreur (attention, on parle bien d'erreur et pas de verdict « échec » pour le test). Expliquez (indication : c'est un test unitaire qu'on tente d'exécuter...).
2. Résolvez le problème précédent à l'aide de l'annotation `@SpringBootTest` qui permet de charger complètement le contexte Spring de l'application. Vous testerez que la requête GET sur l'URL associée au contrôleur retourne un status 200, que le type MIME du contenu retourné est bien du `text/html` encodé en UTF8, que la vue retournée d'appelle bien « utilisateurs » et que la chaîne de caractères « `<h2>Liste des Utilisateurs</h2>` » est bien présente dans la réponse.
3. On souhaite réaliser le même type de test pour ActiviteController mais en limitant le test à la couche « web ». Cela requiert cette fois d'utiliser `@WebMvcTest` à la place de `@SpringBootTest` et d'utiliser Mockito pour simuler la dépendance. Ecrire la classe de test ActiviteControllerTest.

## Questions de réflexion (liens cours/TP)

En cours, nous avons vu qu'une application web dynamique est embarquée sur un conteneur web de type tomcat. Où se cache tomcat dans le projet FriendsOfMine ?   
On a aussi vu que le traitement des requêtes se faisait par le biais de servlet. Où se cachent les servlets dans le projet FriendsOfMine ?

## Exercice additionnel (optionnel)

1. Changez le style CSS des templates Thymeleaf de votre application FriendsOfMine en utilisant le w3.css de w3schools.com. Pour cela, vous téléchargerez le fichier



Université  
Paul Sabatier  
TOULOUSE III

M2 Info DL

Développement orienté plateforme backoffice Java EE

suivant :

<https://www.w3schools.com/lib/w3.css>

... et vous le placerez dans le répertoire « resources/static » de votre projet.

Personnalisez le code de vos pages en utilisant la documentation suivante :

<https://www.w3schools.com/w3css/default.asp>

2. Créez un fragment Thymeleaf appelé header.html qui sera affiché dans tous les templates Thymeleaf de votre application et affichant un logo, un lien vers la liste des utilisateurs et un lien vers la liste des activités.



Ce(tte) oeuvre est mise à disposition selon les termes de la [Licence Creative Commons Paternité - Pas d'Utilisation Commerciale 3.0 France](#).