# PRISE EN MAIN DE SPRING BOOT

## PRÉPARER L’ENVIRONNEMENT

Il est temps de préparer le terrain pour accueillir notre projet de spring.

Pour cela, il convient de mettre en place certains outils sur votre machine.

Il vous faut installer dans un premier temps Java, puisque Spring est un framework Java. Nous vous conseillons d’installer la version 11. Elle est ce que nous avons utilisé et ce qui marche pour nous. Vous pouvez le télécharger [ici](https://www.oracle.com/java/technologies/downloads/) .

Ensuite, il vous faut un IDE. Personnellement, nous avons opté pour Intellij que vous pouvez télécharger sur le [site officiel](https://www.jetbrains.com/fr-fr/idea/download/).

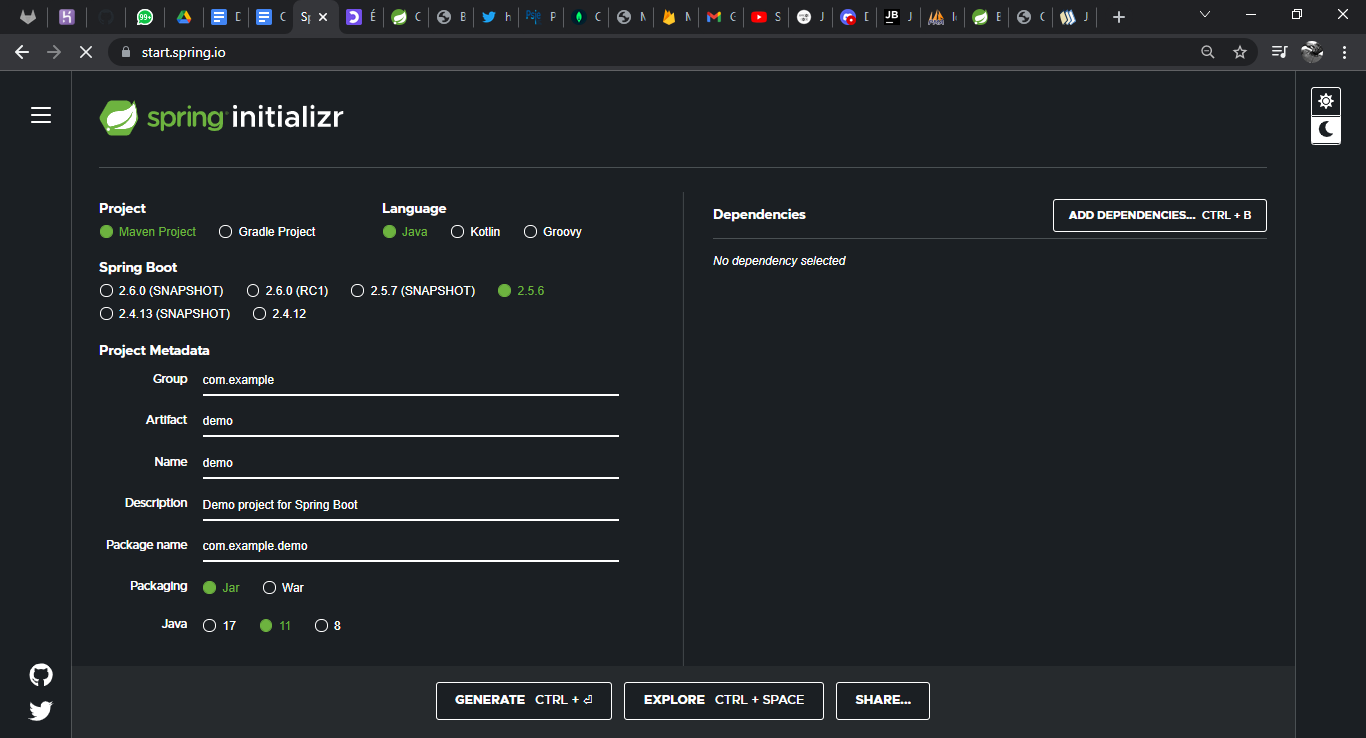
A priori tout est bon pour l’instant, puisque Intellij détectera automatiquement après l’ouverture de votre projet(nous reviendrons sur ça un peu plus tard dans ce document) votre Jdk et le renseignera par défaut. Si tel n’est pas le cas, rendez vous sur ce [site](https://mkyong.com/intellij/how-to-change-the-intellij-idea-jdk-version/).

## CRÉATION D’UN PROJET SPRING

La création d’un projet spring est relativement simple. Il suffit juste de se rendre sur le site [Spring Initialzr](https://start.spring.io/).

Une fois sur le site, renseignez les informations relatives à votre projet comme un grand. Rien de compliqué mon expert ;).

La capture suivante vous montre un aperçu des informations à renseigner.

*fig.1.Spring Initializr*

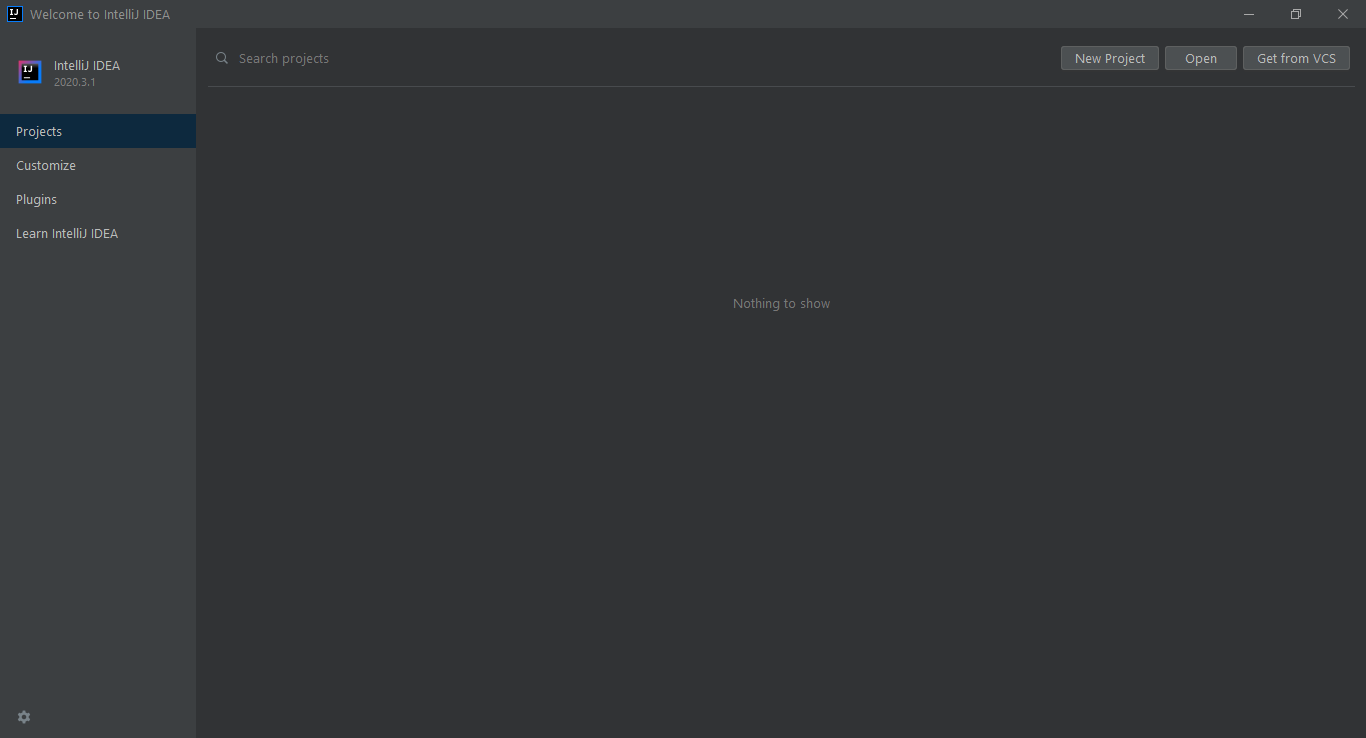
Une fois fait, cliquez sur le bouton [GENERATE] pour télécharger le projet (un fichier .zip).

## OUVERTURE DU PROJET

Une fois le téléchargement fini, dézippez le fichier et ouvrez le avec votre IDE que vous avez téléchargé. Comme vous êtes des experts, je vous laisse le faire. Ce n’est pas du tout compliqué. Mais je vais quand même vous guider.

* Lancez intellij

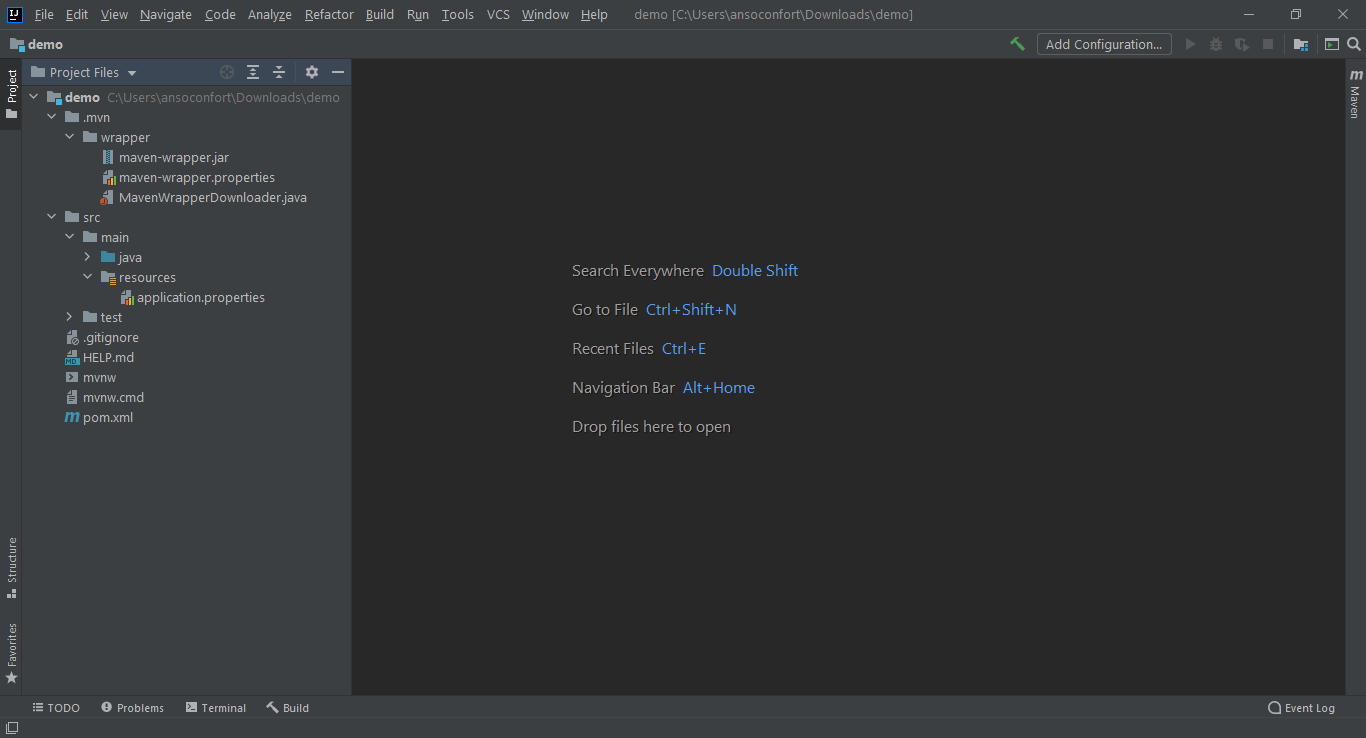
Vous tomberez ici. Vous pouvez ne pas avoir le même aperçu. Ce n’est pas la première fois, nous ouvrons notre intellij, Mais toujours est-il que vous aurez “open” ou “ouvrir” sur l’interface qui sera présentée.



#### fig.2.Accueil de Intellij

* Appuyez sur “open” ou “ouvrir”

Vous serez dirigé vers l’explorateur de fichier intégré de Intellij. Trouvez votre fichier dézippé et sélectionnez ce dernier puis sur “OK”. Et bingo vous avez ouvert votre projet. Vous tomberez donc sur l'interface ci- après.



*fig.3.Vue d’un projet Spring*

## STRUCTURE D’UN PROJET SPRING

Un projet nouvellement comporte une structure bien connue, mais à notre niveau actuel d’expertise, on est amené à plus utiliser les dossiers et fichiers ci-après:

### /src/main/java

C’est à cet endroit qu’ira tout notre code java que nous aurons à écrire.

### /src/main/resources

Il contient le fichier *application.properties.* Ce fichier contient des configurations liées à notre application. Nous parlons là de la connexion à la base de données , le comportement du serveur en cas d’une requête erronée et beaucoup de choses. Pour consulter toutes les propriétés disponibles, visitez vous même ce site. Ah oui , nous n’allons quand même pas toutes les écrire dans notre document. Allez maillez vous.

### /src/test

Dans ce dossier, iront toutes nos classes de test que nous écriront. Vous savez au moins ce qu’on appelle test en développement. Bon c’est pas grave. Vous pouvez en lire davantage [ici](https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-le-test-driven-development/).

### pom.xml

Ce fichier contient des informations telles que la version de java que vous utilisez sur le projet, la version de spring, les dépendances utilisées et bien de choses. Sentez vous libre et visitez ce fichier pour le connaître.

## DESIGN PATTERN

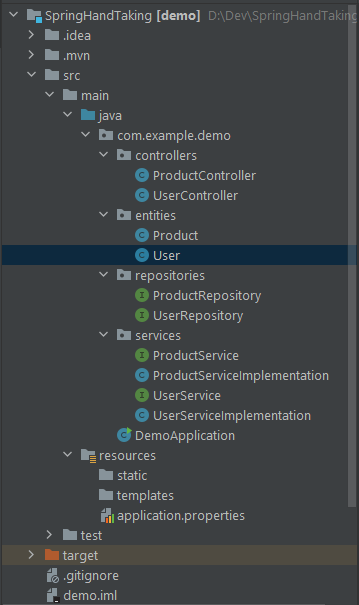
Nous allons adopter un [design pattern](https://fr.wikipedia.org/wiki/Patron_de_conception) bien précis et concis pour bien démarrer. Non seulement ceci donne une scalabilité à notre application , mais facilite le débogage et la détection des erreurs.

Dans un projet , on a souvent besoin de classes. Ces dernières sont la base des applications. Elles ont des actions à effectuer dans le système. Pour cela on sépare chaque partie de notre code, de l’écriture des classes jusqu’aux contrôleurs qui exposent nos routes.

Ainsi, on aura besoin de quatre différents packages:

* **entities** qui contient nos différentes entités bien évidemment
* **repositories** qui contient des interfaces liées à chacune de nos classes et qui connecte notre classe à la base de données via le JpaRepository[].
* **controllers**  qui lui contiendra des classes qui vont exposer nos fonctionnalités.
* **services** qui pour finir contient une pair interface/classe pour chacune de nos entités. L’interface elle définit les méthodes de notre entité tandis que la classe elle implémente ces dernières.

L’image ci contre illustre tout ce qui précède.



*fig.4.Structuration*

*NB:* Notez bien que tous les packages cités ci haut sont bien évidemment créés dans le package principale comme l’indique l’image.

## LES DÉPENDANCES

Les dépendances sont des modules utilisés dans un projet afin d’étendre les fonctionnalités de notre application. Spring sachant bien faire ces choses, fournit plusieurs dépendances qui peuvent être utilisées.

Pour ajouter ces dernières à notre projet, on a deux possibilités:

* Le faire au moment de création du projet.

Il fallait juste cliquer sur le bouton [ADD DEPENDENCIES] pour obtenir les dépendances disponibles puis ajouter avant de cliquer sur le [GENERATE].

* le fichier pom.xml

Pour ajouter une dépendance directement à partir du pom.xml après avoir créer le projet, rendez vous sur le site de [maven](https://mvnrepository.com/).

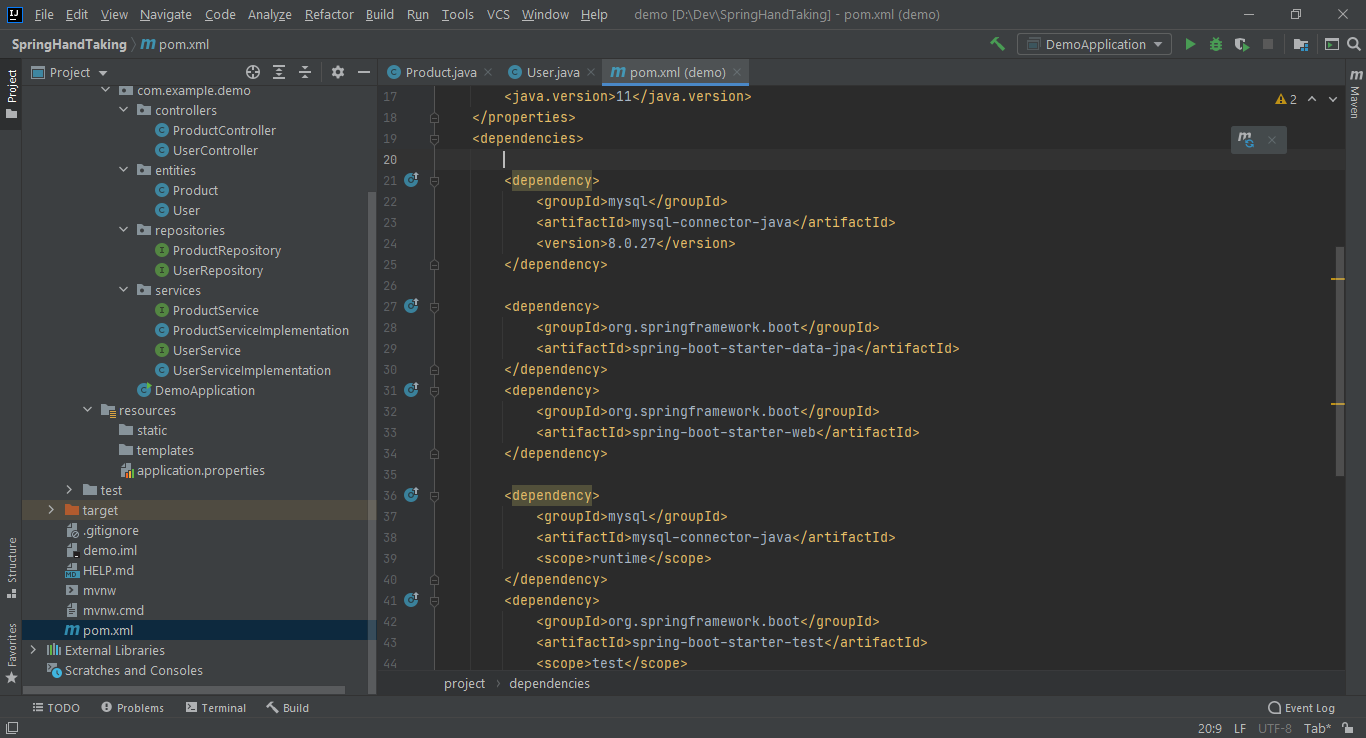
* Cherchez le module que vous désirez
* Sélectionnez le module: ce dernier vous mènera sur une page pour sélectionner la version souhaitée.
* Sélectionnez la version et vous serez redirigé vers une page semblable à celle qui suit.





f*ig.5.Interface de téléchargement de dépendance*

* Cliquez ensuite dans la zone mise en évidence en rouge pour copier le bout de code qui là.
* Ouvrez votre pom.xml et coller le ce dernier dans le <dependencies></dependencies> puis cliquer sur l'icône pointé pour charger la dépendance comme sur l’image qui va suivre.



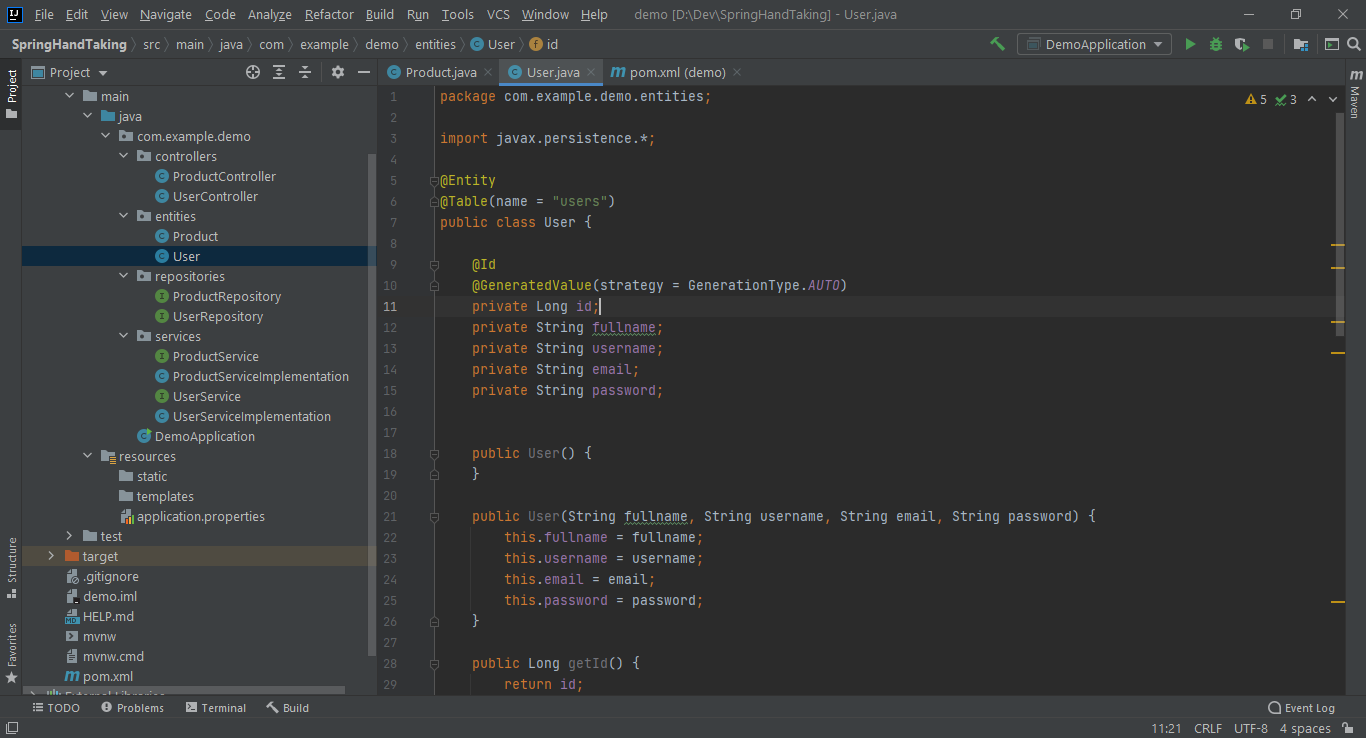
*fig.6.pom.xml*

## LES ANNOTATIONS

Voici quelques annotations nécessaires et souvent utilisées.

* **@Entity**

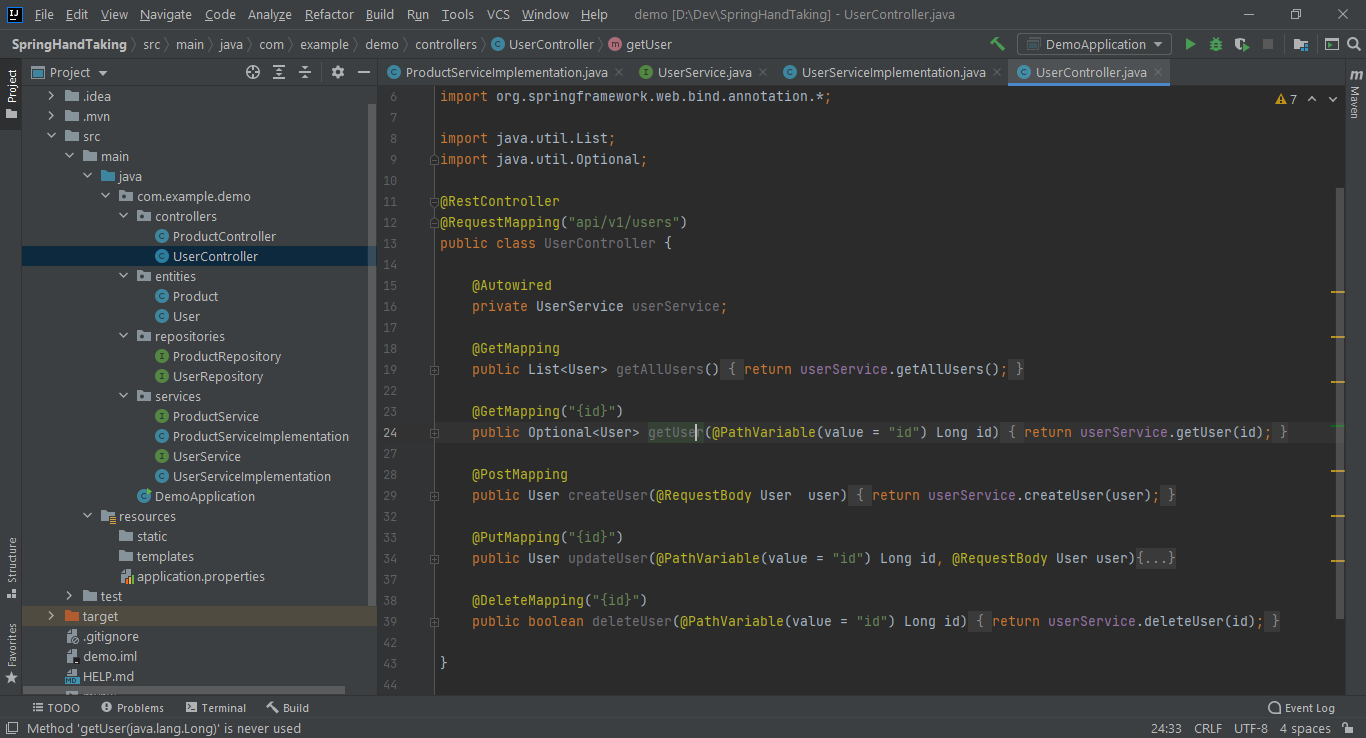
Elle indique à spring que la classe sur laquelle elle a été mise est une entité à insérer dans la base de données.



*fig.7. Une classe*

* **@RestController**

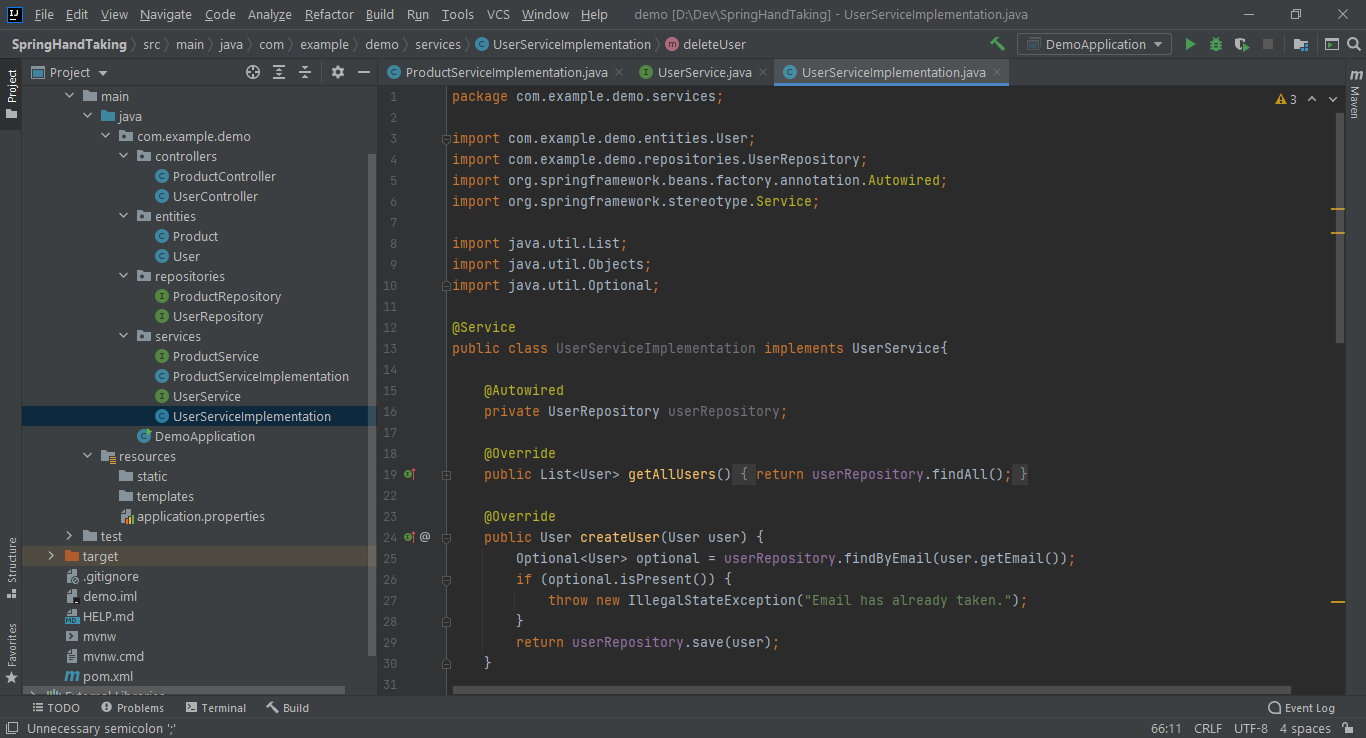
Elle permet de d’indiquer la classe comme un controller



*fig.8. Un controller*

* **@Service**

Elle dit à spring que cette classe est un service



*fig.9. Un service*

Nous utilisons aussi fréquemment pour nos API, des annotations comme:

* @RequestMapping pour indiquer la base URL;
* @PostMapping, @PutMapping, @GetMapping, @DeleteMapping pour indiquer respectivement une requête en post , en put, en get, et en delete.
* @Id pour indiquer que l’attribut est un id
* @GenerateValue pour générer une valeur automatiquement pour l’attribut courant

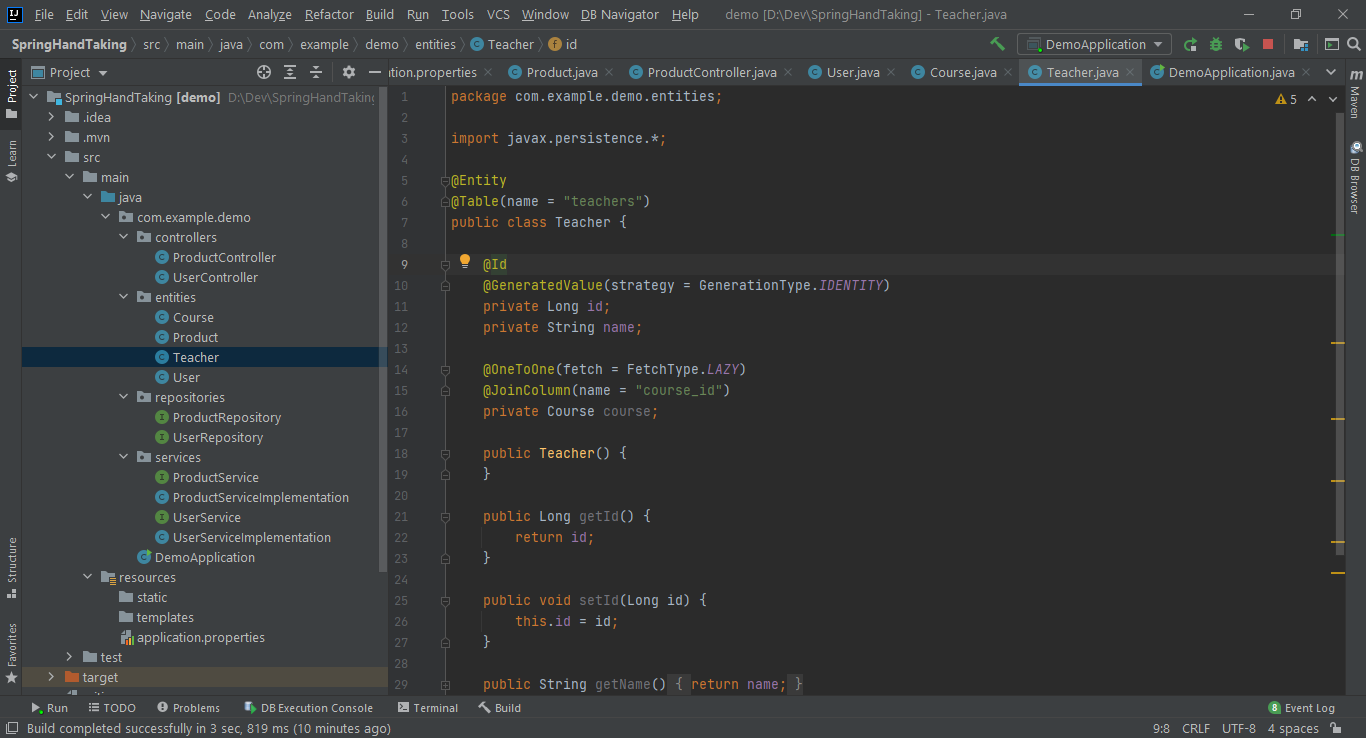
Pour plus d’informations sur les annotations fournies dans spring, référez vous à cette [page](https://www.javatpoint.com/spring-boot-annotations).

## RELATIONS ENTRE LES CLASSES

### ONE-TO-ONE

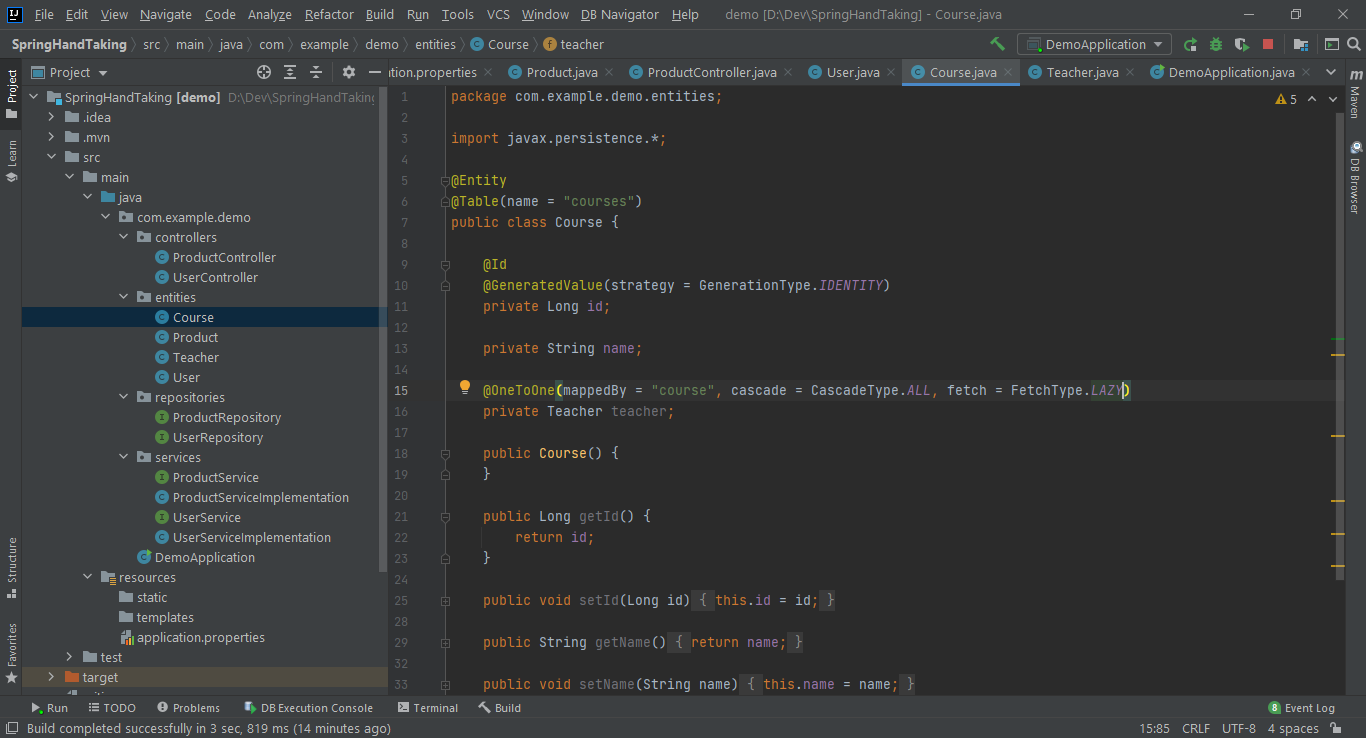
Dans ce cas de figure, une des deux classes contient la clé primaire de l’autre. L’enfant contient la clé primaire de son parent. Dans l’exemple qui va suivre , on suppose qu’un enseignant a un seul cours. Considérons ici le parent comme le cours et l’enseignant comme l’enfant, celui qui contiendra la clé étrangère.

* L’enfant



*fig.10. one-to-one / classe enfant*

-Le parent

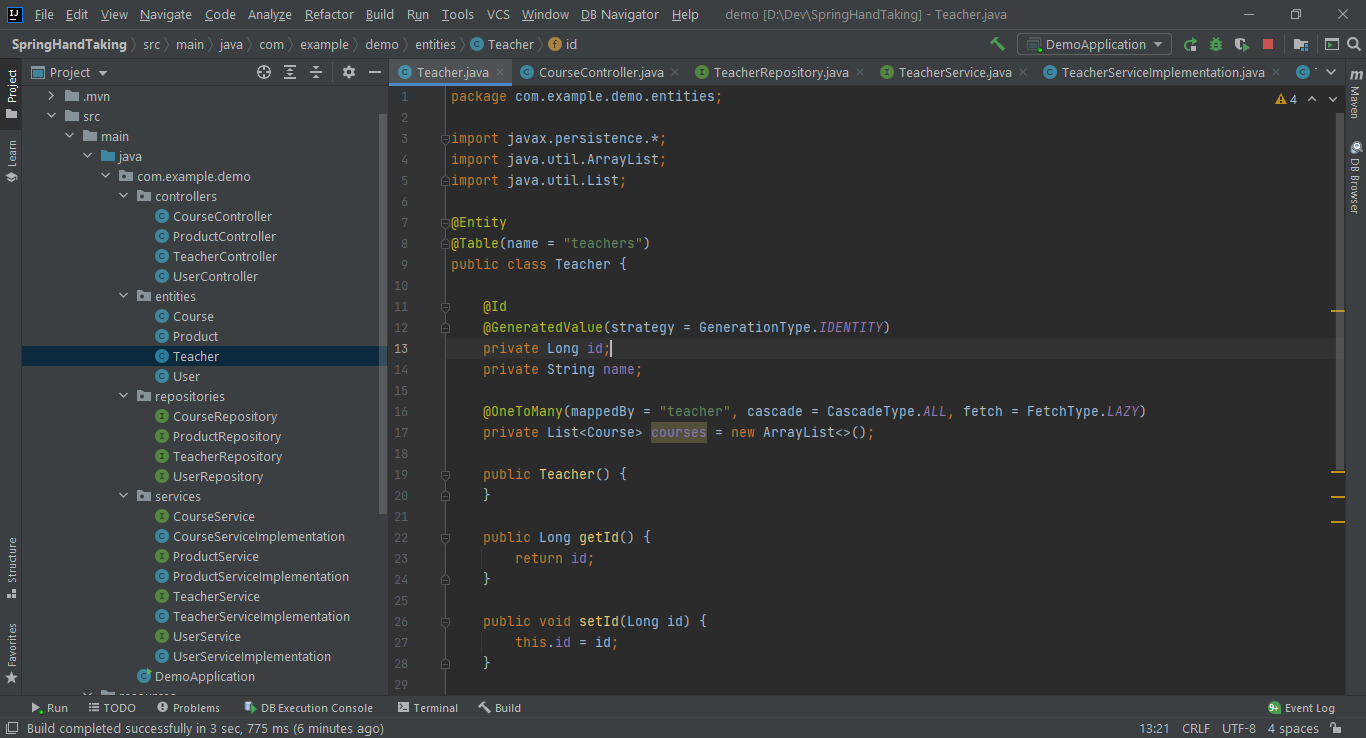


*fig.11. one-to-one / classe parent*

### ONE-TO-MANY, MANY-TO-ONE

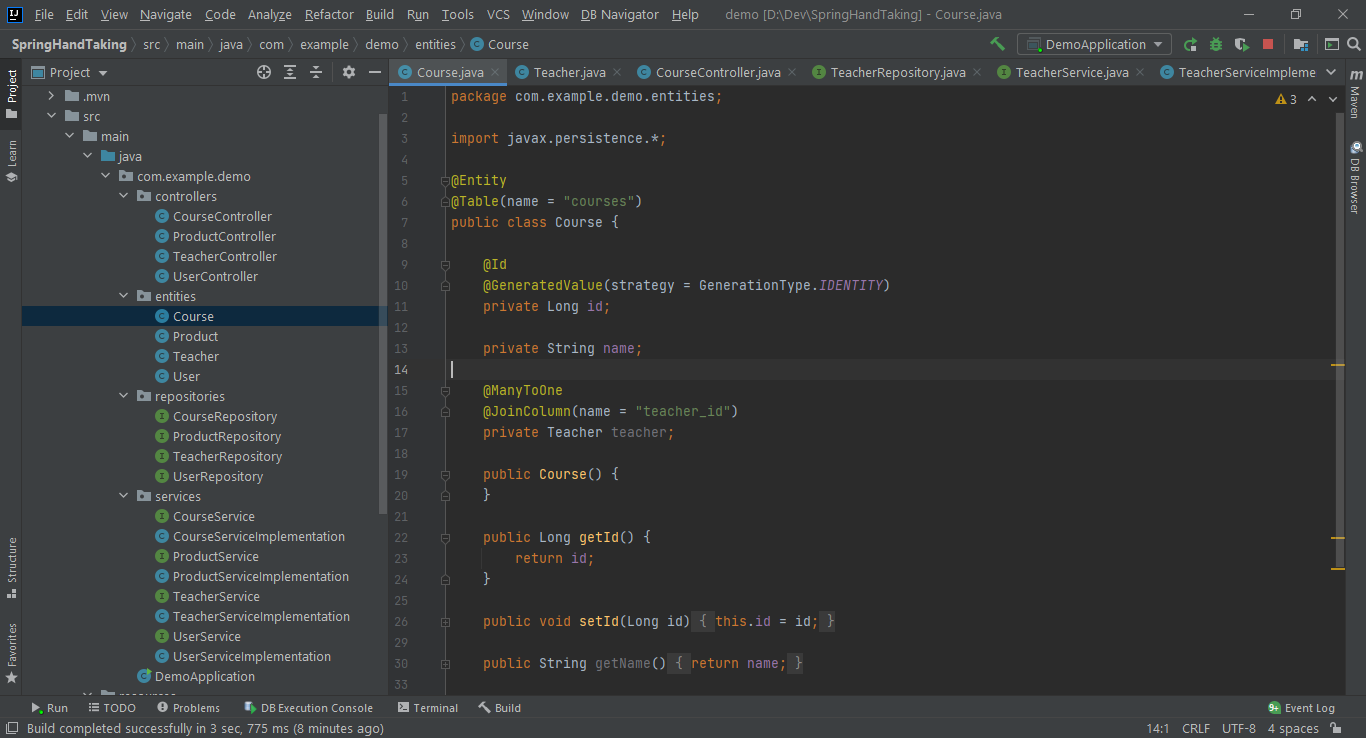
Ici, le scénarion est le même. Prenons le cas de figure où un enseignant doit dispenser plusieurs cours.

* Le parent



*fig.12. one-to-many / classe enfant*

* L’enfant



*fig.13. one-to-many / classe parent*

### MANY-TO-MANY

Dans ce type de relation, le scénario est un peu différent, puisqu’il en résulte une table qui contient les clés primaires des deux tables jointes comme clé étrangère.

Pour commencer, vous devez choisir une classe parent, elle peut être, n’importe laquelle des deux classes.

Une fois ce choix fait, il faudra définir dans la classe parent, le nom de la table nouvellement créée, et les clés étrangères. Ici la clé du parent est définie avec joinColumns = {@JoinColumn(name = “parent\_id”)} tandis que celle de l’enfant est définie avec inverseJoinColumns = { @JoinColumn(name = “child\_id”).

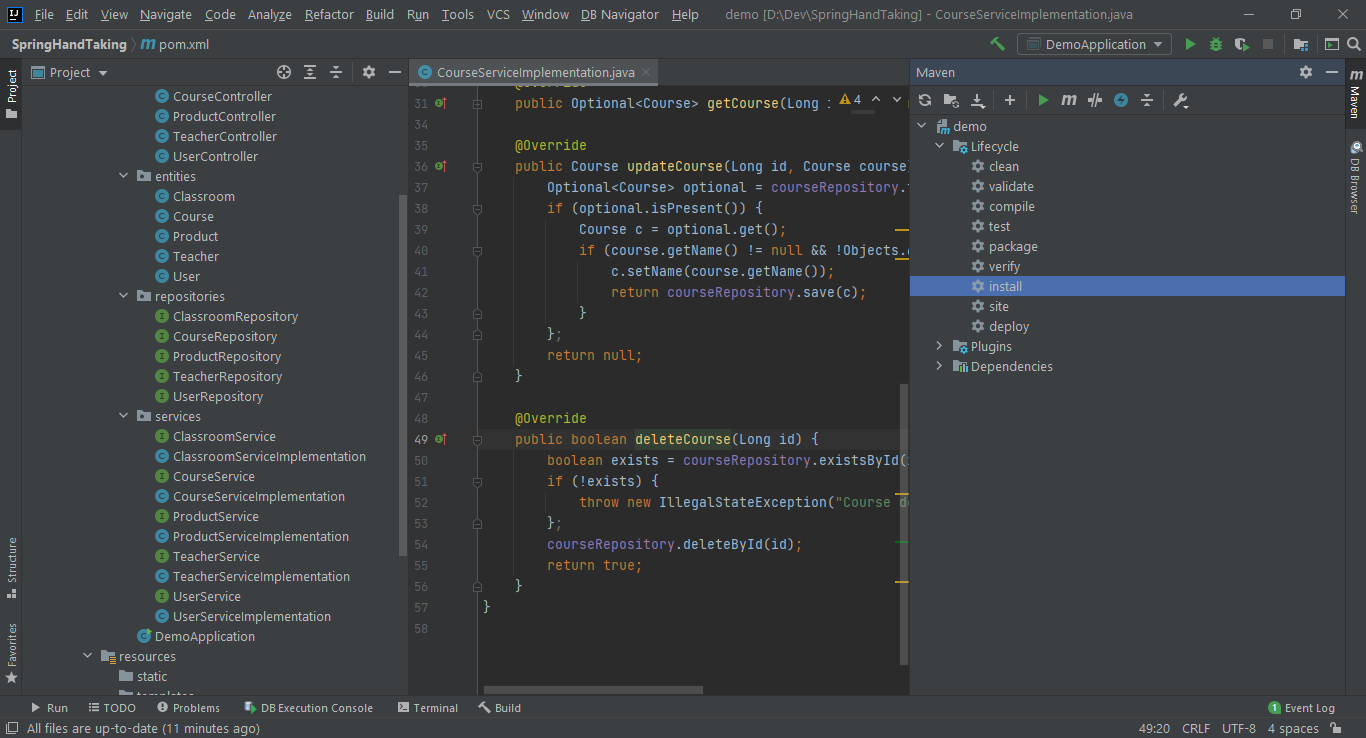
Pour finir, je vous conseille de suivre de façon minutieuse cette [présentation](https://tenmilesquare.com/resources/software-development/spring-boot-jpa-relationship-quick-guide/) sur les relations en spring boot. Une présentation brève et concise mais en anglais, eh oui en anglais cher expert.

## PACKAGING ET RUNNING DE L’APPLICATION

### Packaging

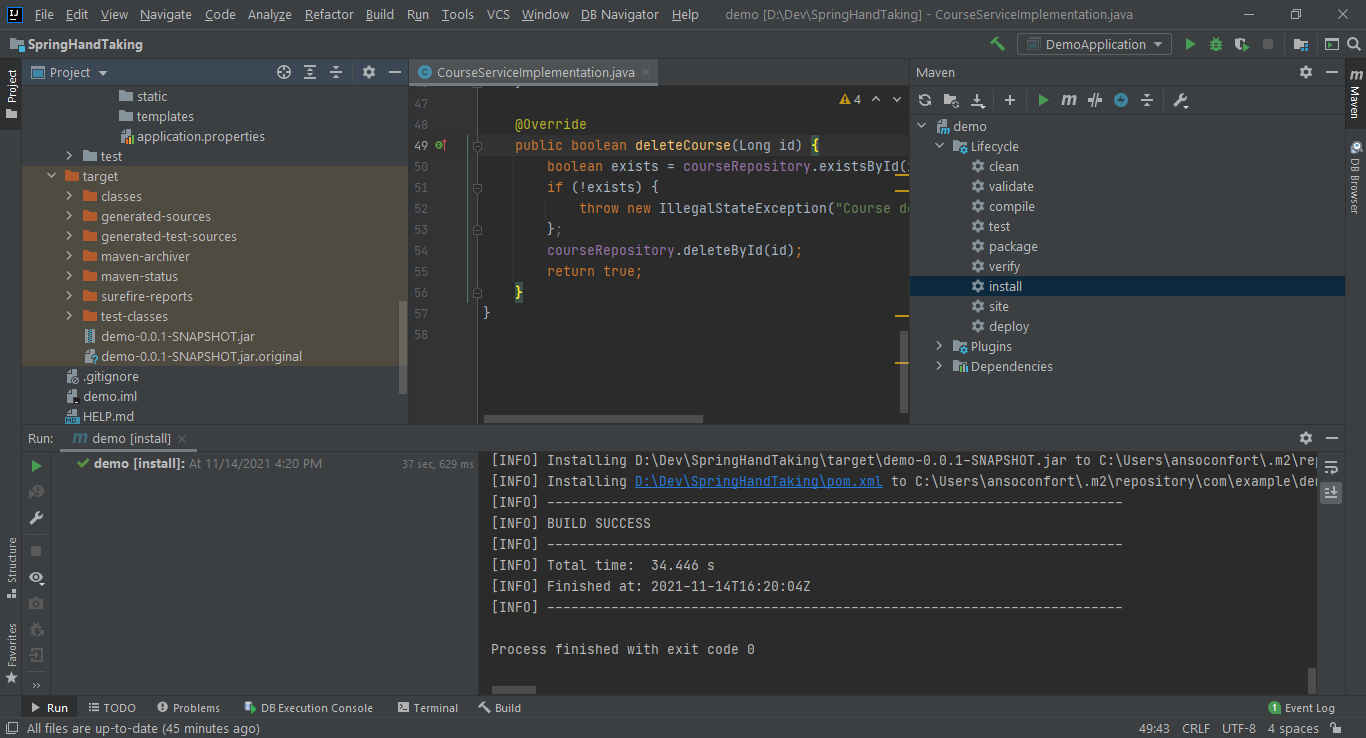
Ce processus consiste à mettre toute notre application dans un fichier .jar situé dans le sous-dossier *target*. Si vous avez déjà lancé votre application une fois, vous l’avez sûrement constaté. Supprimer ce dernier avant de suivre la suite du processus.

Pour packager, il faudra donc suivre l’instruction sur l’image à suivre.



*fig.14.Etape de packaging avec Intellij*

Une fois cette étape bien déroulée, vous verrez que du vert comme sur notre écran ;)



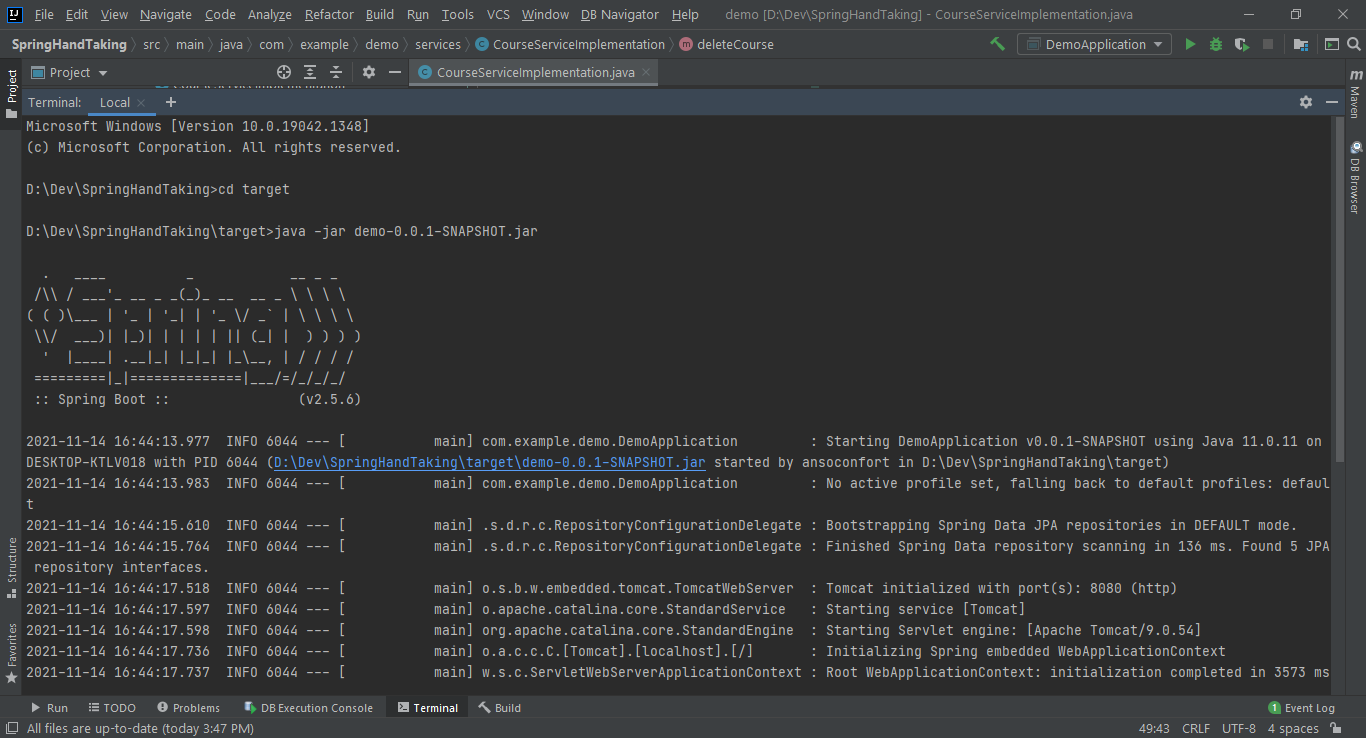
*fig.15.Succès du packaging*

### Running de l’application

Comme vous pouvez le constater également, on a bien notre fichier .jar qui est bien généré dans le dossier *target.*  A présent trève de bavardage. Pour lancer l’application, il vous faut :

* déplacer dans le dossier target
* entrer cette commande : java -jar *nom-du-jar*

Vous pouvez également changer le port 8080 par défaut de l’application en ajoutant -server.port=[votre-port]

**

*fig.16.Lancement de l’application*

Attendez jusqu’à la fin. Une fois l’application lancée, vous pouvez tester ses fonctionnalités. comme vous le faisiez.

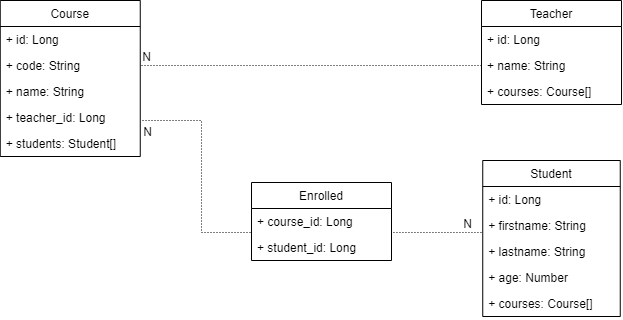
Pour stopper l’exécution, faites juste un ctrl+c dans le terminal.

## PROJET

### Description

Pour la prise en main rapide de spring boot, nous avons réalisé une api de gestion de cours.

### Modélisation



*fig.17.Diagramme de classe*

### Code source

Retrouvez le code source complet de notre projet à l’adresse ci-après <https://gitlab.com/ansoconfort/SpringBootHandTaking>.

## RÉFÉRENCES

[1]<https://docs.google.com/document/d/1uMnfCMG60tI7H1O7KWN3EVznzeUMYAzk/edit#heading=h.35nkun2>

Ceci est un document réalisé sur la prise en main rapide de spring boot

[2]<https://tenmilesquare.com/resources/software-development/spring-boot-jpa-relationship-quick-guide/>

Cette page recense tous les types de relations entre classe en spring boot

[3]<https://www.youtube.com/watch?v=9SGDpanrc8U>

Ce lien mène vers une vidéo qui parle de la sécurité dans tous ces aspects en spring boot.