

MODULE Spring

Exercices / Injection de dépendances

Spring - Exercices

Exercices sur injection de dépendances

- Installation des librairies Spring
- Installation de Spring Tool Suite
- Exercice 01 : instanciación simple d'un bean
- Exercice 02 : invocation de constructeurs avec arguments
- Exercice 03 : instanciación par fabrique
- Exercice 04 : utilisation d'interfaces
- Exercice 05 : définition de bean par annotation
- Exercice 06 : injection de dépendances par annotation

Spring - Exercices

Exercices sur injection de dépendances

- Exercice 07 : injection de collections
- Exercice 08 : injection de valeurs de propriétés
- Exercice 09 : configuration de container basée sur Java

Spring - Exercices

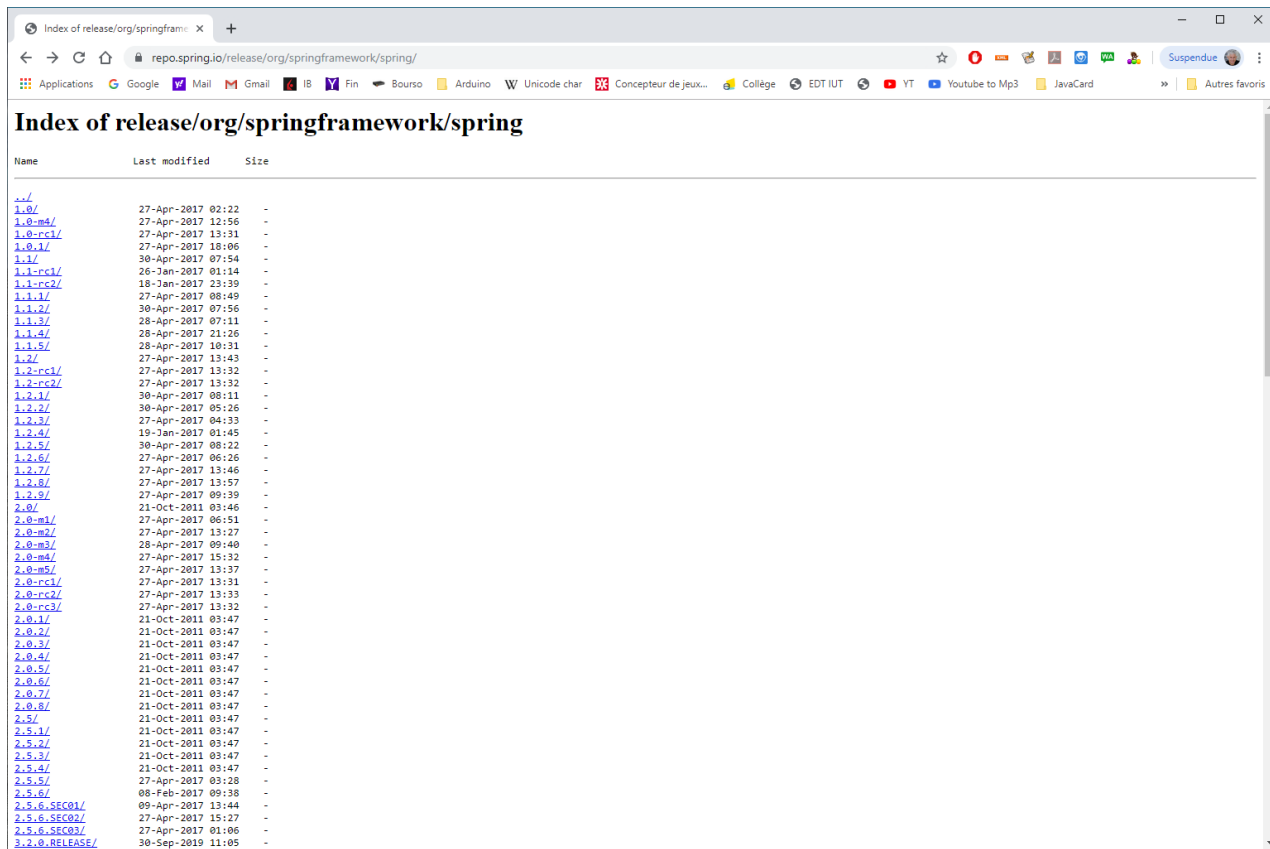
Préliminaire 1 : installation des librairies Spring

Installation des librairies Spring

Récupérer la dernière version RELEASE de l'archive des librairies Spring depuis l'URL :

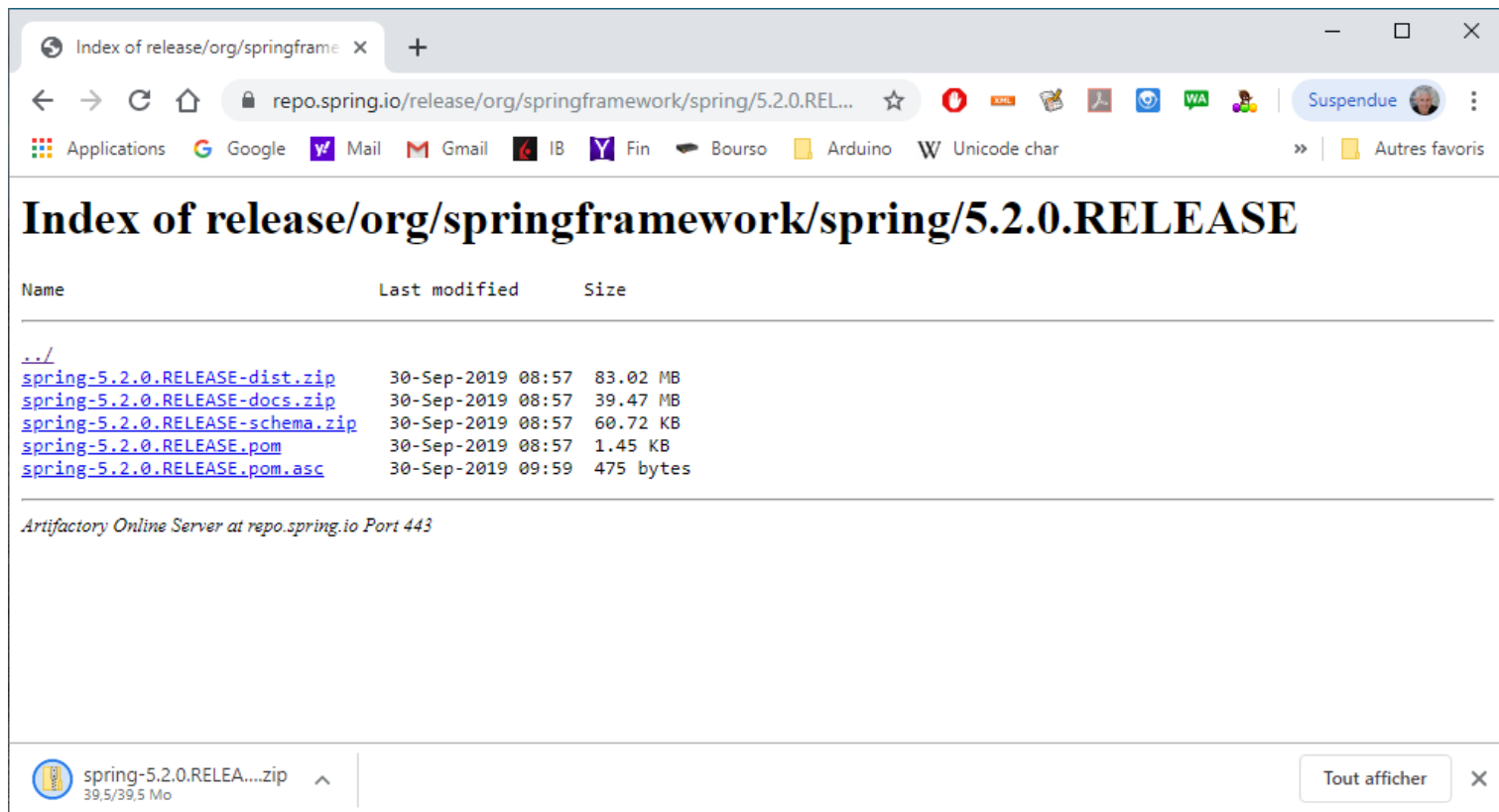
```
https://https://repo.spring.io/release/  
org/springframework/spring/
```

Installation des librairies Spring

A screenshot of a web browser displaying the 'Index of release/org/springframework/spring/' page. The browser's address bar shows the URL 'repo.spring.io/release/org/springframework/spring/'. The page title is 'Index of release/org/springframework/spring'. Below the title is a table with three columns: 'Name', 'Last modified', and 'Size'. The table lists various Spring Framework releases, including versions 1.0, 1.1, 1.2, 2.0, 2.5, and 3.0. Each entry includes a link to the release, the date and time of the last modification, and the size of the release. The releases are listed in descending order of last modified date.

Name	Last modified	Size
./	27-Apr-2017 02:22	-
1.0/	27-Apr-2017 12:56	-
1.0-m4/	27-Apr-2017 13:31	-
1.0-rc1/	27-Apr-2017 18:06	-
1.0.1/	30-Apr-2017 07:54	-
1.1/	26-Jan-2017 01:14	-
1.1-rc1/	18-Jan-2017 23:39	-
1.1-rc2/	27-Apr-2017 08:49	-
1.1.1/	30-Apr-2017 07:56	-
1.1.2/	28-Apr-2017 07:11	-
1.1.4/	28-Apr-2017 21:26	-
1.1.5/	28-Apr-2017 10:31	-
1.2/	27-Apr-2017 13:43	-
1.2-rc1/	27-Apr-2017 13:32	-
1.2-rc2/	27-Apr-2017 13:32	-
1.2.1/	30-Apr-2017 08:11	-
1.2.2/	30-Apr-2017 05:26	-
1.2.3/	27-Apr-2017 04:33	-
1.2.4/	19-Jan-2017 01:45	-
1.2.5/	30-Apr-2017 08:22	-
1.2.6/	27-Apr-2017 06:26	-
1.2.7/	27-Apr-2017 13:46	-
1.2.8/	27-Apr-2017 13:57	-
1.2.9/	27-Apr-2017 09:39	-
2.0/	21-Oct-2011 03:46	-
2.0-m1/	27-Apr-2017 06:51	-
2.0-m2/	27-Apr-2017 13:27	-
2.0-m3/	28-Apr-2017 09:40	-
2.0-m4/	27-Apr-2017 15:32	-
2.0-m5/	27-Apr-2017 13:37	-
2.0-rc1/	27-Apr-2017 13:31	-
2.0-rc2/	27-Apr-2017 13:33	-
2.0-rc3/	27-Apr-2017 13:32	-
2.0.1/	21-Oct-2011 03:47	-
2.0.2/	21-Oct-2011 03:47	-
2.0.3/	21-Oct-2011 03:47	-
2.0.4/	21-Oct-2011 03:47	-
2.0.5/	21-Oct-2011 03:47	-
2.0.6/	21-Oct-2011 03:47	-
2.0.7/	21-Oct-2011 03:47	-
2.0.8/	21-Oct-2011 03:47	-
2.5/	21-Oct-2011 03:47	-
2.5.1/	21-Oct-2011 03:47	-
2.5.2/	21-Oct-2011 03:47	-
2.5.3/	21-Oct-2011 03:47	-
2.5.4/	21-Oct-2011 03:47	-
2.5.5/	27-Apr-2017 03:28	-
2.5.6/	08-Feb-2017 09:38	-
2.5.6.SEC01/	09-Apr-2017 13:44	-
2.5.6.SEC02/	27-Apr-2017 15:27	-
2.5.6.SEC03/	27-Apr-2017 01:06	-
3.0.0.RELEASE/	30-Sep-2019 11:05	-

Installation des librairies Spring



The screenshot shows a web browser window with the address bar displaying `repo.spring.io/release/org/springframework/spring/5.2.0.RELEASE`. The page title is "Index of release/org/springframework/spring/5.2.0.RELEASE". Below the title, there is a table with three columns: "Name", "Last modified", and "Size". The table lists several files for the Spring 5.2.0.RELEASE version, including distribution, documentation, schema, and POM files. At the bottom of the page, there is a footer indicating the server is an "Artifactory Online Server at repo.spring.io Port 443".

Name	Last modified	Size
../		
spring-5.2.0.RELEASE-dist.zip	30-Sep-2019 08:57	83.02 MB
spring-5.2.0.RELEASE-docs.zip	30-Sep-2019 08:57	39.47 MB
spring-5.2.0.RELEASE-schema.zip	30-Sep-2019 08:57	60.72 KB
spring-5.2.0.RELEASE.pom	30-Sep-2019 08:57	1.45 KB
spring-5.2.0.RELEASE.pom.asc	30-Sep-2019 09:59	475 bytes

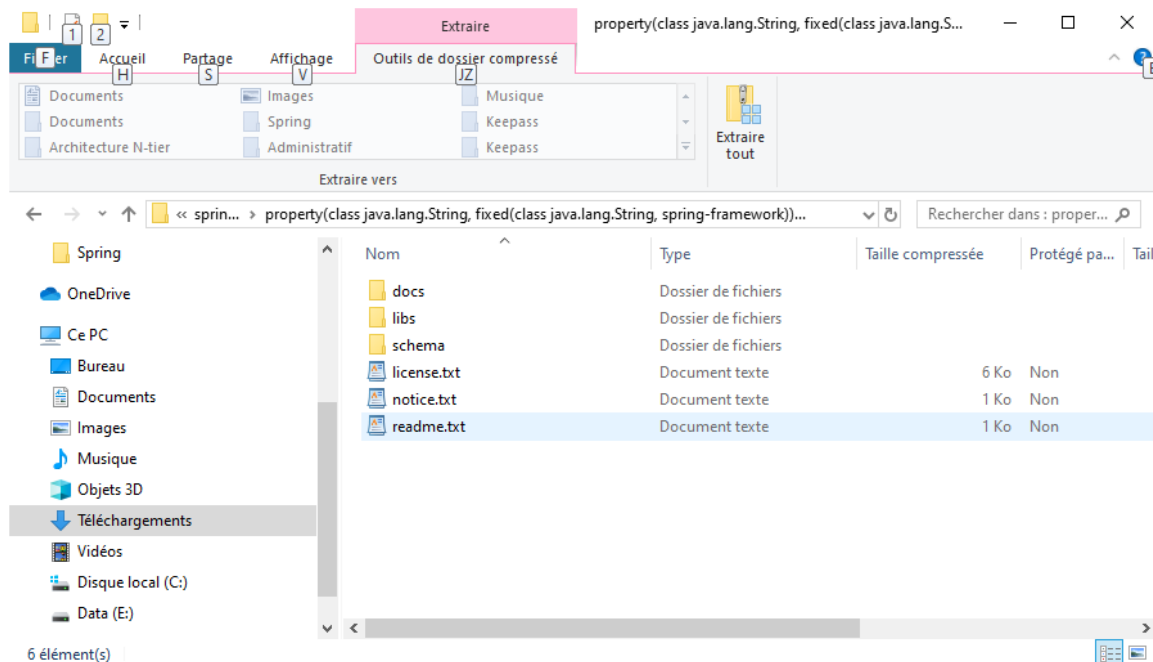
Artifactory Online Server at repo.spring.io Port 443

spring-5.2.0.RELEASE-dist.zip 39,5/39,5 Mo

Tout afficher

Installation des librairies Spring

Extraire tous les fichiers.



Spring - Exercices

Préliminaire 2 : installation de Spring Tool Suite

Installation de Spring Tool Suite

Télécharger la version Windows de Spring Tool Suite 4 :

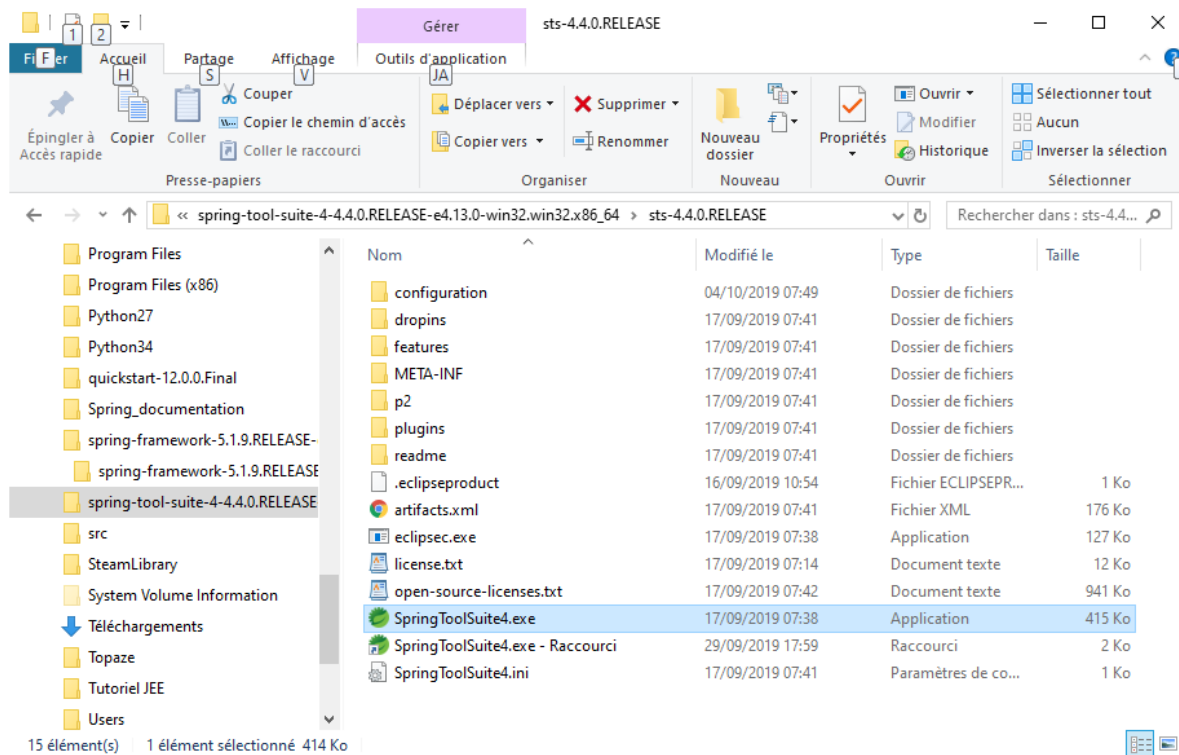
```
https://download.springsource.com/  
release/STS4/4.4.0.RELEASE/dist/e4.13/  
spring-tool-suite-4-4.4.0.RELEASE-  
e4.13.0-win32.win32.x86_64.zip
```

Extraire tous les fichiers (avec 7Zip pour éviter un souci de longueur de nom de fichier).

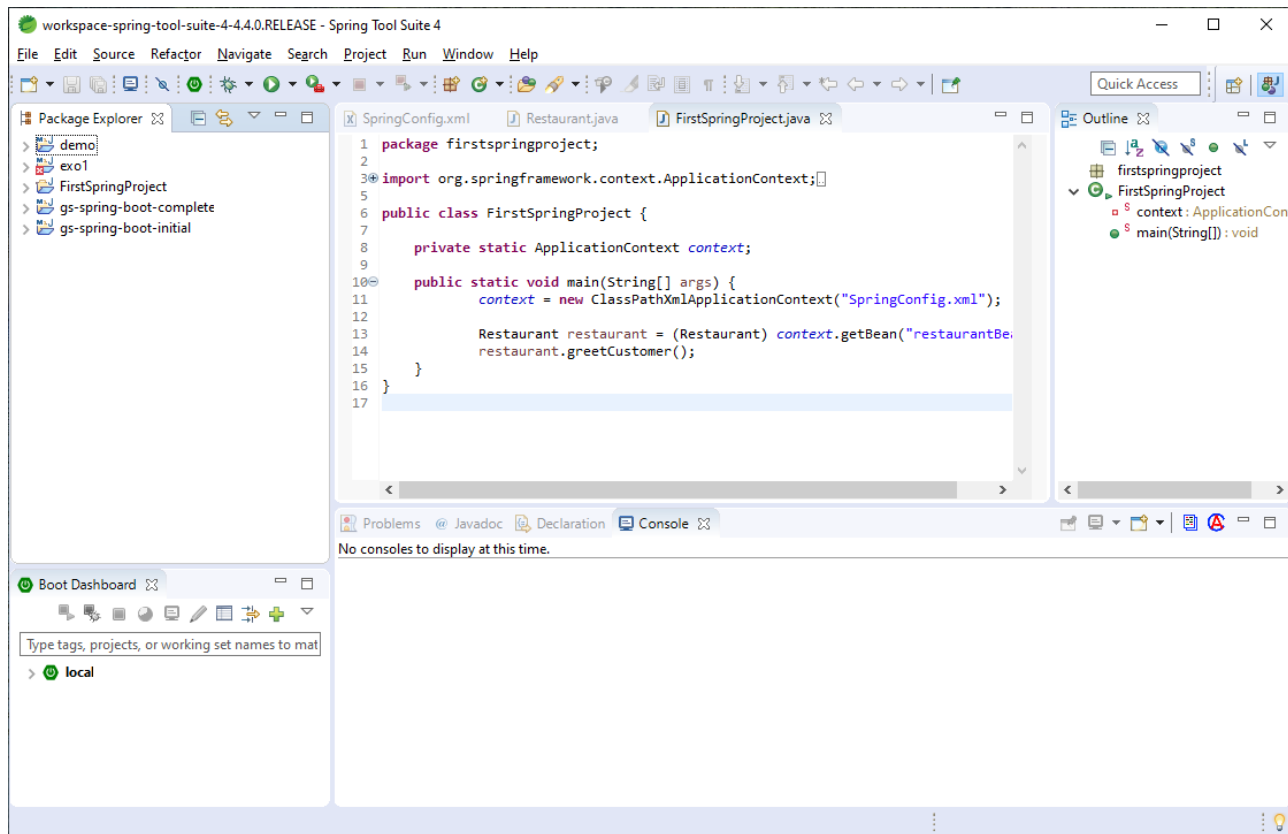
Démarrer l'exécutable Spring Tool Suite 4.

Installation de Spring Tool Suite

Démarrer l'exécutable Spring Tool Suite 4.



Installation de Spring Tool Suite



Spring - Exercices

Exercice 01 : injection de dépendances “classique”

Instanciation simple d'un bean

Le but de l'exercice est de créer une application représentant une propriété immobilière.

Cette propriété comporte un bâtiment et un terrain.

Le bâtiment et le terrain ne portent aucune information mais possèdent une méthode `affiche()` qui affiche quelques informations sur l'élément en question (par exemple, nombre de pièces du bâtiment, superficie du terrain, etc.).

Instanciación simple d'un bean

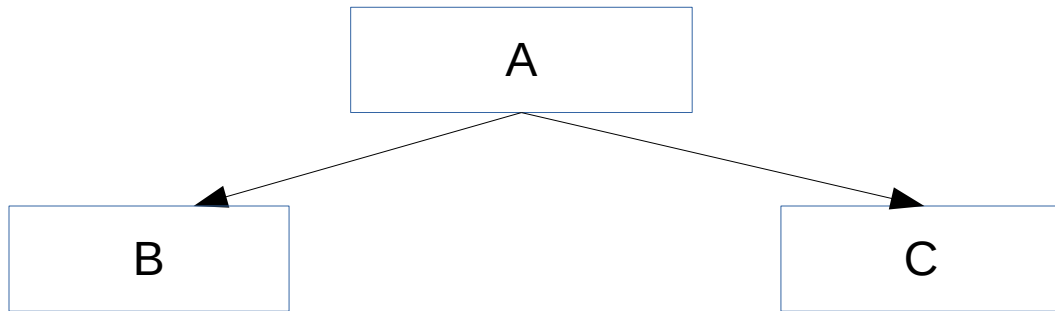
Consignes.

- On veut faire construire la propriété immobilière par l'IOC container, en utilisant son constructeur par défaut
- On veut tester la construction de la propriété immobilière en affichant les informations des éléments qu'elle contient.

Instantiation simple d'un bean

Rappel : exemple de dépendances :

- Un objet A utilise un objet B
- Cet objet A utilise aussi un objet C



Instanciación simple d'un bean

Fichier XML de métadonnées correspondant à cet exemple

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

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="beanB" class="exemples.B"/>

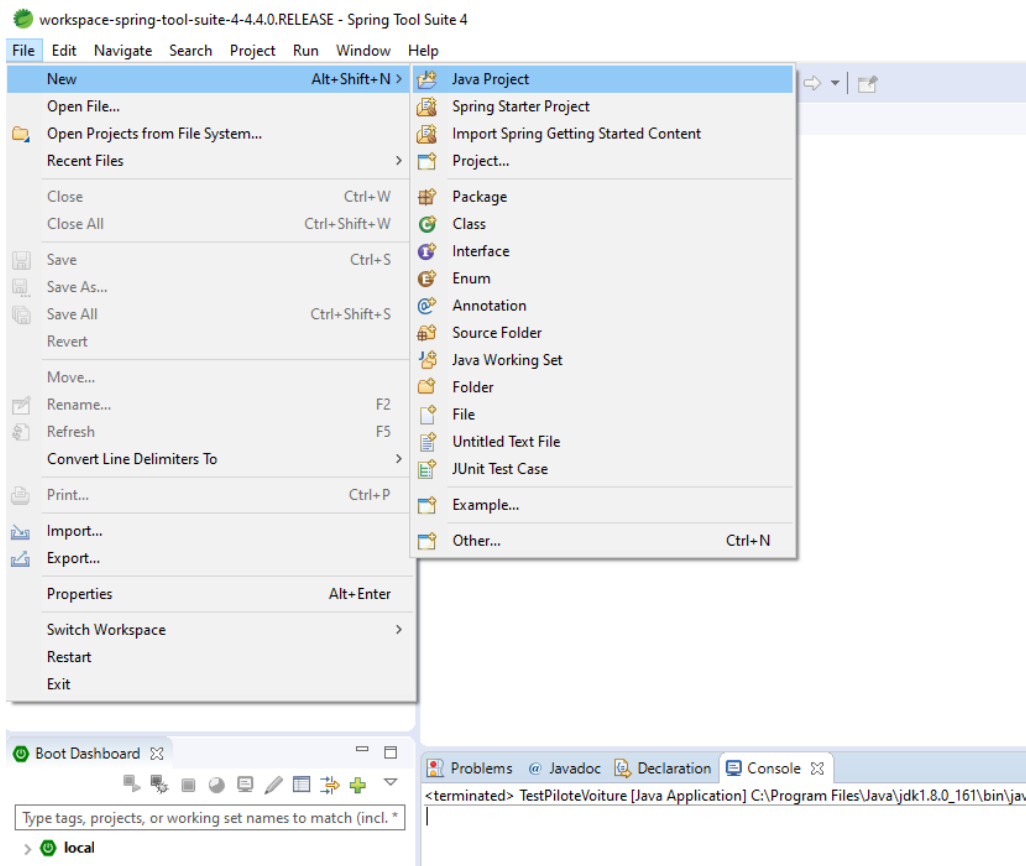
  <bean id="beanC" class="exemples.C"/>

  <bean id="beanA" class="exemples.A">
    <property name="b" ref="beanB"/>
    <property name="c" ref="beanC"/>
  </bean>

</beans>
```

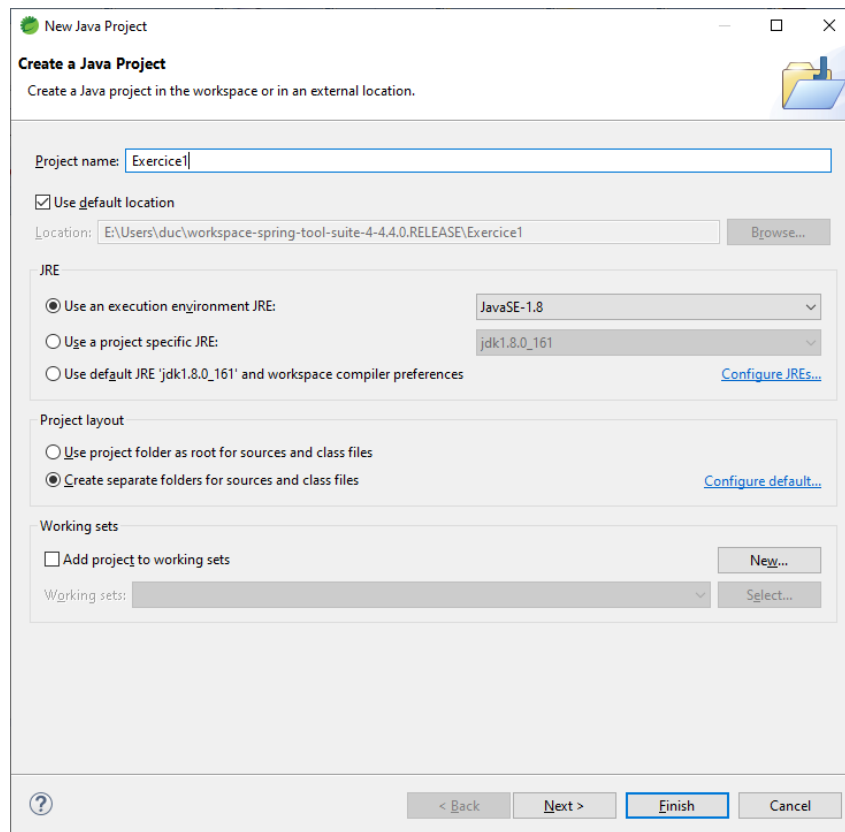
Instanciation simple d'un bean

Créer un nouveau projet Java avec Spring Tool Suite.



Instanciation simple d'un bean

Créer un nouveau projet Java avec Spring Tool Suite.



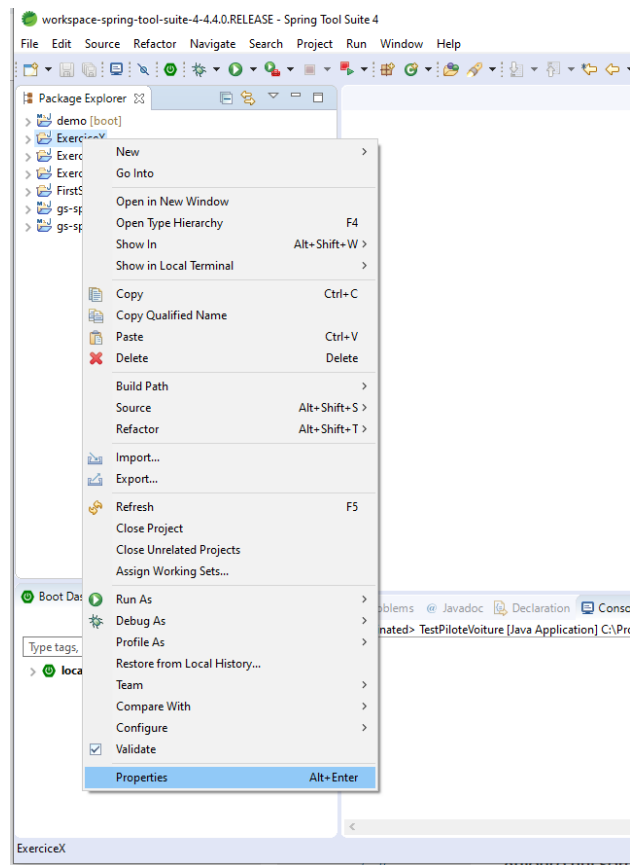
The screenshot shows the 'New Java Project' dialog box in Spring Tool Suite. The dialog is titled 'New Java Project' and has a subtitle 'Create a Java Project'. Below the subtitle, it says 'Create a Java project in the workspace or in an external location.' The dialog is divided into several sections:

- Project name:** A text field containing 'Exercice1'.
- Use default location:** A checked checkbox. Below it, a text field shows the location 'E:\Users\duc\workspace-spring-tool-suite-4-4.0.RELEASE\Exercice1' and a 'Browse...' button.
- JRE:** A section with three radio buttons:
 - ☒ Use an execution environment JRE: A dropdown menu showing 'JavaSE-1.8'.
 - ☐ Use a project specific JRE: A dropdown menu showing 'jdk1.8.0_161'.
 - ☐ Use default JRE 'jdk1.8.0_161' and workspace compiler preferences. A link 'Configure JREs...' is next to it.
- Project layout:** A section with two radio buttons:
 - ☐ Use project folder as root for sources and class files.
 - ☒ Create separate folders for sources and class files. A link 'Configure default...' is next to it.
- Working sets:** A section with a checkbox 'Add project to working sets' and a 'New...' button. Below it, a 'Working sets:' dropdown menu and a 'Select...' button.

At the bottom of the dialog, there are four buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

Instanciation simple d'un bean

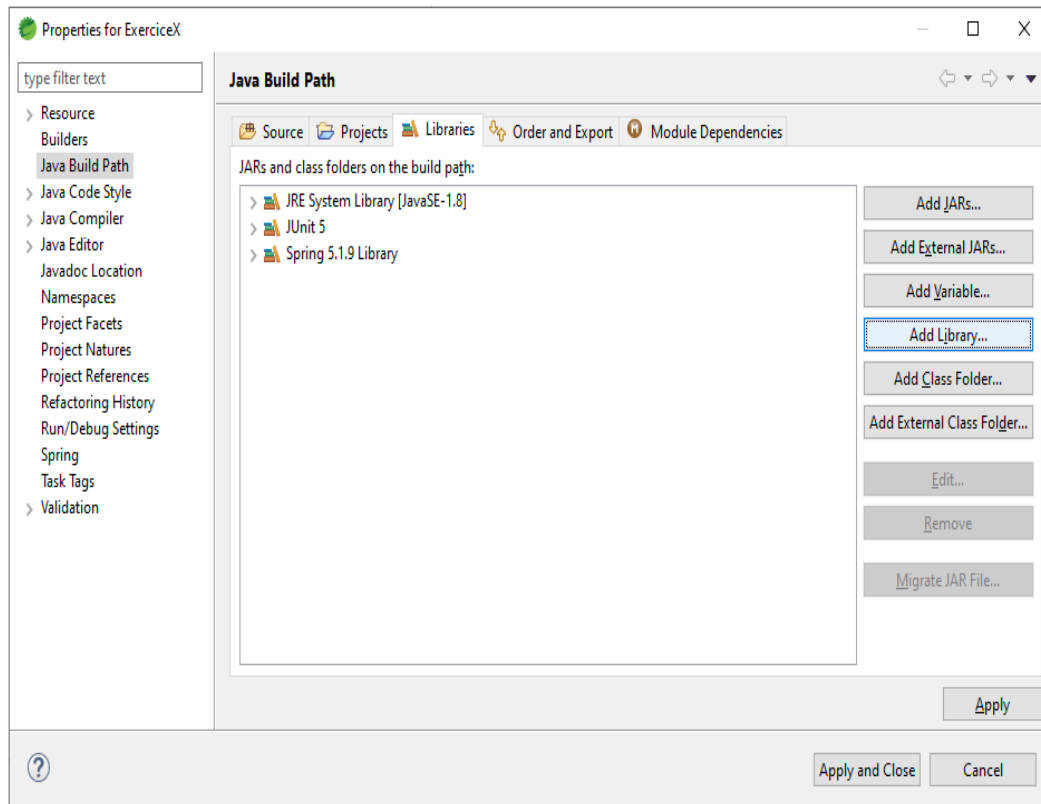
Ajouter les librairies
Spring dans le « Java
build path ».



Instanciation simple d'un bean

Ajouter les librairies Spring dans le « Java build path » :

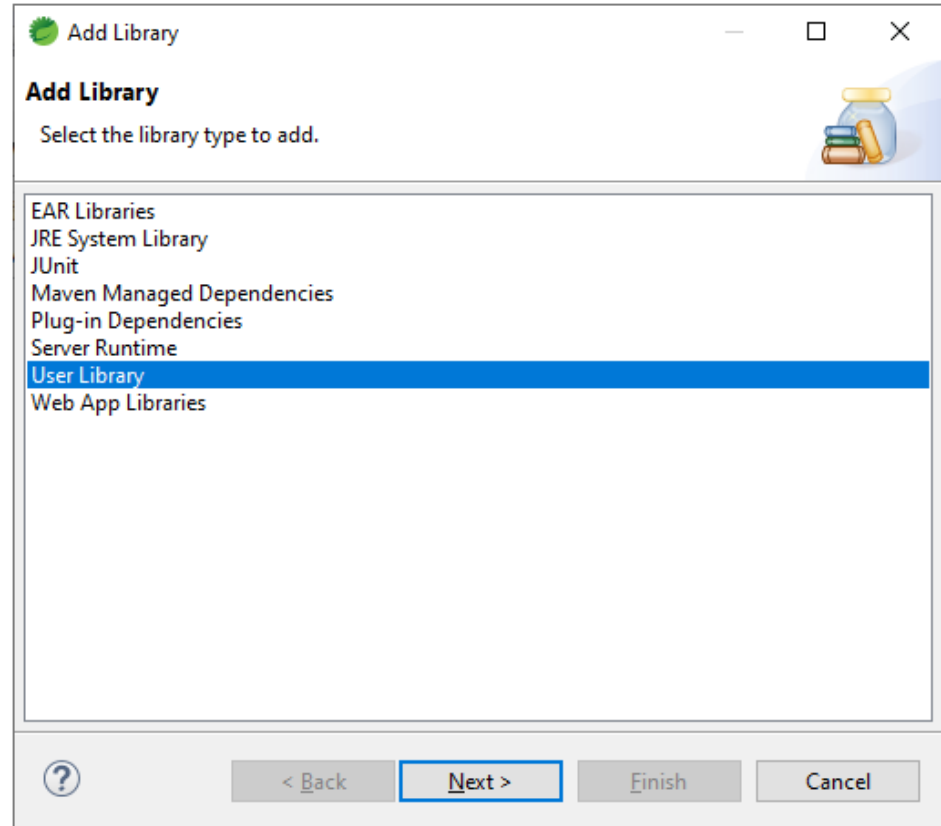
1. Créer une librairie utilisateur...



Instanciation simple d'un bean

Ajouter les librairies
Spring dans le « Java
build path » :

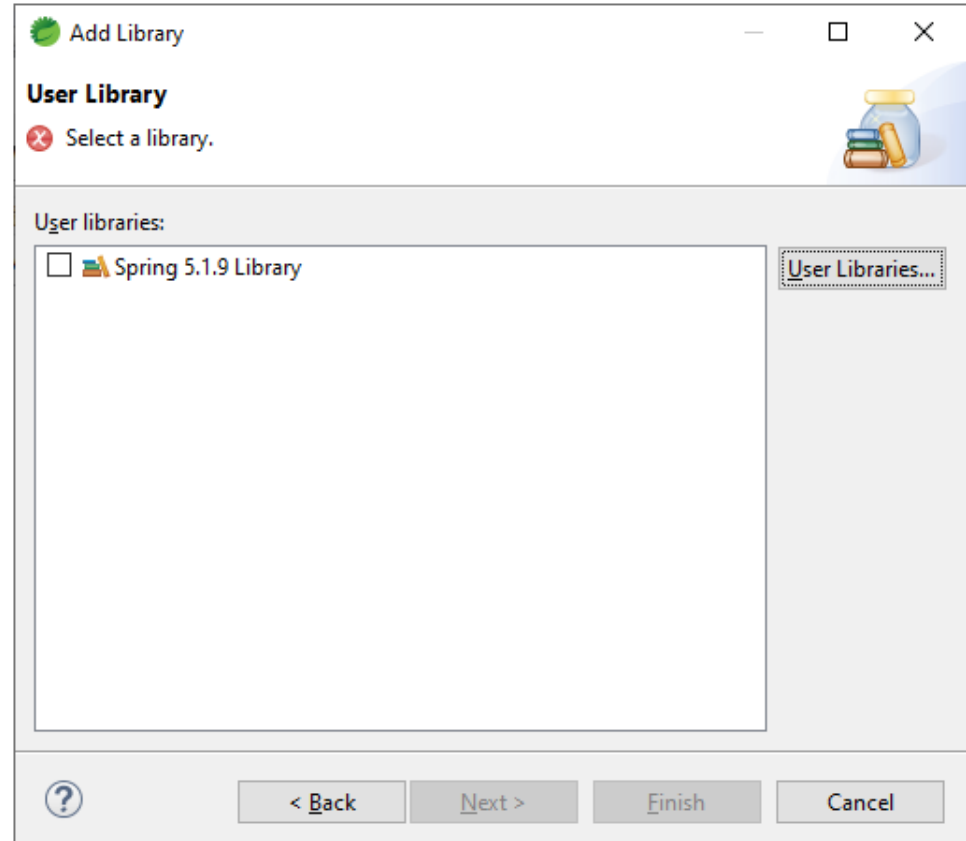
1. Créer une librairie
utilisateur...



Instanciation simple d'un bean

Ajouter les librairies
Spring dans le « Java
build path » :

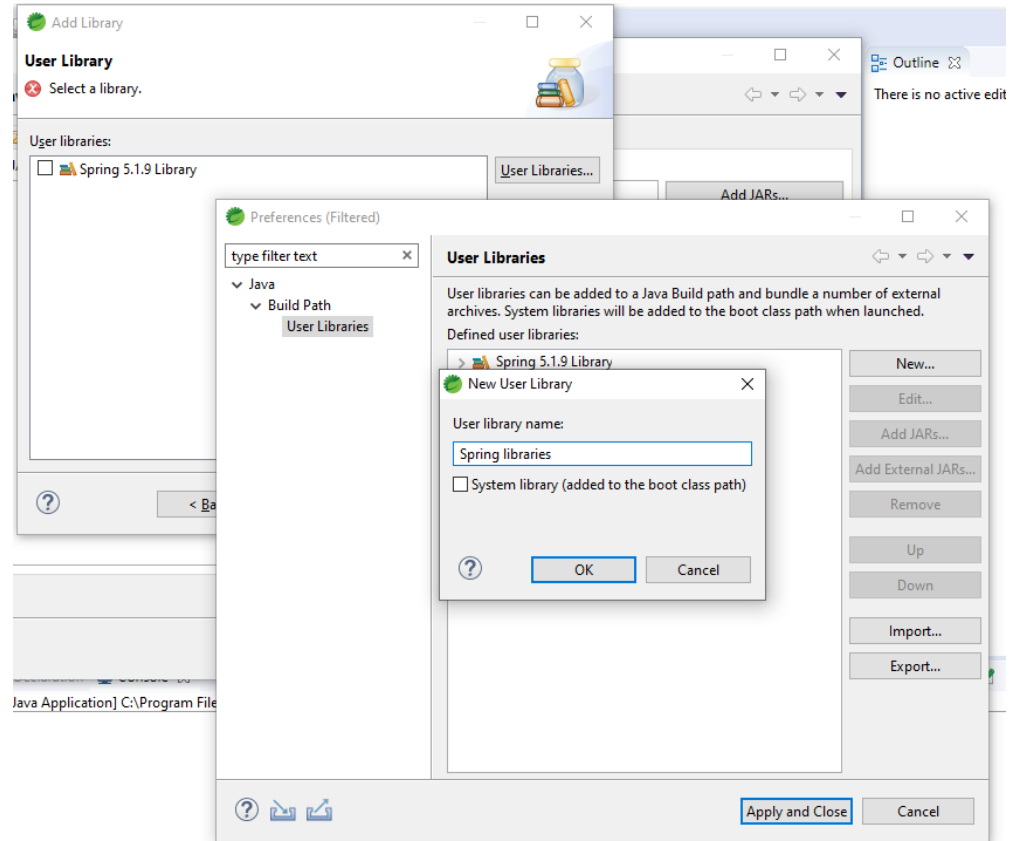
1. Créer une librairie
utilisateur...



Instanciation simple d'un bean

Ajouter les librairies Spring dans le « Java build path » :

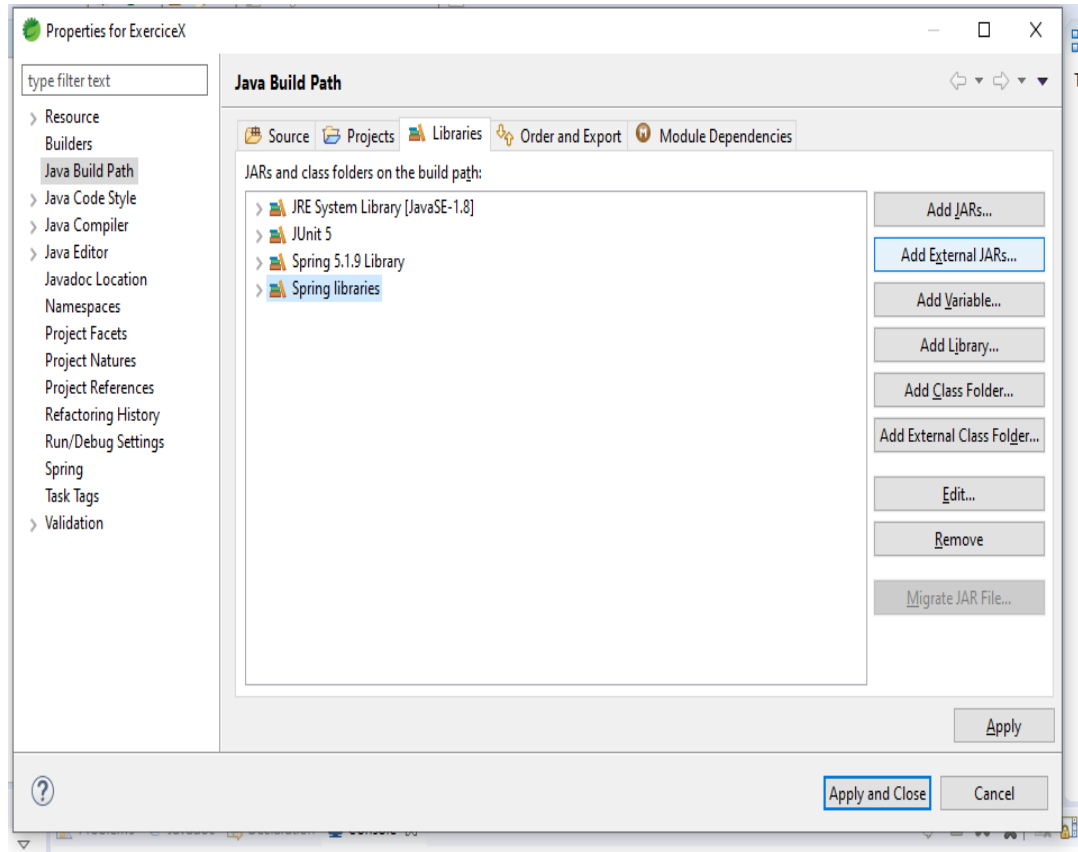
1. Créer une librairie utilisateur en saisissant son nom (Spring libraries ici).



Instanciation simple d'un bean

Ajouter les librairies Spring dans le « Java build path » :

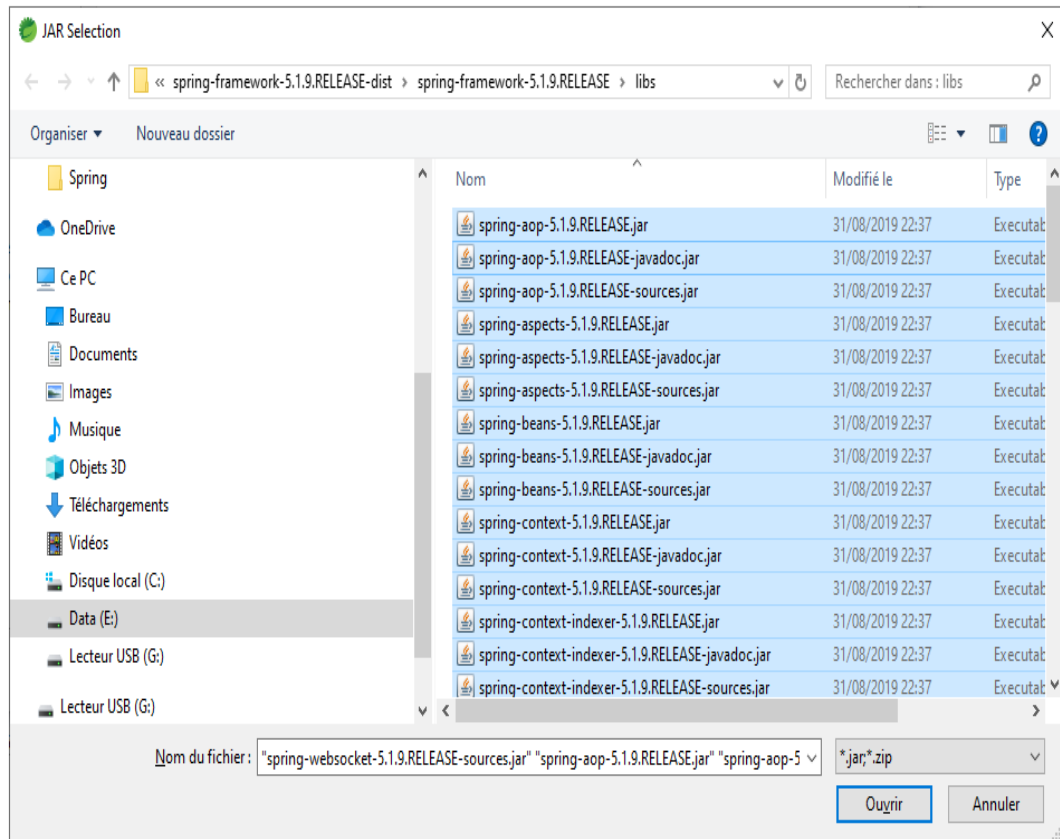
2. Ajouter les fichiers JAR de Spring dans cette librairie utilisateur...



Instanciation simple d'un bean

Ajouter les librairies Spring dans le « Java build path » :

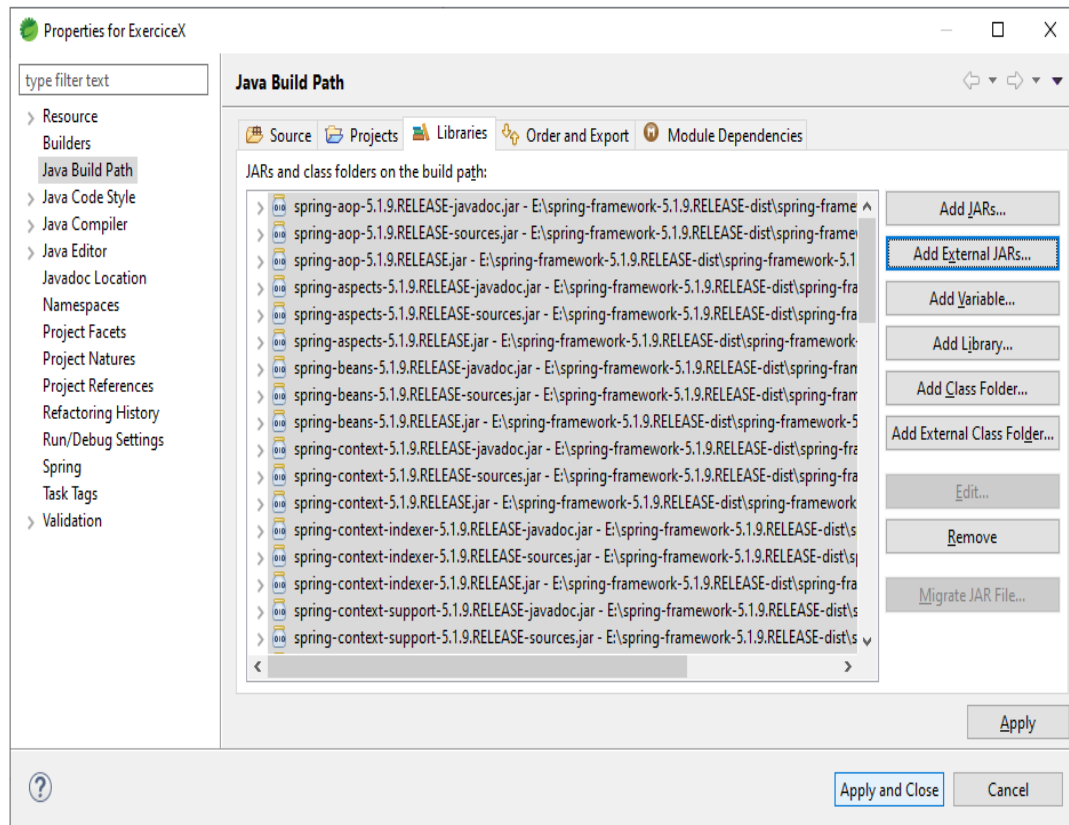
2. Ajouter les fichiers JAR de Spring dans cette librairie utilisateur en sélectionnant tous les fichiers JAR installés.



Instanciation simple d'un bean

Ajouter les librairies Spring dans le « Java build path » :

3. Appliquer les modifications.



Instanciación simple d'un bean

Il reste à développer les beans (classes) demandés puis à créer un (tout petit) programme de test qui instanciera les beans (classes) demandés et affichera ce que contient la propriété immobilière.

A vous !

Spring - Exercices

Exercice 02 : invocation de constructeurs avec arguments

Constructeurs avec arguments

On garde le même sujet : l'instanciation d'une propriété immobilière.

Cette propriété comporte toujours un bâtiment et un terrain.

Cependant, cette fois on ajoute quelques informations sur cette propriété immobilière :

- Adresse de la propriété immobilière
- Nombre de pièces et superficie du bâtiment
- Superficie du terrain

Constructeurs avec arguments

La consigne est d'utiliser des constructeurs avec arguments lorsque c'est indiqué.

On créera un nouveau projet Java.

A vous !

Spring - Exercices

Exercice 03 : instantiation par fabrique

Instanciation par fabrique

On garde toujours le même sujet : l'instanciation d'une propriété immobilière comportant un bâtiment et un terrain avec quelques informations (superficie, nombre de pièces...).

La consigne est toujours de faire créer les différents beans « propriété immobilière », « bâtiment » et « terrain » par l'IoC container, mais cette fois-ci en passant par une fabrique.

Instanciación por fábrica

On créera un nouveau projet Java.

A vous !

Spring - Exercices

Exercice 04 : utilisation d'interfaces

Utilisation d'interfaces

Cette fois, on veut pouvoir changer de type de `Batiment` ou de `Terrain` utilisé dans la `ProprieteImmobiliere` sans modifier son code.

Par conséquent, la classe `ProprieteImmobiliere` utilisera des interfaces pour accéder au `Batiment` et au `Terrain`.

Utilisation d'interfaces

Et pour observer la souplesse qui dérive de cette nouvelle organisation, on fera créer par l'loC container deux instances différentes de `ProprieteImmobiliere` :

- Une qui possèdera un `Batiment` en pierre sur un `Terrain` avec gazon
- Une autre qui possèdera un `Batiment` en brique sur un `Terrain` goudronné.

Utilisation d'interfaces

Un `Batiment` possédera une méthode permettant de connaître sa composition (pierre ou brique), en plus de pouvoir afficher sa superficie et son nombre de pieces.

Un `Terrain` possédera une méthode permettant de savoir ce qui le recouvre (gazon ou béton), en plus de pouvoir afficher sa superficie.

Utilisation d'interfaces

Bien entendu, les choses seront organisées de manière à minimiser les modifications nécessaires et à rendre les choses les plus simples et logiques possible.

Exemple de question à se poser : est-il nécessaire de modifier les constructeurs des classes `Batiment` et `Terrain` ?

On créera un nouveau projet Java.

A vous !

Spring - Exercices

Exercice 05 : définition de bean par annotation

Définition de bean par annotation

On veut pouvoir définir un bean `Batiment` appelé « `Capitole` » et le récupérer depuis l'IoC container mais sans utiliser de fichier XML de métadonnées.

Ce `Batiment` portera les informations suivantes : composition, superficie et nombre de pièces. Elles seront accessibles au travers des mêmes méthodes que dans l'exercice 04 (`affiche()` et `composition()`).

Définition de bean par annotation

Par conséquent, le programme utilisateur de ce bean contiendra les lignes suivantes :

```
Batiment batiment = (Batiment) context.getBean("Capitole");  
  
System.out.println("Batiment en " + batiment.composition()  
+ " : " + batiment.affiche());
```

A vous !

Spring - Exercices

Exercice 06 : injection de dépendances par annotation

Injection de dépendances par annotation

On cherche à créer un satellite. Un satellite est composé de différentes parties :

- Une plateforme mécanique
- Une charge utile
- Deux panneaux solaires
- Une antenne radio

Injection de dépendances par annotation

Certaines contraintes s'appliquent à la manière de construire le satellite :

- La plateforme mécanique doit être injectée par un constructeur
- La charge utile doit aussi être injectée par un constructeur
- Les panneaux solaires doivent être injectés par un setter
- L'antenne radio doit être injectée par propriété

Injection de dépendances par annotation

Chacun de ces éléments possède un identifiant (« antenne radio », « plateforme mécanique », etc.). Pour vérifier la bonne construction du satellite, une fois celui-ci achevé, on affichera l'identifiant de chacune de ses pièces.

On n'utilisera pas de fichier XML de métadonnées, uniquement des annotations.

A vous !

Spring - Exercices

Exercice 07 : injection de collections

Injection de collections

Le but ici est de définir et créer une bibliothèque pouvant recevoir des livres appartenant à différentes catégories :

- Livres d'histoire
- Polars
- Romans de science-fiction
- Essais philosophiques
- ...

Injection de collections

Un livre comporte les informations suivantes :

- Titre
- Auteur
- Catégorie
- ISBN (identifiant international normalisé du livre).

Injection de collections

On veut se simplifier la tâche et donc faire le moins de choses possibles :

- Coder les classes les plus simples possibles (POJO)
- Définir les livres dans un fichier XML de métadonnées
- Ranger automatiquement les livres par catégorie dans la bibliothèque.

Injection de collections

Attention : la bibliothèque étant toute neuve, il est possible qu'il n'y ait pas de livre dans certaines catégories...

A vous !

Spring - Exercices

Exercice 08 : injection de valeurs de propriétés

Injection de valeurs de propriétés

Le but ici est de réaliser un composant simple permettant d'afficher certaines informations système et autres :

- Son class path courant
- L'identifiant du compte utilisateur sous lequel s'exécute le programme
- La racine de l'installation de Java (Java HOME)
- Une valeur récupérée d'un fichier de propriétés
- La valeur par défaut associée à une propriété inconnue

Injection de valeurs de propriétés

A vous !

Spring - Exercices

Exercice 09 : configuration de container basée sur Java

Configuration de container basée sur Java

On veut développer une application permettant de construire une voiture composée d'un moteur et d'un châssis et d'afficher les propriétés du moteur et du châssis.

Le moteur comporte les propriétés suivantes :

- Nom du fabricant
- Cylindrée
- Puissance.

Configuration de container basée sur Java

Le châssis comporte les propriétés suivantes :

- Nom du fabricant
- Nom du modèle
- Année de fabrication.

On veut pouvoir indiquer les valeurs des propriétés du moteur et du châssis au moyen de deux fichiers de propriétés séparés et placés dans un répertoire `conf/` sous la racine du projet .

Configuration de container basée sur Java

L'absence d'un ou des deux fichiers de propriétés ne doit pas entraîner de dysfonctionnement de l'application.

Pour ce projet, la configuration de l'IoC container doit être basée sur Java. Il n'y aura donc pas besoin d'un fichier XML de métadonnées.

L'injection de dépendances doit être effectuée automatiquement et le fichier Java de configuration du projet doit être minimal.

A vous !