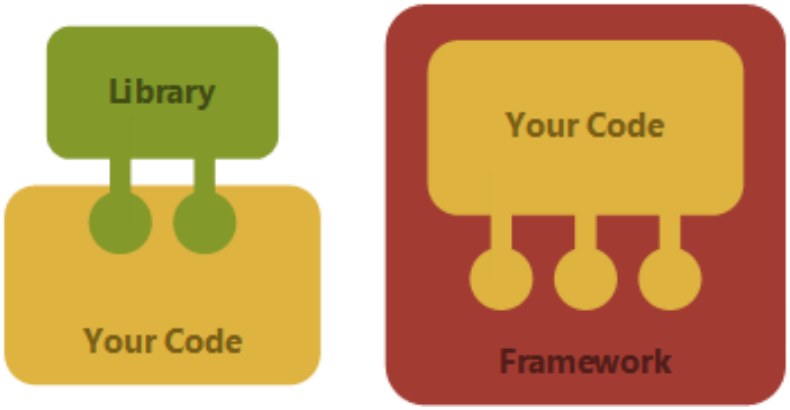
Why Framework?

Framework	it is the one who is calling your code.
Library	you are making a call to a code.



The main difference is in the **“Inversion of Control”**

When we call a method from a library, we are **in control**.
But in framework, the control is inverted i.e. the framework calls us.

Spring Framework

- The **Spring Framework** is an application framework for java.
- It has become popular in the Java community as replacement of EJB.
- First version was written by **Rod Johnson** in 2002.
- It is developed by **Pivotal Software, Inc.** company.

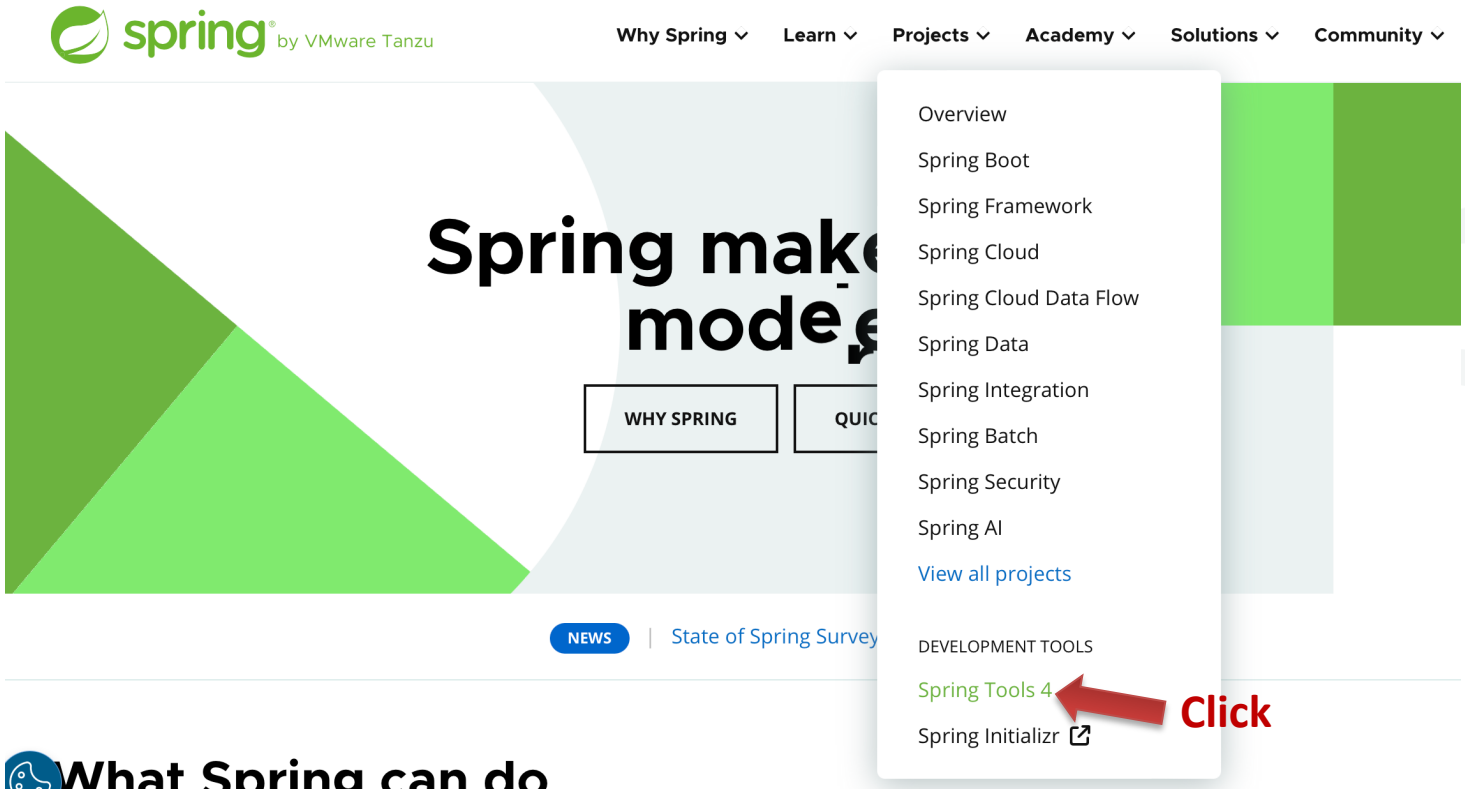
Tools Required

Spring IDE

Spring Tool Suite

Download STS

Go to https://www.spring.io



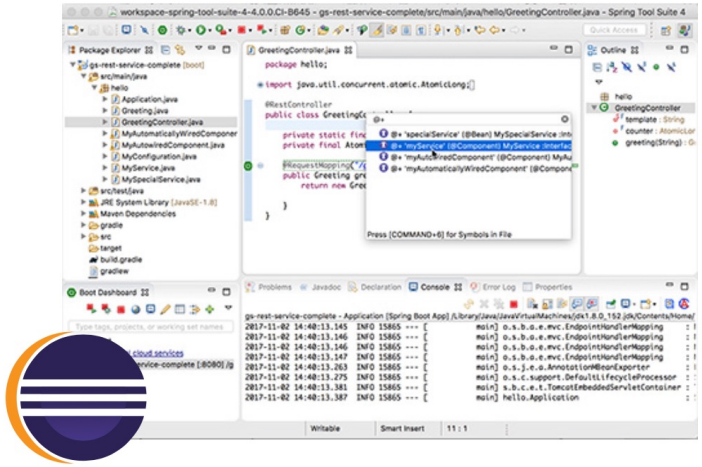
Download STS

Download according to your OS.

Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4. Free. Open source.

- 4.25.0 - LINUX X86_64
- 4.25.0 - LINUX ARM_64
- 4.25.0 - MACOS X86_64
- 4.25.0 - MACOS ARM_64
- 4.25.0 - WINDOWS X86_64



Click, According to your OS.

IoC Container

Spring IoC is the mechanism to achieve loose-coupling between Objects dependencies.
Spring IoC container **injects** dependencies into an object and make it ready for our use.
Types of IoC container:

- **BeanFactory**
- **ApplicationContext**

Using **BeanFactory :** *[Not in use now]

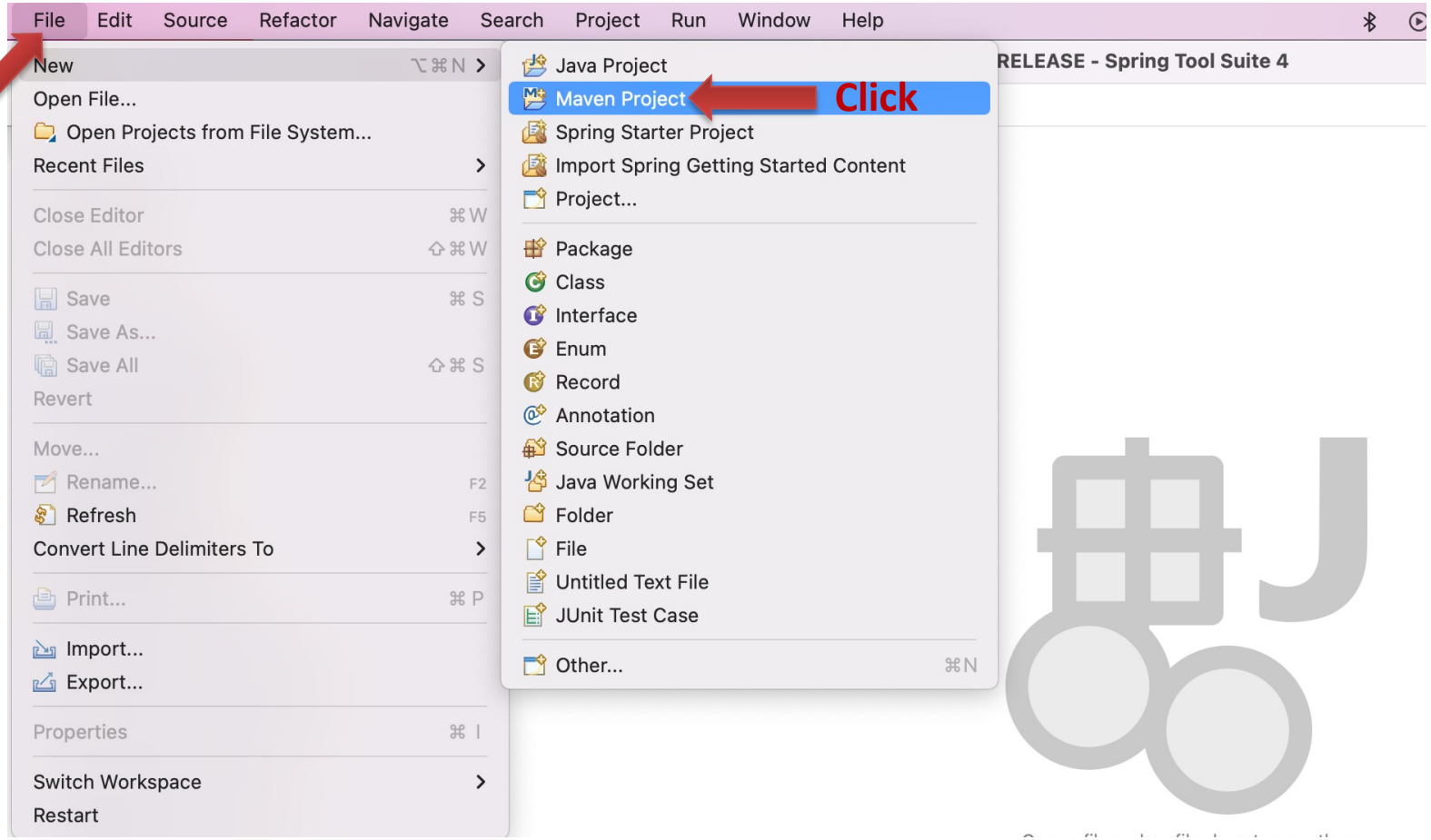
```
Resource resource=new ClassPathResource("Beans.xml");
BeanFactory factory=new XmlBeanFactory(resource);
```

Using **ApplicationContext** : 

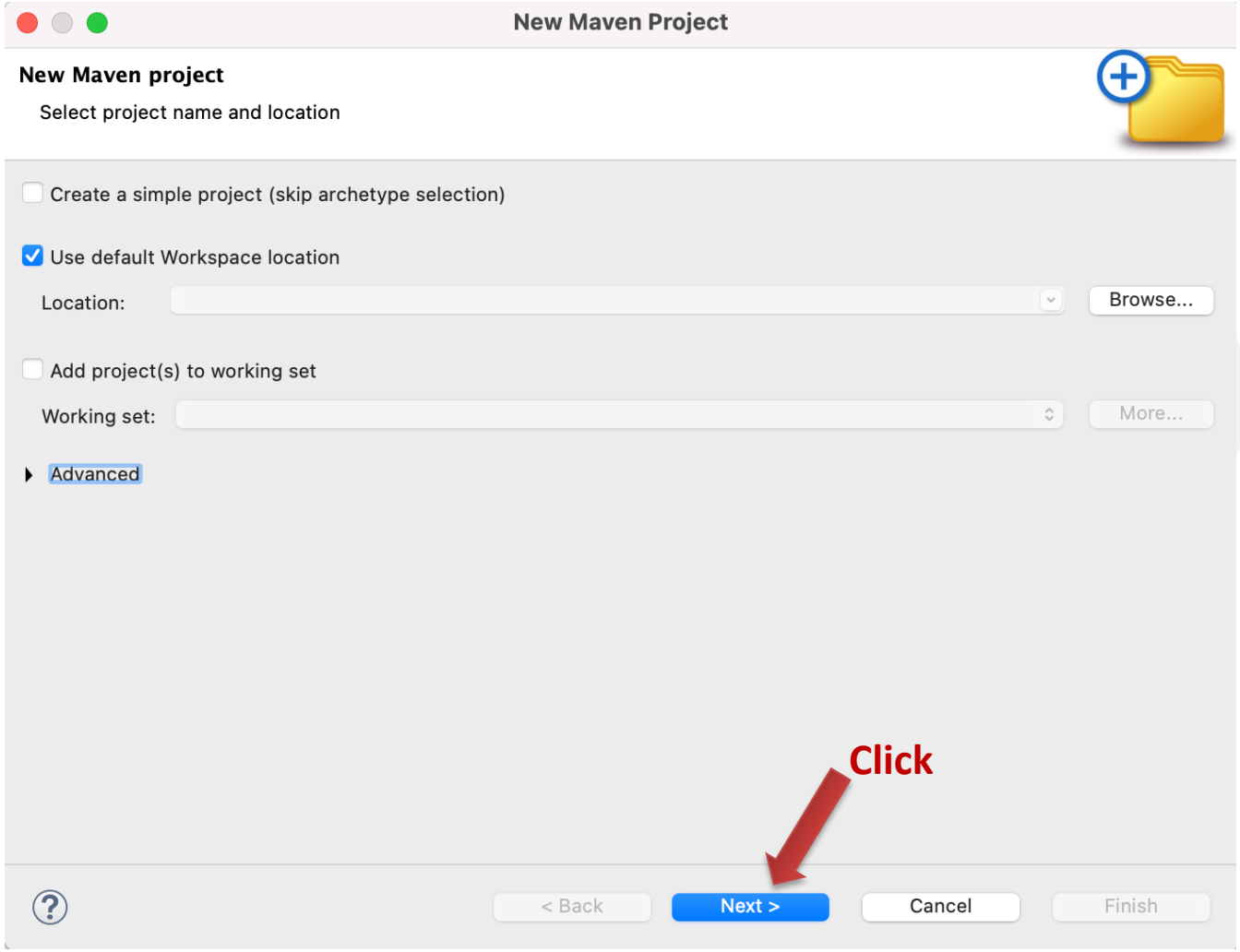
```
ApplicationContext ctx=new ClassPathXmlApplicationContext("Beans.xml");
```

Step by Step: First Spring Project

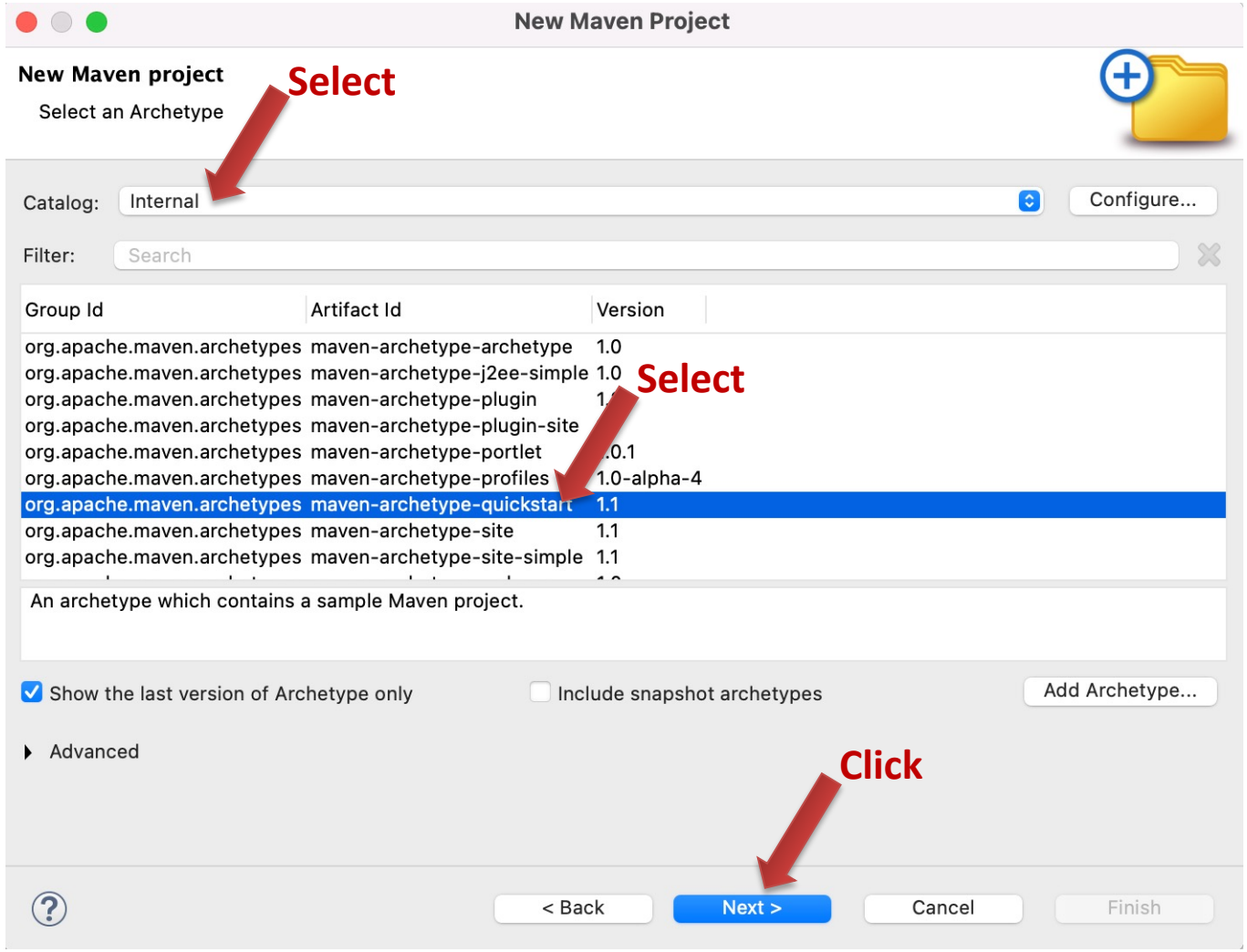
Open Spring Tool Suit or Eclipse. Create new maven project:



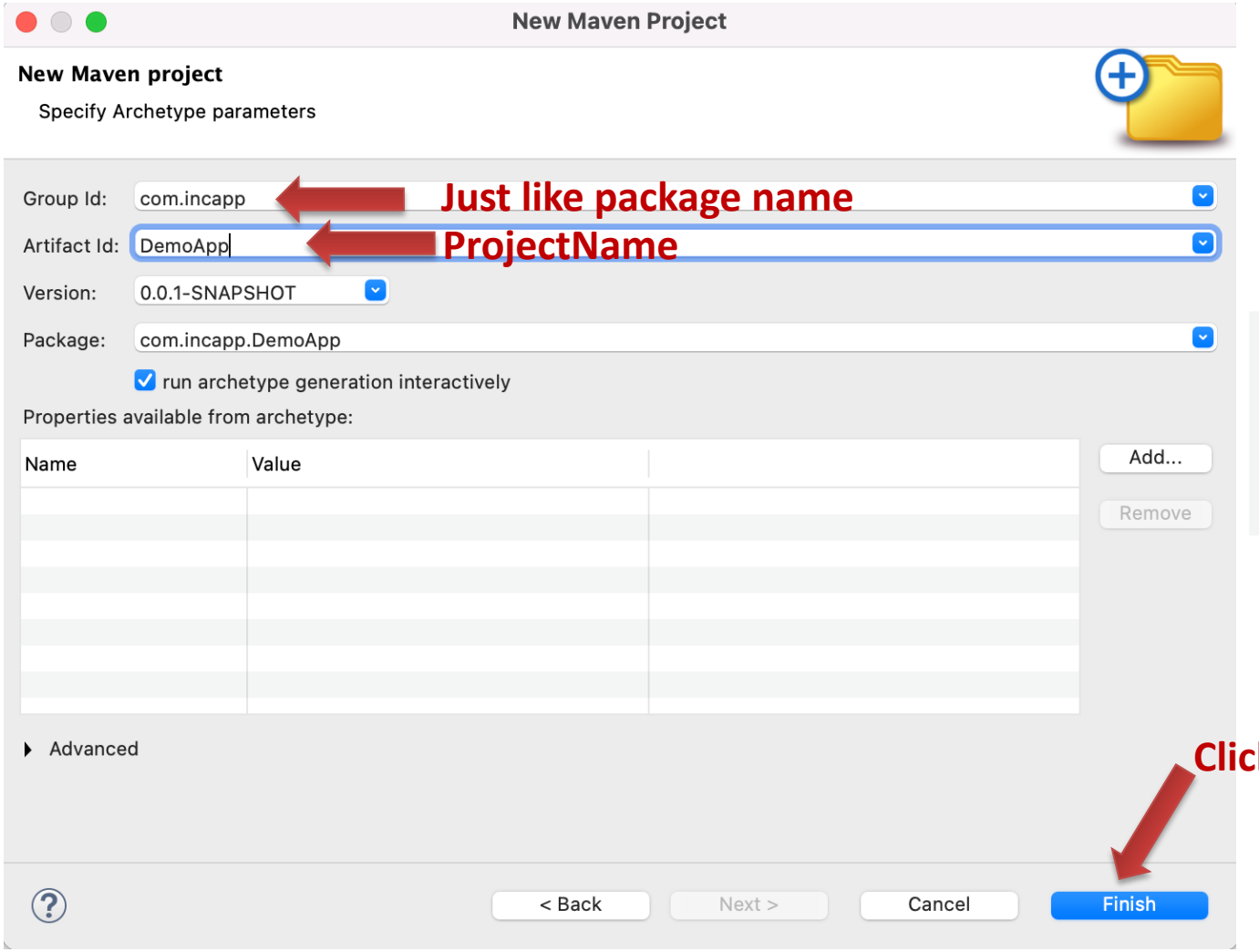
Step by Step: First Spring Project



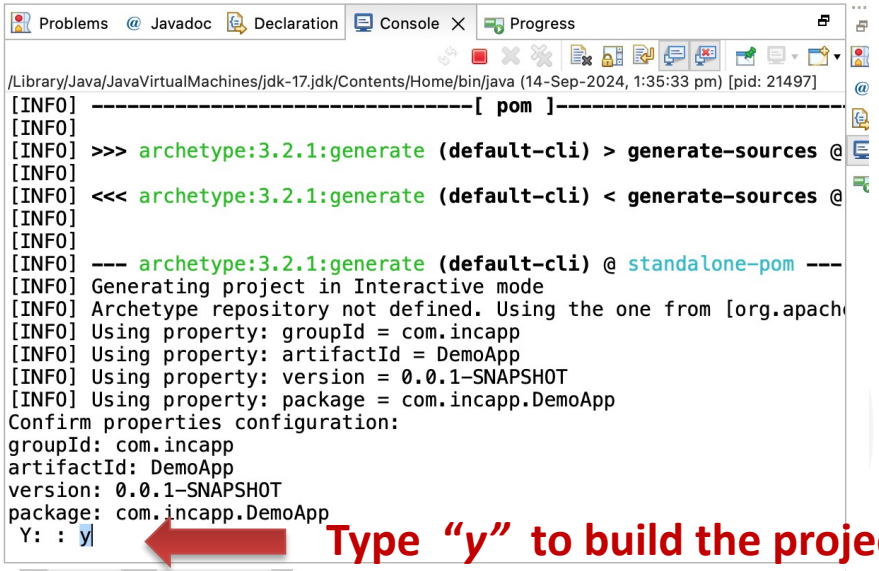
Step by Step: First Spring Project



Step by Step: First Spring Project

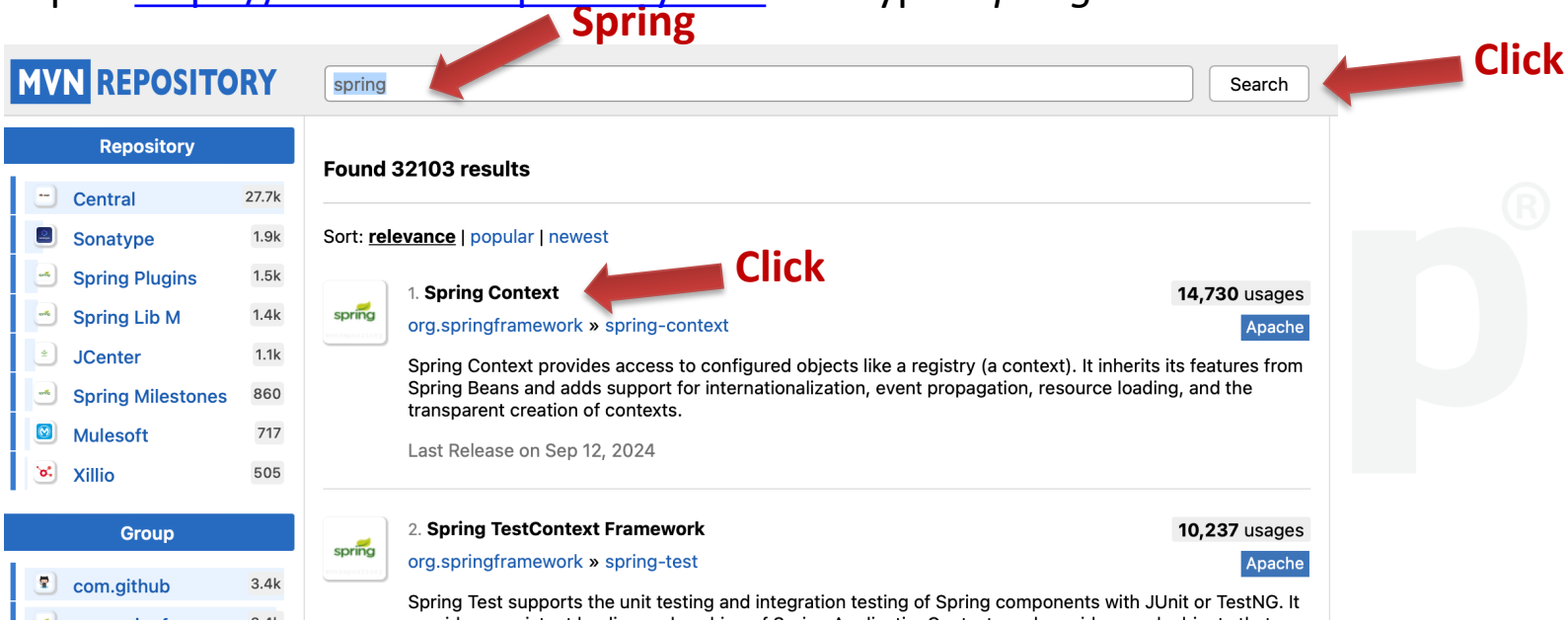


Step by Step: First Spring Project

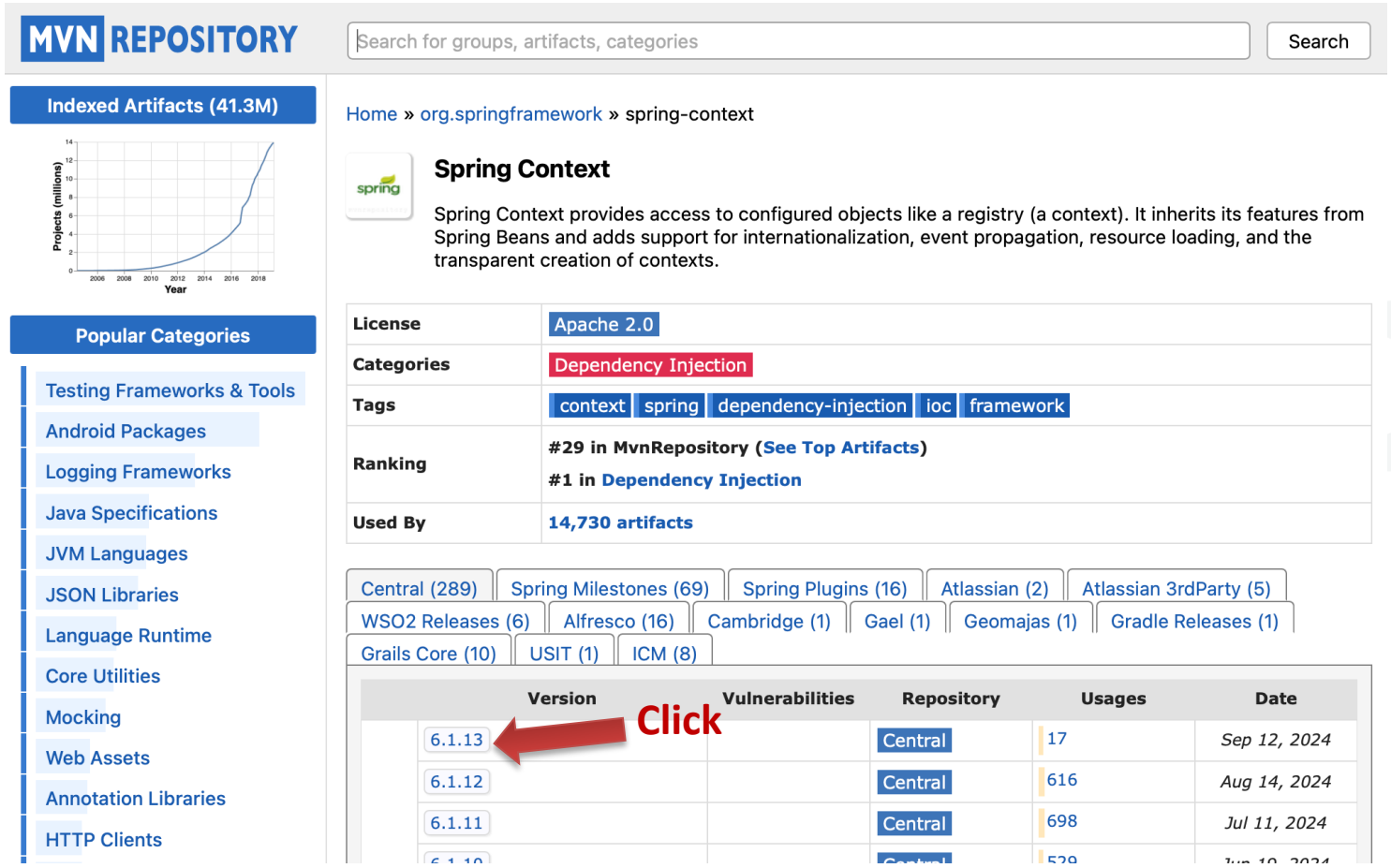


Step by Step: First Spring Project

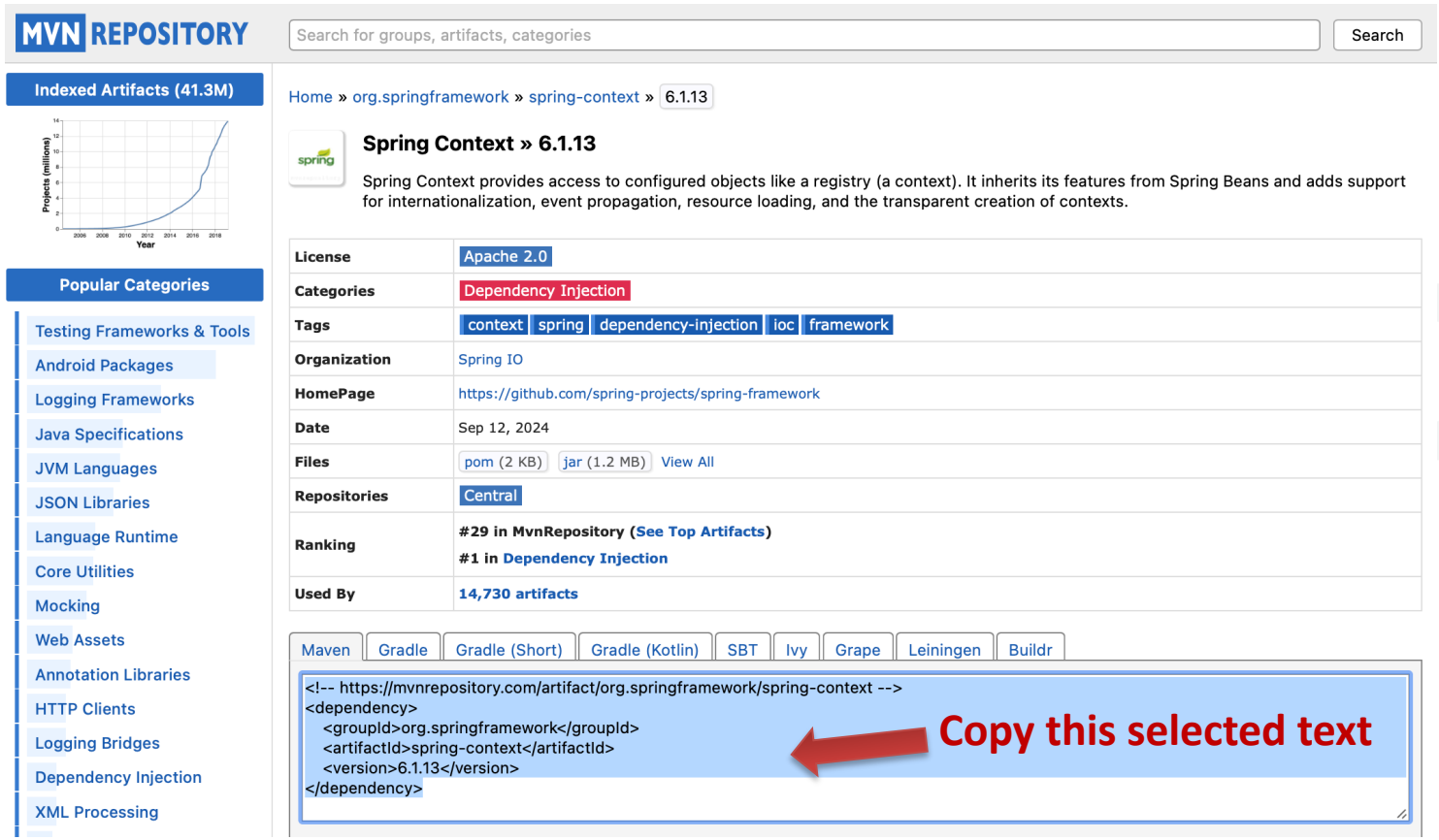
To add Spring framework in the project, go to maven repository.
Open https://www.mvnrepository.com and type "*spring*" in search section:



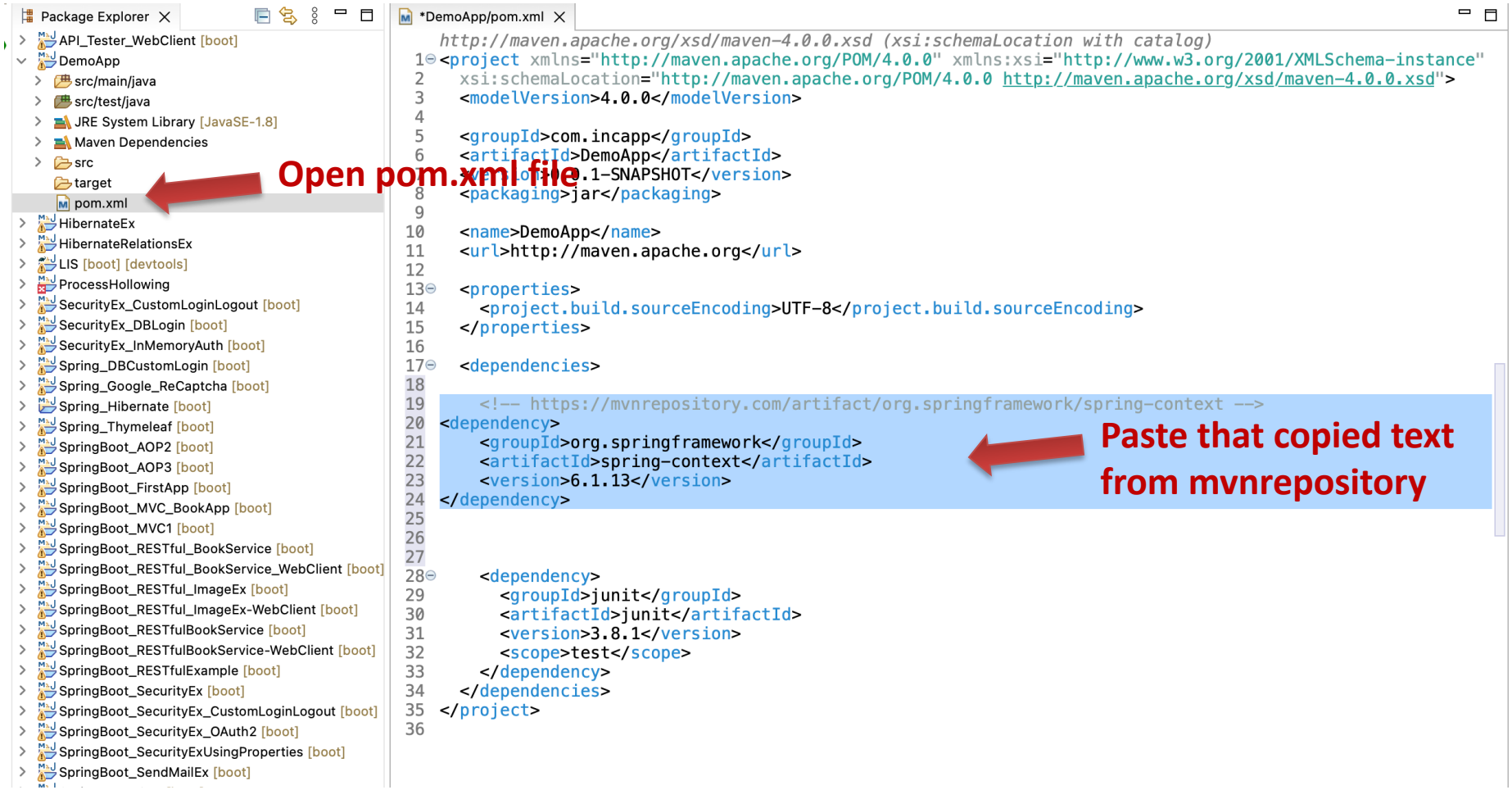
Step by Step: First Spring Project



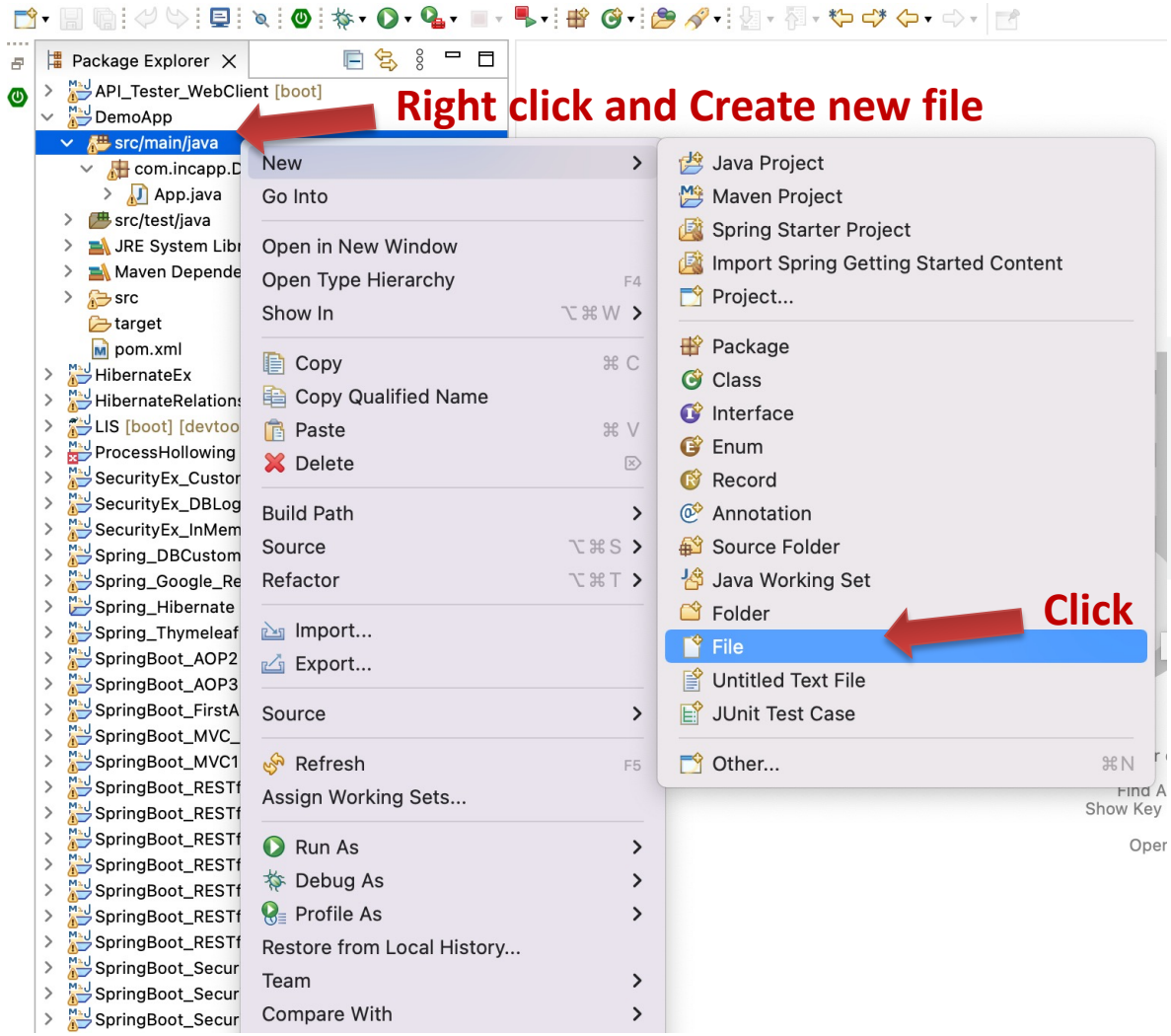
Step by Step: First Spring Project



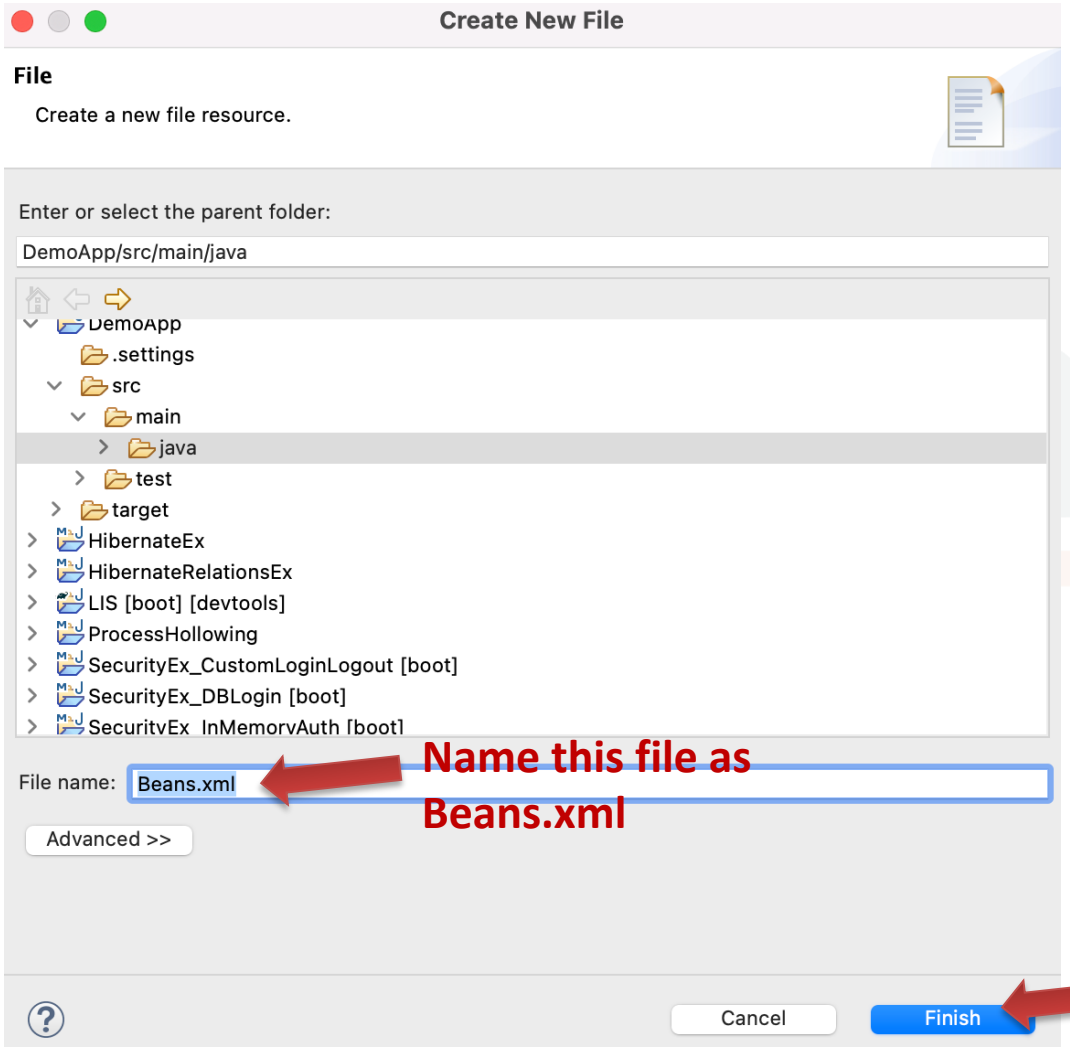
Step by Step: First Spring Project



Step by Step: First Spring Project

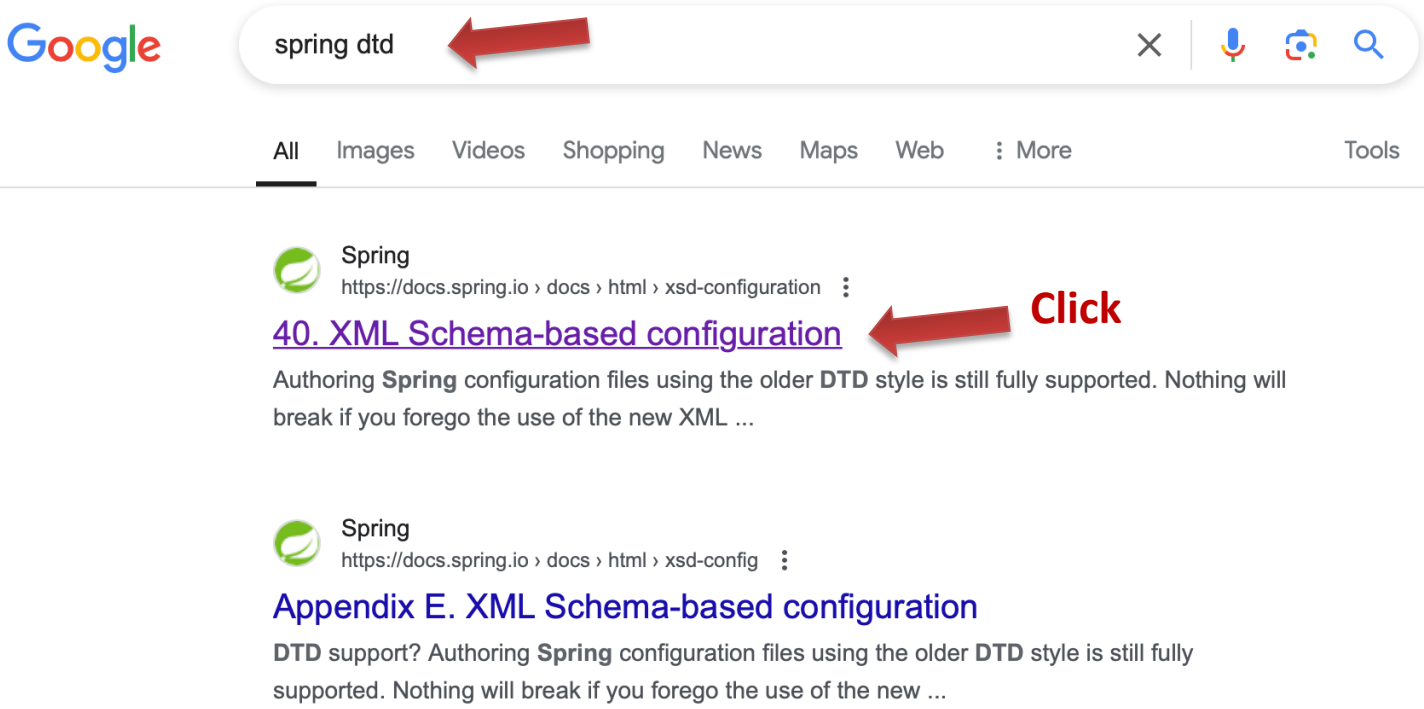


Step by Step: First Spring Project



Step by Step: First Spring Project

Open https://www.google.com and type "*spring dtd*"



Step by Step: First Spring Project

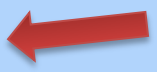
40.2.1 Referencing the schemas

To switch over from the DTD-style to the new XML Schema-style, you need to make the following change.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
        "http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans>

<!-- bean definitions here -->

</beans>
```

The equivalent file in the XML Schema-style would be…

```
<?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 definitions here -->

</beans>
```

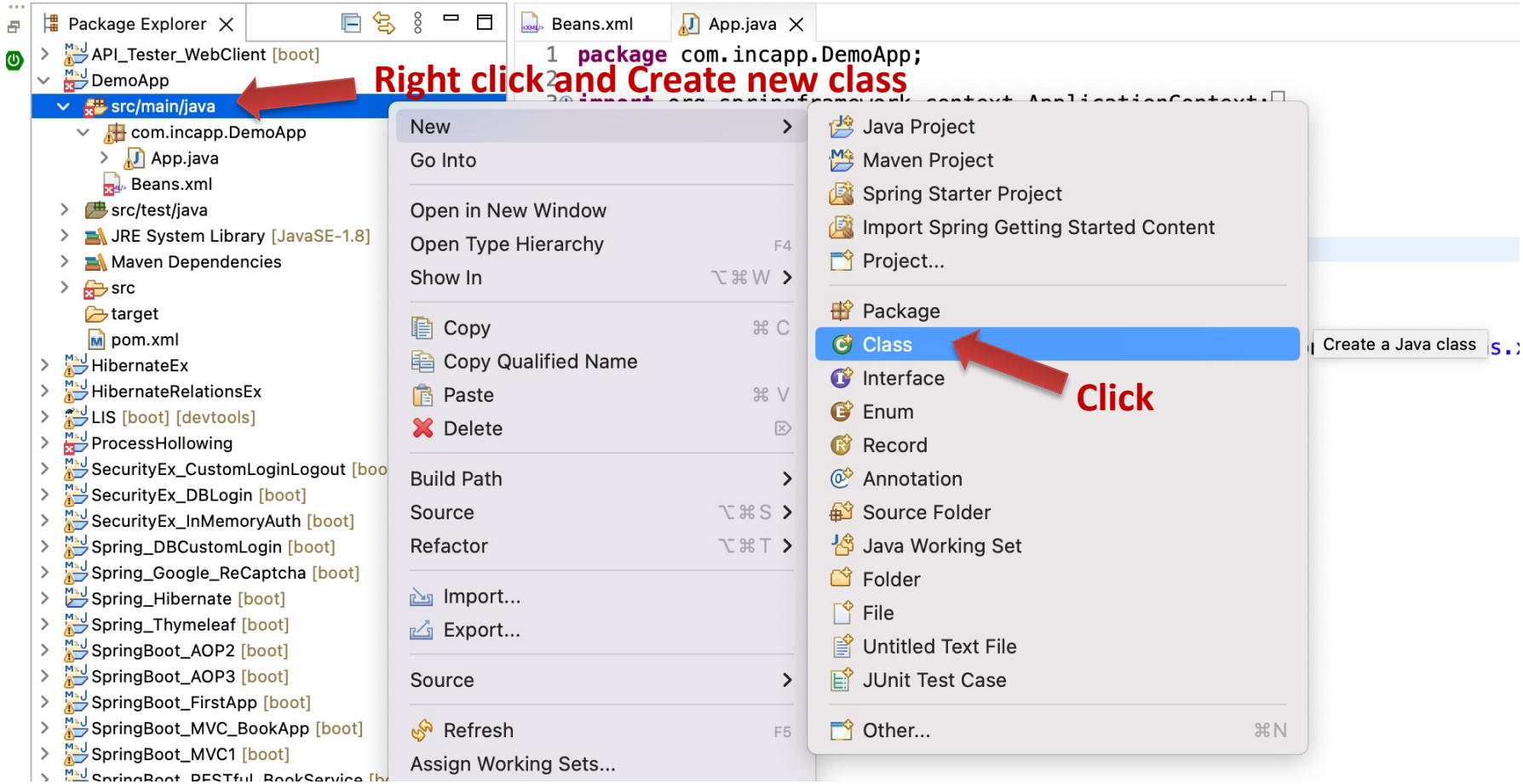


Copy this selected text

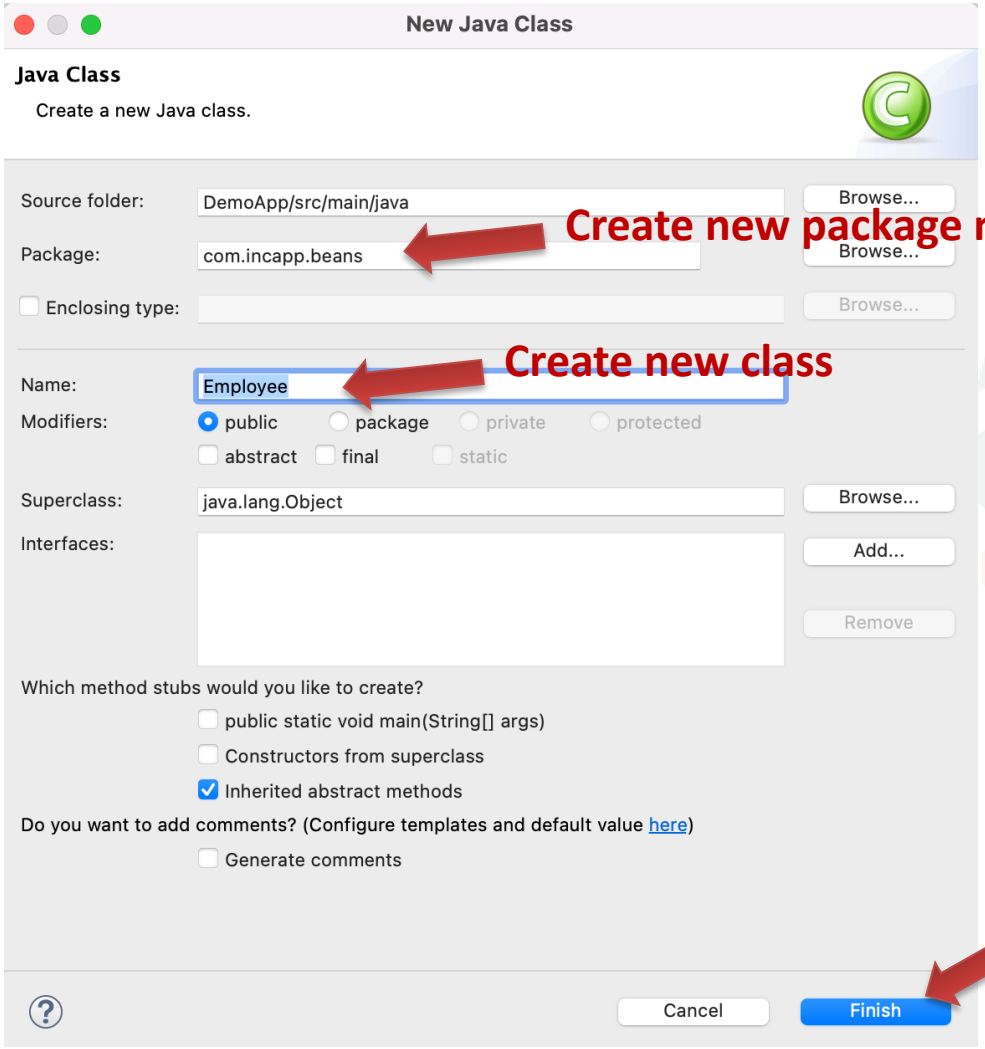
Step by Step: First Spring Project

Step by Step: First Spring Project

Create new Java Bean named as Employee.

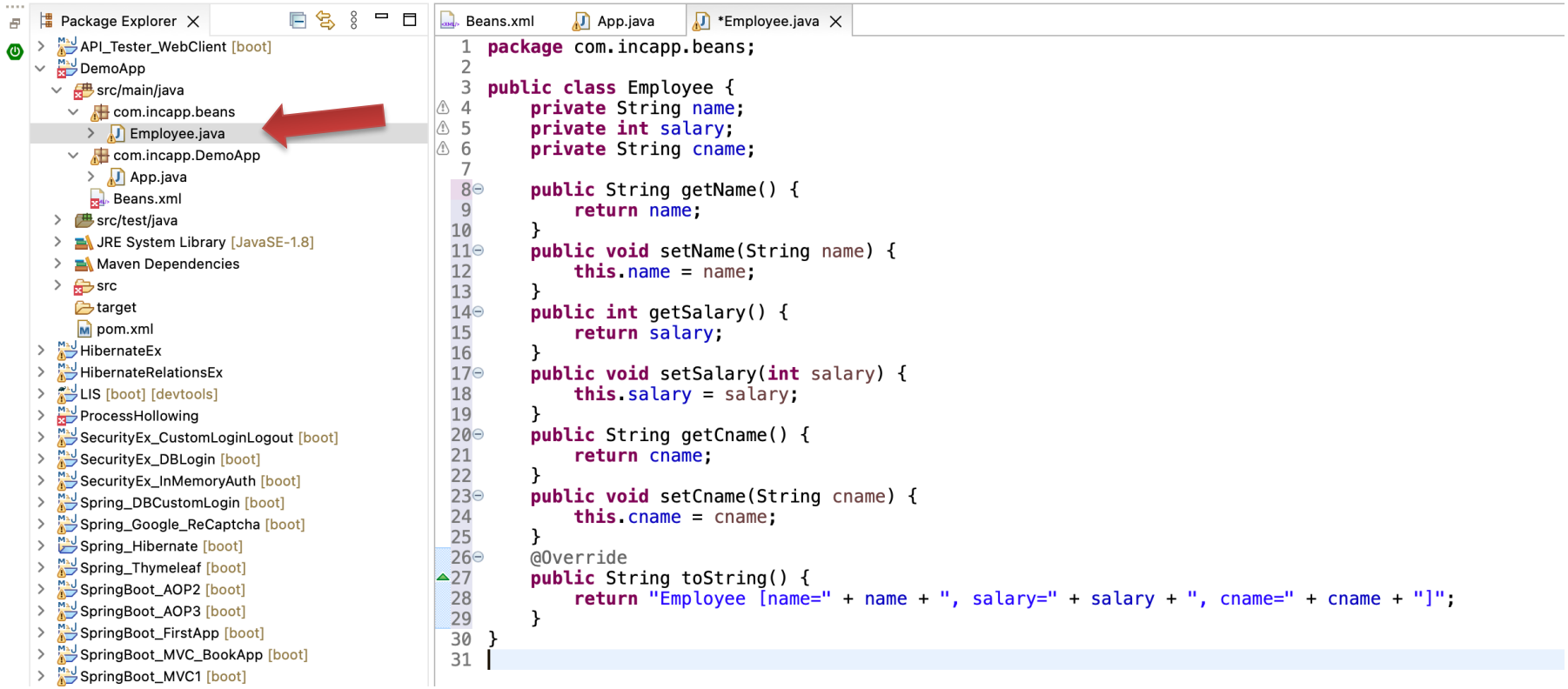


Step by Step: First Spring Project



Step by Step: First Spring Project

Open Employee class.
Create some fields and generate getter/setter & toString() method.

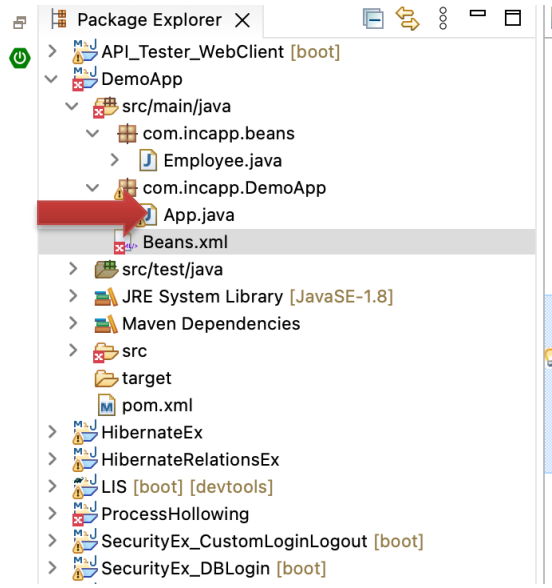


Step by Step: First Spring Project

Open Beans.xml file.
Declare the Employee bean inside the beans tag.

Step by Step: First Spring Project

Step by Step: First Spring Project



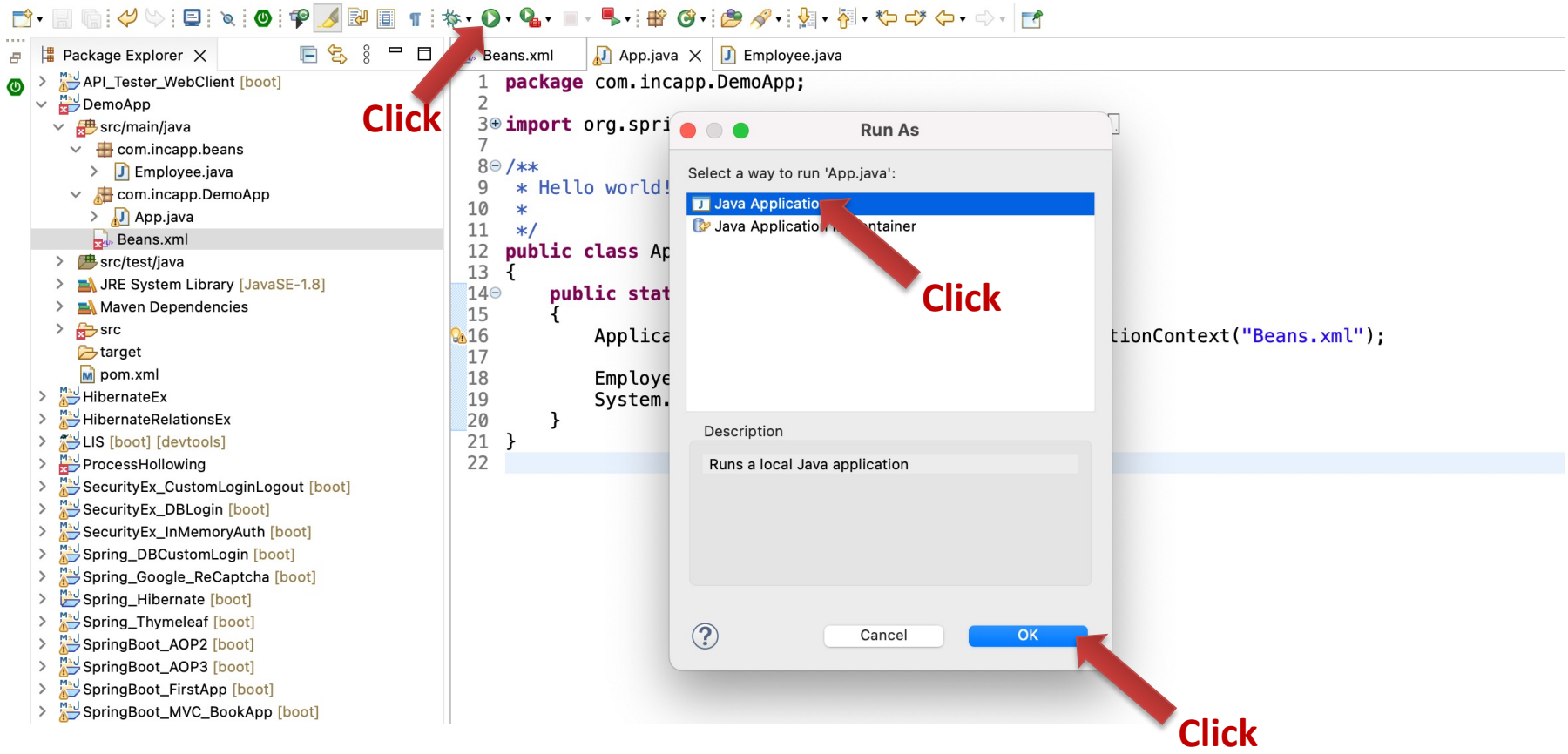
Beans.xml | App.java | Employee.java

```
1  package com.incapp.DemoApp;
2
3  import org.springframework.context.ApplicationContext;
7
8  /**
9   * Hello world!
10  *
11  */
12 public class App
13 {
14     public static void main( String[] args )
15     {
16         ApplicationContext ctx=new ClassPathXmlApplicationContext("Beans.xml");
17
18         Employee a=ctx.getBean("e",Employee.class);
19         System.out.println(a);
20     }
21 }
22
```

Getting Employee object from Spring framework

Step by Step: First Spring Project

Run this project



Step by Step: First Spring Project

Output of this project

Beans.xml | App.java | Employee.java

```
1  package com.incapp.DemoApp;
2
3  import org.springframework.context.ApplicationContex
7
8  /**
9   * Hello world!
10  *
11  */
12 public class App
13 {
14     public static void main( String[] args )
15     {
16         ApplicationContext ctx=new ClassPathXmlAppli
17
18         Employee a=ctx.getBean("e",Employee.class);
19         System.out.println(a);
20     }
21 }
22
```

Problems | Javadoc | Declaration | Console | Progress

Restore

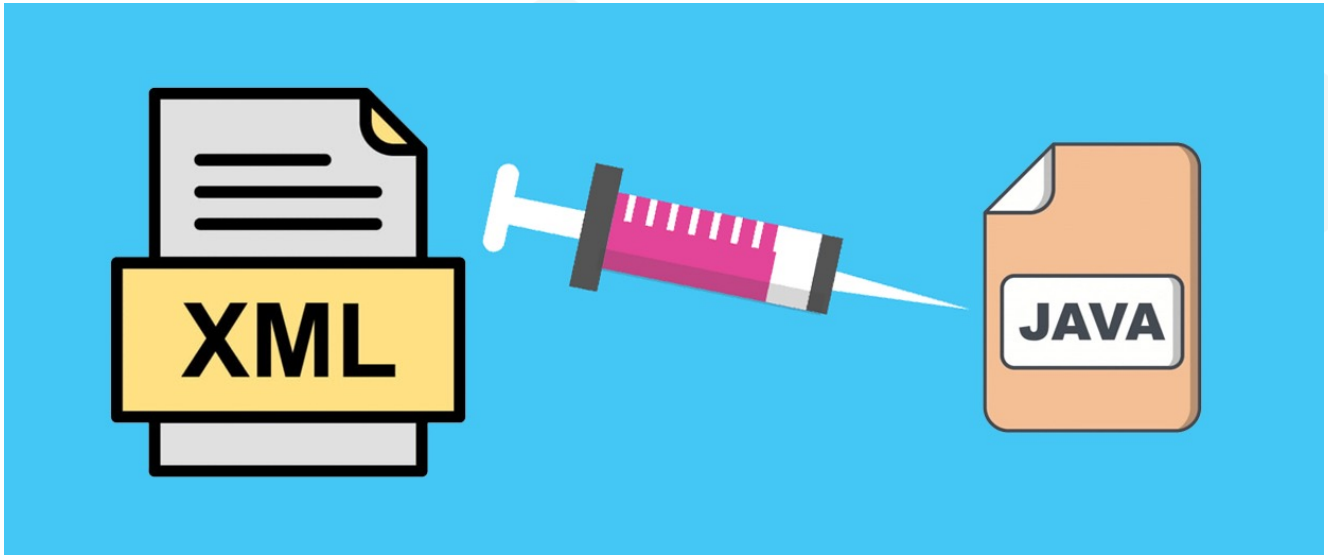
<terminated> App (7) [Java Application] /Library/Java/JavaVirtualMachines/jdk-

```
Employee [name=Ram Kumar, salary=150000, cname=XYZ]
```

Output

DI(Dependency Injection)

You can provide information from external source such as XML file. Here, you do not create the objects instead you just define how they should be created and IoC container will create the objects for you.



Scope Attribute

Scope attribute is used to specify scope of a bean. We have two options:

- **singleton**
- **prototype**

A **singleton bean** is instantiated only once.
A **prototype bean** is instantiated for each time, when it is requested from container.

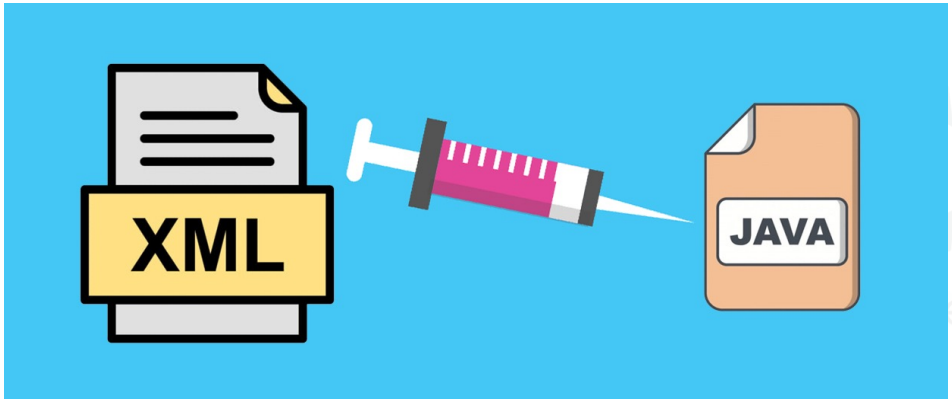
Note:- Default scope of bean is singleton.

DI Approaches

Dependencies can be injected in two ways:

By setter method: *<property name="id" value="10" />*

By constructor: *<constructor-arg type="int" value="10" />*



Setter Method DI Approach

<property name="name of the variable" **value=**"your data" **/>**

<property name="name of the variable" **ref=**"Name of Bean to be Injected" **/>**

Constructor DI Approach

<constructor-arg type="data type**" value="**your data**" />**

Note: **type** attribute is optional, required only if there is some ambiguity among constructor parameters.

<constructor-arg value="your data**" />**

<constructor-arg ref="Name of Bean to be Injected**" />**

Autowiring

Autowiring is the facility through which IoC container detects and injects the dependencies on its own.
Note: can not be used for primitive type dependencies.

It can be of following types:

- no
- byName
- byType and byType with primary attribute
- constructor

Note:- Default value is no.