

# Why Framework?

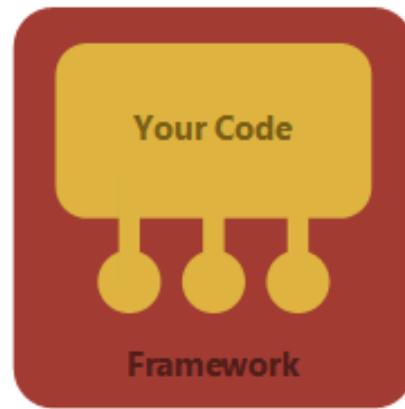
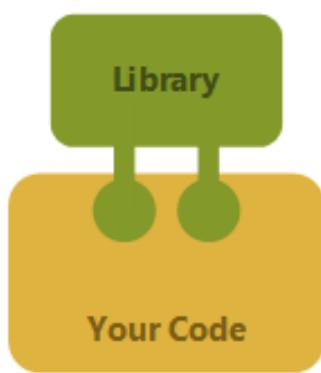


Framework

it is the one who is calling your code.

Library

you are making a call to a code.



capp<sup>®</sup>  
stitute

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>

The screenshot shows the official Spring website (<https://www.spring.io>). At the top, there's a navigation bar with links: Why Spring, Learn, Projects, Academy, Solutions, and Community. The 'Projects' link is currently active, opening a dropdown menu. This menu lists various Spring projects: Overview, Spring Boot, Spring Framework, Spring Cloud, Spring Cloud Data Flow, Spring Data, Spring Integration, Spring Batch, Spring Security, Spring AI, and a link to 'View all projects'. Below the dropdown, there's a section titled 'DEVELOPMENT TOOLS' with links to 'Spring Tools 4' and 'Spring Initializr'. A red arrow points from the text 'Click' to the 'Spring Initializr' link.

- Overview
- Spring Boot
- Spring Framework
- Spring Cloud
- Spring Cloud Data Flow
- Spring Data
- Spring Integration
- Spring Batch
- Spring Security
- Spring AI
- [View all projects](#)

DEVELOPMENT TOOLS

[Spring Tools 4](#) **Click**

[Spring Initializr](#)

What Spring can do

# Download STS

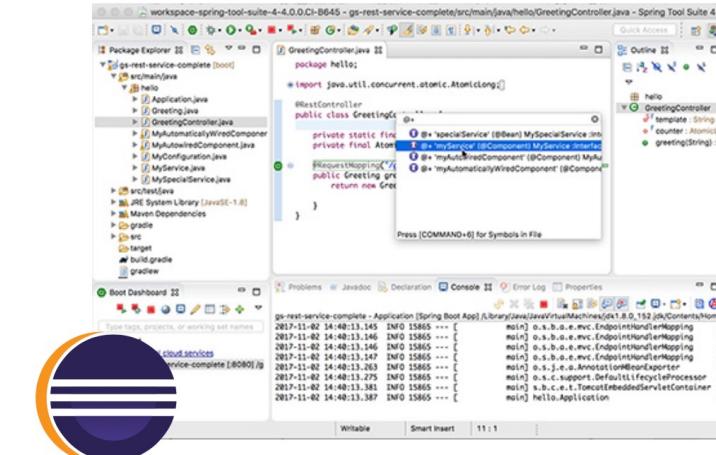


Download according to your OS.

## 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);
```

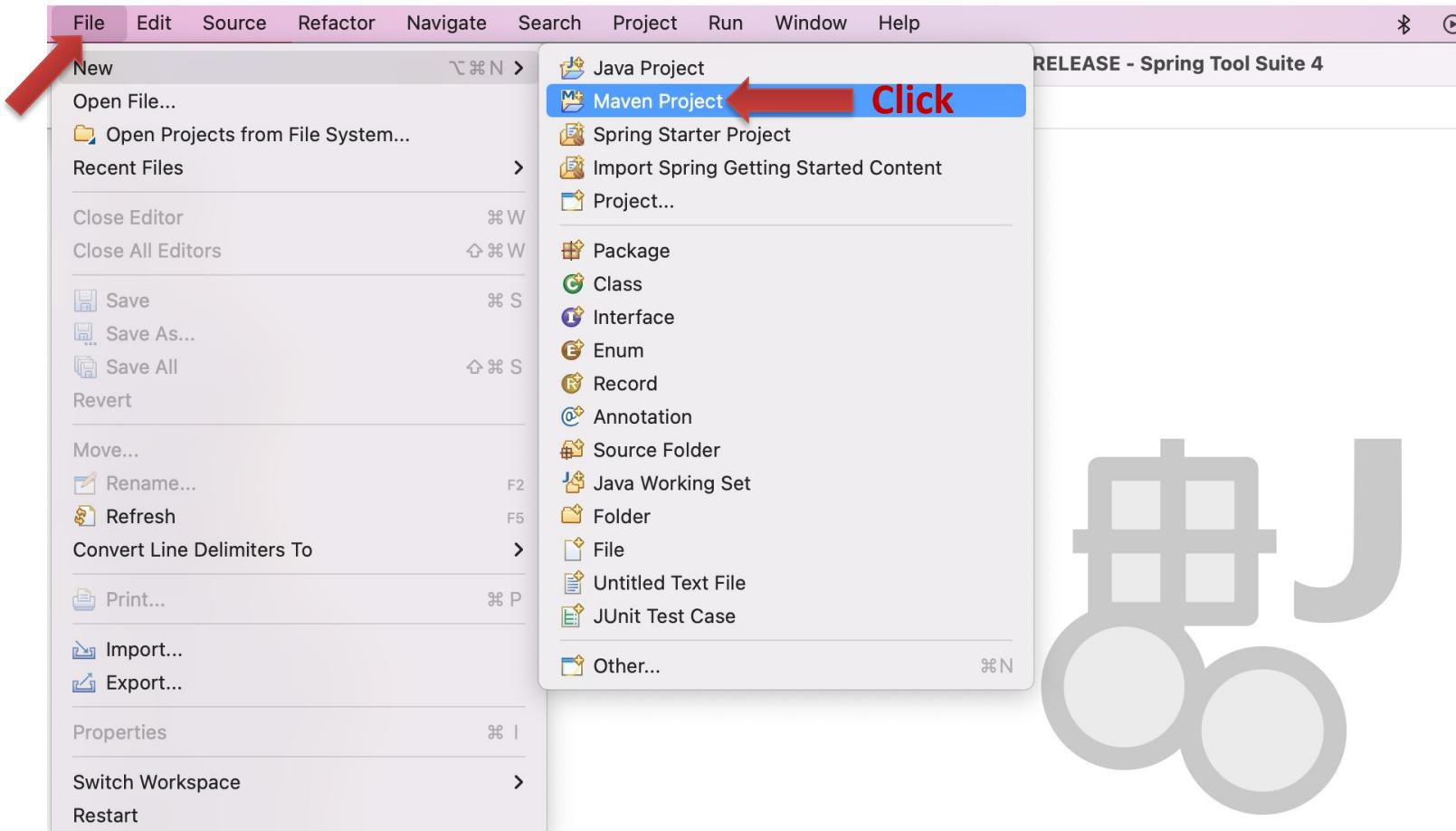
Using **ApplicationContext** : 

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

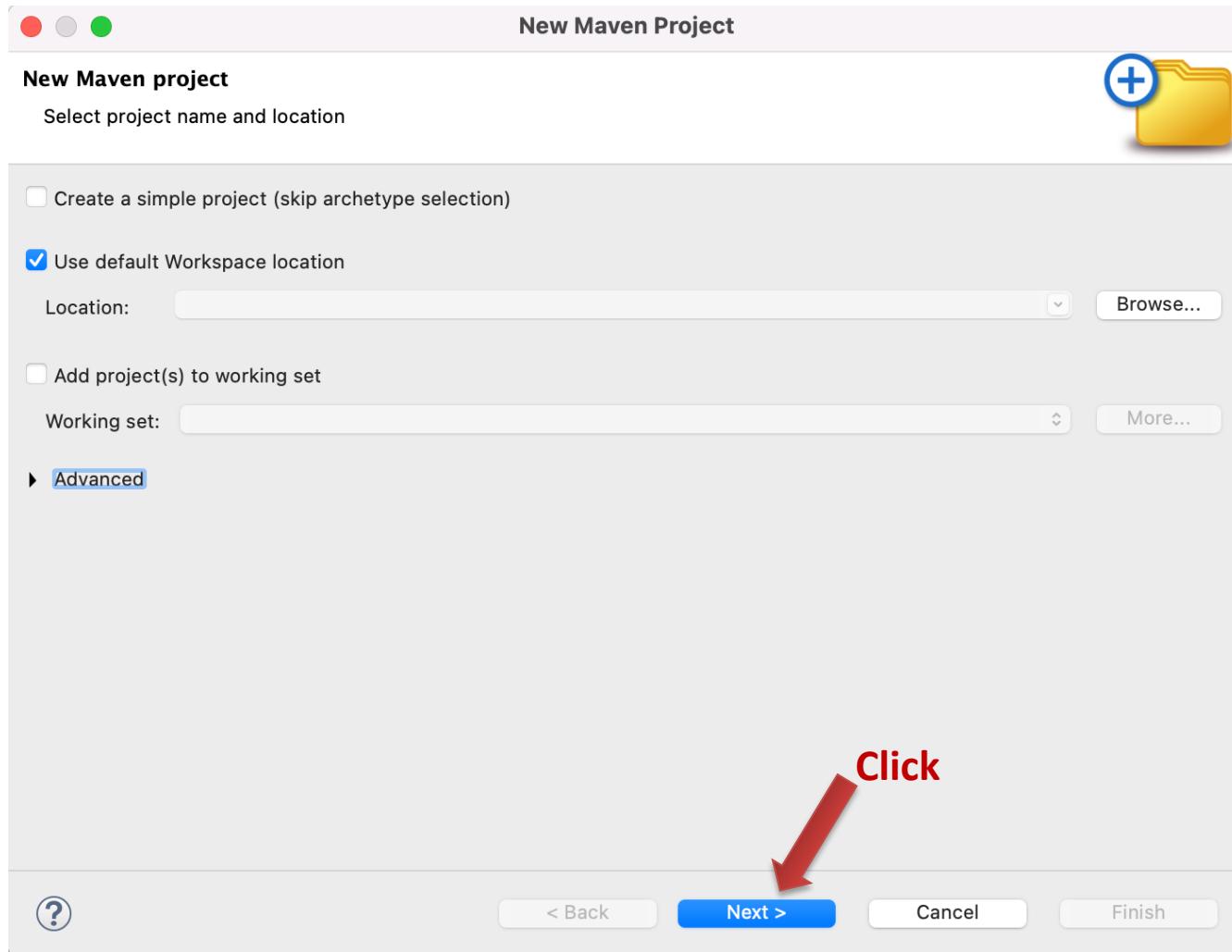
# Step by Step: First Spring Project



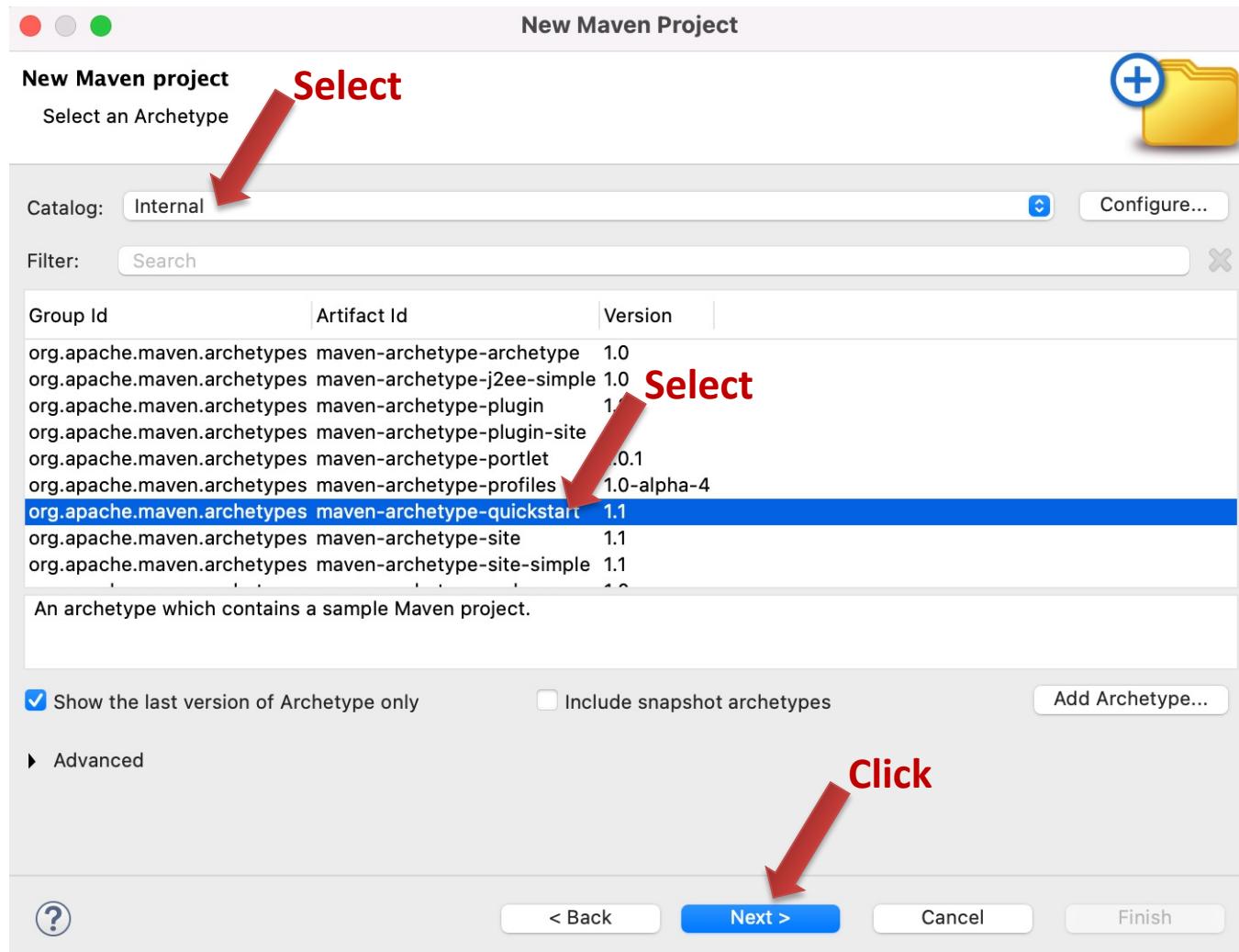
Open Spring Tool Suit or Eclipse. Create new maven project:



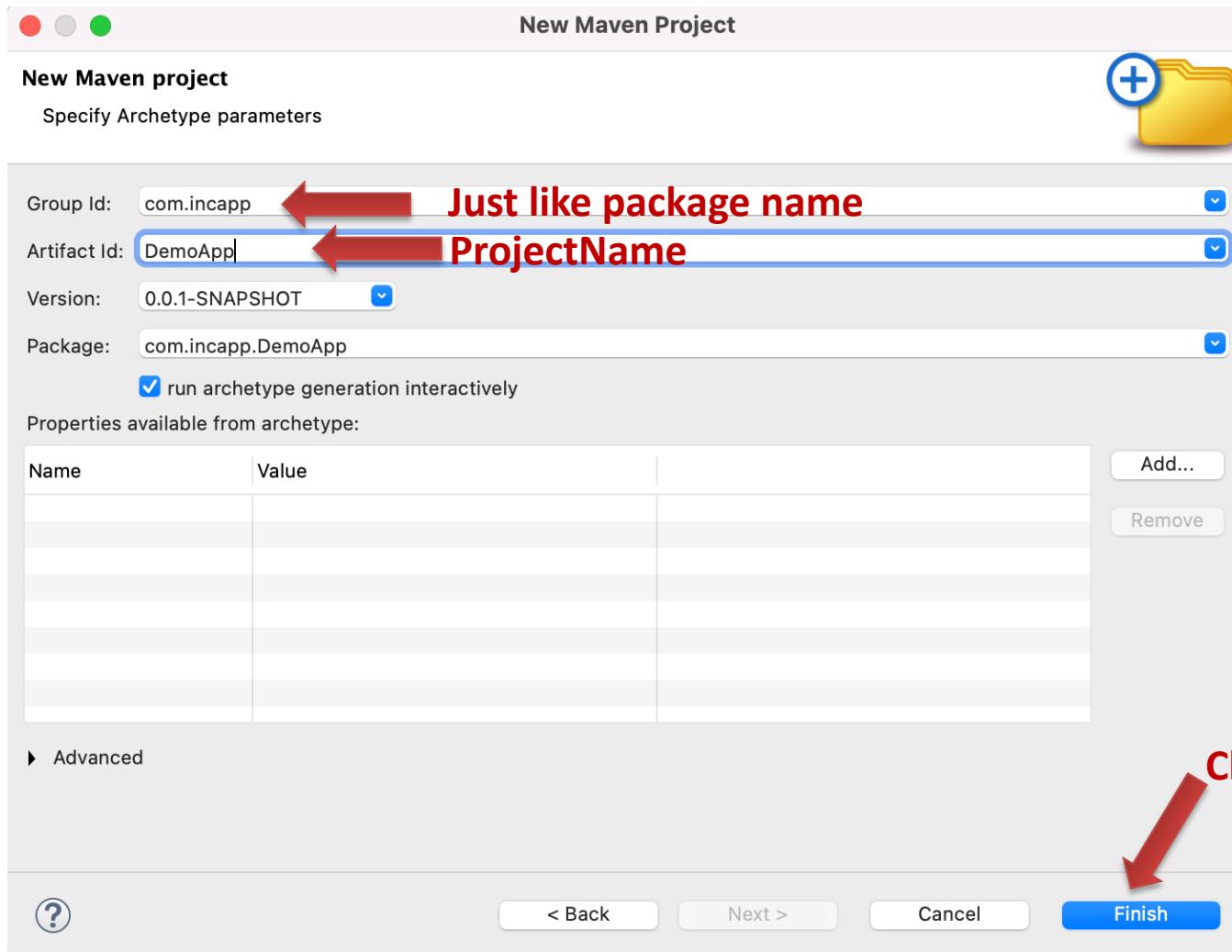
# Step by Step: First Spring Project



# Step by Step: First Spring Project



# Step by Step: First Spring Project



# Step by Step: First Spring Project



```
[INFO] -----[ pom ]-----  
[INFO]  
[INFO] >>> archetype:3.2.1:generate (default-cli) > generate-sources @  
[INFO]  
[INFO] <<< archetype:3.2.1:generate (default-cli) < generate-sources @  
[INFO]  
[INFO]  
[INFO] --- archetype:3.2.1:generate (default-cli) @ standalone-pom ---  
[INFO] Generating project in Interactive mode  
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:apache-archetype-bundle:3.2.1]  
[INFO] Using property: groupId = com.incapp  
[INFO] Using property: artifactId = DemoApp  
[INFO] Using property: version = 0.0.1-SNAPSHOT  
[INFO] Using property: package = com.incapp.DemoApp  
Confirm properties configuration:  
groupId: com.incapp  
artifactId: DemoApp  
version: 0.0.1-SNAPSHOT  
package: com.incapp.DemoApp  
Y: : y
```

Type “y” to build the project using maven.

# Step by Step: First Spring Project



To add Spring framework in the project, go to maven repository.

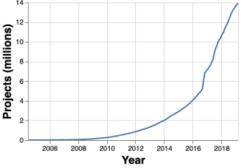
Open <https://www.mvnrepository.com> and type “spring” in search section:

The screenshot shows the MVN Repository search interface. A red arrow points to the search bar containing the text "spring". Another red arrow points to the "Search" button. A third red arrow points to the link "1. Spring Context" under the search results. The results page displays 32103 items. The first result is "Spring Context" from org.springframework, with 14,730 usages and an Apache license. It provides a brief description of Spring Context and its last release date. The second result is "Spring TestContext Framework" from org.springframework, with 10,237 usages and an Apache license. It describes Spring Test's support for unit and integration testing. On the left sidebar, there are sections for "Repository" (Central, Sonatype, Spring Plugins, Spring Lib M, JCenter, Spring Milestones, Mulesoft, Xillio) and "Group" (com.github).

# Step by Step: First Spring Project

**MVN REPOSITORY**

Indexed Artifacts (41.3M)



Search for groups, artifacts, categories

Search

Home » org.springframework » spring-context

**Spring Context**

Spring Context provides access to configured objects like a registry (a context). It inherits its features from Spring Beans and adds support for internationalization, event propagation, resource loading, and the transparent creation of contexts.

License	Apache 2.0
Categories	Dependency Injection
Tags	context   spring   dependency-injection   ioc   framework
Ranking	#29 in MvnRepository (See Top Artifacts) #1 in Dependency Injection
Used By	14,730 artifacts

Central (289) Spring Milestones (69) Spring Plugins (16) Atlassian (2) Atlassian 3rdParty (5)  
WSO2 Releases (6) Alfresco (16) Cambridge (1) Gael (1) Geomajas (1) Gradle Releases (1)  
Grails Core (10) USIT (1) ICM (8)

Version	Vulnerabilities	Repository	Usages	Date
6.1.13		Central	17	Sep 12, 2024
6.1.12		Central	616	Aug 14, 2024
6.1.11		Central	698	Jul 11, 2024
6.1.10		Central	520	Jun 10, 2024

# Step by Step: First Spring Project

**MVN REPOSITORY**

Search for groups, artifacts, categories

Indexed Artifacts (41.3M)

Projects (million) vs Year

Year	Projects (million)
2006	0.5
2008	1.0
2010	2.0
2012	3.0
2014	5.0
2016	8.0
2018	12.0
2020	14.0

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- HTTP Clients
- Logging Bridges
- Dependency Injection
- XML Processing

Home » org.springframework » spring-context » 6.1.13

**Spring Context » 6.1.13**

Spring Context provides access to configured objects like a registry (a context). It inherits its features from Spring Beans and adds support for internationalization, event propagation, resource loading, and the transparent creation of contexts.

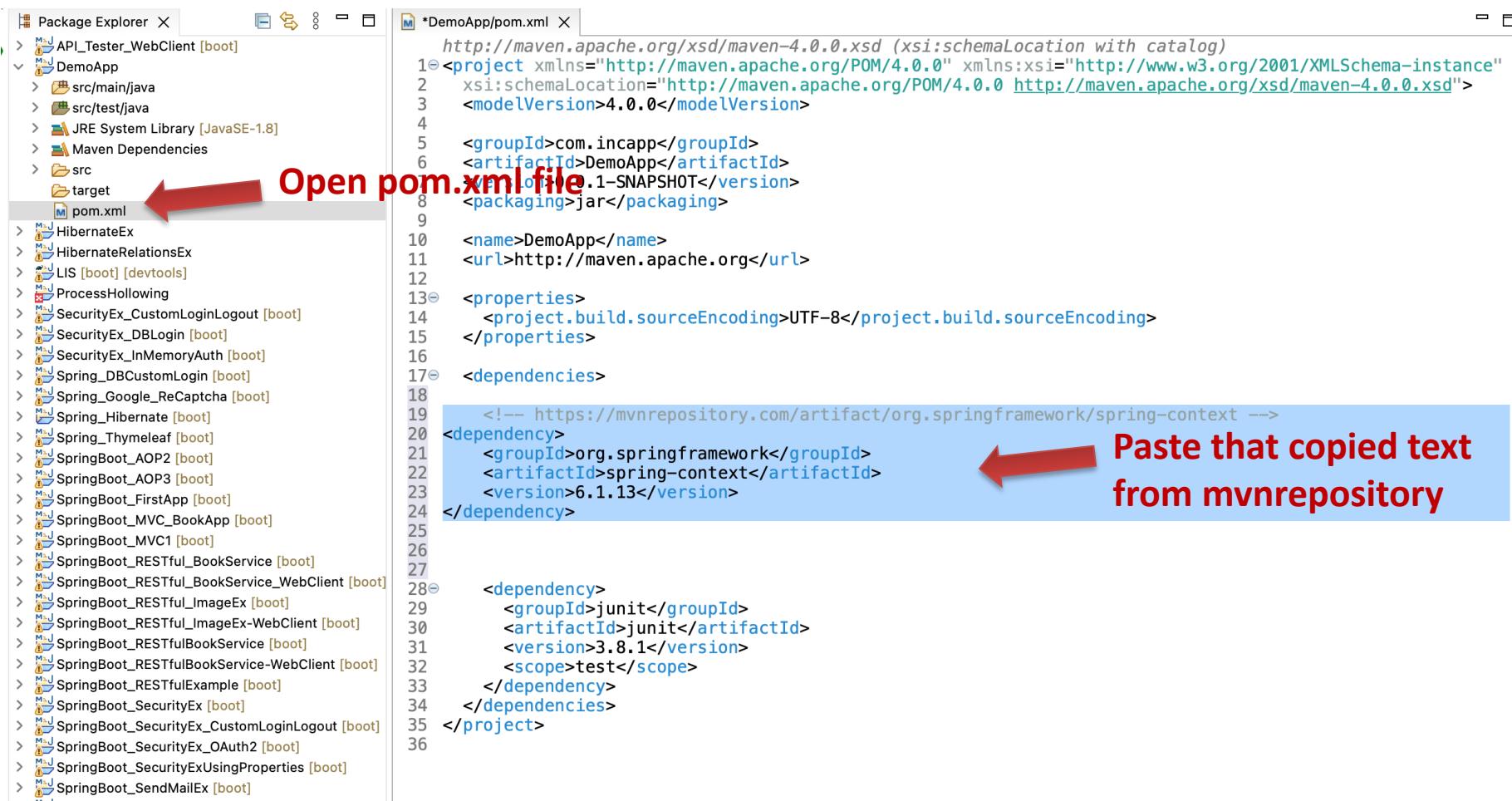
License	Apache 2.0
Categories	Dependency Injection
Tags	context   spring   dependency-injection   ioc   framework
Organization	Spring IO
HomePage	<a href="https://github.com/spring-projects/spring-framework">https://github.com/spring-projects/spring-framework</a>
Date	Sep 12, 2024
Files	pom (2 KB) jar (1.2 MB) <a href="#">View All</a>
Repositories	Central
Ranking	#29 in MvnRepository ( <a href="#">See Top Artifacts</a> ) #1 in Dependency Injection
Used By	14,730 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>6.1.13</version>
</dependency>
```

Copy this selected text

# Step by Step: First Spring Project

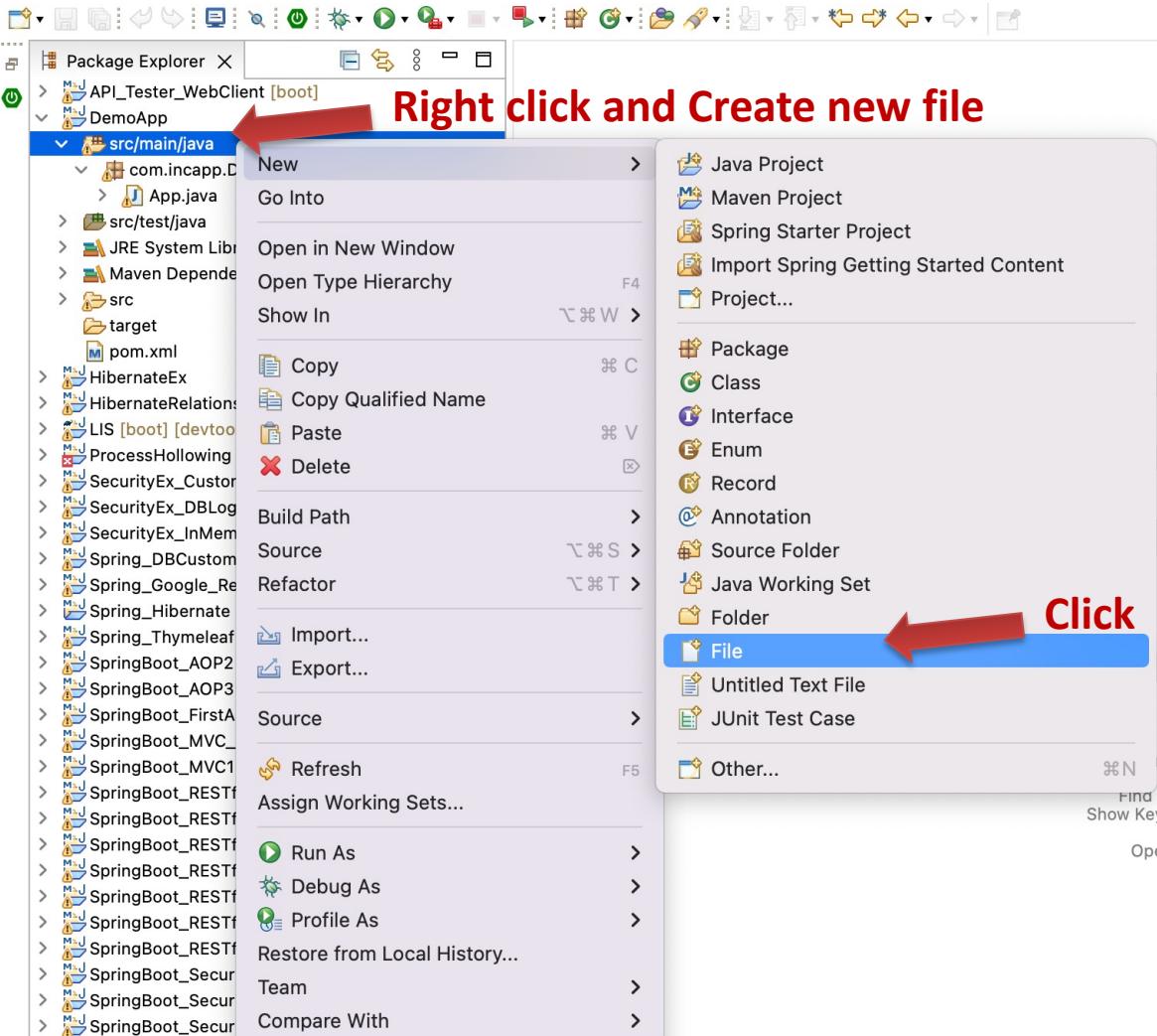


Open pom.xml file

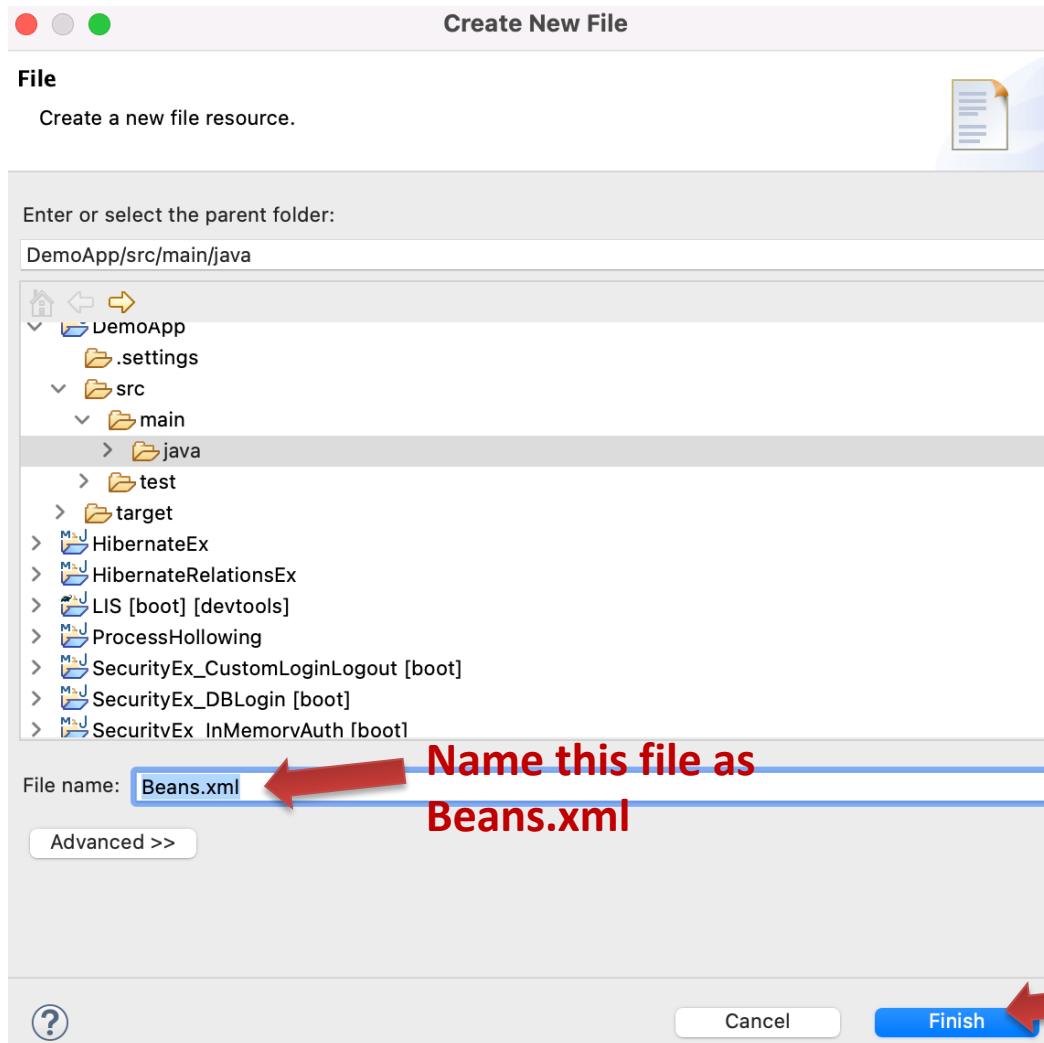
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.incapp</groupId>
  <artifactId>DemoApp</artifactId>
  <version>0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>DemoApp</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>6.1.13</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Paste that copied text from mvnrepository

# 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*”

A screenshot of a Google search results page. The search bar at the top contains the query "spring dtd". A red arrow points to the search bar. Below the search bar, the "All" tab is selected, followed by other categories: Images, Videos, Shopping, News, Maps, Web, More, and Tools. The first search result is from the Spring documentation website, titled "40. XML Schema-based configuration". A red arrow points to this title with the text "Click" next to it.

 Spring  
<https://docs.spring.io/docs/html/xsd-configuration.html> ::

[40. XML Schema-based configuration](#)

Authoring **Spring** configuration files using the older **DTD** style is still fully supported. Nothing will break if you forego the use of the new XML ...

 Spring  
<https://docs.spring.io/docs/html/xsd-config.html> ::

[Appendix E. XML Schema-based configuration](#)

**DTD support?** Authoring **Spring** configuration files using the older **DTD** style is still fully supported. Nothing will break if you forego the use of the new ...

# 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">

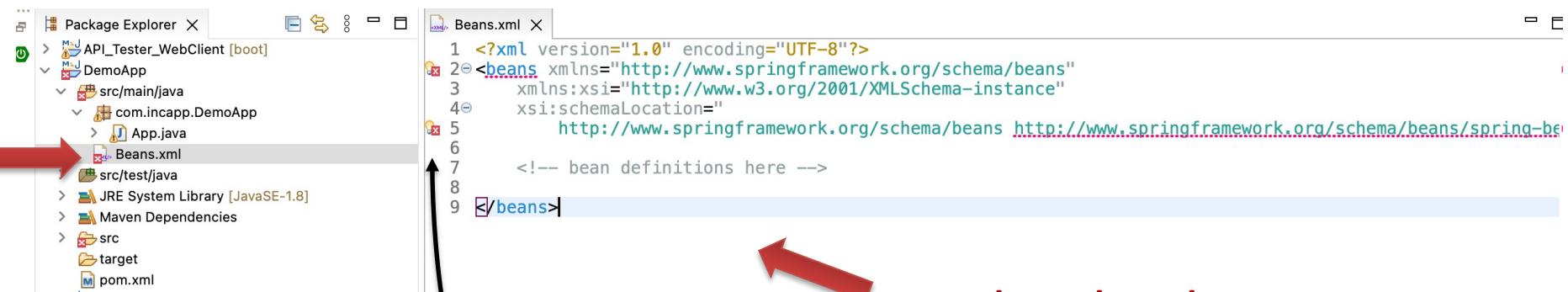
<!-- bean definitions here -->

</beans>
```



**Copy this selected text**

# Step by Step: First Spring Project



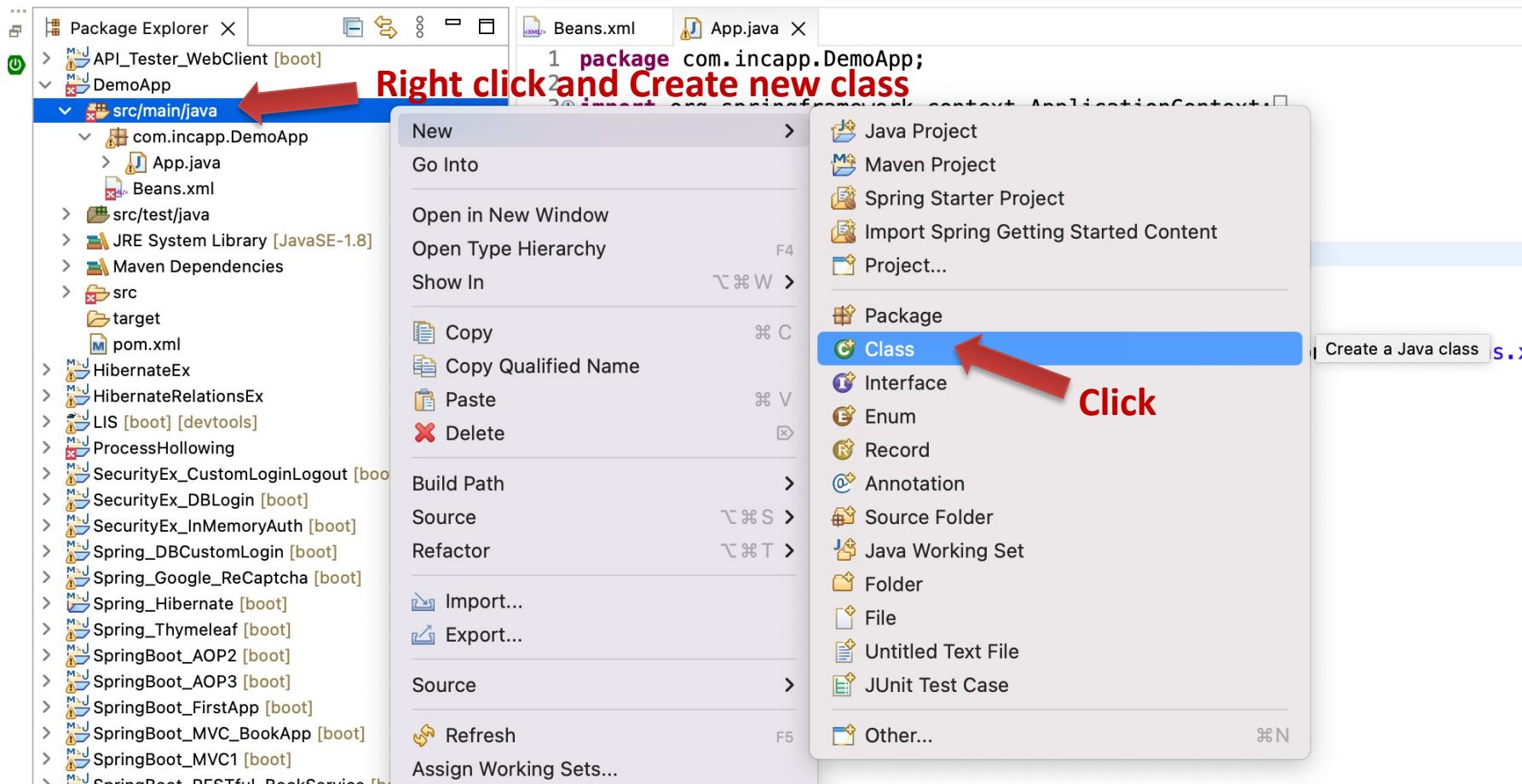
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- bean definitions here -->
</beans>
```

Paste that selected text

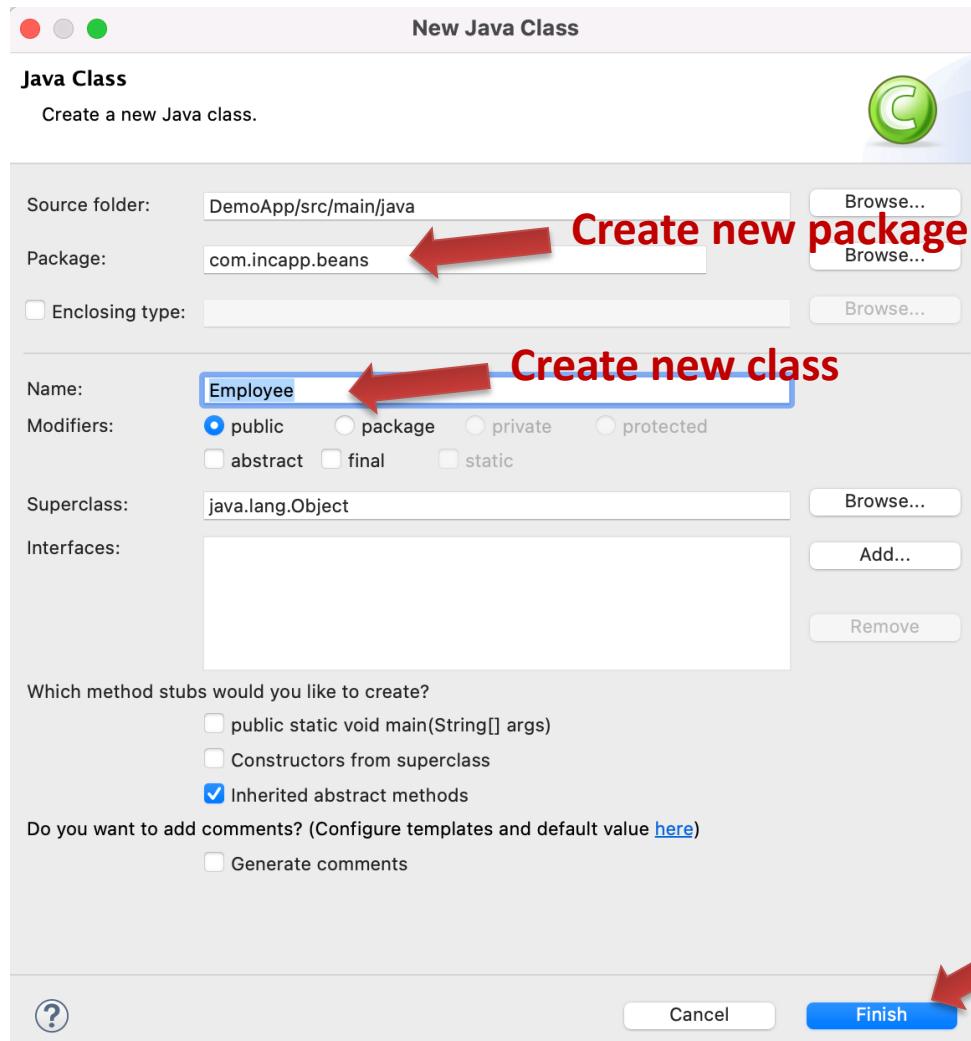
Do not worry about the errors

# Step by Step: First Spring Project

Create new Java Bean named as Employee.



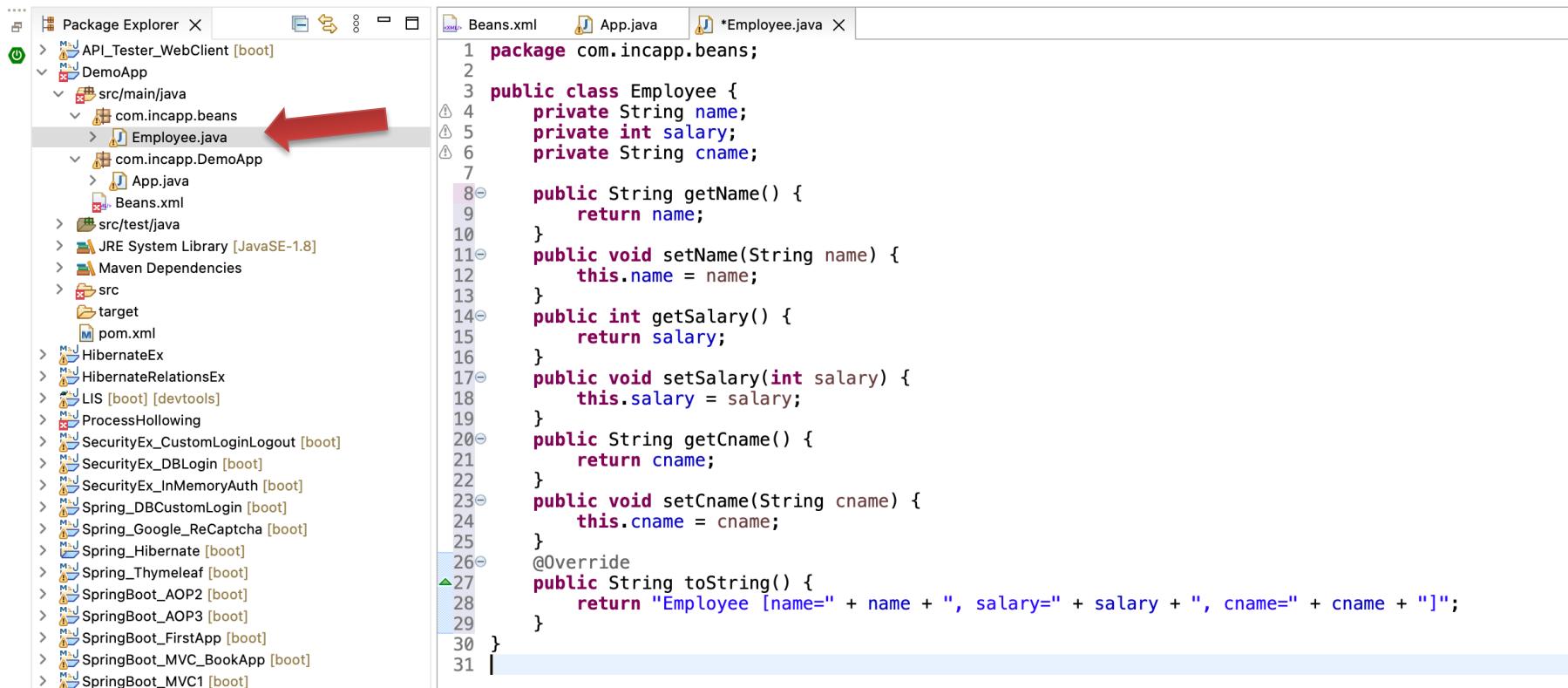
# Step by Step: First Spring Project



# Step by Step: First Spring Project

Open Employee class.

Create some fields and generate getter/setter & `toString()` method.



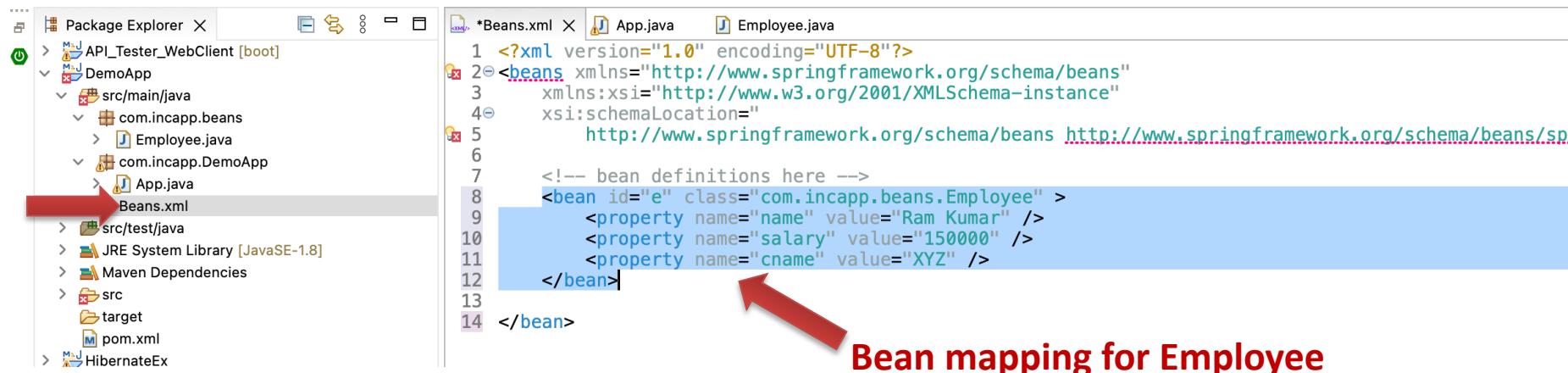
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a project structure under 'API\_Tester\_WebClient [boot]'. A red arrow points to the 'Employee.java' file within the 'src/main/java/com.incapp.beans' package. On the right, the code editor window shows the Java code for the 'Employee' class:

```
1 package com.incapp.beans;
2
3 public class Employee {
4     private String name;
5     private int salary;
6     private String cname;
7
8     public String getName() {
9         return name;
10    }
11    public void setName(String name) {
12        this.name = name;
13    }
14    public int getSalary() {
15        return salary;
16    }
17    public void setSalary(int salary) {
18        this.salary = salary;
19    }
20    public String getCname() {
21        return cname;
22    }
23    public void setCname(String cname) {
24        this.cname = cname;
25    }
26    @Override
27    public String toString() {
28        return "Employee [name=" + name + ", salary=" + salary + ", cname=" + cname + "]";
29    }
30}
31
```

# Step by Step: First Spring Project

Open Beans.xml file.

Declare the Employee bean inside the beans tag.



```
*Beans.xml X App.java Employee.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="
      http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/sp...
5
6
7     <!-- bean definitions here -->
8     <bean id="e" class="com.incapp.beans.Employee" >
9         <property name="name" value="Ram Kumar" />
10        <property name="salary" value="150000" />
11        <property name="cname" value="XYZ" />
12    </bean>
13
14 </bean>
```

Bean mapping for Employee

# Step by Step: First Spring Project



The screenshot shows a Java development environment with the following structure:

- Package Explorer:** Shows a project named "DemoApp" containing "src/main/java/com.incapp.DemoApp" which includes "App.java" and "Beans.xml". Other projects like "API\_Tester\_WebClient" and "HibernateEx" are also listed.
- Code Editor:** Displays the content of "App.java".

```
1 package com.incapp.DemoApp;
2
3 import org.springframework.context.ApplicationContext;
4
5 /**
6  * Hello world!
7  *
8  */
9
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         ApplicationContext ctx=new ClassPathXmlApplicationContext("Beans.xml");
15     }
16 }
17 }
```

A red arrow points from the text "Write this statement to get Spring IoC container" to the line "ApplicationContext ctx=new ClassPathXmlApplicationContext("Beans.xml");".

**Write this statement to get Spring IoC container**

# Step by Step: First Spring Project

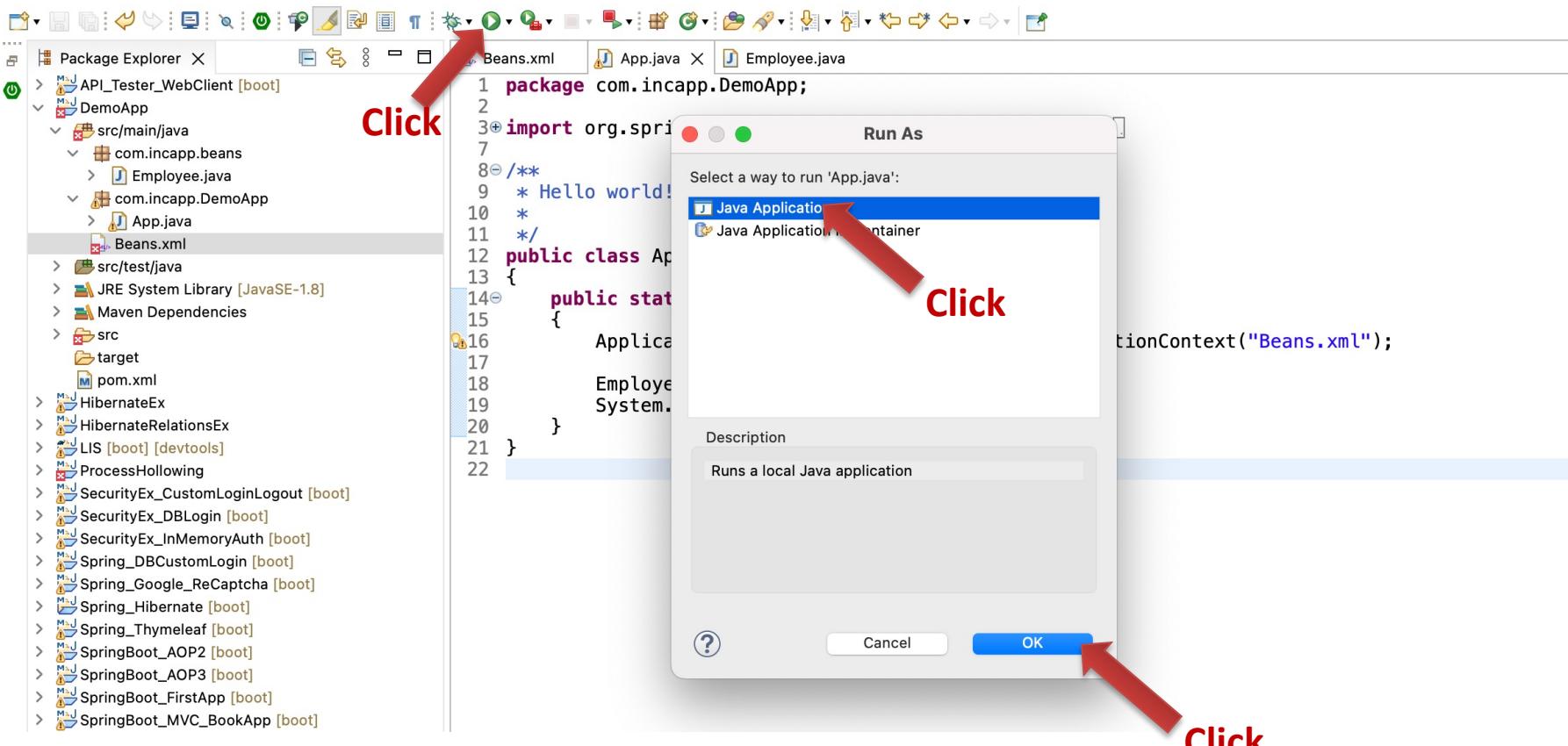


```
1 package com.incapp.DemoApp;
2
3 import org.springframework.context.ApplicationContext;
4
5 /**
6 * Hello world!
7 */
8
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         ApplicationContext ctx=new ClassPathXmlApplicationContext("Beans.xml");
14         Employee a=ctx.getBean("e",Employee.class);
15         System.out.println(a);
16     }
17 }
18
19
20 }
```

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

The screenshot shows a Java application running in an IDE. On the left, the code editor displays `App.java` with the following content:

```
1 package com.incapp.DemoApp;
2
3 import org.springframework.context.ApplicationContext
4
5 /**
6  * Hello world!
7  *
8  */
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         ApplicationContext ctx=new ClassPathXmlApplication
14         Employee a=ctx.getBean("e",Employee.class);
15         System.out.println(a);
16     }
17 }
```

On the right, the IDE's toolbars and windows are visible. A red arrow points from the word "Output" to the `Console` tab, which displays the output of the application's execution:

```
<terminated> App (7) [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk/Contents/Home/bin/java -Dfile.encoding=UTF-8 com.incapp.DemoApp
Employee [name=Ram Kumar, salary=150000, cname=XYZ]
```

# 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.

The CODING Institute

# 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" />



app<sup>®</sup>

The word "app" is written in a large, light gray sans-serif font. A registered trademark symbol (®) is located above the letter "p". Below the "app" text, there is a faint, horizontal, light-colored bar.

# 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.