

Lesson 01 Demo 05

Inheritance Relationship in Spring Core

Objective: To demonstrate inheritance in Spring Core through the creation of bean classes and utilizing Inversion of Control (IOC)

Tool required: Eclipse IDE

Prerequisites: None

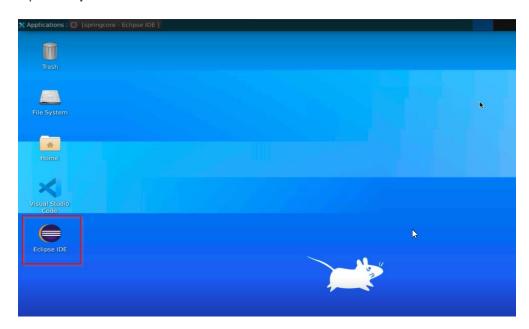
Steps to be followed:

- 1. Creating a Maven project
- 2. Copying files and dependencies
- 3. Creating the FoodItem bean
- 4. Creating the Pizza bean
- 5. Configuring the **context.xml** for beans
- 6. Writing IOC code in App.java

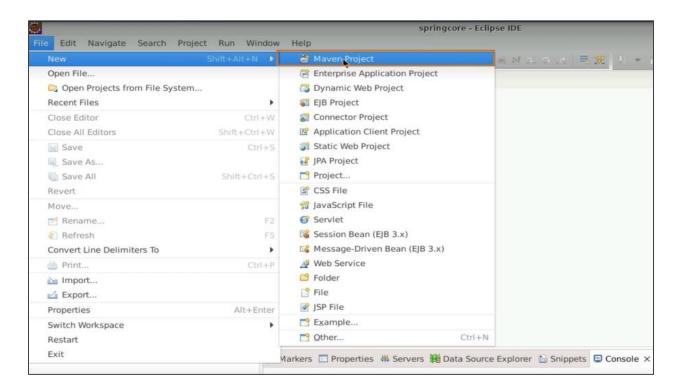


Step 1: Creating a Maven project

1.1 Open Eclipse IDE

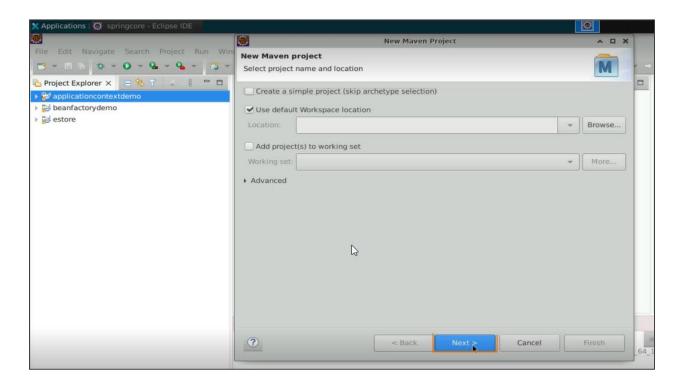


1.2 Click File > New > Maven Project to create a new Maven project

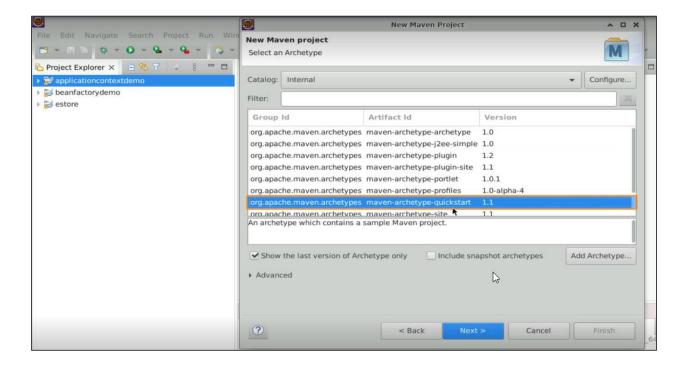




1.3 Select the default workspace location and click Finish

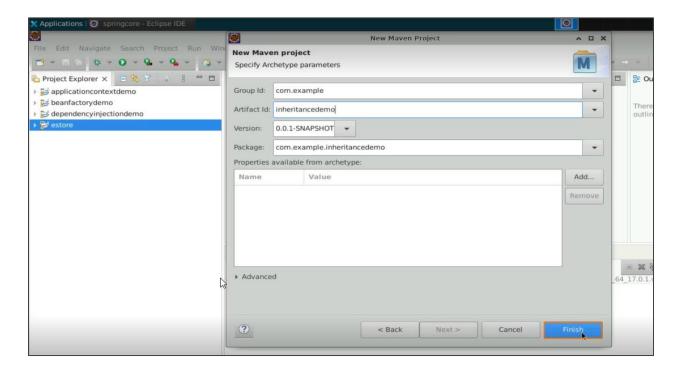


1.4 Select the maven-archetype-quickstart from the Internal catalogs and click Next





1.5 Provide the Group Id, typically the company's domain name in reverse order, and the Artifact Id as **inheritancedemo** and click **Finish**



Step 2: Copying files and dependencies

2.1 Copy the **spring-context** dependency from the **pom.xml** of the **estore** project and paste it into the current project's **pom.xml**



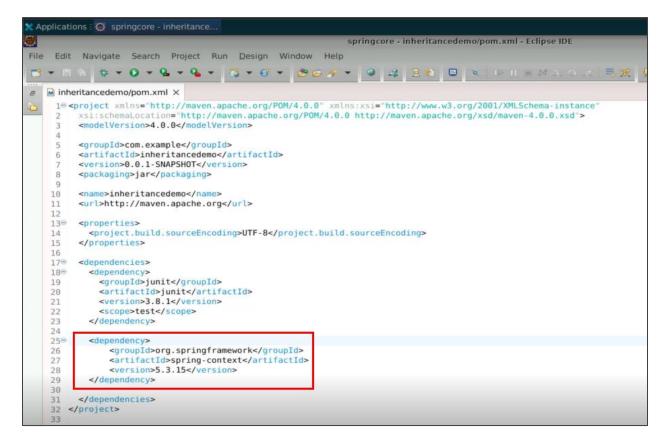
```
springcore - estore/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Design Window Help
Project Explorer × □ 🥫 🍸 👵 🖇 📟 🗖 📓 estore/pom.xml ×
                                                                                                                                        - -
▶ अ applicationcontextdemo
                                                    <denendencies>
▶ 👺 beanfactorydemo
                                               185
                                                        <groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
▶ B dependencyinjectiondemo
                                               19

→ I estore

                                               21
                                               22 23
 ▶ @ src/main/java
                                                         <scope>test</scope>
                                                       </dependency>
 ▶ # src/test/java
 ▶ M JRE System Library [J2SE-1.5]
                                               25
                                                           <groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.3.15</version>
 Maven Dependencies
                                               26
27
                                               28
  ▶ ⊜ target
   m pom.xr
                                               30
                                               31 </depend
32 </project>
                                                     </dependencies>

→ I inheritancedemo

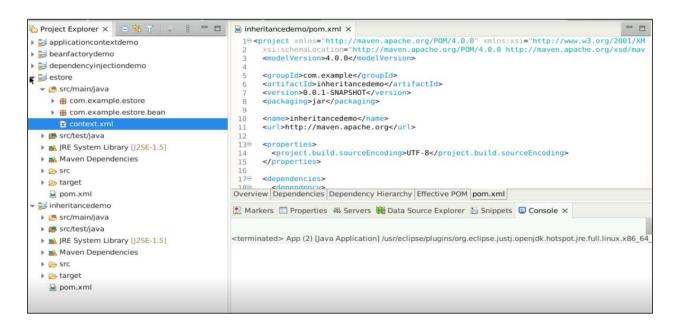
  ▶ @ src/main/java
 src/test/java
                                              Overview Dependencies Dependency Hierarchy Effective POM pom.xml
 ▶ M JRE System Library [J25E-1.5]
                                              Markers Properties & Servers Data Source Explorer Snippets Console X
  Maven Dependencies
  ► B STC
                                              <terminated> App (2) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64
  > = target
    pom.xml
```

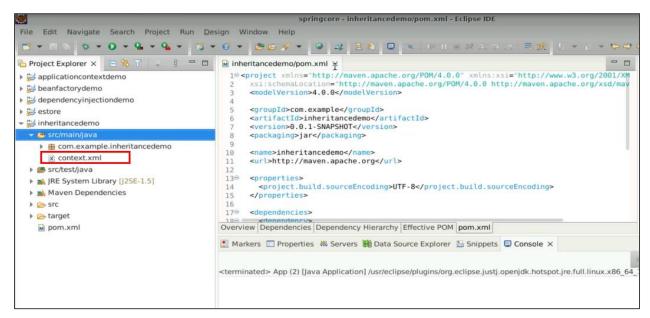


Note: Please refer to the previous demos on how to create the **estore** project



2.2 Copy the context.xml file from the estore project and paste it into the src/main/java package of the current project

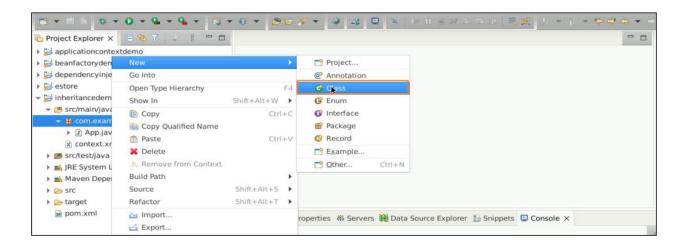




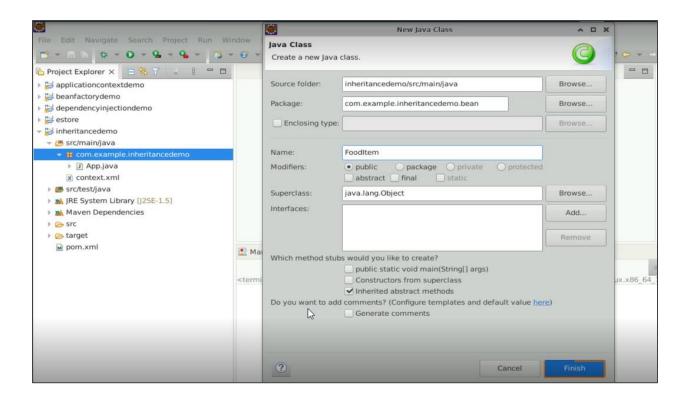


Step 3: Creating the FoodItem bean

3.1 Right-click on the package and select **New > Class** to create a new class



3.2 Provide a name for the class, such as **FoodItem**. Add **.bean** to the package name and click **Finish**





3.3 In the FoodItem class, define attributes such as name and price

3.4 Create a default constructor for the class and add a print statement: [FoodItem] - Object Created

```
Applications = Springcore - Inheritance...

springcore - inheritancedemo/src/main/java/com/example/inheritancedemo/bean/FoodItem.java -
File Edit Source Refactor Navigate Search Project Run Window Help

FoodItem.java ×

package com.example.inheritancedemo.bean;

spublic class FoodItem {

String name;
int price;

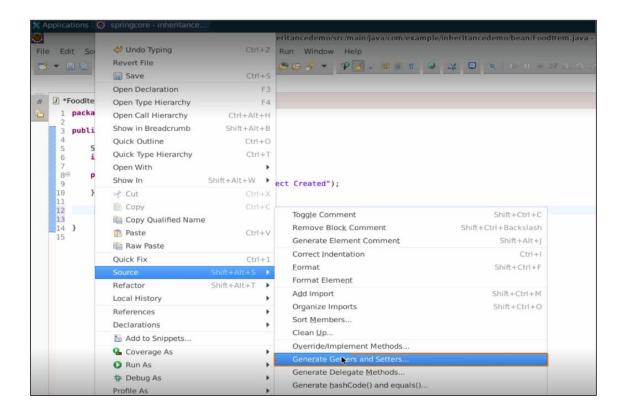
public FoodItem() {

System.out.println("[FoodItem] - Object Created");
}

13
```

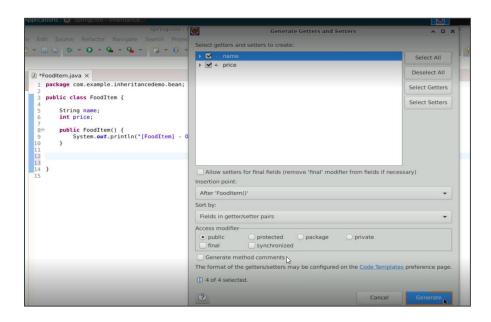


3.5 Right-click inside the FoodItem class, select Source, and Generate Getters and Setters





3.6 Select all the attributes and click Generate



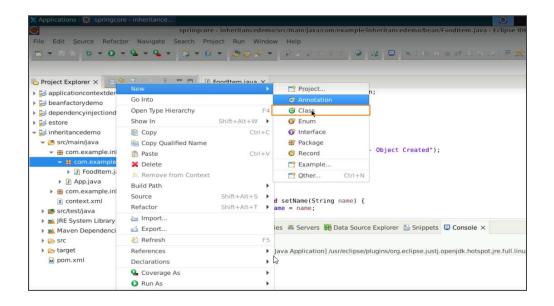
3.7 Repeat the same step to generate a toString() method that returns all the attributes

```
springcore - inheritancedemo/src/main/java/com/example/inheritancedemo/bean/FoodItem.java
File Edit Source Refactor Navigate Search Project Run Window Help
package com.example.inheritancedemo.bean;
     3 public class FoodItem {
            String name;
           int price;
        public FoodItem() {
               System.out.println("[FoodItem] - Object Created");
    10
    11
         public String getName() {
    13
               return name;
    14
          public void setName(String name) {
                this.name = name;
    18
    20⊖
          public int getPrice() {
    21
               return price;
    22
            public void setPrice(int price) {
   24<sup>©</sup>
25
               this.price = price;
    26
    27
   28<sup>©</sup>
29
30
            @Override
           public String toString() {|
    return "FoodItem [name=" + name + ", price=" + price + "]";
```

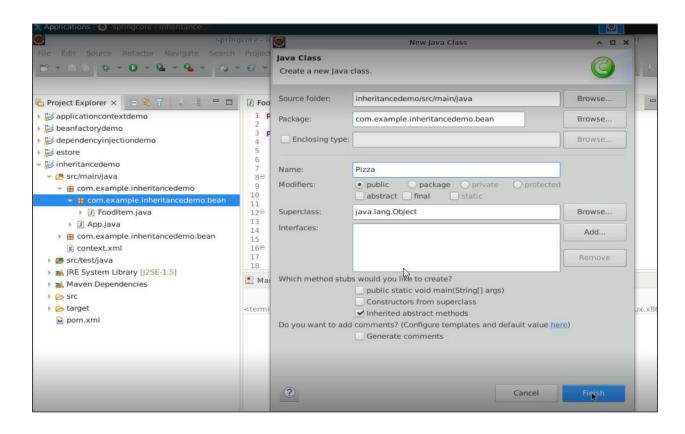


Step 4: Creating the Pizza bean

4.1 Create a new class. Right-click on the bean package, select New > Class



4.2 Name the class as Pizza and click Finish





4.3 Extend the **FoodItem** class to inherit all its properties and methods using the **extends** keyword

4.4 Define a few more attributes such as size and toppings

```
Applications : Springcore - inheritance...

Springcore - inheritancedemo/src/main/java/com/example/inheritancedemo/bean/Pizza
File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Refactor Navigate Search Project Run Window Help

File Edit Source Run Window Help

F
```



4.5 Create a default constructor for the class and add a print statement: [Pizza] - Object Created

4.6 Generate getters and setters for the attributes and a toString() method for the class

```
☑ Pizza.java ×

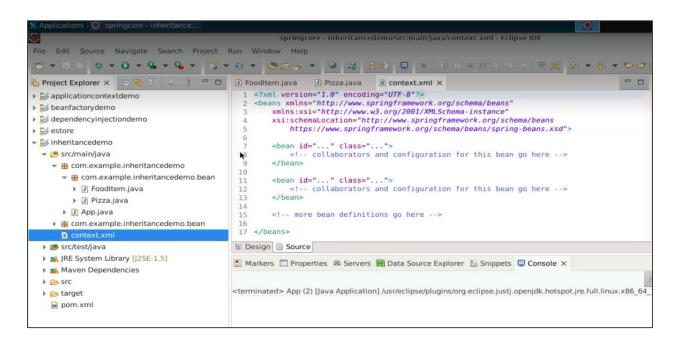
☑ FoodItem.java

    package com.example.inheritancedemo.bean;
  3 public class Pizza extends FoodItem{
         int size;
         String toppings;
         public Pizza() {
              System.out.println("[Pizza] - Object Created");
 10
 11
        public int getSize() {
 13
              return size;
15
16⊖
         public void setSize(int size) {
              this.size = size;
         }
18
19
         public String getToppings() {
20<sup>©</sup>
21
22
23
24<sup>©</sup>
25
26
27
              return toppings;
         public void setToppings(String toppings) {
              this.toppings = toppings;
         @Override
public String toString() {
    return "Pizza [size=" + size + ", toppings=" + toppings + ", name=" + name + ", price=" + price + "]";
```



Step 5: Configuring the context.xml for beans

5.1 Open the context.xml file



5.2 Define a bean for the **FoodItem** class with an id **foodItem** and set the key-value pairs for its attributes

```
Pizza.java
                              x context.xml ×
1 <?xml version="1.0" encoding="UTF-8"?>
 20 <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
           https://www.springframework.org/schema/beans/spring-beans.xsd">
6
       <bean id="foodItem" class="com.example.inheritancedemo.bean.FoodItem">
           property name="name" value="Veggie Pizza"/>
q
            property name="price" value="300"/>
       </bean>
10
11
       <br/><bean id="I.." class="...">
           <!-- collaborators and configuration for this bean go here -->
       </bean>
15
       <!-- more bean definitions go here -->
16
18 </beans>
```



5.3 Define another bean for the Pizza class with an id **pizza** and set the key-value pairs for its attributes. Use the **parent** attribute to link it to the **FoodItem** bean

```
Source Navigate Search Project Run Window Help
                    FoodItem.java Pizza.java 🗷 context.xml 🗙
   1 <?xml version="1.0" encoding="UTF-8"?>
20 <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
            https://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="foodItem" class="com.example.inheritancedemo.bean.FoodItem">
            10
        </bean>
  11
         <bean id="pizza" class="com.example.inheritancedemo.bean.Pizza" parent="foodItem">
  129
             13
 15
             operty name="toppings" value="cheese, bell peppers, corns and brocolli"/>
  16
17
         <!-- more bean definitions go here -->
  18
  20 </beans>
```

Step 6: Writing IOC code in App.java

6.1 Navigate to the **App.java** class and update the print statement to **Welcome to Spring**Core Inheritance Configuration

```
Project Explorer X 🖃 🕏 🥫 📮 📮 🗓 FoodItem.java 🗓 Pizza.java 🕱 context.xml 🗓 App.java X
                                          package com.example.inheritancedemo;
► ₩ applicationcontextdemo
▶ 👺 beanfactorydemo
                                            * Hello world!
▶ 🔀 dependencyinjectiondemo
▶ ﷺ estore
→ ﷺ inheritancedemo
                                           public class App
{
 public static void main( String[] args )

→ ⊕ com.example.inheritancedemo

     ▶ D Appliava
                                                  System.out.println( "Hello World!" );

→ ⊕ com.example.inheritancedemo.bean

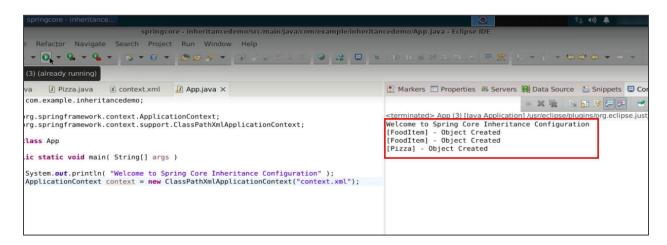
                                               }
                                        13 }
     x context.xml
 src/test/java
 ▶ ■ IRE System Library [J2SE-1.5]
 ▶ ■ Maven Dependencies
 ▶ ⊜ src
                                       Markers ☐ Properties ₩ Servers 🏙 Data Source Explorer 🔁 Snippets 📮 Console X
 ▶ ⊜ target
   pom.xml
                                       <terminated> App (2) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64
```



6.2 Create an instance of the ApplicationContext interface using the ClassPathXmlApplicationContext and pass the context.xml file to the ApplicationContext constructor



6.3 Run the project by clicking on the green play button



You may notice in the console logs that the FoodItem bean object is instantiated twice due to the Pizza class inheriting from the FoodItem bean.



6.4 For better representation, print the hash codes in the default constructors of the **FoodItem** and **Pizza** classes

```
Edit Source Refactor Navigate Search Project Run Window Help
₽ I *FoodItem.java × I Pizza.java
                                  x context.xml
                                                    App.java
    package com.example.inheritancedemo.bean;
    3 public class FoodItem {
           String name;
          int price;
          public FoodItem() {
               System.out.println("[FoodItem] - Object Created "+hashCode());
   10
    11
         public String getName() {
    120
               return name;
    15
   16⊖ public void setName(String name) {
               this.name = name;
    17
    19
         public int getPrice() {
   20⊖
                                                                            2
    21
               return price;
   22
    23
          public void setPrice(int price) {
   240
   25
               this.price = price;
   26
    27
    289
          public String toString() {
    return "FoodItem [name=" + name + ", price=" + price + "]";
   29
   30
    31
```

6.5 Rerun the project

In the console logs, you can observe that the FoodItem bean object has been instantiated with a distinct hash code, while both the parent and child relationship beans, such as Pizza inheriting from FoodItem, share the same hash code.



6.6 Use the **getBean()** method to retrieve the **Pizza** bean instance by its reference ID and print the pizza

6.7 Run the project by clicking on the green play button

```
Markers Properties & Servers Data Source Snippets Console X

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp.
Welcome to Spring Core Inheritance Configuration
[FoodItem] - Object Created 1390869998
[FoodItem] - Object Created 40075281
[Pizza] - Object Created 40075281
Pizza [size=10, toppings=cheese, bell peppers, corns and brocolli, name=Ve

I
```

You can now observe that the Pizza details are listed on the console, and the name and size values are coming from the parent class, **FoodItem**.