Lesson 01 Demo 03

IOC with Application Context

Objective: To demonstrate the usage of IOC (Inversion of Control) with the Application Context in a Spring framework project

Tool required: Eclipse IDE

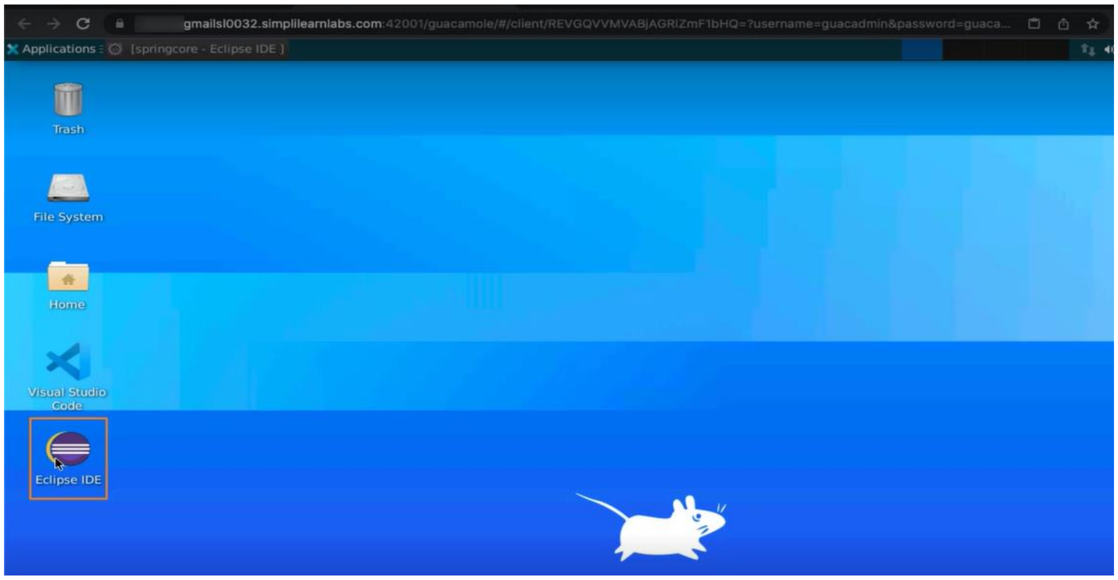
Prerequisites: None

Steps to be followed:

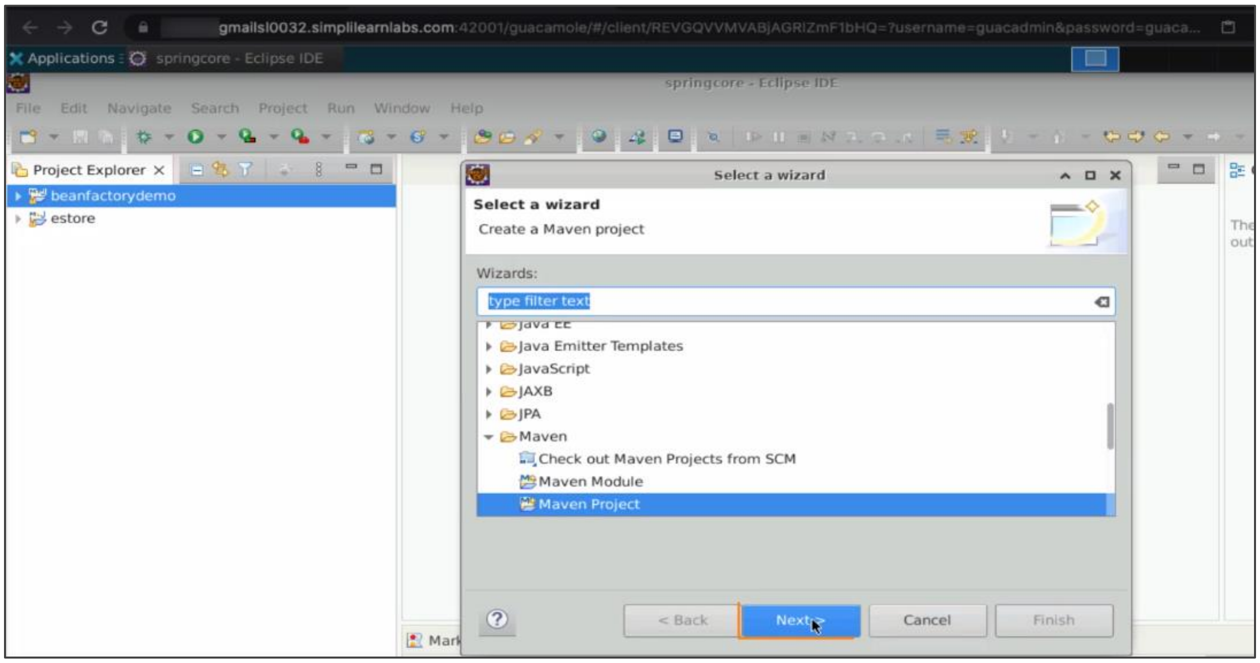
1. Setting up the Maven project
2. Copying files and dependencies
3. Configuring ApplicationContext and retrieving beans
4. Modifying the bean scope
5. Implementing methods for the bean lifecycle

Step 1: Setting up the Maven project

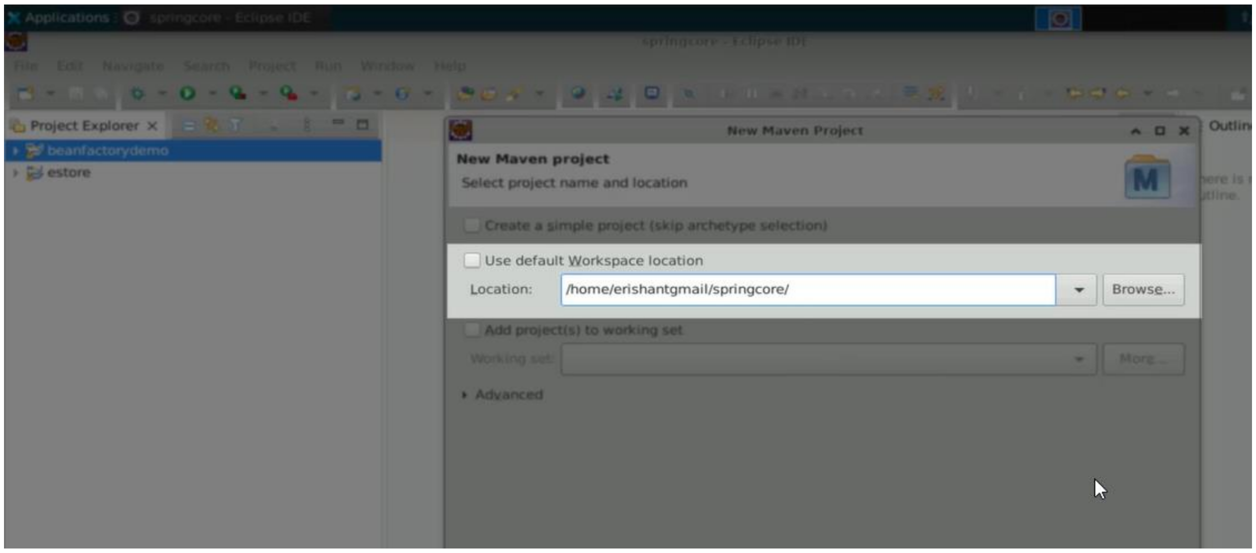
1.1 Open **Eclipse IDE**

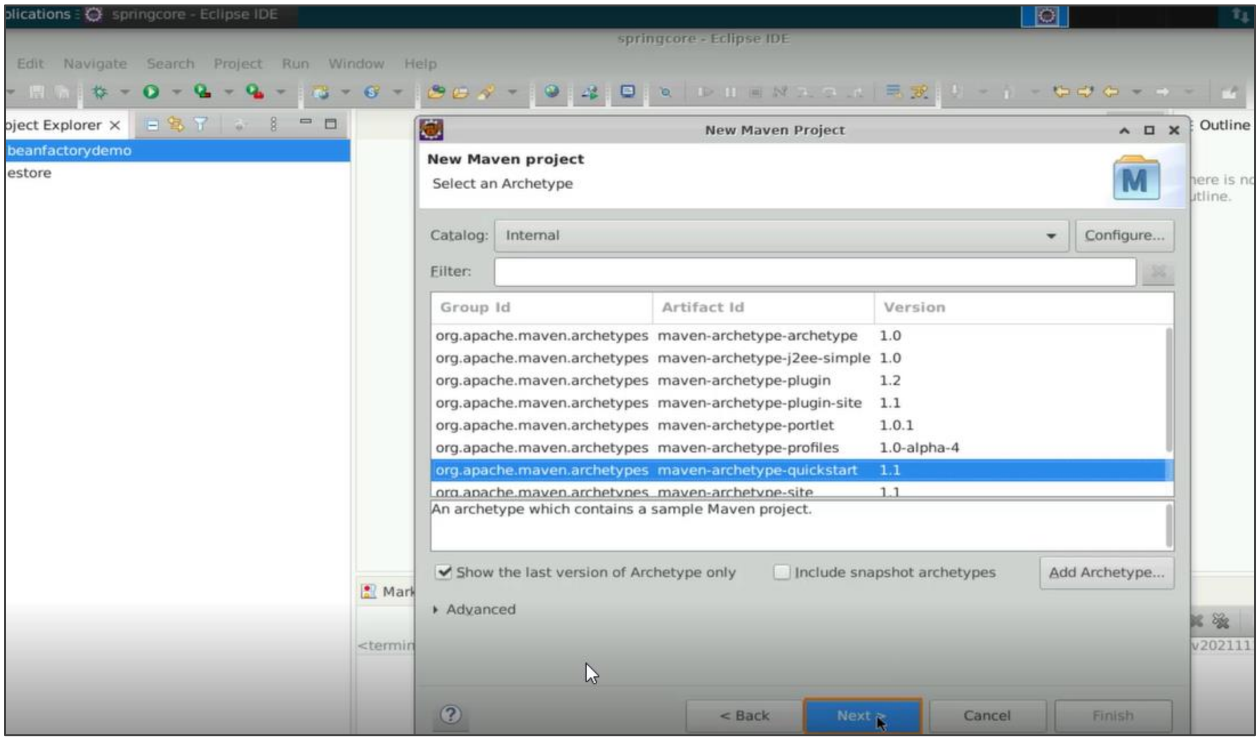


1.2 Create a new **Maven** project



1.3 Choose the desired location and select the **maven-archetype-quickstart** archetype


New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
/home/erishantgmail/springcore/
Browse...
Add project(s) to working set
Working set:
More...
Advanced



New Maven project

Select an Archetype

Catalog: Internal

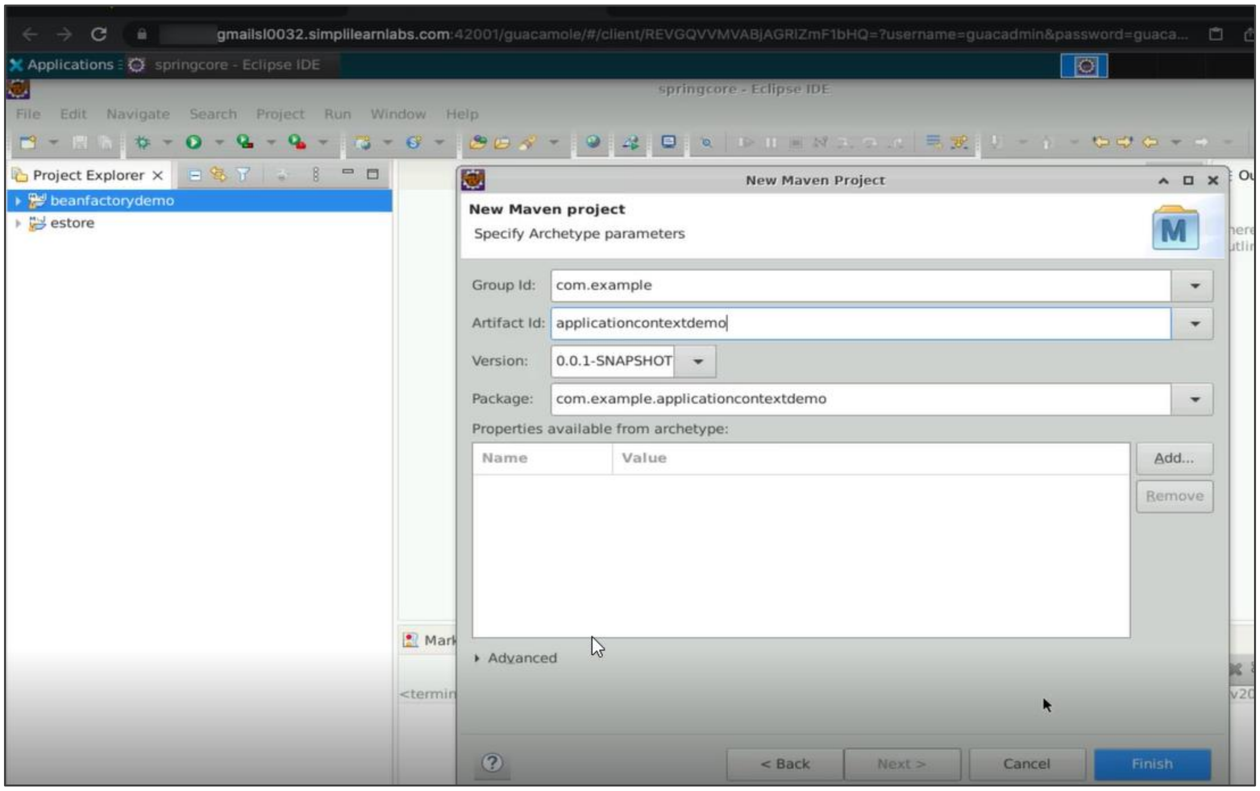
Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-archetype	1.0
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1

An archetype which contains a sample Maven project.

[✓] Show the last version of Archetype only   [ ] Include snapshot archetypes

1.4 Specify the artifact ID as **applicationcontextdemo** and click **Finish**



New Maven project

Specify Archetype parameters

Group Id: com.example

Artifact Id: applicationcontextdemo

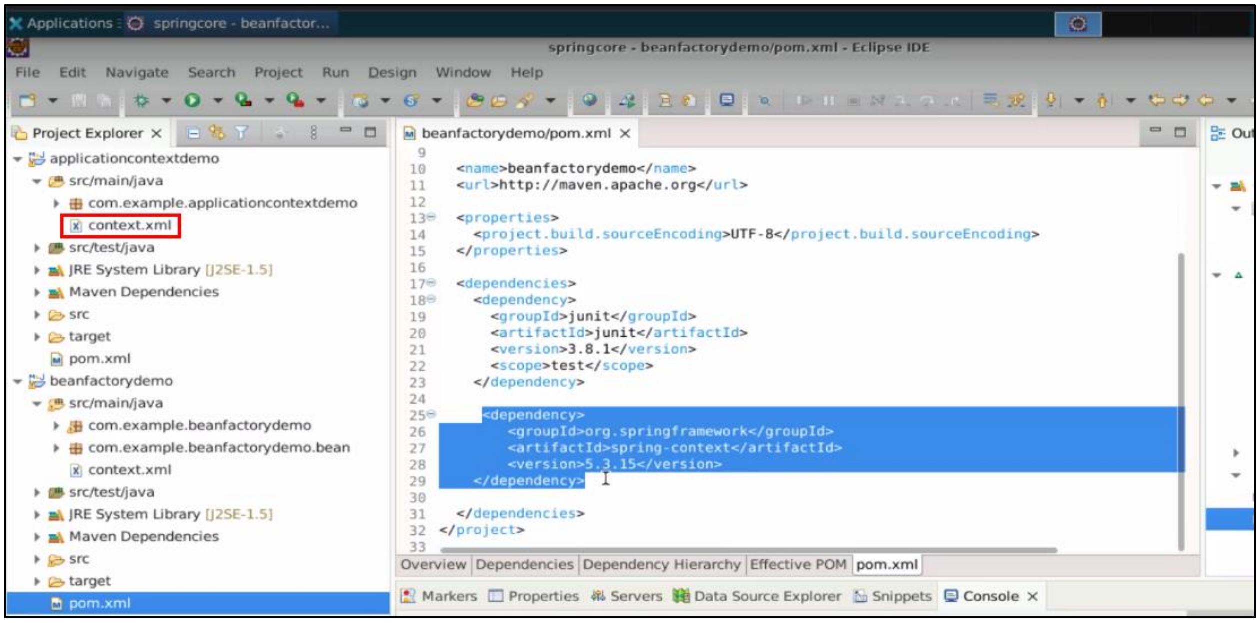
Version: 0.0.1-SNAPSHOT

Package: com.example.applicationcontextdemo

Properties available from archetype:

Step 2: Copying files and dependencies

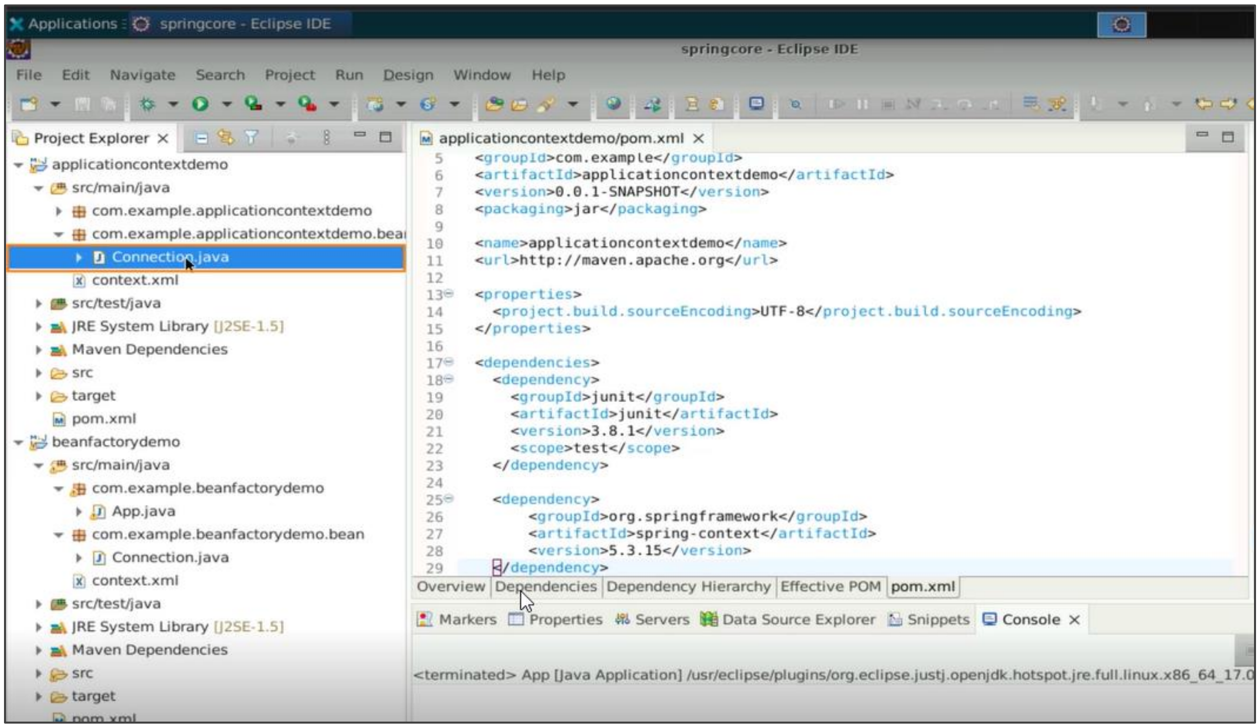
2.1 Copy the **Connection.java** and **context.xml** files from previous projects


Applications
springcore - beanfactor...
springcore - beanfactorydemo/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Design Window Help
Project Explorer
applicationcontextdemo
src/main/java
com.example.applicationcontextdemo
context.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
beanfactorydemo
src/main/java
com.example.beanfactorydemo
com.example.beanfactorydemo.bean
context.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
beanfactorydemo/pom.xml
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Markers Properties Servers Data Source Explorer Snippets Console

```
  <name>beanfactorydemo</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
       <groupId>org.springframework</groupId>
       <artifactId>spring-context</artifactId>
       <version>5.3.15</version>
    </dependency>

  </dependencies>
</project>
```

Note: Please refer to the previous demos on how to create the Maven project with the Spring framework

2.2 Copy the **Spring-context** dependency into the Maven project

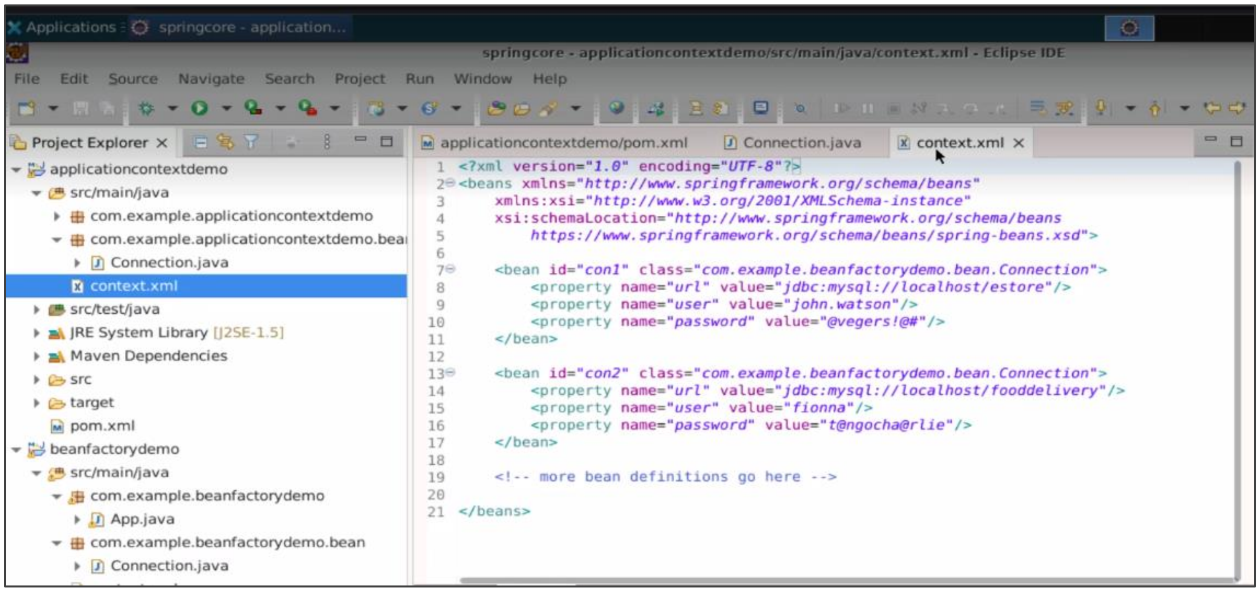

Applications
springcore - Eclipse IDE
File Edit Navigate Search Project Run Design Window Help
Project Explorer
applicationcontextdemo
src/main/java
com.example.applicationcontextdemo
com.example.applicationcontextdemo.bea
Connection.java
context.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
beanfactorydemo
src/main/java
com.example.beanfactorydemo
App.java
com.example.beanfactorydemo.bean
Connection.java
context.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
applicationcontextdemo/pom.xml
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Markers Properties Servers Data Source Explorer Snippets Console

```
  <groupId>com.example</groupId>
  <artifactId>applicationcontextdemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>applicationcontextdemo</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.15</version>
    </dependency>
```

```
<terminated> App [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0
```

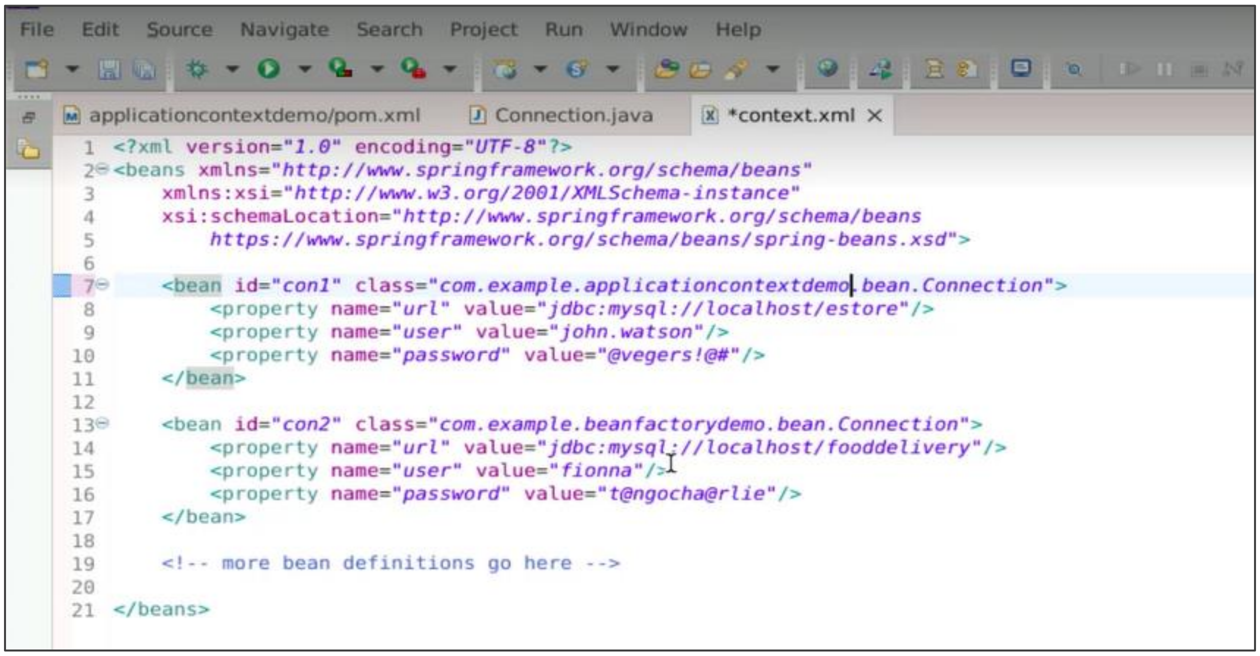
Step 3: Configuring ApplicationContext and retrieving beans

3.1 Open the **context.xml** file


Applications
springcore - application...
springcore - applicationcontextdemo/src/main/java/context.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
Project Explorer
applicationcontextdemo
src/main/java
com.example.applicationcontextdemo
com.example.applicationcontextdemo.bea
Connection.java
context.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
beanfactorydemo
src/main/java
com.example.beanfactorydemo
App.java
com.example.beanfactorydemo.bean
Connection.java
applicationcontextdemo/pom.xml
Connection.java
context.xml

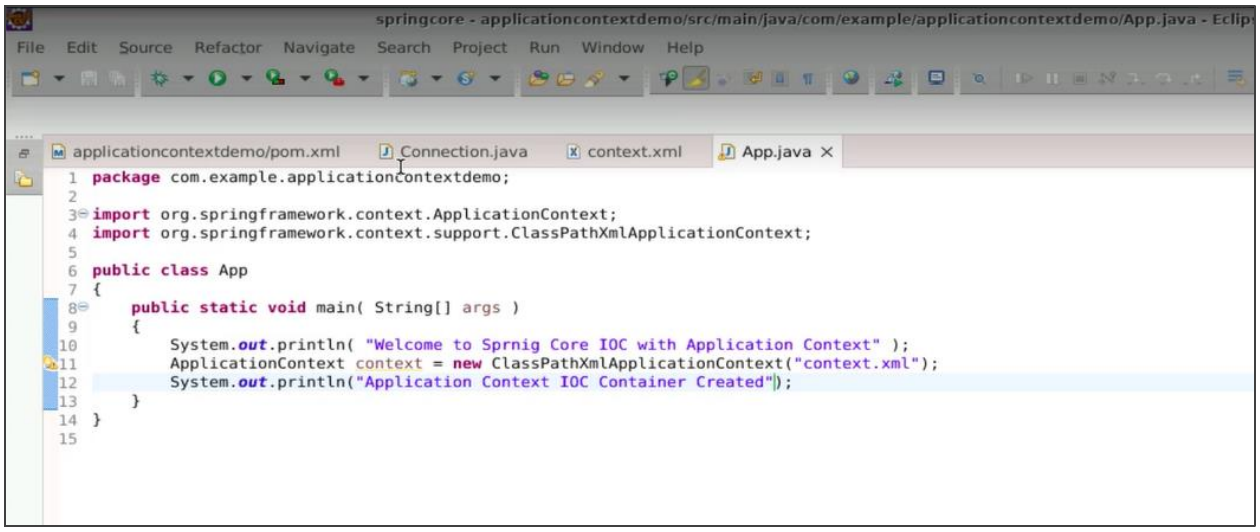
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="con1" class="com.example.beanfactorydemo.bean.Connection">
        <property name="url" value="jdbc:mysql://localhost/estore"/>
        <property name="user" value="john.watson"/>
        <property name="password" value="@vegers!@#"/>
    </bean>

    <bean id="con2" class="com.example.beanfactorydemo.bean.Connection">
        <property name="url" value="jdbc:mysql://localhost/fooddelivery"/>
        <property name="user" value="fionna"/>
        <property name="password" value="t@ngocha@rlie"/>
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

3.2 Update the class name and package to match the project's package



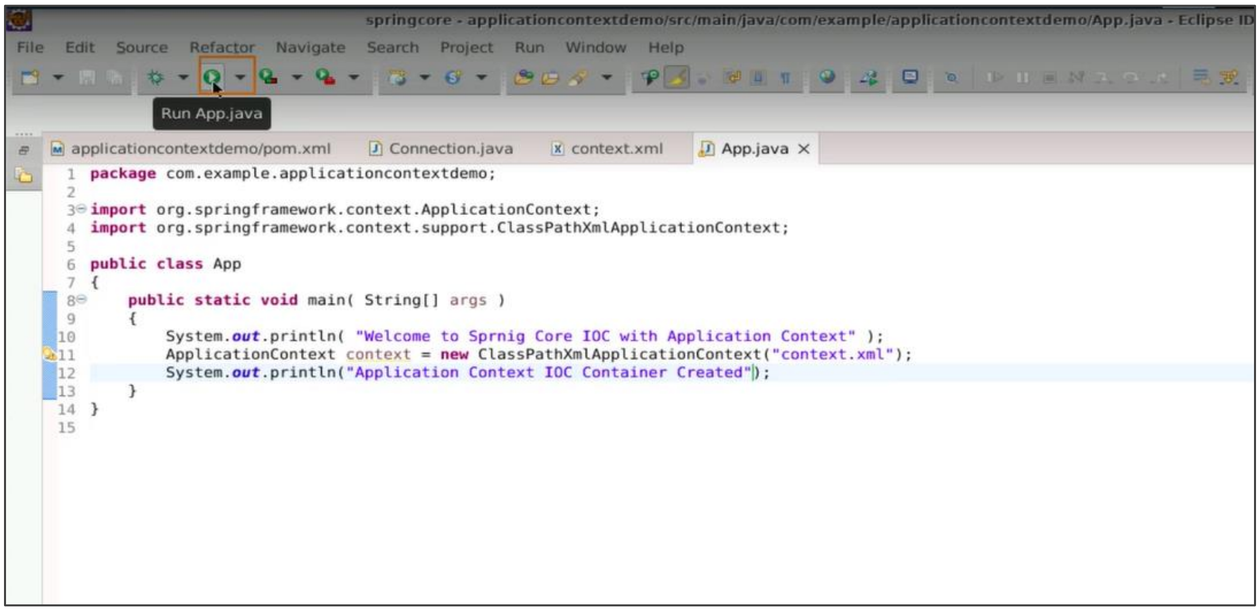
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="con1" class="com.example.applicationcontextdemo.bean.Connection">
        <property name="url" value="jdbc:mysql://localhost/estore"/>
        <property name="user" value="john.watson"/>
        <property name="password" value="@vegers!@#"/>
    </bean>

    <bean id="con2" class="com.example.beanfactorydemo.bean.Connection">
        <property name="url" value="jdbc:mysql://localhost/fooddelivery"/>
        <property name="user" value="fionna"/>
        <property name="password" value="t@ngocha@rlie"/>
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

3.3 In the **App.java**, import the necessary Spring Framework packages, create an instance of the **ApplicationContext** interface using the **ClassPathXmlApplicationContext,** and pass the **context.xml** file to the **ApplicationContext** constructor

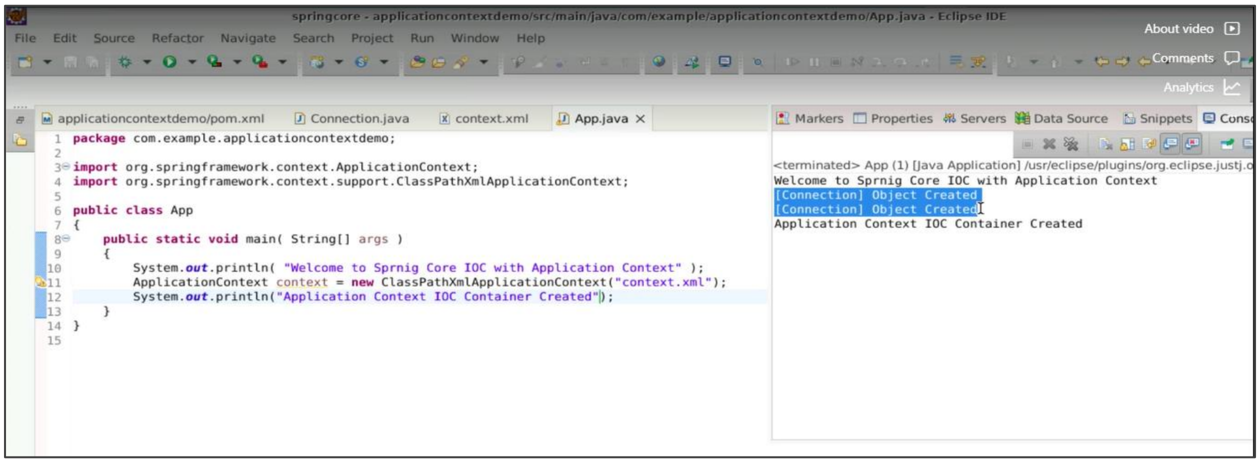


```java
package com.example.applicationcontextdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
        System.out.println("Application Context IOC Container Created");
    }
}
```

3.4 Run the project


springcore - applicationcontextdemo/src/main/java/com/example/applicationcontextdemo/App.java - Eclipse ID
File Edit Source Refactor Navigate Search Project Run Window Help
Run App.java
applicationcontextdemo/pom.xml
Connection.java
context.xml
App.java

```java
package com.example.applicationcontextdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
        System.out.println("Application Context IOC Container Created");
    }
}
```

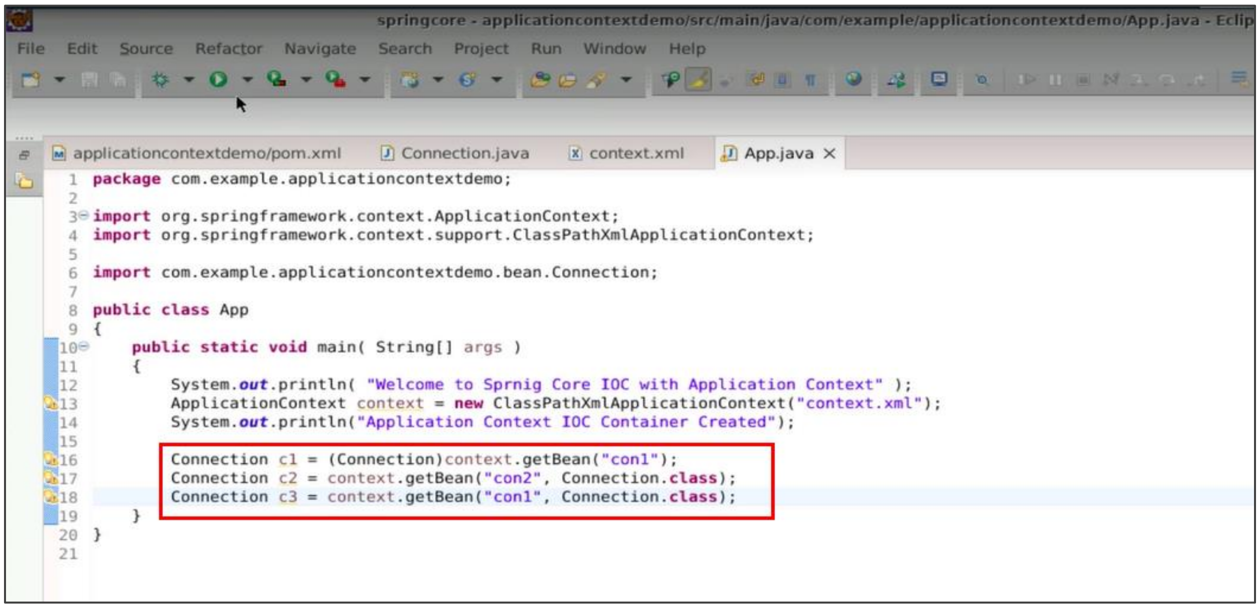

springcore - applicationcontextdemo/src/main/java/com/example/applicationcontextdemo/App.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
About video
Comments
Analytics
applicationcontextdemo/pom.xml
Connection.java
context.xml
App.java
Markers
Properties
Servers
Data Source
Snippets
Cons

```java
package com.example.applicationcontextdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
        System.out.println("Application Context IOC Container Created");
    }
}
```

```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.o
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created
[Connection] Object Created
Application Context IOC Container Created
```

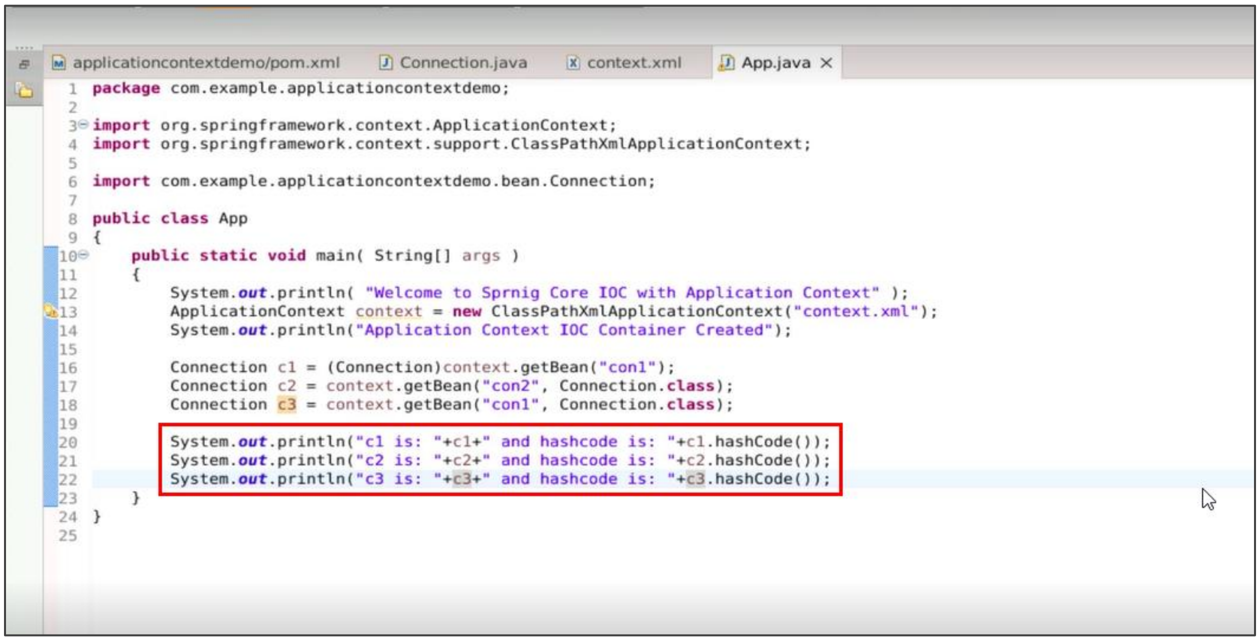
You can see two Connection objects are created along with the print statements.

3.5 Use the **getBean()** method to retrieve the bean instances by their IDs and cast the retrieved bean objects to the appropriate class


springcore - applicationcontextdemo/src/main/java/com/example/applicationcontextdemo/App.java - Eclip
File Edit Source Refactor Navigate Search Project Run Window Help
applicationcontextdemo/pom.xml
Connection.java
context.xml
App.java

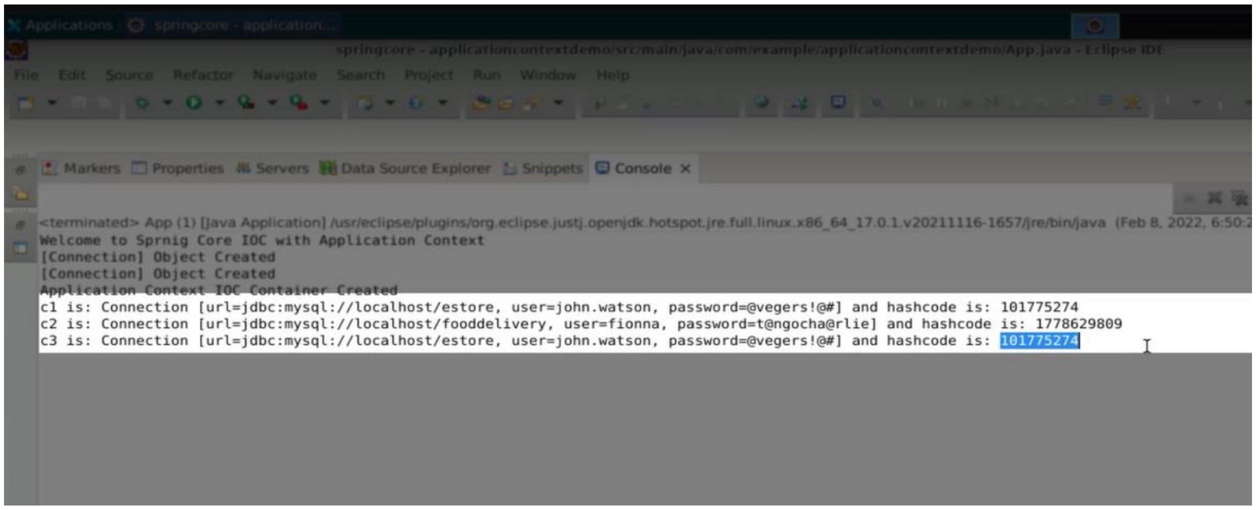
```java
package com.example.applicationcontextdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.applicationcontextdemo.bean.Connection;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
        System.out.println("Application Context IOC Container Created");

        Connection c1 = (Connection)context.getBean("con1");
        Connection c2 = context.getBean("con2", Connection.class);
        Connection c3 = context.getBean("con1", Connection.class);
    }
}
```

3.6 Assign retrieved bean objects to variables, cast them to appropriate class types, and add print statements to display data along with hash codes



```java
package com.example.applicationcontextdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.applicationcontextdemo.bean.Connection;

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
        ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
        System.out.println("Application Context IOC Container Created");

        Connection c1 = (Connection)context.getBean("con1");
        Connection c2 = context.getBean("con2", Connection.class);
        Connection c3 = context.getBean("con1", Connection.class);

        System.out.println("c1 is: "+c1+" and hashcode is: "+c1.hashCode());
        System.out.println("c2 is: "+c2+" and hashcode is: "+c2.hashCode());
        System.out.println("c3 is: "+c3+" and hashcode is: "+c3.hashCode());
    }
}
```

3.7 Rerun the project

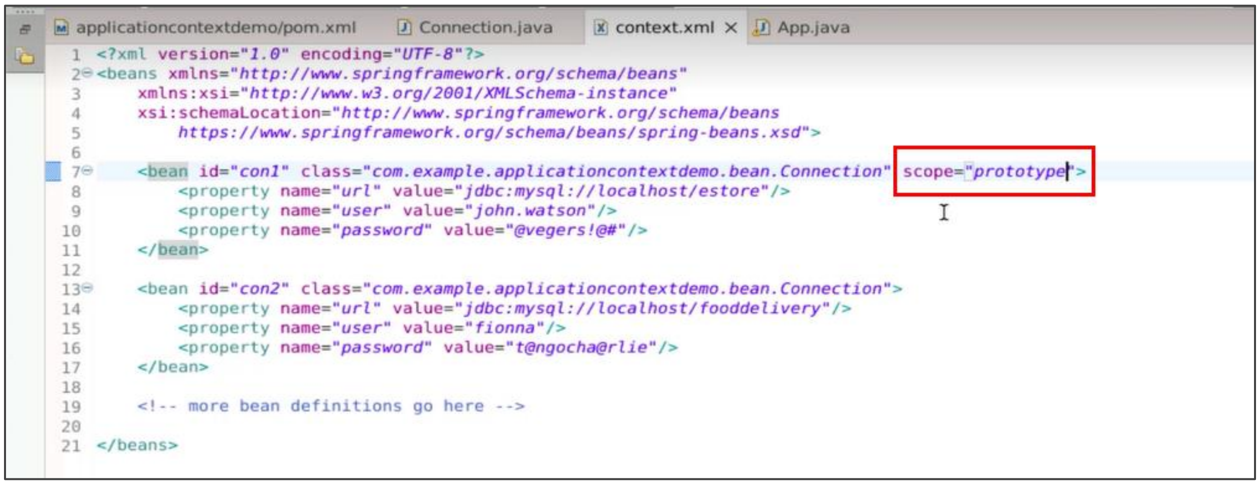


```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 6:50:2
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created
[Connection] Object Created
Application Context IOC Container Created
c1 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 101775274
c2 is: Connection [url=jdbc:mysql://localhost/fooddelivery, user=fionna, password=t@ngocha@rlie] and hashcode is: 1778629809
c3 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 101775274
```

The two connection objects are created: **c1** and **c3**. They are sharing the same data and objects and their hash codes are the same.

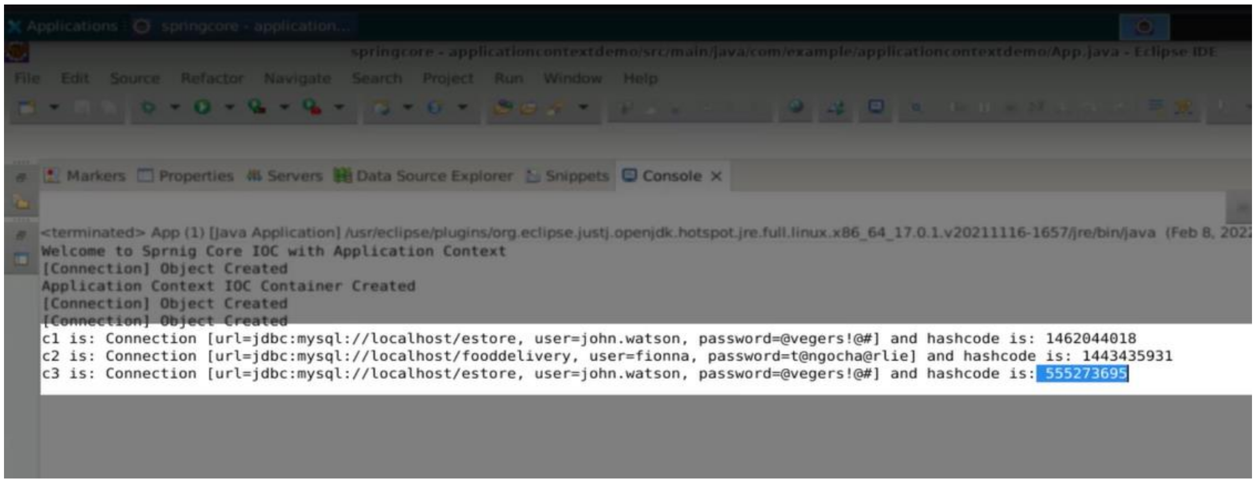
Step 4: Modifying the bean scope

4.1 Go to the **context.xml** file and add a **scope** attribute with a value of **prototype** to the bean **con1**



```xml
<?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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="con1" class="com.example.applicationcontextdemo.bean.Connection" scope="prototype">
        <property name="url" value="jdbc:mysql://localhost/estore"/>
        <property name="user" value="john.watson"/>
        <property name="password" value="@vegers!@#"/>
    </bean>

    <bean id="con2" class="com.example.applicationcontextdemo.bean.Connection">
        <property name="url" value="jdbc:mysql://localhost/fooddelivery"/>
        <property name="user" value="fionna"/>
        <property name="password" value="t@ngocha@rlie"/>
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

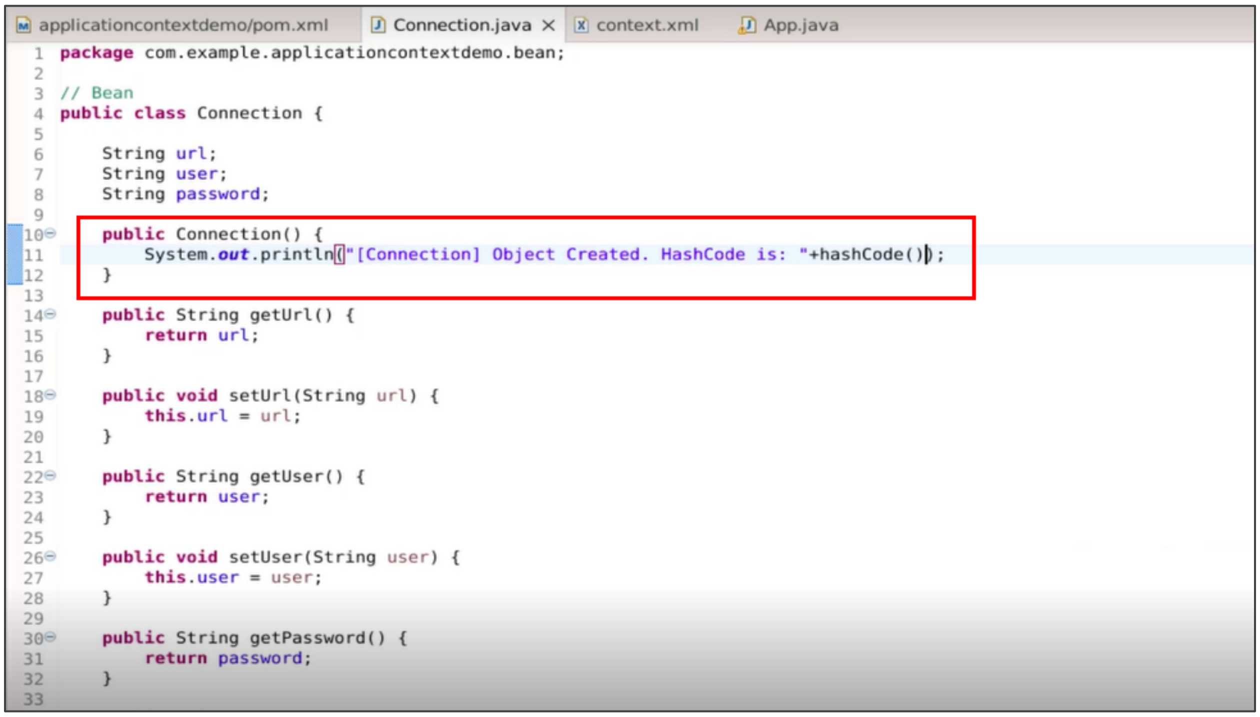
4.2 Run the project



```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created
Application Context IOC Container Created
[Connection] Object Created
[Connection] Object Created
c1 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 1462044018
c2 is: Connection [url=jdbc:mysql://localhost/fooddelivery, user=fionna, password=t@ngocha@rlie] and hashcode is: 1443435931
c3 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 555273695
```

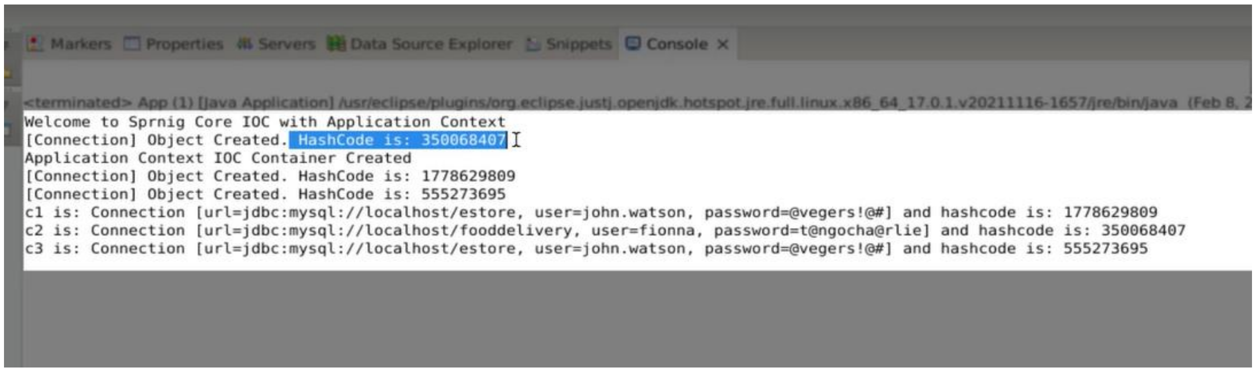
Three connection objects were created in total, and while **c1** and **c2** share the same data, their hash codes now differ. If the scope is set as a **prototype**, new objects will be created every time the **getBean** method is called.

4.3 In the **Connection.java** class, include a print statement in the default constructor to display the hash code of each created object. This will provide information about the object's creation order.



```java
package com.example.applicationcontextdemo.bean;

// Bean
public class Connection {

    String url;
    String user;
    String password;

    public Connection() {
        System.out.println("[Connection] Object Created. HashCode is: "+hashCode());
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPassword() {
        return password;
    }

```

4.4 Rerun the project

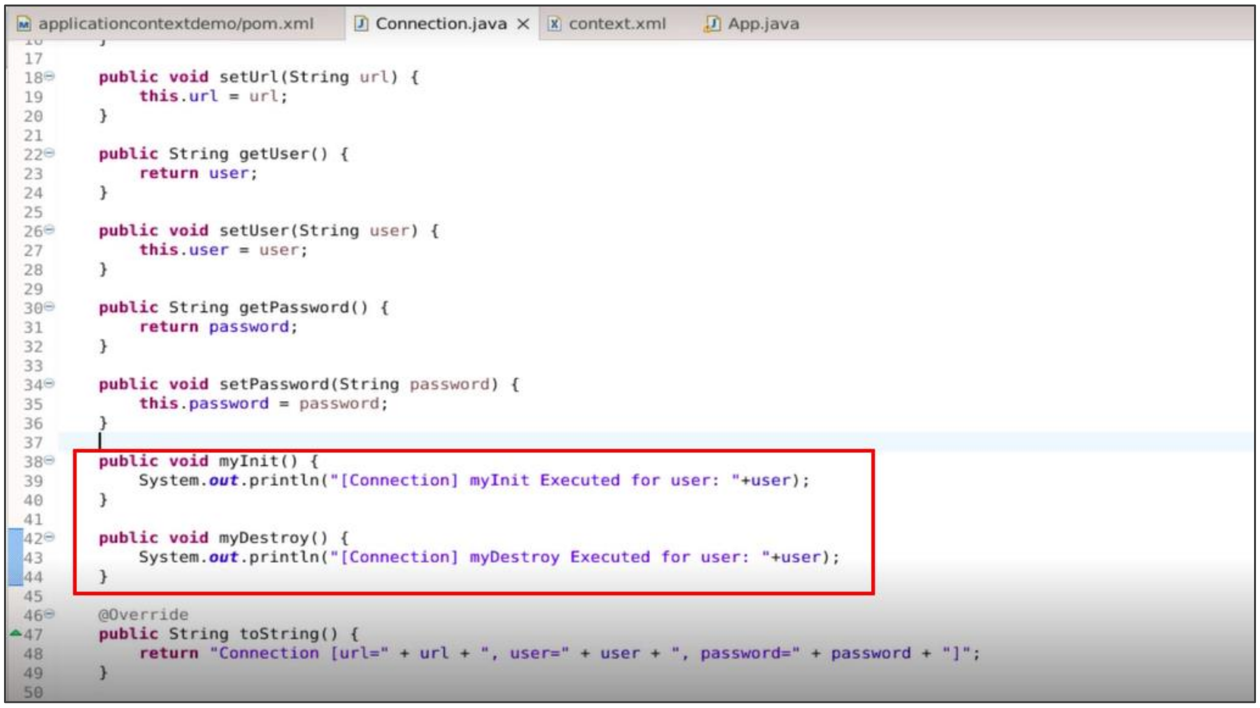


```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created. HashCode is: 350068407
Application Context IOC Container Created
[Connection] Object Created. HashCode is: 1778629809
[Connection] Object Created. HashCode is: 555273695
c1 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 1778629809
c2 is: Connection [url=jdbc:mysql://localhost/fooddelivery, user=fionna, password=t@ngocha@rlie] and hashcode is: 350068407
c3 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 555273695
```

In the console, you can see that **c2** is created first with the default **singleton** scope, while **c1** and **c3** are created later when the **getBean()** methods are called. The hash codes differ once again.

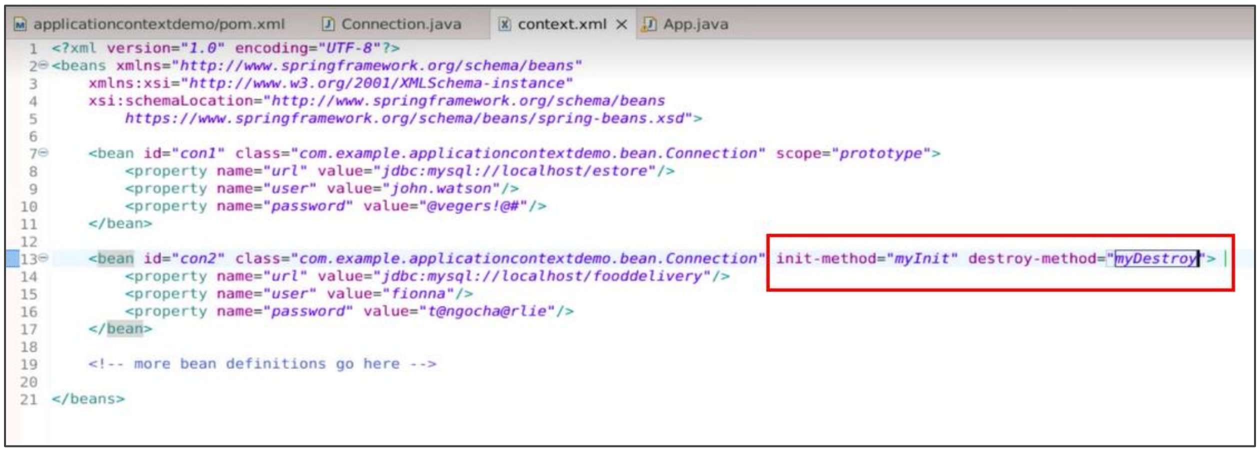
Step 5: Implementing methods for the bean lifecycle

5.1 Implement **myInit()** and **myDestroy()** methods in the **Connection.java** class with print statements to track the bean lifecycle, including initialization and destruction



```java
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void myInit() {
        System.out.println("[Connection] myInit Executed for user: "+user);
    }

    public void myDestroy() {
        System.out.println("[Connection] myDestroy Executed for user: "+user);
    }

    @Override
    public String toString() {
        return "Connection [url=" + url + ", user=" + user + ", password=" + password + "]";
    }
```

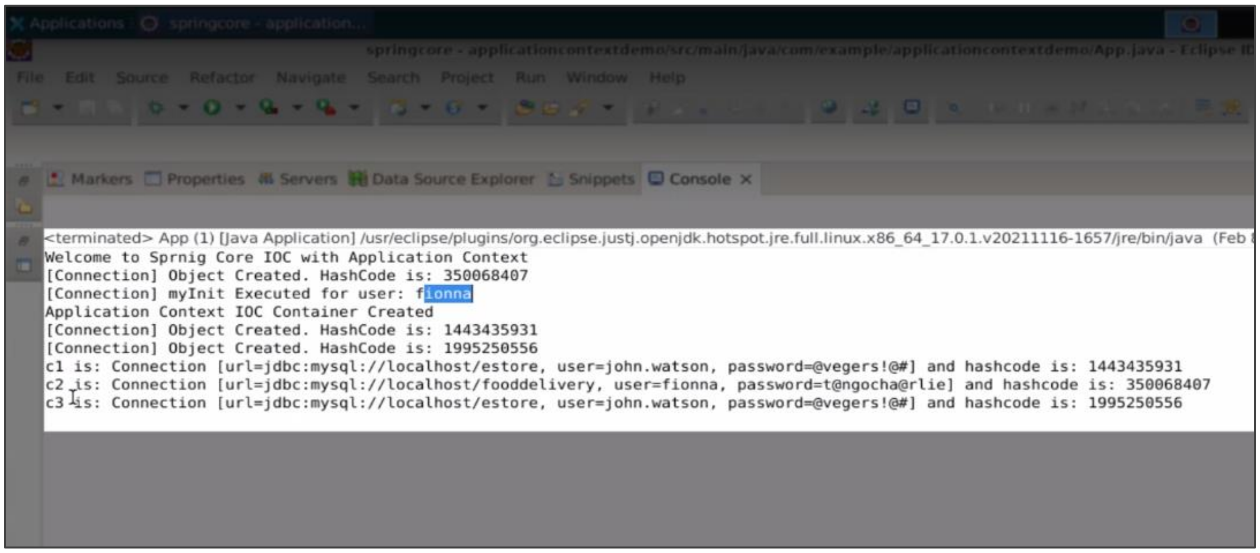
5.2 Configure attributes for the **con2** bean in the **context.xml** file to enable the usage of **myInit()** and **myDestroy()** methods



applicationcontextdemo/pom.xml | Connection.java | context.xml | App.java

```xml
<?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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="con1" class="com.example.applicationcontextdemo.bean.Connection" scope="prototype">
        <property name="url" value="jdbc:mysql://localhost/estore"/>
        <property name="user" value="john.watson"/>
        <property name="password" value="@vegers!@#"/>
    </bean>

    <bean id="con2" class="com.example.applicationcontextdemo.bean.Connection" init-method="myInit" destroy-method="myDestroy">
        <property name="url" value="jdbc:mysql://localhost/fooddelivery"/>
        <property name="user" value="fionna"/>
        <property name="password" value="t@ngocha@rlie"/>
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

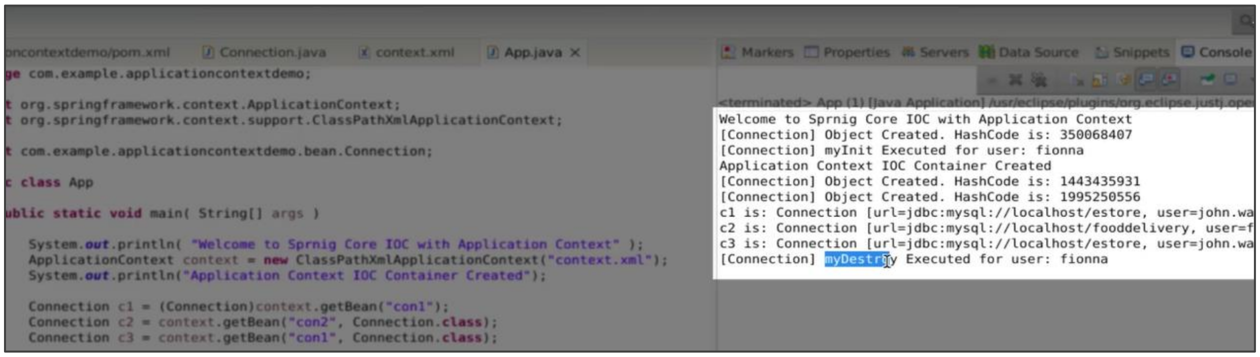
5.3 Run the project



```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java  (Feb
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created. HashCode is: 350068407
[Connection] myInit Executed for user: fionna
Application Context IOC Container Created
[Connection] Object Created. HashCode is: 1443435931
[Connection] Object Created. HashCode is: 1995250556
c1 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 1443435931
c2 is: Connection [url=jdbc:mysql://localhost/fooddelivery, user=fionna, password=t@ngocha@rlie] and hashcode is: 350068407
c3 is: Connection [url=jdbc:mysql://localhost/estore, user=john.watson, password=@vegers!@#] and hashcode is: 1995250556
```

You can observe that the **myInit()** method is executed and the username **fionna** is printed only for the **con2** bean.

5.4 Close the **ApplicationContext** by creating a reference variable **cxt** of type **ClassPathXmlApplicationContext**, downcasting the **ApplicationContext** interface, and invoking the **close()** method to trigger the execution of **myDestroy()**



```
ge com.example.applicationcontextdemo;

t org.springframework.context.ApplicationContext;
t org.springframework.context.support.ClassPathXmlApplicationContext;

t com.example.applicationcontextdemo.bean.Connection;

c class App

ublic static void main( String[] args )

  System.out.println( "Welcome to Sprnig Core IOC with Application Context" );
  ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
  System.out.println("Application Context IOC Container Created");

  Connection c1 = (Connection)context.getBean("con1");
  Connection c2 = context.getBean("con2", Connection.class);
  Connection c3 = context.getBean("con1", Connection.class);
```

```
<terminated> App (1) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.ope
Welcome to Sprnig Core IOC with Application Context
[Connection] Object Created. HashCode is: 350068407
[Connection] myInit Executed for user: fionna
Application Context IOC Container Created
[Connection] Object Created. HashCode is: 1443435931
[Connection] Object Created. HashCode is: 1995250556
c1 is: Connection [url=jdbc:mysql://localhost/estore, user=john.wa
c2 is: Connection [url=jdbc:mysql://localhost/fooddelivery, user=f
c3 is: Connection [url=jdbc:mysql://localhost/estore, user=john.wa
[Connection] myDestroy Executed for user: fionna
```

You can notice that the **myDestroy()** method is executed at the end, resulting in the elimination of all objects associated with the user **fionna**.

In conclusion, we have explored the ApplicationContext API as an alternative to BeanFactory for Spring IOC and delved into the lifecycle methods of a bean, including the constructor, init, and destroy phases.