

Person.java

—--------------

**package** com.codegnan.beans;

**public** **class** Person {

**int** id;

String name;

String email;

**public** Person() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Person(**int** id, String name, String email) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.email = email;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Person [id=" + id + ", name=" + name + ", email=" + email + "]";

}

}

PersonDbService.java

—---------------------------

**package** com.codegnan.service;

**import** com.codegnan.beans.Person;

**public** **class** PersonDbService {

String driver;

String url;

String username;

String password;

**public** PersonDbService() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** PersonDbService(String driver, String url, String username, String password) {

**super**();

**this**.driver = driver;

**this**.url = url;

**this**.username = username;

**this**.password = password;

}

**public** String getDriver() {

**return** driver;

}

**public** **void** setDriver(String driver) {

**this**.driver = driver;

}

**public** String getUrl() {

**return** url;

}

**public** **void** setUrl(String url) {

**this**.url = url;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

@Override

**public** String toString() {

**return** "PersonDbService [driver=" + driver + ", url=" + url + ", username=" + username + ", password="

+ password + "]";

}

**public** **void** save(Person person) {

System.***out***.println("Saving a Person to Db Is : " + person);

}

**public** **void** find(**int** id) {

System.***out***.println("Finding A person from db with id : " + id);

}

}

App.java

—--------------

**package** com.codegnan;

**import** com.codegnan.beans.Person;

**import** com.codegnan.service.PersonDbService;

/\*\*

\* Hello world!

\*

\*/

**public** **class** App

{

**public** **static** **void** main( String[] args )

{

PersonDbService personDbService=**new** PersonDbService();

personDbService.setDriver("com.mysql..");

personDbService.setUrl("jdbc:mysql...");

personDbService.setUsername("root");

personDbService.setPassword("root");

Person person=**new** Person(101,"malli","malli@gmail");

personDbService.save(person);

}

}

Pom.xml

—--------------

<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.codegnan</groupId>

<artifactId>SpringDemo\_01</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringDemo\_01</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>5.3.31</version>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.16.22</version>

<scope>provided</scope>

</dependency>

</dependencies>

</project>

**Spring History:**

1. Home: Interface1

2. Author: Mr. Rod Janson

3. Objective: To simplify and Accelerate the complete Enterprise Applications.

4. Type: Open Source Software.

5. Type of Framework: Application Framework.

6. Initial Version: Spring1.0[Oct, 2004]

7. Used Version: Spring5.x[Dec, 2014]

8. Latest Version: Spring6.x[April, 2017]

9. Website: [http://spring.io](http://spring.io/)

10. Designed on: Java[JAVA SE API, Servlets API] 11.Compatibility:Supported by All IDEs and all Servers

12:Designi Tool: STS[Spring tool Suit][Designed on the top of Eclipse]

what is spring:

----------------------------

1. spring is a factory

2. spring is a dependency injection

3. spring is singleton pattern class.

**Steps to Prepare Spring Application**

**—-----------------------------------------------**

**1.Prepare the persistence class or pojo class**

**2.prepare Spring configuration File**

**3.prepare the client application**

**1.Prepare the persistence class or pojo class**

**—-------------------------------------------------------------**

**POJO is Plain Old Java Objects, these are beans which must not extend or implement predefined Libraries except java.io.Serializable.**

**If we want to use Beans in java applications then we have to use the following rules and regulations.**

1. **Every Bean class must be declared with public, non abstract and non final.**

**Where the main purpose to declare bean class as public is to make available the scope of bean classes to the Containers, Frameworks,..... Inorder to create objects.**

**Where the main purpose to declare a bean class as non abstract is to allow Frameworks or containers to create objects.**

**Where the main purpose to declare a bean class as non-final is to allow extending a bean class to another bean class in order to reuse one bean class behavior in another bean class.**

1. **In the Bean class, we must provide a separate setXXX() method and getXXX() method for every property.**

1. **In Bean class, declare all properties as private and declare all methods as public.**

**Spring Configuration File:**

**—-------------------------------**

Spring Configuration file is an xml file, it must have all the bean classes configuration details which are required by the Container in order to create Bean objects and in order to supply that bean objects to the application.

To provide beans configuration in the Spring configuration file we have to use the following tags.

<beans —-------XSD —------>

<bean id=”---” class=”---”/>

—----

—---

</beans>

Where “<beans>” is a root tag, it includes a number of bean configurations.

Where “<bean> tag is able to provide single bean class configuration.

Where “id” attribute in <bean> tag will provide identity for the bean object.

Where “class” attribute will take the fully qualified name of the bean class.

Note: In the Spring Configuration file we have to provide XSD , here we will get XSD from the following resource.

E:\softwares\frameworks\Spring\spring-framework-5.3.9\docs\reference\html\core.html [As per my File system]

<?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>">

—------

</beans>

EX:

SpringConfig.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=”helloBean” class=”com.durgasoft.beans.Hello”/>

</beans>

**Prepare Test Application:**

**—-------------------------------**

The main intention of Test application is

1. Create a Container.
2. Get a Bean object from Container.
3. Access Business methods of the respective Bean.

**Creating a Container:**

**—--------------------------**

The Main purpose of Container is to create bean objects and to manage bean objects along with their identities.

Spring Framework has provided the following two types of Containers.

1. BeanFactory
2. ApplicationContext

To represent ApplicationContext containers, Spring Framework has provided a predefined interface in the form of “ApplicationContext” and Spring Framework has provided an implementation class for ApplicationContext in the form of “ClassPathXmlApplicationContext”.

To create a ClassPathXmlApplicationContext object we have to use the following Constructor.

public ClassPathXmlApplicationContext(String configFilePath)

EX:

ApplicationContext context = new ClassPathXmlApplicationContext(“SpringConfig.xml”);

When we create a Container object like above, Container will perform the following actions.

1. Identify the spring configuration file.
2. When Spring Configuration is identified , Container will perform loading Configuration file, parsing configuration file and reading the data from Configuration file.
3. After getting data from the Configuration file, Container will recognize all the bean configurations and Container will create objects for all the bean classes which we configured in the spring configuration file.

1. After creating bean objects, Container will manage all the bean objects along with their identity values [id attribute values].

**Get a Bean object from Container**

**—------------------------------------------**

To get a particular Bean object from Container we have to use the following method from ApplicationContext.

public Object getBean(String idValue)

EX:

Hello hello = context.getBean(“helloBean”);

**Access Business Method:**

**—--------------------------------**

String msg = hello.sayHello();

System.out.println(msg);

In the above program is not a spring we have to convert the above program in Spring

===================================================================

Person.java

—--------------

**package** com.codegnan.beans;

**public** **class** Person {

**int** id;

String name;

String email;

**public** Person() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Person(**int** id, String name, String email) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.email = email;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Person [id=" + id + ", name=" + name + ", email=" + email + "]";

}

}

Spring.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*

*http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"person"* class=*"com.codegnan.beans.Person" scope=”prototype”*>

<property name=*"id"* value=*"101"* />

<property name=*"name"* value=*"codegnan"* />

<property name=*"email"* value=*"codegnan@gmail.com"* />

</bean>

</beans>

App.java

—-----------

package com.codegnan;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.codegnan.beans.Person;

public class App

{

public static void main( String[] args )

{

ApplicationContext context=new ClassPathXmlApplicationContext("Spring.xml");

Person person =(Person)context.getBean("person");

Person person1 =(Person)context.getBean("person");

System.out.println(person);

if(person==person1) {

System.out.println("both are same");

}else {

System.out.println("Both are diffrent");

}

}

}

Pom.xml

—------------

<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.codegnan</groupId>

<artifactId>SpringDemo\_02</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringDemo\_02</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>

<!--

https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.31</version>

</dependency>

</dependencies>

</project>

Annotation Based Container Configuration

==================================

Person.java

—-----------------

**package** com.codegnan.beans;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.stereotype.Component;

@Component("person")

//@Component

**public** **class** Person {

@Value("111")

**private** **int** id;

@Value("Ram")

**private** String name;

@Value("ram@gmail.com")

**private** String email;

**public** Person() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Person(**int** id, String name, String email) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.email = email;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Person [id=" + id + ", name=" + name + ", email=" + email + "]";

}

}

Spring.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"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*https://www.springframework.org/schema/context/spring-context.xsd"*>

<context:annotation-config />

<context:component-scan

base-package=*"com.codegnan"* />

</beans>

App.java

============

package com.codegnan;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.codegnan.beans.Person;

public class App

{

public static void main( String[] args )

{

ApplicationContext context=new ClassPathXmlApplicationContext("Spring.xml");

Person person =(Person)context.getBean("person");

System.out.println(person);

}

}

Pom.xml

—-----------

<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.codegnan</groupId>

<artifactId>SpringDemo\_02</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringDemo\_02</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>

<!--

https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.31</version>

</dependency>

</dependencies>

</project>

Dependency Injection:

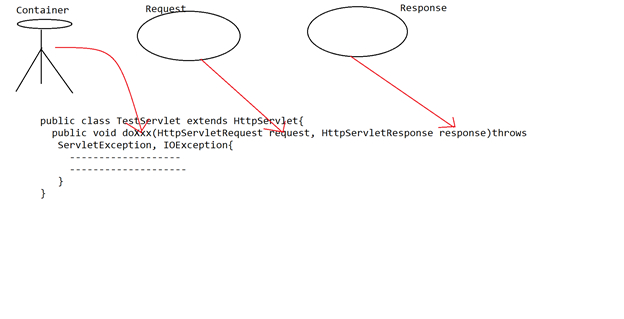
-----------------------------

In this approach, Service provider will create the services and service provider will send the services to the main application directly.

EX:

In Servlet application execution, the container will execute Servlet classes by its lifecycle, where in Request Processing phase Container will access service() method in order to process the request, to access service() method container must create and pass request and response objects, it is called as "Dependency Injection".

public void service(ServletRequest request, ServletResponse response)throws ServletException, IOException



There are two types of Dependency Injection:

1. Constructor Dependency Injection

2. Setter Method Dependency Injection

1. Constructor Dependency Injection:

--------------------------------------------------

The process of injecting a dependency object through a constructor is called "Constructor Dependency Injection".

If we want to implement Constructor Dependency Injection in Spring applications we have to use the following tags in the Spring Configuration file under <bean> tag.

<beans>

<bean id=”--” class=”--”>

<constructor-arg value=”--” name=”--” index=”---”/>

</bean>

</beans>

Ex:

Customer.java

—---------------------

**package** com.codegnan.beans;

**public** **class** Customer {

**private** String customerId;

**private** String customerName;

**private** String customerAddress;

**private** String customerMobileNo;

**public** Customer(String customerId, String customerName, String customerAddress, String customerMobileNo) {

**this**.customerId = customerId;

**this**.customerName = customerName;

**this**.customerAddress = customerAddress;

**this**.customerMobileNo = customerMobileNo;

}

**public** **void** displayCustomerDetails() {

System.***out***.println("Customer Details");

System.***out***.println("-----------------------");

System.***out***.println("Customer Id : " + customerId);

System.***out***.println("Customer Name : " + customerName);

System.***out***.println("Customer Address : " + customerAddress);

System.***out***.println("Customer Mobile No : " + customerMobileNo);

}

}

Item.java

—-----------

**package** com.codegnan.beans;

**public** **class** Item {

**private** String itemId;

**private** String itemName;

**private** **int** itemPrice;

**public** Item(String itemId, String itemName, **int** itemPrice) {

**this**.itemId = itemId;

**this**.itemName = itemName;

**this**.itemPrice = itemPrice;

}

**public** **void** displayItemDetails() {

System.***out***.println("Item Details");

System.***out***.println("--------------------");

System.***out***.println("Item Id : " + itemId);

System.***out***.println("Item Name : " + itemName);

System.***out***.println("Item Proce : " + itemPrice);

}

}

Order.java

—-------------

**package** com.codegnan.beans;

**public** **class** Order {

**private** String orderId;

**private** String orderName;

**private** Customer customer;

**private** Item item;

**public** Order(String orderId, String orderName, Customer customer, Item item) {

**this**.orderId = orderId;

**this**.orderName = orderName;

**this**.customer = customer;

**this**.item = item;

}

**public** **void** displayOrderDetails() {

System.***out***.println("Order Details");

System.***out***.println("-------------------");

System.***out***.println("Order Id : " + orderId);

System.***out***.println("Order Name : " + orderName);

System.***out***.println();

customer.displayCustomerDetails();

System.***out***.println();

item.displayItemDetails();

}

}

SpringConfig.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=*"customer"* class=*"com.codegnan.beans.Customer"*>

<constructor-arg value=*"C111"* name=*"customerId"*/>

<constructor-arg value=*"malli"* name=*"customerName"*/>

<constructor-arg value=*"JNTU Metro, Hyd-38"* name=*"customerAddress"*/>

<constructor-arg value=*"9988776655"* name=*"customerMobileNo"*/>

</bean>

<bean id=*"item"* class=*"com.codegnan.beans.Item"*>

<constructor-arg name=*"itemId"* value=*"I-111"*/>

<constructor-arg name=*"itemName"* value=*"Mobile"*/>

<constructor-arg name=*"itemPrice"* value=*"25000"*/>

</bean>

<bean id=*"order"* class=*"com.codegnan.beans.Order"*>

<constructor-arg name=*"orderId"* value=*"O-111"*/>

<constructor-arg name=*"orderName"* value=*"Mobile-Order"*/>

<constructor-arg name=*"customer"* ref=*"customer"*/>

<constructor-arg name=*"item"* ref=*"item"*/>

</bean>

</beans>

App.java

—---------

**package** com.codegnan.Spring01;

**import** org.springframework.context.ApplicationContext;

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

**import** com.codegnan.beans.Order;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("SpringConfig.xml");

Order order = applicationContext.getBean(Order.**class**);

order.displayOrderDetails();

}

}

Pom.xml

—----------

<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.codegnan</groupId>

<artifactId>Spring01</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Spring01</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>5.3.25</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

</project>

2. Setter Method Dependency Injection:

-----------------------------------------------------

The process of injecting a dependency object through its respective setter method is called "Setter Method Dependency Injection".

EX:

Item.java

—---------------

**package** com.codegnan.beans;

**public** **class** Item {

**private** String itemId;

**private** String itemName;

**private** **int** itemPrice;

**public** String getItemId() {

**return** itemId;

}

**public** **void** setItemId(String itemId) {

**this**.itemId = itemId;

}

**public** String getItemName() {

**return** itemName;

}

**public** **void** setItemName(String itemName) {

**this**.itemName = itemName;

}

**public** **int** getItemPrice() {

**return** itemPrice;

}

**public** **void** setItemPrice(**int** itemPrice) {

**this**.itemPrice = itemPrice;

}

}

Cart.java

—-----------

**package** com.codegnan.beans;

**public** **class** Cart {

**private** String cartId;

**private** String cartName;

**private** Item item;

**public** String getCartId() {

**return** cartId;

}

**public** **void** setCartId(String cartId) {

**this**.cartId = cartId;

}

**public** String getCartName() {

**return** cartName;

}

**public** **void** setCartName(String cartName) {

**this**.cartName = cartName;

}

**public** Item getItem() {

**return** item;

}

**public** **void** setItem(Item item) {

**this**.item = item;

}

**public** **void** displayCartDetails() {

System.***out***.println("Cart Details");

System.***out***.println("-------------------");

System.***out***.println("Cart Id : " + cartId);

System.***out***.println("Cart Name : " + cartName);

System.***out***.println();

System.***out***.println("Item Details");

System.***out***.println("----------------");

System.***out***.println("Item Id : " + item.getItemId());

System.***out***.println("Item Name : " + item.getItemName());

System.***out***.println("Item Price : " + item.getItemPrice());

}

}

SpringConfig.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=*"item"* class=*"com.codegnan.beans.Item"*>

<property name=*"itemId"* value=*"I-111"* />

<property name=*"itemName"* value=*"MI Mobile"* />

<property name=*"itemPrice"* value=*"25000"* />

</bean>

<bean id=*"cart"* class=*"com.codegnan.beans.Cart"*>

<property name=*"cartId"* value=*"C-111"* />

<property name=*"cartName"* value=*"Mobile-Cart"* />

<property name=*"item"* ref=*"item"* />

</bean>

</beans>

Client.java

—--------------

**package** com.codegnan.client;

**import** org.springframework.context.ApplicationContext;

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

**import** com.codegnan.beans.Cart;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("SpringConfig.xml");

Cart cart = applicationContext.getBean(Cart.**class**);

cart.displayCartDetails();

}

}

Pom.xml

—------------

<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.codegnan</groupId>

<artifactId>Spring01</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Spring01</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>5.3.25</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

</project>

Ex

—--

Review.java

—-------------

**package** com.codegnan.beans;

**public** **class** Review {

**private** String reviewId;

**private** String criticsName;

**private** String description;

**private** **float** rating;

**public** **float** getRating() {

**return** rating;

}

**public** **void** setRating(**float** rating) {

**this**.rating = rating;

}

**public** String getCriticsName() {

**return** criticsName;

}

**public** **void** setCriticsName(String criticsName) {

**this**.criticsName = criticsName;

}

**public** String getReviewId() {

**return** reviewId;

}

**public** **void** setReviewId(String reviewId) {

**this**.reviewId = reviewId;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

}

Movie.java

—-----------------

**package** com.codegnan.beans;

**public** **class** Movie {

**private** String movieId;

**private** String movieName;

**private** Review review;

**public** String getMovieId() {

**return** movieId;

}

**public** **void** setMovieId(String movieId) {

**this**.movieId = movieId;

}

**public** String getMovieName() {

**return** movieName;

}

**public** **void** setMovieName(String movieName) {

**this**.movieName = movieName;

}

**public** Review getReview() {

**return** review;

}

**public** **void** setReview(Review review) {

**this**.review = review;

}

**public** **void** getMovieDetails() {

System.***out***.println("Movie Details");

System.***out***.println("----------------------");

System.***out***.println("Movie Id : " + movieId);

System.***out***.println("Movie Name : " + movieName);

System.***out***.println("Review Details");

System.***out***.println("-----------------------");

System.***out***.println("Review Id : " + review.getReviewId());

System.***out***.println("Critics Name: " + review.getCriticsName());

System.***out***.println("Description : " + review.getDescription());

System.***out***.println("Rating : " + review.getRating());

}

}

SpringConfig.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=*"review"* class=*"com.codegnan.beans.Review"*>

<property name=*"reviewId"* value=*"R-111"* />

<property name=*"criticsName"* value=*"Malli"* />

<property name=*"description"* value=*"Blockbuster"* />

<property name=*"rating"* value=*"4.0"* />

</bean>

<bean id=*"movie"* class=*"com.codegnan.beans.Movie"*>

<property name=*"movieId"* value=*"M-111"* />

<property name=*"movieName"* value=*"PUSHPA"* />

<property name=*"review"* ref=*"review"* />

</bean>

</beans>

Client.java

—--------------

**package** com.codegnan.client;

**import** org.springframework.context.ApplicationContext;

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

**import** com.codegnan.beans.Movie;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("SpringConfig.xml");

Movie movie = (Movie) applicationContext.getBean("movie");

movie.getMovieDetails();

}

}

POM.XML

—--------

<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.codegnan</groupId>

<artifactId>Spring01</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Spring01</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>5.3.25</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

</project>