Loading Country Data from Spring XML Configuration (Spring Core)

Step 1: Create the Country Class

package com.example.model;

public class Country {

private String code;

private String name;

// Constructors

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

Step 2: Create Spring Configuration XML

Create src/main/resources/applicationContext.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">

<!-- Define Country bean with property injection -->

<bean id="country" class="com.example.model.Country">

<property name="code" value="US" />

<property name="name" value="United States" />

</bean>

<!-- Alternative with constructor injection -->

<bean id="countryWithConstructor" class="com.example.model.Country">

<constructor-arg index="0" value="IN" />

<constructor-arg index="1" value="India" />

</bean>

</beans>

Step 3: Load the Application Context and Retrieve Beans

Create a main class to test the configuration:

package com.example;

import com.example.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load the Spring configuration file

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

// Retrieve bean by name

Country country = (Country) context.getBean("country");

System.out.println("Country from property injection: " + country);

Country countryConst = (Country) context.getBean("countryWithConstructor");

System.out.println("Country from constructor injection: " + countryConst);

}

}

Step 4: Maven Dependencies

Ensure your pom.xml includes Spring Core:

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.20</version> <!-- Use appropriate version -->

</dependency>

</dependencies>

Step 5: Running the Application

When you run MainApp, you should see output like:

text

Country from property injection: Country [code=US, name=United States]

Country from constructor injection: Country [code=IN, name=India]

Advanced Configuration Options

1. Using Collections in XML Configuration

xml

<bean id="countries" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="country1"/>

<ref bean="country2"/>

</list>

</constructor-arg>

</bean>

<bean id="country1" class="com.example.model.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

<bean id="country2" class="com.example.model.Country">

<property name="code" value="FR"/>

<property name="name" value="France"/>

</bean>

2. Using Property Files with XML

Create src/main/resources/countries.properties:

usa.code=US

usa.name=United States

india.code=IN

india.name=India

Update XML configuration:

<context:property-placeholder location="classpath:countries.properties"/>

<bean id="usa" class="com.example.model.Country">

<property name="code" value="${usa.code}"/>

<property name="name" value="${usa.name}"/>

</bean>

3. Bean Scopes in XML

<bean id="prototypeCountry" class="com.example.model.Country" scope="prototype">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

Migrating to Annotation-Based Configuration

While XML configuration is still supported, modern Spring applications typically use annotations:

@Configuration

public class AppConfig {

@Bean

public Country country() {

return new Country("UK", "United Kingdom");

}

}