**Hands on 4**

**Spring Core – Load Country from Spring Configuration XML**

An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country. 

|  |  |
| --- | --- |
| **Code** | **Name** |
| US | United States |
| DE | Germany |
| IN | India |
| JP | Japan |

Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.

**Country.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private static final Logger LOGGER = LoggerFactory.getLogger(Country.class);

private String code;

private String name;

public Country() {

LOGGER.debug("Inside Country Constructor.");

}

public String getCode() {

LOGGER.debug("Inside getCode()");

return code;

}

public void setCode(String code) {

LOGGER.debug("Inside setCode()");

this.code = code;

}

public String getName() {

LOGGER.debug("Inside getName()");

return name;

}

public void setName(String name) {

LOGGER.debug("Inside setName()");

this.name = name;

}

@Override

public String toString() {

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

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

public static void main(String[] args) {

displayCountry();

}

public static void displayCountry() {

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

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

LOGGER.debug("Country : {}", country.toString());

}

}

**Country.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="country" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

</beans>

**Output**

DEBUG com.cognizant.springlearn.Country - Inside Country Constructor.

DEBUG com.cognizant.springlearn.Country - Inside setCode()

DEBUG com.cognizant.springlearn.Country - Inside setName()

DEBUG com.cognizant.springlearn.Country - Inside getCode()

DEBUG com.cognizant.springlearn.Country - Inside getName()

DEBUG com.cognizant.springlearn.SpringLearnApplication - Country : Country [code=IN, name=India]

**SME detailing**

**Bean tag :**

**<bean id="country" class="com.cognizant.spring\_learn.Country">**

* Defines a spring-managed object (bean).
* “id” is the name used to reference the bean.
* “class” is the fully-qualified class name to instantiate.

**Property tag :**

**<property name= “code” value= “IN”/>**

* Injects the value “IN” into the “code” field using the setCode() method.

**ApplicationContext :** Spring’s central interface for accessing beans.

**ClassPathXMLApplicationContext :** Loads XML config from “src/main/resources” (classpath).

**Q. What exactly happens when context.getBean() is invoked?**

**- context.getBean(“country”, Country.class)**

- Loads the bean with id= “country”.

- Spring creates an object of type “Country”.

- It calls: 1. Constructor -> logs “Inside Country Constructor”.

2. setCode(“IN”)

3. setName(“India”)

- Returns the fully-configured “Country” object.