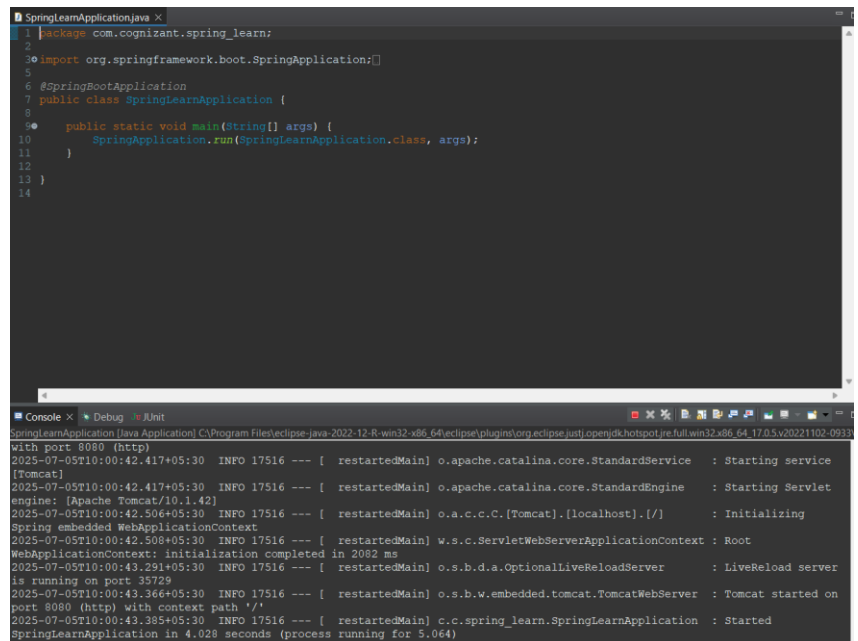
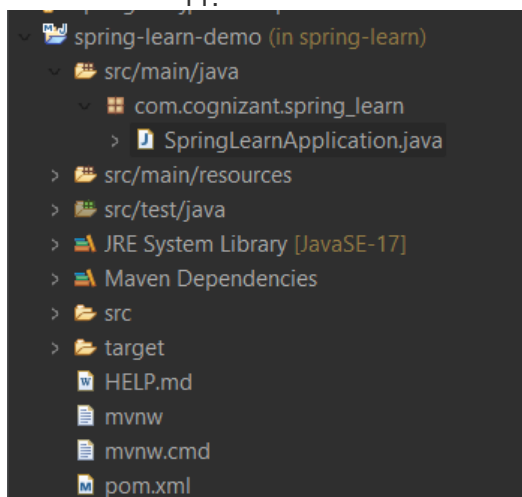


Hands on 1

Create a Spring Web Project using Maven

Follow steps below to create a project:

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.
- 11.



SME to walk through the following aspects related to the project created:

1. **src/main/java** - Folder with application code
2. **src/main/resources** - Folder for application configuration
3. **src/test/java** - Folder with code for testing the application
4. **SpringLearnApplication.java** - Walkthrough the main() method.

5. Purpose of `@SpringBootApplication` annotation
6. `pom.xml`
 1. Walkthrough all the configuration defined in XML file
 2. Open 'Dependency Hierarchy' and show the dependency tree.

Hands on 2

Spring Core – Load SimpleDateFormat from Spring Configuration XML

`SimpleDateFormat` with the pattern '`dd/MM/yyyy`' is created in multiple places of an application. To avoid creation of `SimpleDateFormat` in multiple places, define a bean in Spring XML Configuration file and retrieve the date.

Follow steps below to implement:

- Create spring configuration file `date-format.xml` in `src/main/resources` folder of 'spring-learn' project
- Open <https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-metadata>
- Copy the XML defined in the section of previous step URL and paste it into `date-format.xml`
- Define bean tag in the XML with for date format. Refer code below.

```
<?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="dateFormat" class="java.text.SimpleDateFormat">
        <constructor-arg value="dd/MM/yyyy" />
    </bean>

</beans>
```

- Create new method `displayDate()` in `SpringLearnApplication.java`

- In displayDate() method create the **ApplicationContext**. Refer code below:

```
ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");
```

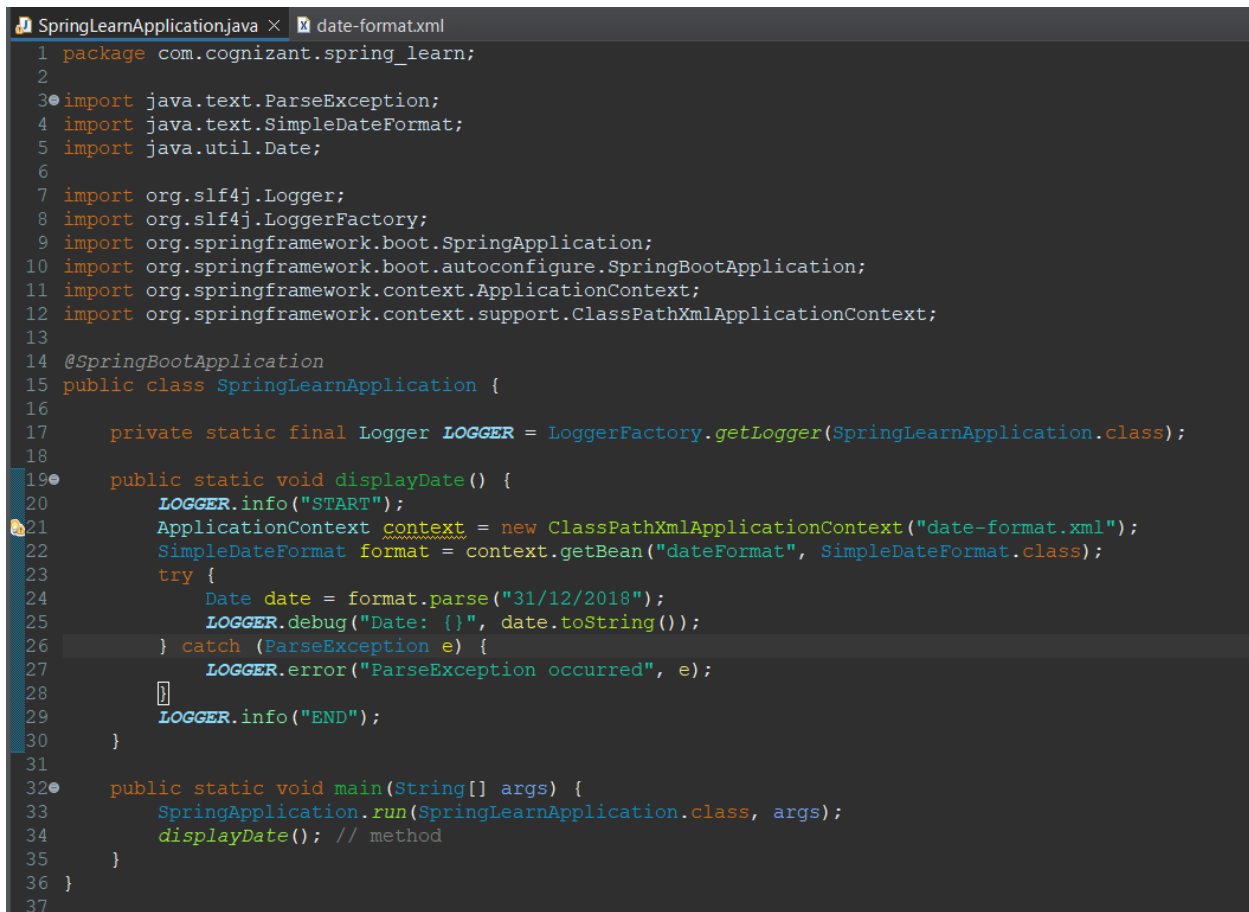
- Get the **dateFormat** using **getBean()** method. Refer code below.

```
SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);
```

- Using the format variable try to parse string '31/12/2018' to **Date** class and display the result using **System.out.println**.
- Run the application as 'Java Application' and check the result in console log output.

Troubleshooting Tips

If the tomcat port has a conflict and the server is not starting include the below property in **application.properties** file in **src/main/resources** folder.



```
SpringLearnApplication.java × date-format.xml
1 package com.cognizant.spring_learn;
2
3 import java.text.ParseException;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 import org.slf4j.Logger;
8 import org.slf4j.LoggerFactory;
9 import org.springframework.boot.SpringApplication;
10 import org.springframework.boot.autoconfigure.SpringBootApplication;
11 import org.springframework.context.ApplicationContext;
12 import org.springframework.context.support.ClassPathXmlApplicationContext;
13
14 @SpringBootApplication
15 public class SpringLearnApplication {
16
17     private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);
18
19     public static void displayDate() {
20         LOGGER.info("START");
21         ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");
22         SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);
23         try {
24             Date date = format.parse("31/12/2018");
25             LOGGER.debug("Date: {}", date.toString());
26         } catch (ParseException e) {
27             LOGGER.error("ParseException occurred", e);
28         }
29         LOGGER.info("END");
30     }
31
32     public static void main(String[] args) {
33         SpringApplication.run(SpringLearnApplication.class, args);
34         displayDate(); // method
35     }
36 }
37
```

Hands on 3

Spring Core - Incorporate Logging

Incorporate logging in the Spring Boot project created in previous hands on. Refer steps below:

- Create **application.properties** if not yet created in **src/main/resources** folder
- Add below lines in **application.properties**

```
logging.level.org.springframework=info
logging.level.com.cognizant.springlearn=debug
logging.pattern.console=%d{yyMMdd} | %d{HH:mm:ss.SSS} | %-20.20thread | %5p | %-25.25logger{25} | %25M | %m%n
```

- In **SpringLearnApplication.java** include the following imports:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

- Include the below static variable in **SpringLearnApplication.java**:

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

- Include info log on start and end of method. Debug log for displaying the date (refer code below)

```
public void displayDate() {
    LOGGER.info("START");
    //..
    LOGGER.debug(date);
    //..
    LOGGER.info("END");
}
```

IMPORTANT NOTE: Going forward all methods should incorporate logging as specified above. **Never** use `System.out.println()`.

```
SpringLearnApplication.java  date-format.xml  application.properties ×
1 spring.application.name=spring-learn
2 logging.level.org.springframework=info
3 logging.level.com.cognizant.springlearn=debug
4 logging.pattern.console=%d{yyMMdd} | %d{HH:mm:ss.SSS} | %-20.20thread | %5p | %-25.25logger{25} | %25M | %m%n
5 |
```

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.

Steps to implement

- Pick any one of your choice country to configure in Spring XML configuration named **country.xml**.
- Create a bean tag in spring configuration for country and set the property and values

```
<bean id="country" class="com.cognizant.springlearn.Country">
    <property name="code" value="IN" />
    <property name="name" value="India" />
</bean>
```

- Create **Country** class with following aspects:
 - Instance variables for **code** and **name**
 - Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “**Inside Country Constructor.**”

- Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.
 - Generate `toString()` method
- Create a method `displayCountry()` in `SpringLearnApplication.java`, which will read the country bean from spring configuration file and display the country details. `ClassPathXmlApplicationContext`, `ApplicationContext` and `context.getBean("beanId", Country.class)`. Refer sample code for `displayCountry()` method below.

```
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
Country country = (Country) context.getBean("country", Country.class);
LOGGER.debug("Country : {}", country.toString());
```

- Invoke `displayCountry()` method in `main()` method of `SpringLearnApplication.java`.
- Execute `main()` method and check the logs to find out which constructors and methods were invoked.

SME to provide more detailing about the following aspects:

- bean tag, id attribute, class attribute, property tag, name attribute, value attribute
- `ApplicationContext`, `ClassPathXmlApplicationContext`
- What exactly happens when `context.getBean()` is invoked

```
public static void displayCountry() {
    ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
    Country country = context.getBean("country", Country.class);
    LOGGER.debug("Country : {}", country.toString());
}
```

```
SpringLearnApplication.java  date-format.xml  application.properties  country.xml  Country.java
1 <beans xmlns="http://www.springframework.org/schema/beans"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.springframework.org/schema/beans
4     http://www.springframework.org/schema/beans/spring-beans.xsd">
5
6   <bean id="country" class="com.cognizant.springlearn.Country">
7     <property name="code" value="IN" />
8     <property name="name" value="India" />
9   </bean>
10 </beans>
11
```

```
SpringLearnApplication.java  date-format.xml  application.properties  country.xml  Country.java
1 package com.cognizant.spring_learn;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5
6 public class Country {
7
8     private static final Logger LOGGER = LoggerFactory.getLogger(Country.class);
9
10    private String code;
11    private String name;
12
13    public Country() {
14        LOGGER.debug("Inside Country Constructor.");
15    }
16
17    public String getCode() {
18        LOGGER.debug("Getting code");
19        return code;
20    }
21
22    public void setCode(String code) {
23        LOGGER.debug("Setting code");
24        this.code = code;
25    }
26
27    public String getName() {
28        LOGGER.debug("Getting name");
29        return name;
30    }
31
32    public void setName(String name) {
33        LOGGER.debug("Setting name");
34        this.name = name;
35    }
36
37    @Override
38    public String toString() {
39        return "Country [code=" + code + ", name=" + name + "];"
40    }
41 }
42
```

Spring Core – Demonstration of Singleton Scope and Prototype Scope

The Country bean done in the previous hands on will be used to demonstrate the scopes in Spring. Implement the steps below.

Follow steps below to demonstrate Singleton Scope

- Include a line in `displayCountry()` to get country bean reference one more time from the same application context. Only the third line of the below code snippet should be copied and pasted.

```
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
Country country = context.getBean("country", Country.class);
Country anotherCountry = context.getBean("country", Country.class);
```

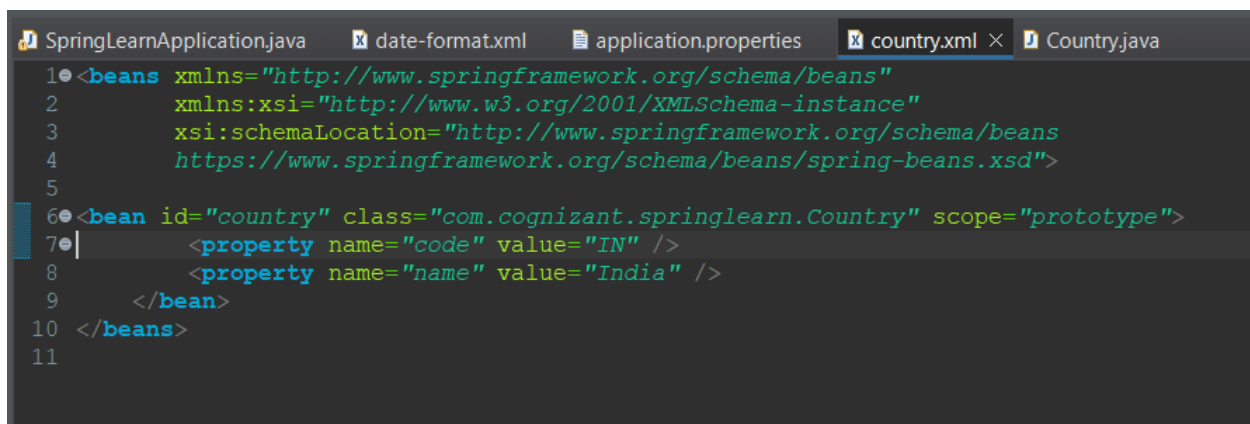
- The constructor will be called only once, which means that only one instance of Country bean is created

Follow steps below to demonstrate Prototype Scope

- Include `scope="prototype"` attribute in bean definition xml.

```
<bean id="country" class="com.cognizant.springlearn.Country" scope="prototype">
```

- Run the application
- Constructor will be called twice, which means that two instances of country is created.



```
SpringLearnApplication.java  date-format.xml  application.properties  country.xml  Country.java
1 <beans xmlns="http://www.springframework.org/schema/beans"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.springframework.org/schema/beans
4     https://www.springframework.org/schema/beans/spring-beans.xsd">
5
6 <bean id="country" class="com.cognizant.springlearn.Country" scope="prototype">
7   <property name="code" value="IN" />
8   <property name="name" value="India" />
9 </bean>
10 </beans>
11
```

Spring Core – Load list of countries from Spring Configuration XML

Our main objective was to retrieve the list of four countries for the airlines website. Refer steps below to get this incorporated.

- Create a separate bean for each of the four country in country.xml.
- Create an ArrayList of Country in country.xml. Refer code below.

```
<bean id="countryList" class="java.util.ArrayList">
    <constructor-arg>
        <list>
            <ref bean="in"></ref>
            <ref bean="us"></ref>
            <ref bean="de"></ref>
            <ref bean="jp"></ref>
        </list>
    </constructor-arg>
</bean>
```

- Include new method displayCountries() in SpringLearnApplication.java
- In displayCountries() read the country list created above
- Display the list of countries as debug log.

SME to provide detailing on below aspects:


- <list>
- <ref>
- bean attribute

IMPORTANT NOTE: Do not forget to include the start and end logs in this new method.


```

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.109 s -- in com.cognizant.spring_learn.SpringLearnApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ spring-learn ---
[INFO] Building jar: C:\Users\HP\Desktop\Spring-Learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.5.3:repackage (repackage) @ spring-learn ---
[INFO] Replacing main artifact C:\Users\HP\Desktop\Spring-Learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\HP\Desktop\Spring-Learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.4:install (default-install) @ spring-learn ---
[INFO] Installing C:\Users\HP\Desktop\Spring-Learn\spring-learn\pom.xml to C:\Users\HP\.m2\repository\com\cognizant\spring-learn\0.0.1-SNAPSHOT\spring-learn-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\HP\Desktop\Spring-Learn\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar to C:\Users\HP\.m2\repository\com\cognizant\spring-learn\0.0.1-SNAPSHOT\spring-learn-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 14.190 s
[INFO] Finished at: 2025-07-05T10:27:29+05:30
[INFO] -----

```



```

:: Spring Boot :: (v3.5.3)

250705|10:27:22.166|restartedMain|INFO|s.SpringLearnApplication|logStarting|Starting SpringLearnApplication using Java 17.0.5 with PID 12880 (C:\Users\HP\Desktop\Spring-Learn\spring-learn\target\classes started by HP in C:\Users\HP\Desktop\Spring-Learn\spring-learn)
250705|10:27:22.167|restartedMain|INFO|s.SpringLearnApplication|logStartupProfileInfo|No active profile set, falling back to 1 default profile: "default"
250705|10:27:22.534|restartedMain|INFO|s.b.w.e.t.TomcatWebServer|initialize|Tomcat initialized with port 8080 (http)
250705|10:27:22.535|restartedMain|INFO|o.a.c.c.c.StandardService|log|Starting service [Tomcat]
250705|10:27:22.536|restartedMain|INFO|o.a.c.c.c.core.StandardEngine|log|Starting Servlet engine: [Apache Tomcat/10.1.42]
250705|10:27:22.572|restartedMain|INFO|o.a.c.c.c.C.[.].[/]|log|Initializing Spring embedded WebApplicationContext
250705|10:27:22.574|restartedMain|INFO|bServerApplicationContext|prepareWebApplicationContext|Root WebApplicationContext: initialization completed in 403 ms
250705|10:27:22.680|restartedMain|INFO|OptionalLiveReloadServer|startServer|LiveReload server is running on port 35729
250705|10:27:22.695|restartedMain|INFO|s.b.w.e.t.TomcatWebServer|start|Tomcat started on port 8080 (http) with context path '/'
250705|10:27:22.701|restartedMain|INFO|s.SpringLearnApplication|logStarted|Started SpringLearnApplication in 0.597 seconds (process running for 116.964)
250705|10:27:22.703|restartedMain|INFO|ationDeltaLoggingListener|onApplicationEvent|Condition evaluation unchanged
250705|10:27:22.704|restartedMain|INFO|s.SpringLearnApplication|displayDate|START
250705|10:27:22.730|restartedMain|INFO|s.SpringLearnApplication|displayDate|END
250705|10:27:22.731|restartedMain|INFO|s.SpringLearnApplication|displayCountry|START
250705|10:27:22.766|restartedMain|INFO|s.SpringLearnApplication|displayCountry|END
250705|10:27:22.766|restartedMain|INFO|s.SpringLearnApplication|displayCountries|START
250705|10:27:22.791|restartedMain|INFO|s.SpringLearnApplication|displayCountries|END

```

Submitted By:

Name : Lingaraj Nayak

Superset ID : 6387607