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.

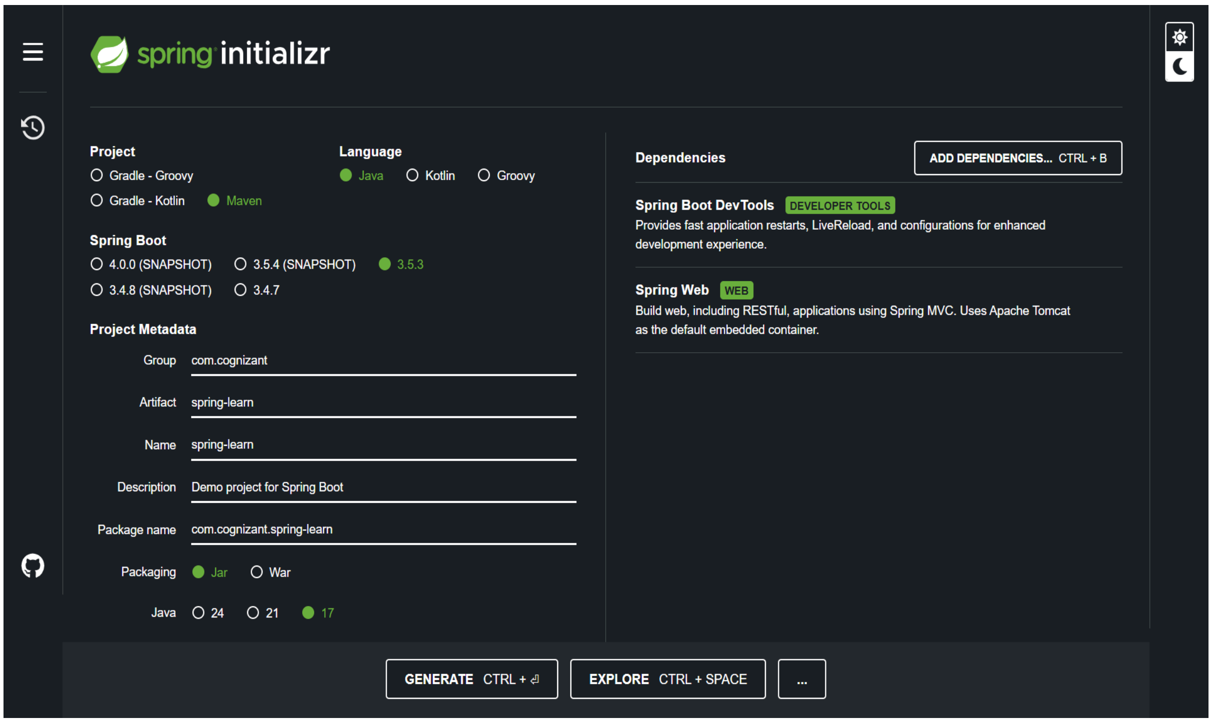
Solution

`SpringLearnApplication.java`

```java
package com.cognizant.spring_learn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringLearnApplication {
	private static final Logger logger =
LoggerFactory.getLogger(SpringLearnApplication.class);

	public static void main(String[] args) {
		SpringApplication.run(SpringLearnApplication.class, args);
		logger.info("Hello Spring REST!");
	}
}
```

Output Screenshots

Setting up the project using Spring Intializr



Project
○ Gradle - Groovy
○ Gradle - Kotlin
● Maven

Language
● Java
○ Kotlin
○ Groovy

Spring Boot
○ 4.0.0 (SNAPSHOT)
○ 3.5.4 (SNAPSHOT)
● 3.5.3
○ 3.4.8 (SNAPSHOT)
○ 3.4.7

Project Metadata

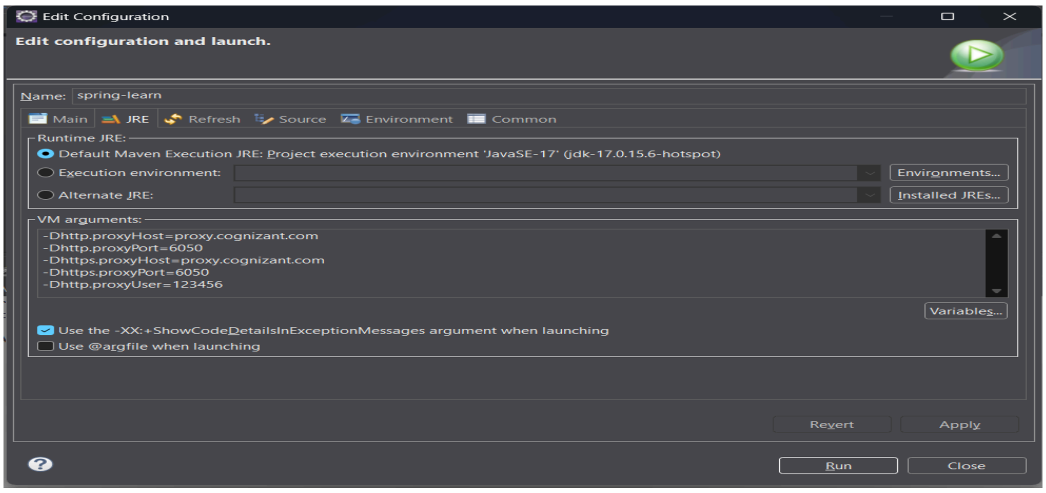
Group	com.cognizant
Artifact	spring-learn
Name	spring-learn
Description	Demo project for Spring Boot
Package name	com.cognizant.spring-learn
Packaging	● Jar ○ War
Java	○ 24 ○ 21 ● 17

Dependencies

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Building the project



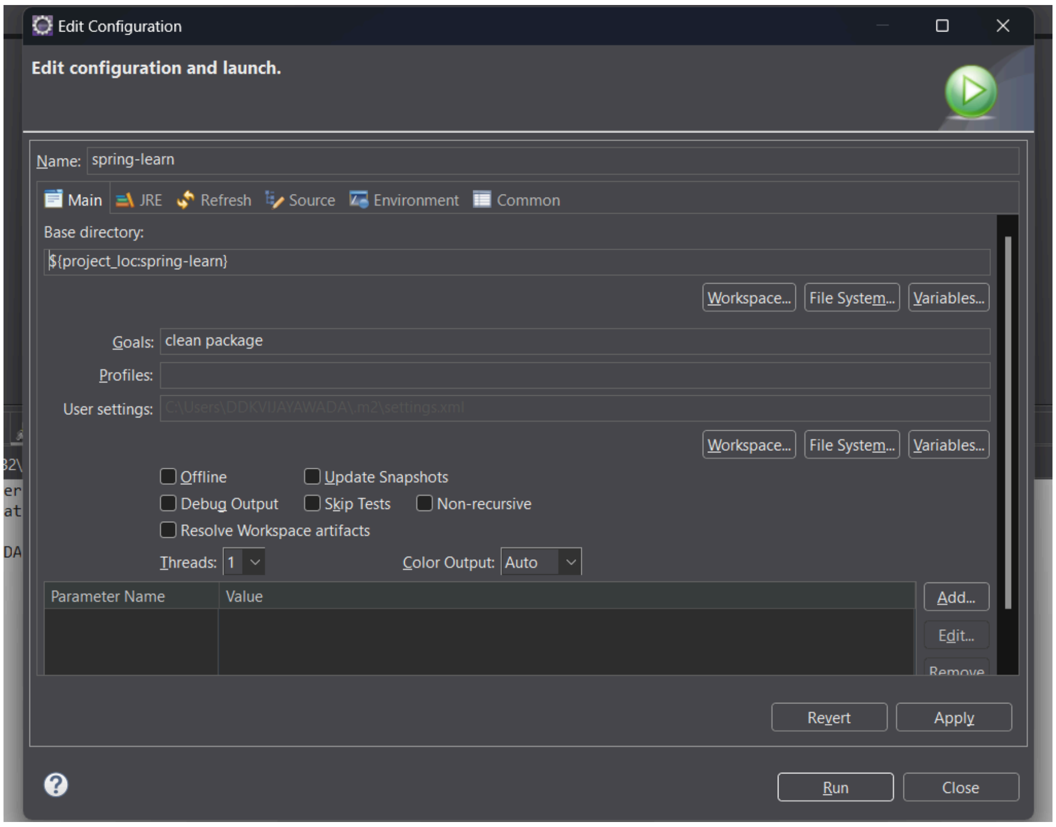
Name: spring-learn

Runtime JRE:
[✓ Default Maven Execution JRE: Project execution environment 'JavaSE-17' (jdk-17.0.15.6-hotspot)]
Execution environment:
Alternate JRE:

VM arguments:

```
-Dhttp.proxyHost=proxy.cognizant.com
-Dhttp.proxyPort=6050
-Dhttps.proxyHost=proxy.cognizant.com
-Dhttps.proxyPort=6050
-Dhttp.proxyUser=123456
```

[✓] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching
[ ] Use @argfile when launching



Name: spring-learn

Base directory: ${project_loc:spring-learn}

Goals: clean package

Profiles:

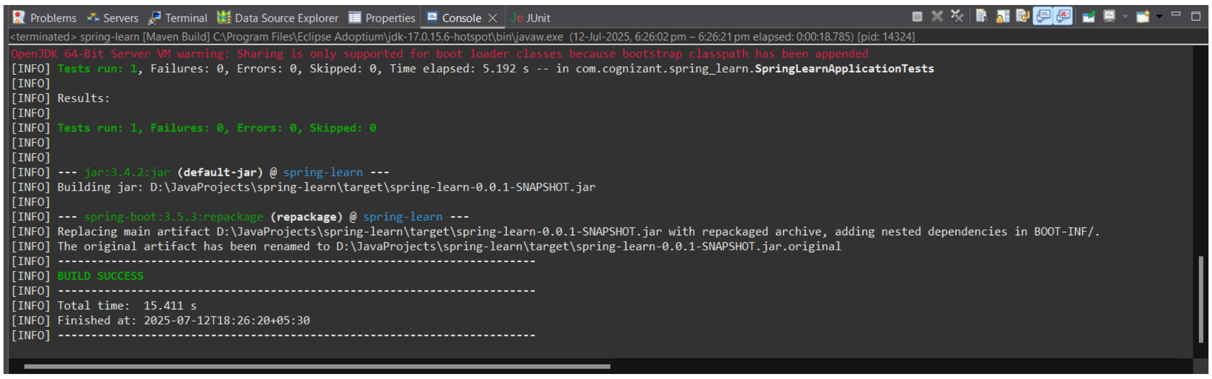
User settings: C:\Users\DDKVIJAYAWADA\.m2\settings.xml

[ ] Offline [ ] Update Snapshots
[ ] Debug Output [ ] Skip Tests [ ] Non-recursive
[ ] Resolve Workspace artifacts

Threads: 1 Color Output: Auto

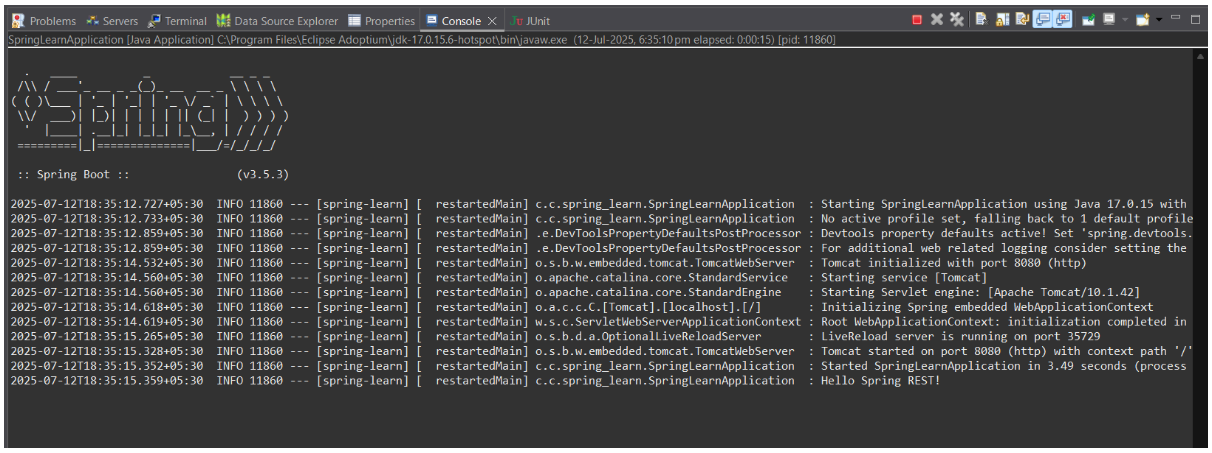
Parameter Name	Value

Project successfully built



```
<terminated> spring-learn [Maven Build] C:\Program Files\Eclipse Adoptium\jdk-17.0.15.6-hotspot\bin\javaw.exe  (12-Jul-2025, 6:26:02 pm – 6:26:21 pm elapsed: 0:00:18.785) [pid: 14324]
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.192 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: D:\JavaProjects\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 D:\JavaProjects\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 D:\JavaProjects\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar.original
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  15.411 s
[INFO] Finished at: 2025-07-12T18:26:20+05:30
[INFO] ------------------------------------------------------------------------
```

Output of SpringLearnApplication.java



```
SpringLearnApplication [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.15.6-hotspot\bin\javaw.exe  (12-Jul-2025, 6:35:10 pm elapsed: 0:00:15) [pid: 11860]

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-12T18:35:12.727+05:30  INFO 11860 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Starting SpringLearnApplication using Java 17.0.15 with
2025-07-12T18:35:12.733+05:30  INFO 11860 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : No active profile set, falling back to 1 default profile
2025-07-12T18:35:12.859+05:30  INFO 11860 --- [spring-learn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.
2025-07-12T18:35:12.859+05:30  INFO 11860 --- [spring-learn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the
2025-07-12T18:35:14.532+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-12T18:35:14.560+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-12T18:35:14.560+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-12T18:35:14.618+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-12T18:35:14.619+05:30  INFO 11860 --- [spring-learn] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2025-07-12T18:35:15.265+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-12T18:35:15.328+05:30  INFO 11860 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-07-12T18:35:15.352+05:30  INFO 11860 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 3.49 seconds (process
2025-07-12T18:35:15.359+05:30  INFO 11860 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Hello Spring REST!
```

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.

Solution

`application.properties`

```
spring.application.name=spring-learn
server.port=9095
logging.level.com.cognizant.spring_learn=DEBUG
```

`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
        http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="country" class="com.cognizant.spring_learn.Country">
       <property name="countryCode" value="JP" />
       <property name="countryName" value="Japan" />
   </bean>
</beans>
```

`Country.java`

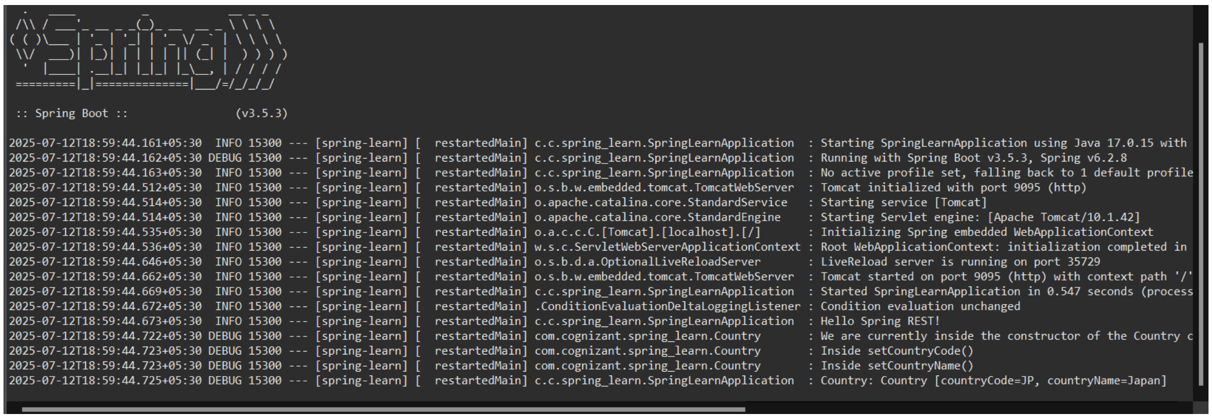
```
package com.cognizant.spring_learn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class Country {
	private static final Logger logger =
LoggerFactory.getLogger(Country.class);

	private String countryCode;
	private String countryName;
	public Country() {
		logger.debug("We are currently inside the constructor of the
Country class!");
	}
	public String getCountryCode() {
		logger.debug("Inside getCountryCode()");
		return countryCode;
	}
	public void setCountryCode(String countryCode) {
		logger.debug("Inside setCountryCode()");
		this.countryCode = countryCode;
	}
	public String getCountryName() {
		logger.debug("Inside getCountryName()");
		return countryName;
	}
	public void setCountryName(String countryName) {
		logger.debug("Inside setCountryName()");
		this.countryName = countryName;
	}
	@Override
	public String toString() {
		return "Country [countryCode=" + countryCode + ", countryName=" +
countryName + "]";
	}



}
```

SpringLearnApplication.java

```
package com.cognizant.spring_learn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
@SpringBootApplication
public class SpringLearnApplication {
	private static final Logger logger =
LoggerFactory.getLogger(SpringLearnApplication.class);

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

```java
		logger.debug("Country: {}", country.toString());
	}

	public static void main(String[] args) {
		SpringApplication.run(SpringLearnApplication.class, args);
		logger.info("Hello Spring REST!");
		displayCountry();
	}
}
```

Output Screenshots



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-12T18:59:44.161+05:30  INFO 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Starting SpringLearnApplication using Java 17.0.15 with
2025-07-12T18:59:44.162+05:30 DEBUG 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Running with Spring Boot v3.5.3, Spring v6.2.8
2025-07-12T18:59:44.163+05:30  INFO 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : No active profile set, falling back to 1 default profile
2025-07-12T18:59:44.512+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 9095 (http)
2025-07-12T18:59:44.514+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-12T18:59:44.514+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-12T18:59:44.535+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-12T18:59:44.536+05:30  INFO 15300 --- [spring-learn] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2025-07-12T18:59:44.646+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-12T18:59:44.662+05:30  INFO 15300 --- [spring-learn] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 9095 (http) with context path '/'
2025-07-12T18:59:44.669+05:30  INFO 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 0.547 seconds (process
2025-07-12T18:59:44.672+05:30  INFO 15300 --- [spring-learn] [  restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
2025-07-12T18:59:44.673+05:30  INFO 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Hello Spring REST!
2025-07-12T18:59:44.722+05:30 DEBUG 15300 --- [spring-learn] [  restartedMain] com.cognizant.spring_learn.Country       : We are currently inside the constructor of the Country c
2025-07-12T18:59:44.723+05:30 DEBUG 15300 --- [spring-learn] [  restartedMain] com.cognizant.spring_learn.Country       : Inside setCountryCode()
2025-07-12T18:59:44.723+05:30 DEBUG 15300 --- [spring-learn] [  restartedMain] com.cognizant.spring_learn.Country       : Inside setCountryName()
2025-07-12T18:59:44.725+05:30 DEBUG 15300 --- [spring-learn] [  restartedMain] c.c.spring_learn.SpringLearnApplication  : Country: Country [countryCode=JP, countryName=Japan]
```

Hello World RESTful Web Service

Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:

Method: GET
URL: /hello
Controller: com.cognizant.spring-learn.controller.HelloController
Method Signature: public String sayHello()
Method Implementation: return hard coded string "Hello World!!"
Sample Request: http://localhost:8083/hello
Sample Response: Hello World!!

IMPORTANT NOTE: Don't forget to include start and end log in the sayHello() method.

Try the URL http://localhost:8083/hello in both chrome browser and postman.

Solution

`application.properties`

```
spring.application.name=spring-learn
server.port=8083
logging.level.com.cognizant.spring_learn=DEBUG
```

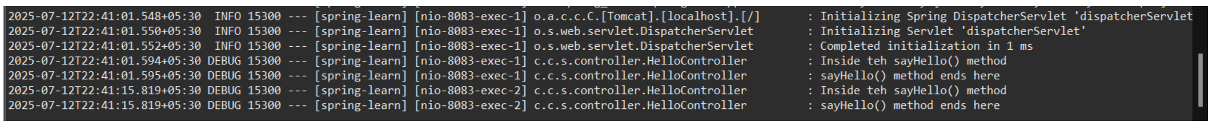
`HelloController.java`

```
package com.cognizant.spring_learn.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
```

```java
public class HelloController {
	private static final Logger logger =
LoggerFactory.getLogger(HelloController.class);

	@GetMapping("/hello")
	public String sayHello() {
		logger.debug("Inside teh sayHello() method");
		String response = "Hello Spring REST!!";
		logger.debug("sayHello() method ends here");
		return response;
	}
}
```

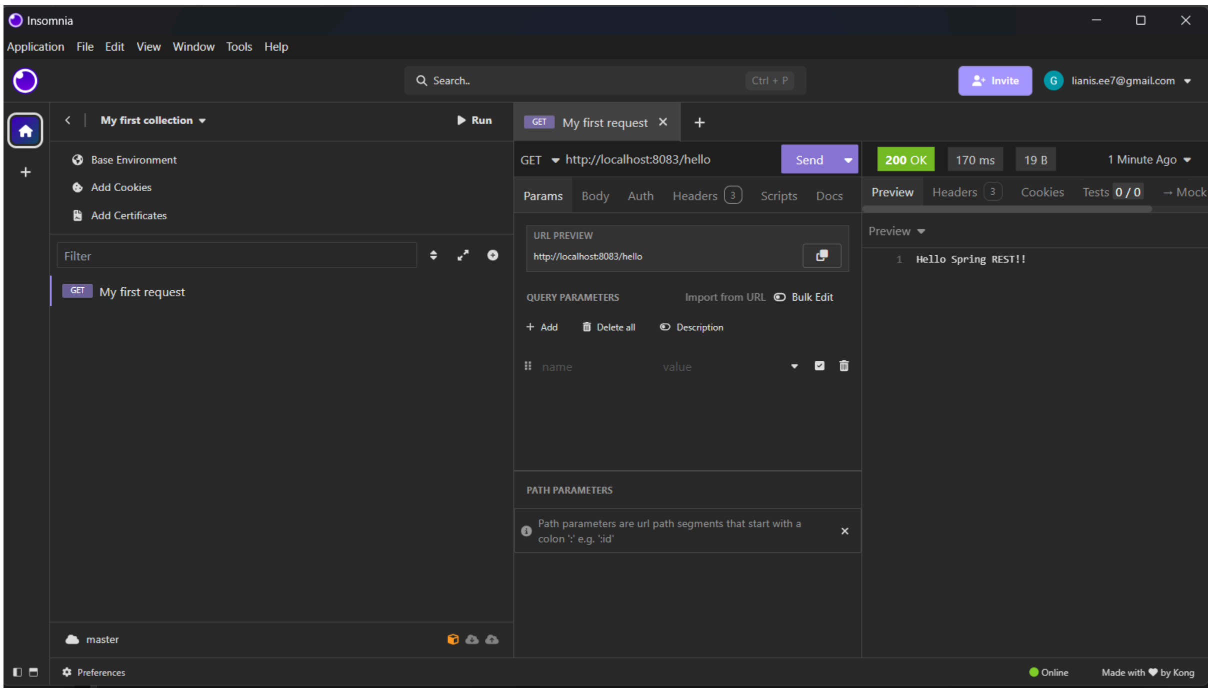
Output Screenshots

Debug logs



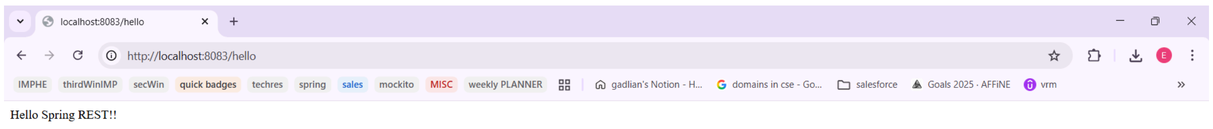
```
2025-07-12T22:41:01.548+05:30  INFO 15300 --- [spring-learn] [nio-8083-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet
2025-07-12T22:41:01.550+05:30  INFO 15300 --- [spring-learn] [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-12T22:41:01.552+05:30  INFO 15300 --- [spring-learn] [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2025-07-12T22:41:01.594+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-1] c.c.s.controller.HelloController         : Inside teh sayHello() method
2025-07-12T22:41:01.595+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-1] c.c.s.controller.HelloController         : sayHello() method ends here
2025-07-12T22:41:15.819+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-2] c.c.s.controller.HelloController         : Inside teh sayHello() method
2025-07-12T22:41:15.819+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-2] c.c.s.controller.HelloController         : sayHello() method ends here
```

Sending request and receiving response in Insomnia


Insomnia
Application File Edit View Window Tools Help
Search.. Ctrl + P
Invite
lianis.ee7@gmail.com
My first collection
Run
Base Environment
Add Cookies
Add Certificates
Filter
GET My first request
master
Preferences
GET My first request
GET http://localhost:8083/hello
Send
Params Body Auth Headers 3 Scripts Docs
URL PREVIEW
http://localhost:8083/hello
QUERY PARAMETERS
Import from URL Bulk Edit
Add Delete all Description
name value
PATH PARAMETERS
Path parameters are url path segments that start with a colon ':' e.g. ':id'
200 OK 170 ms 19 B 1 Minute Ago
Preview Headers 3 Cookies Tests 0 / 0 → Mock
Preview
Online
Made with ♥ by Kong

```
1  Hello Spring REST!!
```

Sending request and receiving response in Chrome Browser



```
Hello Spring REST!!
```

REST - Country Web Service

Write a REST service that returns India country details in the earlier created spring learn application.

URL: /country
Controller: com.cognizant.spring-learn.controller.CountryController
Method Annotation: @RequestMapping
Method Name: getCountryIndia()
Method Implementation: Load India bean from spring xml configuration and return
Sample Request: http://localhost:8083/country
Sample Response:

```
{

  "code": "IN",

  "name": "India"

}
```

Solution

application.properties

```
spring.application.name=spring-learn
server.port=8083
logging.level.com.cognizant.spring_learn=DEBUG
```

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
        http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="country" class="com.cognizant.spring_learn.Country">
       <property name="countryCode" value="IN" />
       <property name="countryName" value="India" />
   </bean>

<bean id="countryJapan" class="com.cognizant.spring_learn.Country">
       <property name="countryCode" value="JP" />
```

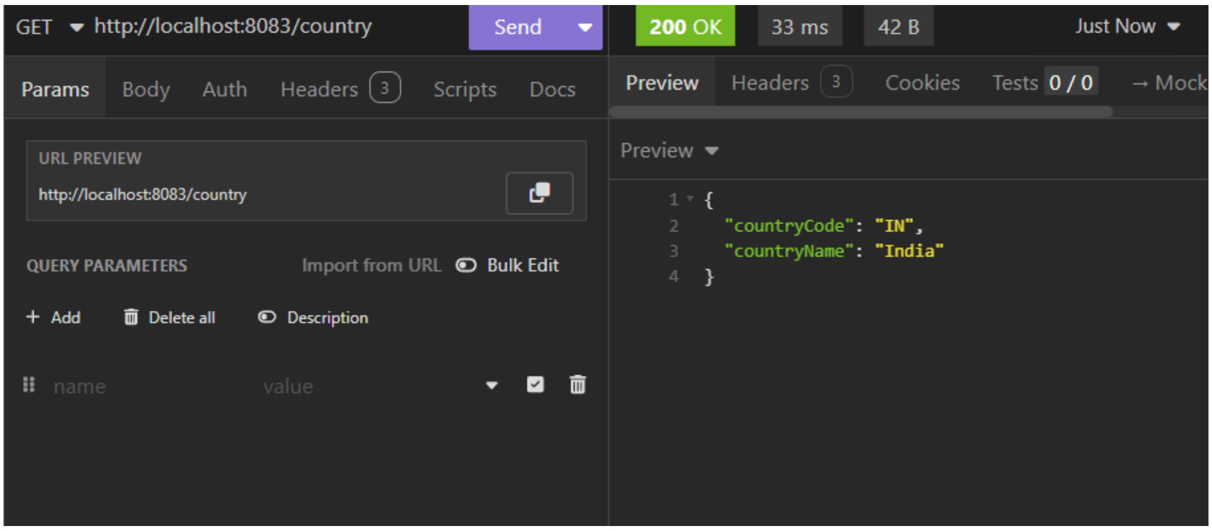
```xml
        <property name="countryName" value="Japan" />
    </bean>
</beans>
```

CountryController.java

```java
package com.cognizant.spring_learn.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.cognizant.spring_learn.Country;
@RestController
public class CountryController {
	private static final Logger logger = 
LoggerFactory.getLogger(CountryController.class);
	ApplicationContext context = new 
ClassPathXmlApplicationContext("country.xml");

	@RequestMapping("/country")
	public Country getCountryIndia() {
		logger.debug("Inside the method getCountryIndia()");
		Country india = context.getBean("country", Country.class);
		logger.debug("End of getCountryIndia() method");
		return india;
	}

	@RequestMapping("/country/japan")
	public Country getCountryJapan() {
		logger.debug("Inside the method to fetch Japan details!");
		Country japan = context.getBean("countryJapan", Country.class);
		logger.debug("End of getCountryJapan()");
		return japan;
	}
}
```

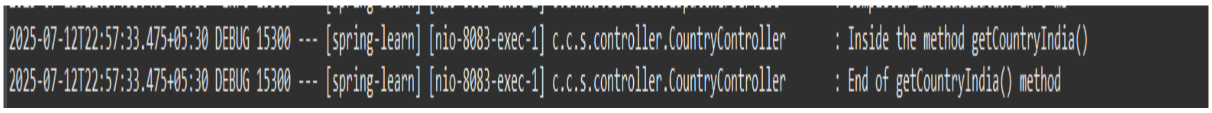
Output Screenshots

Get “/country” in Insomnia



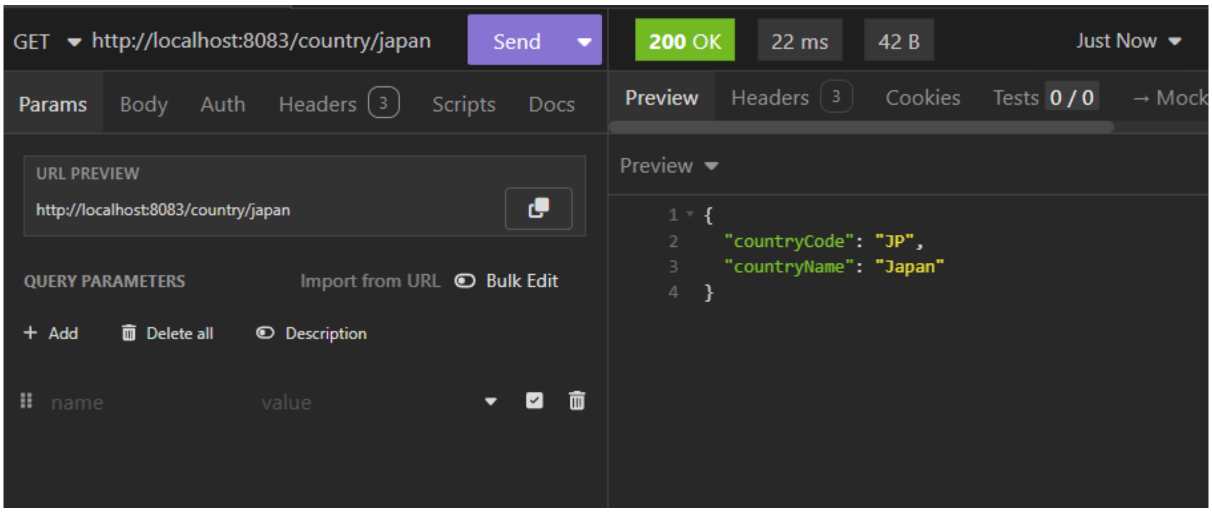
```
{
  "countryCode": "IN",
  "countryName": "India"
}
```

Logs of get http://localhost:8083/country



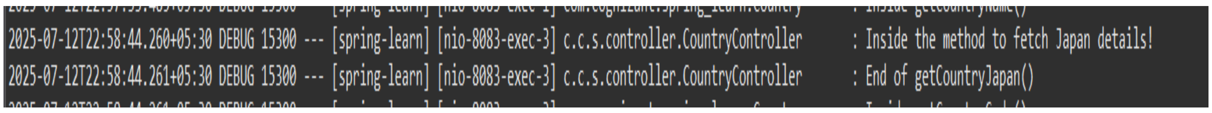
```
2025-07-12T22:57:33.475+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-1] c.c.s.controller.CountryController       : Inside the method getCountryIndia()
2025-07-12T22:57:33.475+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-1] c.c.s.controller.CountryController       : End of getCountryIndia() method
```

Get “/country/japan” in Insomnia



```
{
  "countryCode": "JP",
  "countryName": "Japan"
}
```

Logs of get http://localhost:8083/country/japan



```
2025-07-12T22:58:44.260+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-3] c.c.s.controller.CountryController       : Inside the method to fetch Japan details!
2025-07-12T22:58:44.261+05:30 DEBUG 15300 --- [spring-learn] [nio-8083-exec-3] c.c.s.controller.CountryController       : End of getCountryJapan()
```

REST - Get country based on country code

Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

Controller:
com.cognizant.spring-learn.controller.CountryController
Method Annotation: @GetMapping("/countries/{code}")
Method Name: getCountry(String code)
Method Implemetation: Invoke countryService.getCountry(code)
Service Method:
com.cognizant.spring-learn.service.CountryService.getCountry(String code)

Service Method Implementation:

- Get the country code using @PathVariable
- Get country list from country.xml
- Iterate through the country list
- Make a case insensitive matching of country code and return the country.
- Lambda expression can also be used instead of iterating the country list

Sample Request: http://localhost:8083/country/in

Sample Response:

```
{
 "code": "IN",
 "name": "India"
}
```

Solution

application.properties

```
spring.application.name=spring-learn
server.port=8083
logging.level.com.cognizant.spring_learn=DEBUG
```

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
        http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="countryIndia" class="com.cognizant.spring_learn.Country">
       <property name="countryCode" value="IN" />
       <property name="countryName" value="India" />
   </bean>

<bean id="countryJapan" class="com.cognizant.spring_learn.Country">
       <property name="countryCode" value="JP" />
       <property name="countryName" value="Japan" />
   </bean>
<bean id="countryList" class="java.util.ArrayList">
       <constructor-arg>
           <list>
               <ref bean="countryIndia"/>
               <ref bean="countryJapan"/>
           </list>
       </constructor-arg>
 </bean>
</beans>
```

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 countryCode;
      private String countryName;
```

```
	public Country() {
		logger.debug("We are currently inside the constructor of the 
Country class!");
	}
	public String getCountryCode() {
		logger.debug("Inside getCountryCode()");
		return countryCode;
	}
	public void setCountryCode(String countryCode) {
		logger.debug("Inside setCountryCode()");
		this.countryCode = countryCode;
	}
	public String getCountryName() {
		logger.debug("Inside getCountryName()");
		return countryName;
	}
	public void setCountryName(String countryName) {
		logger.debug("Inside setCountryName()");
		this.countryName = countryName;
	}
	@Override
	public String toString() {
		return "Country [countryCode=" + countryCode + ", countryName=" + 
countryName + "]";
	}



}
```

CountryService.java

```
package com.cognizant.spring_learn.service;
import com.cognizant.spring_learn.Country;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class CountryService {
	private static final Logger logger = 
LoggerFactory.getLogger(CountryService.class);

	public Country getCountry(String code) {

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

```
	List <Country> countryList = context.getBean("countryList", List.class);

	return countryList.stream().filter(c ->
c.getCountryCode().equalsIgnoreCase(code)).findFirst().orElse(null);
	}
}
```

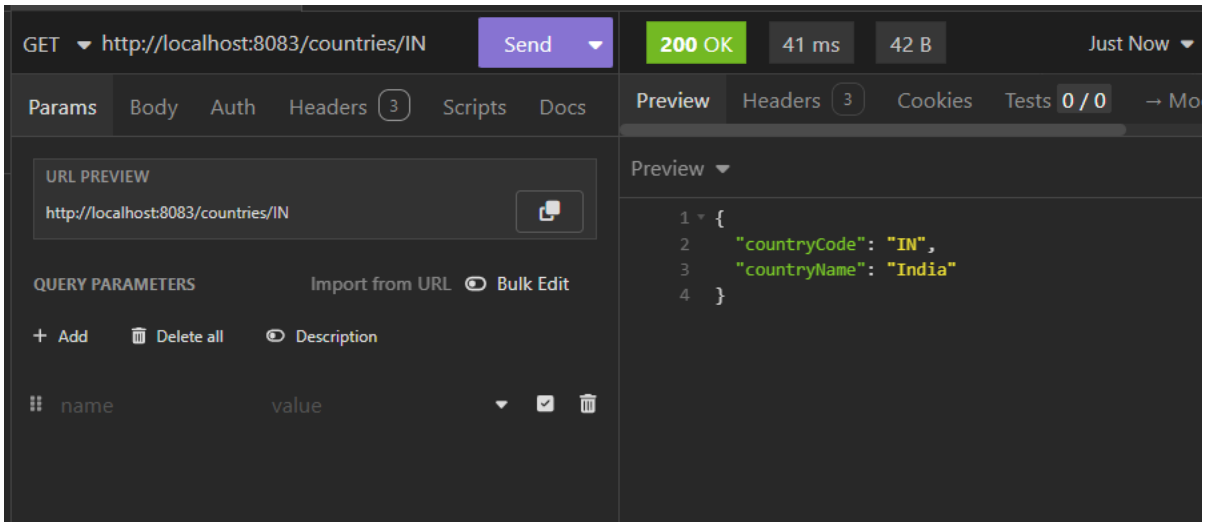
CountryController.java

```
package com.cognizant.spring_learn.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.cognizant.spring_learn.Country;
import com.cognizant.spring_learn.service.CountryService;
@RestController
public class CountryController {
	private static final Logger logger =
LoggerFactory.getLogger(CountryController.class);
	ApplicationContext context = new
ClassPathXmlApplicationContext("country.xml");

	@RequestMapping("/country")
	public Country getCountryIndia() {
		logger.debug("Inside the method getCountryIndia()");
		Country india = context.getBean("country", Country.class);
		logger.debug("End of getCountryIndia() method");
		return india;
	}

	@RequestMapping("/country/japan")
	public Country getCountryJapan() {
		logger.debug("Inside the method to fetch Japan details!");
		Country japan = context.getBean("countryJapan", Country.class);
		logger.debug("End of getCountryJapan()");
		return japan;
	}

	@Autowired
	private CountryService countryservice;
	@GetMapping("/countries/{code}")
	public Country getCountry(@PathVariable String code) {
```

```
            logger.info("Start getCountry()");
            Country c = countryservice.getCountry(code);
            return c;
        }
    }
```

Output Screenshots

Output of get http://localhost:8083/countries/IN

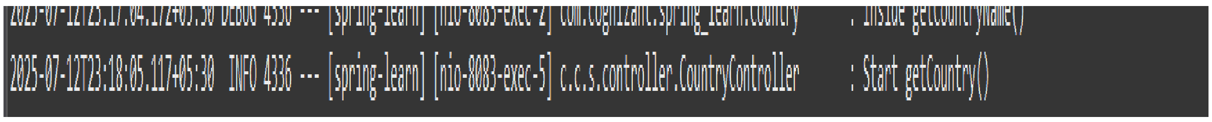


GET http://localhost:8083/countries/IN

URL PREVIEW: http://localhost:8083/countries/IN

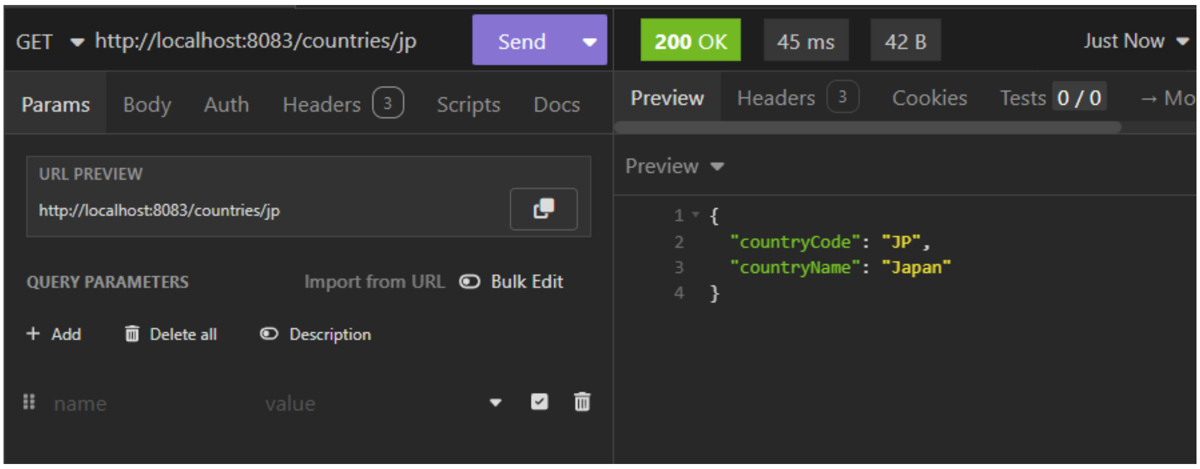
```
{
  "countryCode": "IN",
  "countryName": "India"
}
```

Logs Output



```
2025-07-12T23:17:04.172+05:30 DEBUG 4336 --- [spring-learn] [nio-8083-exec-2] com.cognizant.spring_learn.Country        : Inside getCountryName()
2025-07-12T23:18:05.117+05:30  INFO 4336 --- [spring-learn] [nio-8083-exec-5] c.c.s.controller.CountryController       : Start getCountry()
```

Output of get http://localhost:8083/countries/jp



GET http://localhost:8083/countries/jp

URL PREVIEW: http://localhost:8083/countries/jp

```
{
  "countryCode": "JP",
  "countryName": "Japan"
}
```

Create authentication service that returns JWT

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.

Request

```
curl -s -u user:pwd http://localhost:8090/authenticate
```

Response

```
{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJle
HAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}
```

Solution

Output Screenshots