HTTP Request Response

To get a granular level of details about HTTP Request and Response, follow the steps below:

- Open the link https://tools.ietf.org/html/rfc7230 in browser. This document contains the standard definition for HTTP request response.
- Refer sample HTTP request and response in page number 7. This is the actual bytes of data that is transferred between the browser and server.
- Specific details about the request and response:
    - Request

```
GET /hello.txt HTTP/1.1

User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3

Host: www.example.com

Accept-Language: en, mi
```

- Line 1 contains:
    - Method type - GET
    - Resource - /hello.txt
    - HTTP Version - HTTP/1.1
- Line 2 contains the details about the client
- Line 3 contains the server that will respond to this request
- The URL given in the browser is broken into Resource and Host in the HTTP Request

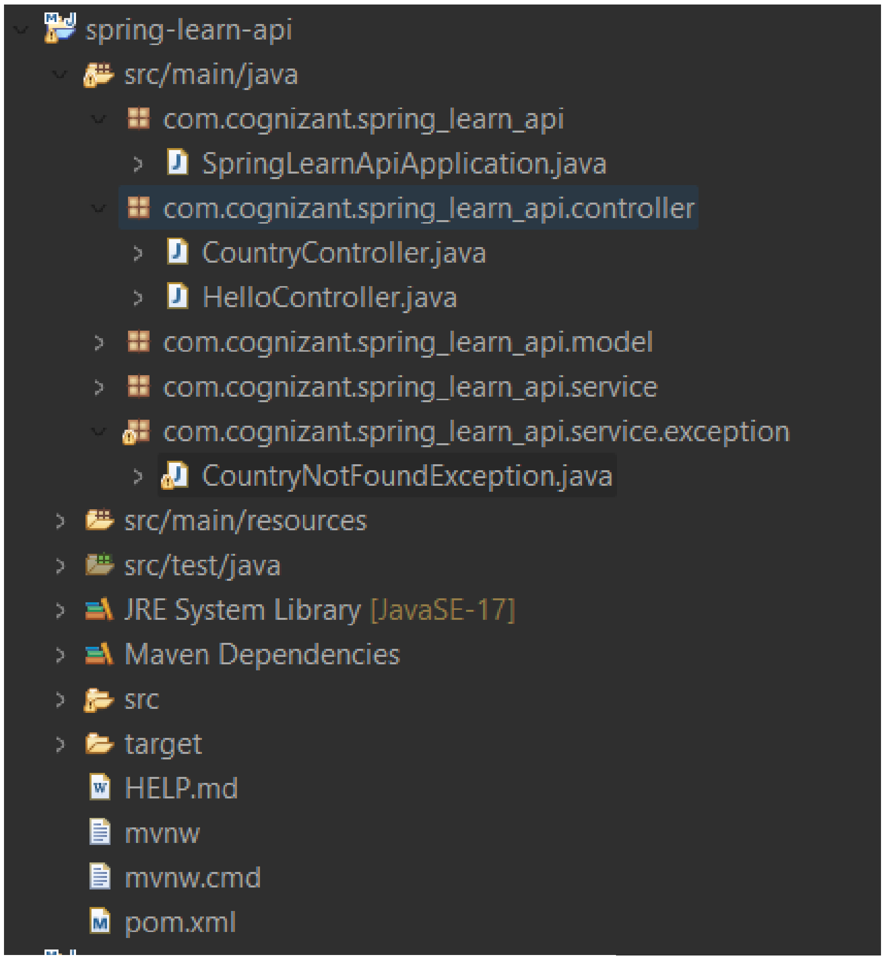
- Response

```
HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Accept-Ranges: bytes

Content-Length: 51

Vary: Accept-Encoding
```

```
Content-Type: text/plain


Hello World! My payload includes a trailing CRLF.
```

- Line 1
  - HTTP Version - HTTP/1.1
  - Response Status - 200 (this is means the request is responded successfully)
  - Response Message - Contains the response message
- Line 2 - Date of request
- Line 9 - Type of content returned. There is a list of predefined Content-Types. Based on Content-Type browser decides how the conent has to be visually displayed. Few examples below:
  - text/plain - Text content
  - text/html - HTML Document
  - application/json - JSON content
  - image/png - Image content of type PNG
- Last line contains the content of the resource.
  - In case of text/html, this will contain the HTML tags
  - In case of application/json, this will contain the JSON response
  - In case of image/png, this will contain the bytes to render the image

- To view the request and response details in browser, follow the steps below:
  - Open Chrome Browser
  - Press F12 to open the Developer Tools
  - Go to 'Network' table in Developer Tools
  - Open google search website in this browser window
  - Click on the first link available in the 'Network' tab
  - A new window will open in the right hands side. Observe the following details:
    - It will contain 3 sections. The data displayed will be similar to the HTTP request, response given above.
      - General
      - Response Headers
      - Request Headers


```
spring-learn-api
  src/main/java
    com.cognizant.spring_learn_api
      SpringLearnApiApplication.java
    com.cognizant.spring_learn_api.controller
      CountryController.java
      HelloController.java
    com.cognizant.spring_learn_api.model
    com.cognizant.spring_learn_api.service
    com.cognizant.spring_learn_api.service.exception
      CountryNotFoundException.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

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!!

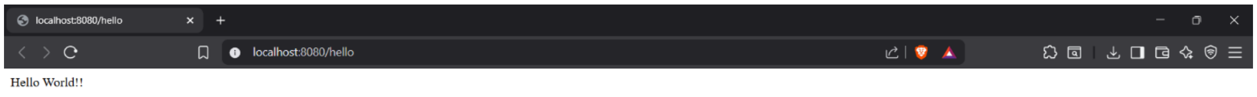
IMPORTANT NOTE: Don't forget to include start and end log in the sayHello() method.

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

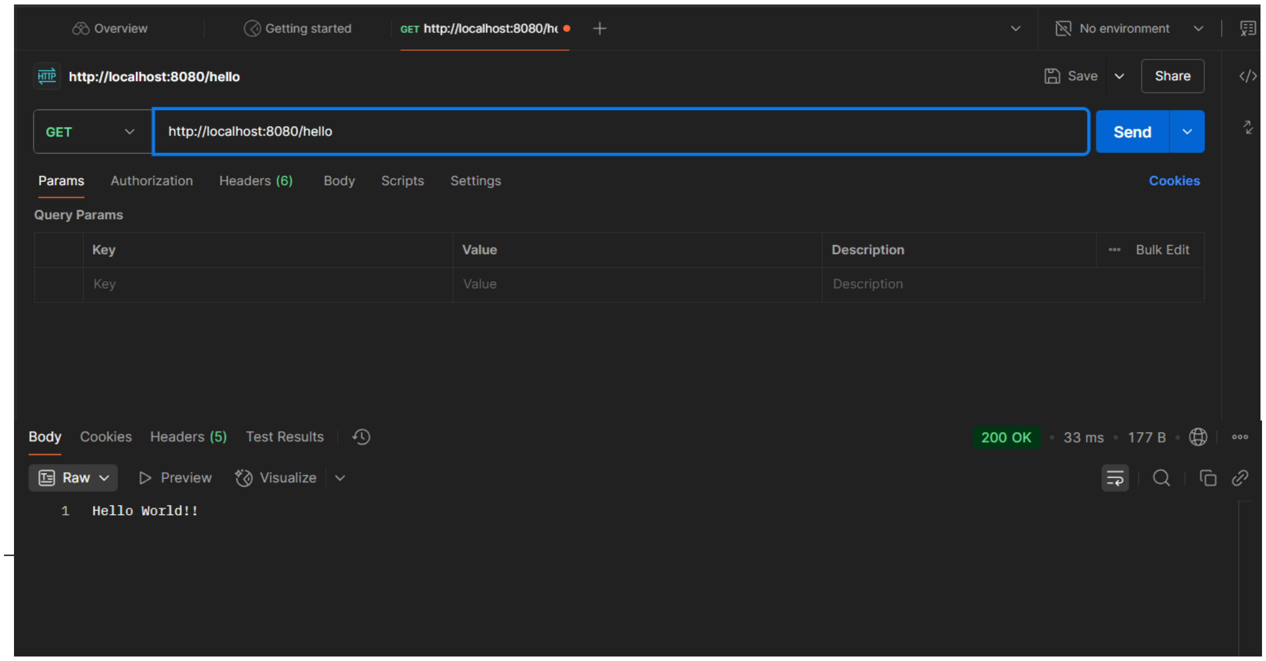
SME to explain the following aspects:

- In network tab of developer tools show the HTTP header details received

- In postman click on "Headers" tab to view the HTTP header details received



Hello World!!



GET http://localhost:8080/hello

Key	Value	Description
Key	Value	Description

```
Hello World!!
```

REST - Country Web Service

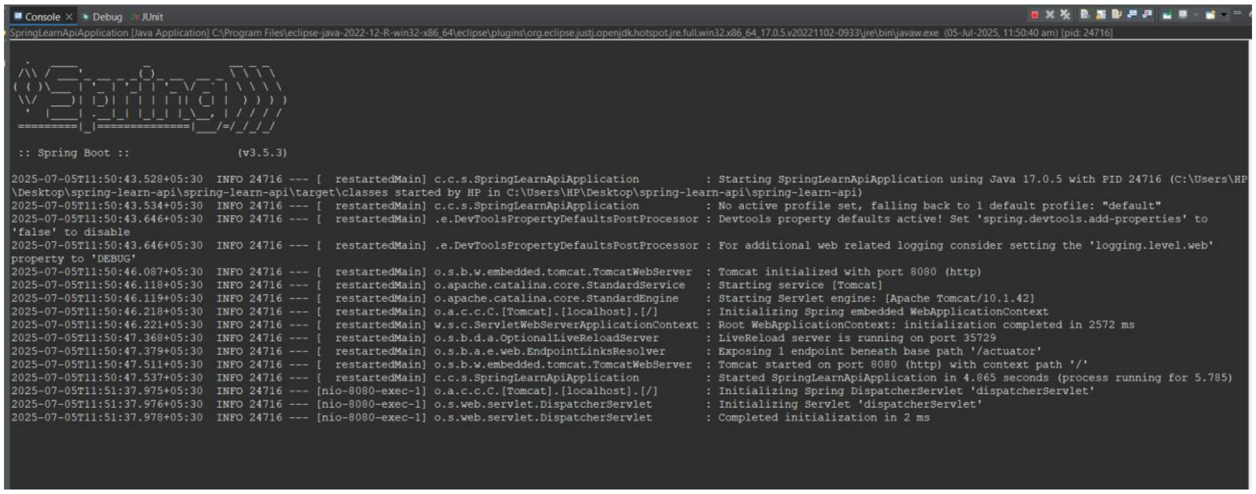
Write a REST service that returns India country details in the earlier created spring learn application.

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"
}
```

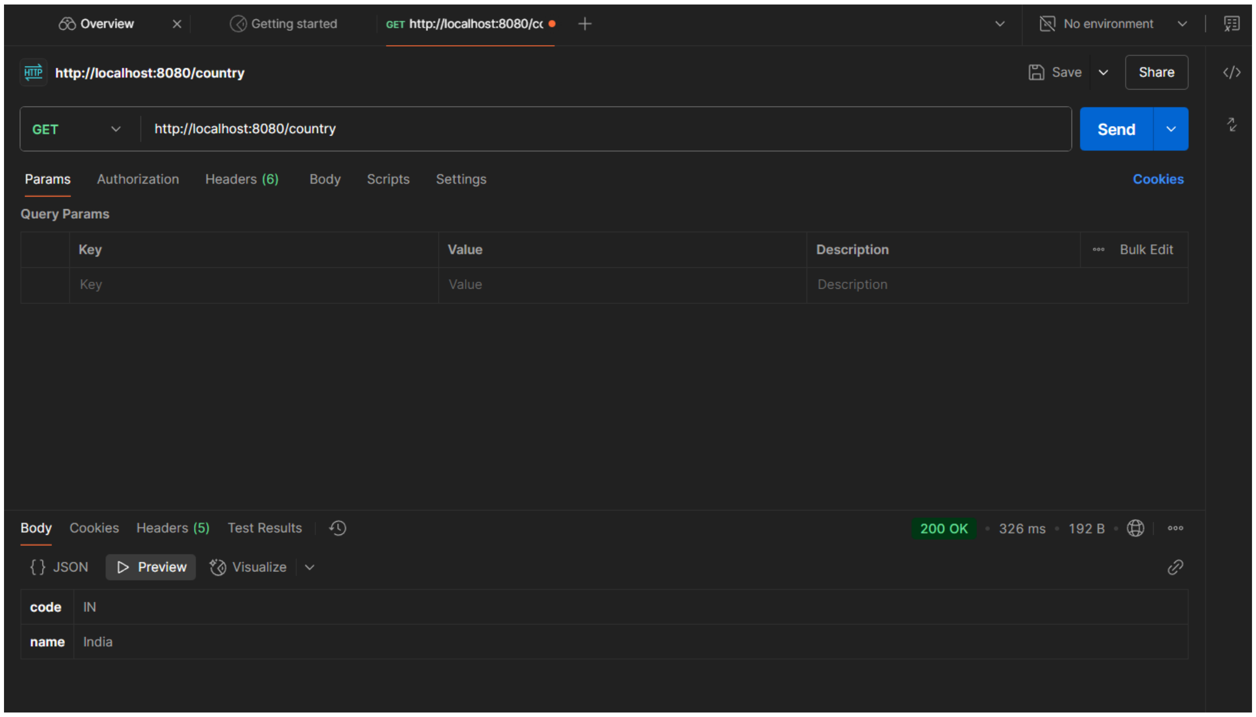
SME to explain the following aspects:

- What happens in the controller method?
- How the bean is converted into JSON reponse?
- In network tab of developer tools show the HTTP header details received
- In postman click on "Headers" tab to view the HTTP header details received



SpringLearnApiApplication [Java Application] C:\Program Files\eclipse-java-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (05-Jul-2025, 11:50:40 am) [pid: 24716]

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

 :: Spring Boot ::                (v3.5.3)

2025-07-05T11:50:43.528+05:30  INFO 24716 --- [  restartedMain] c.c.s.SpringLearnApiApplication          : Starting SpringLearnApiApplication using Java 17.0.5 with PID 24716 (C:\Users\HP
\Desktop\spring-learn-api\spring-learn-api\target\classes started by HP in C:\Users\HP\Desktop\spring-learn-api\spring-learn-api)
2025-07-05T11:50:43.534+05:30  INFO 24716 --- [  restartedMain] c.c.s.SpringLearnApiApplication          : No active profile set, falling back to 1 default profile: "default"
2025-07-05T11:50:43.646+05:30  INFO 24716 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to
'false' to disable
2025-07-05T11:50:43.646+05:30  INFO 24716 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web'
property to 'DEBUG'
2025-07-05T11:50:46.087+05:30  INFO 24716 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-05T11:50:46.118+05:30  INFO 24716 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-05T11:50:46.119+05:30  INFO 24716 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-05T11:50:46.218+05:30  INFO 24716 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-07-05T11:50:46.221+05:30  INFO 24716 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2572 ms
2025-07-05T11:50:47.368+05:30  INFO 24716 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-05T11:50:47.379+05:30  INFO 24716 --- [  restartedMain] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1 endpoint beneath base path '/actuator'
2025-07-05T11:50:47.511+05:30  INFO 24716 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-07-05T11:50:47.537+05:30  INFO 24716 --- [  restartedMain] c.c.s.SpringLearnApiApplication          : Started SpringLearnApiApplication in 4.865 seconds (process running for 5.785)
2025-07-05T11:51:37.975+05:30  INFO 24716 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-07-05T11:51:37.976+05:30  INFO 24716 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-05T11:51:37.978+05:30  INFO 24716 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
```



http://localhost:8080/country

GET http://localhost:8080/country

Query Params

	Key	Value	Description
	Key	Value	Description

code	IN
name	India

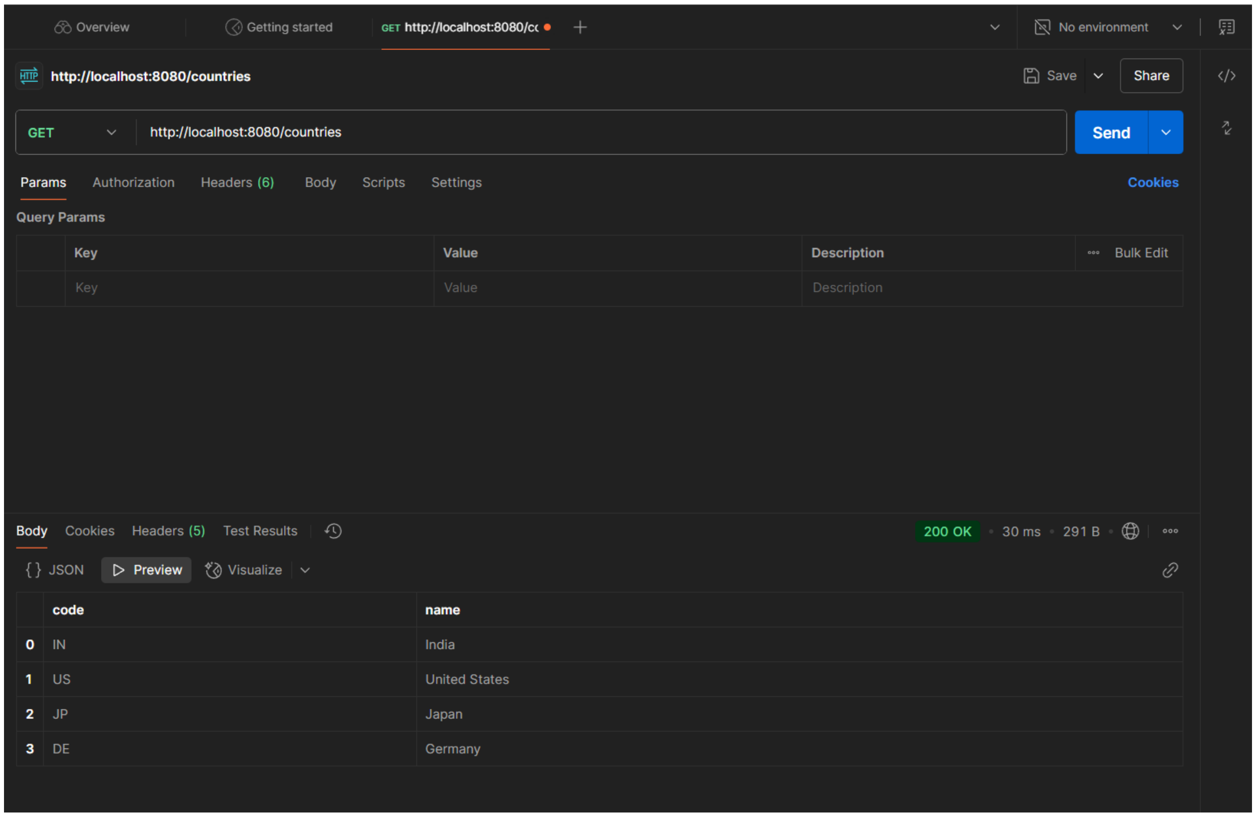
REST - Get all countries

Write a REST service that returns all the countries.

Controller: com.cognizant.spring-learn.controller.CountryController
Method Annotation: @GetMapping("/countries")
Method Name: getAllCountries()
Method Implementation: Load country list from country.xml and return

Sample Request: http://localhost:8083/countries
Sample Response:

```
[
  { "code": "IN", "name": "India"},
  { "code": "US", "name": "United States"},
  { "code": "JP", "name": "Japan"},
  { "code": "DE", "name": "Germany"}
]
```



http://localhost:8080/countries

GET http://localhost:8080/countries

Key	Value	Description
Key	Value	Description

200 OK · 30 ms · 291 B

	code	name
0	IN	India
1	US	United States
2	JP	Japan
3	DE	Germany

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)

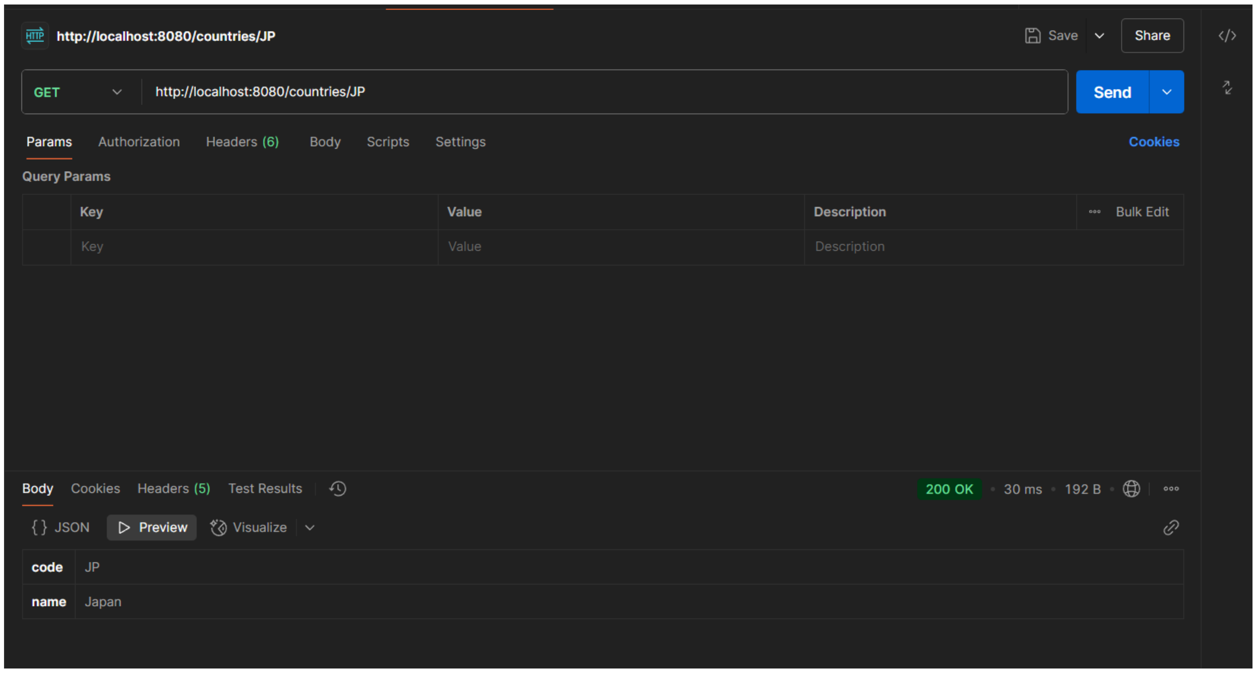
Service Method Implementation:

- 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"
}
```



http://localhost:8080/countries/JP

GET http://localhost:8080/countries/JP

Query Params

Key	Value	Description
Key	Value	Description

200 OK · 30 ms · 192 B

code	JP
name	Japan

REST - Get country exceptional scenario

In the previous hands on where we implemented getting country based on country code, what happens if the country code provided in the URL is not present.

Refer steps below to implement

- Create a new exception class com.cognizant.springlearn.service.exception.CountryNotFoundException
- Include below specified annotation at the class level in CountryNotFoundException class

```
@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "Country not found")
```

- In CountryService.getCountry() method include the logic to throw CountryNotFoundException if the country code does not exists in the list.
- In CountryController.getCountry() method include throws clause in method signature. This will respond to the caller of the web service with appropriate error message in JSON format.
- Test the service in postman and using curl command. Refer below for executing curl command.

Steps to invoke RESTful Web Service using curl command

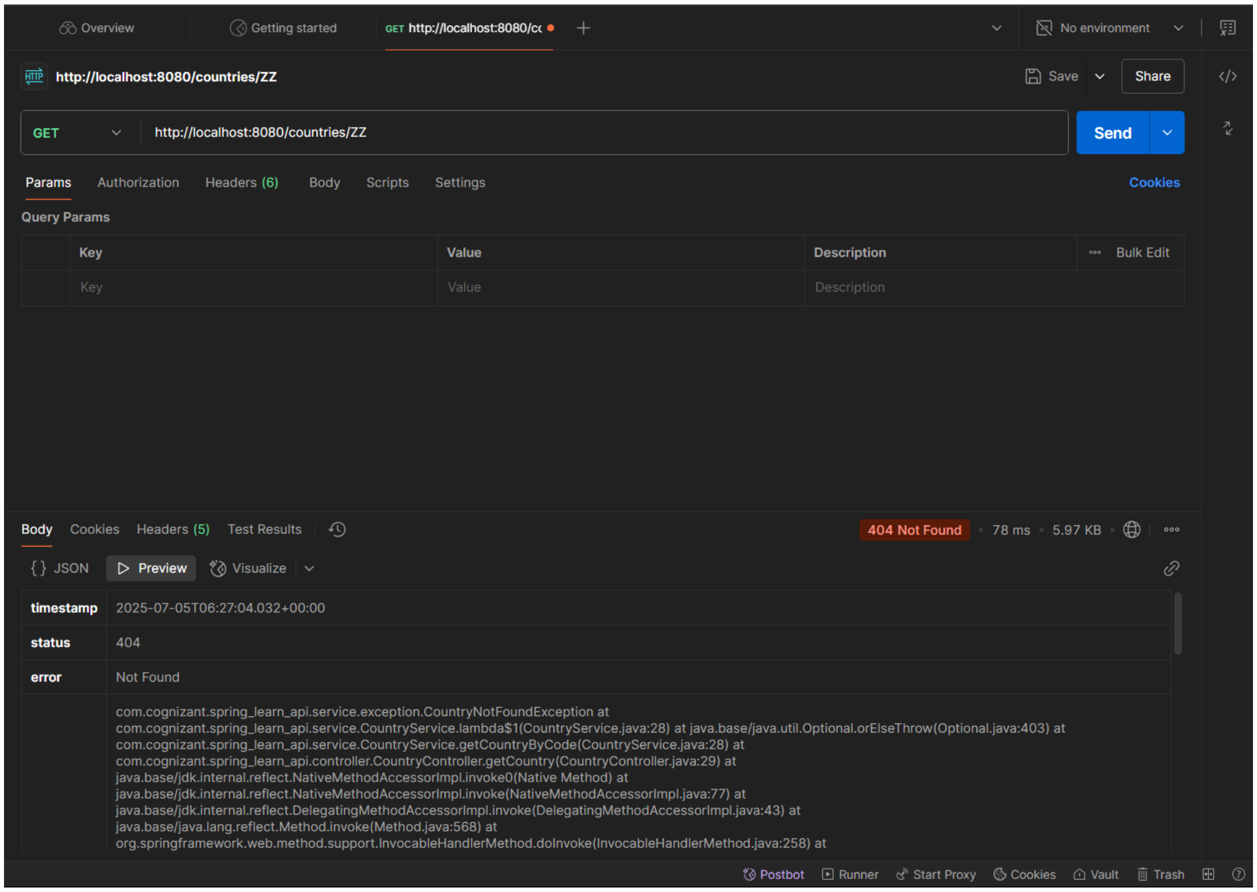
- Open Git Bash
- Execute the below command

```
curl -i http://localhost:8090/country/az
```

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

Sample Response:

```
{
  "timestamp": "2019-10-02T03:27:54.521+0000",
  "status": 404,
  "error": "Not Found",
  "message": "Country not found",
  "path": "/country/az"
}
```


GET http://localhost:8080/countries/ZZ

404 Not Found • 78 ms • 5.97 KB

```
timestamp  2025-07-05T06:27:04.032+00:00
status     404
error      Not Found
           com.cognizant.spring_learn_api.service.exception.CountryNotFoundException at
           com.cognizant.spring_learn_api.service.CountryService.lambda$1(CountryService.java:28) at java.base/java.util.Optional.orElseThrow(Optional.java:403) at
           com.cognizant.spring_learn_api.service.CountryService.getCountryByCode(CountryService.java:28) at
           com.cognizant.spring_learn_api.controller.CountryController.getCountry(CountryController.java:29) at
           java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at
           java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at
           java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at
           java.base/java.lang.reflect.Method.invoke(Method.java:568) at
           org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:258) at
```

MockMVC - Test get country service

Using MockMVC test the get country service.

Create a test cases to test the following aspects:

- Test is the CountryController is loaded
- Invoke the service to get country and check in the response if it contains code as "IN" and name as "India"

Refer steps below to implement

- **Test loading CountryController**
  - Include CountryController instance variable in SpringLearnApplicationTests.java and autowire the instance variable using annotation.

```
@Autowired
private CountryController countryController;
```

- Include assertion in contextLoads() method to check if controller is loaded.

```
    @Test
    public void contextLoads() {
        assertNotNull(countryController);
    }
```

- Run the JUnit testing by right clicking on SpringLearnApplicationTests.java > Run As > JUnit Test
- This test can also be executed in command line using the following maven command in the root folder of the project. (Note: don't forget to include proxy details in the below command)

```
mvn clean test
```

- Check if the log in the constructor of CountryController is called.

- **Test service to get the country**
    - Include below imports

```
import static org.junit.Assert.assertNotNull;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;


import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.ResultActions;
```

- Include @AutoConfigureMockMvc annotation for SpringLearnApplicationTests.java
- Autowire mock mvc in SpringLearnApplicationTests.java

```
@Autowired

private MockMvc mvc;
```

- Include a new test method in SpringLearnApplicationTests.java

```
@Test

public void testGetCountry() throws Exception {


}
```

- Include the following line in the new method that calls the service method. Execute the JUnit test and check if the test case is successful.

```
@Test

public void testGetCountry() throws Exception {

    ResultActions actions = mvc.perform(get("/country"));

}
```

- Include the following line to check if the HTTP Status is 200, which means the call is successful. Execute JUnit test and check if the test case is successful.

```
@Test

public void testGetCountry() throws Exception {

    ResultActions actions = mvc.perform(get("/country"));

    actions.andExpect(status().isOk());

}
```

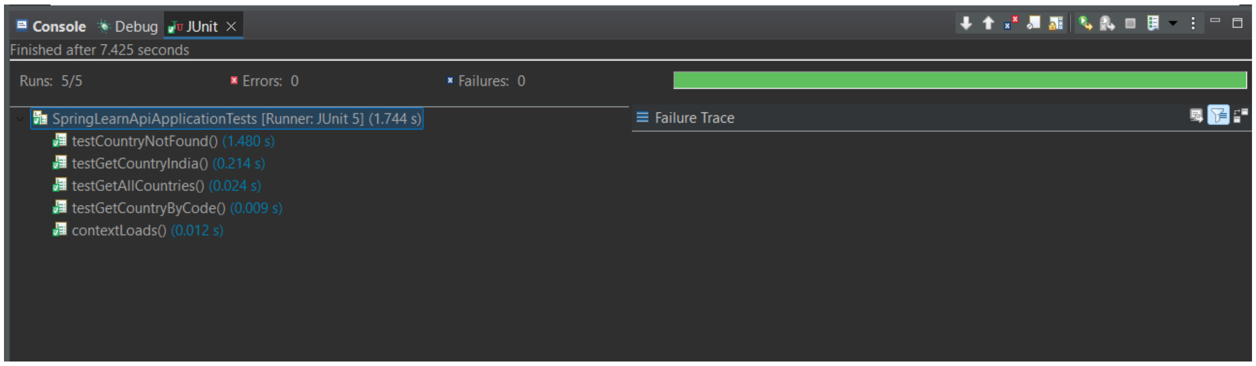
- Include the following line to check if the code is available in the reponse

```
@Test
public void getCountry() throws Exception {
    ResultActions actions = mvc.perform(get("/country"));
    actions.andExpect(status().isOk());
    actions.andExpect(jsonPath("$.code").exists());
}
```

- Include the following line to check if the value of code is "IN"

```
@Test
public void getCountry() throws Exception {
    ResultActions actions = mvc.perform(get("/country"));
    actions.andExpect(status().isOk());
    actions.andExpect(jsonPath("$.code").exists());
    actions.andExpect(jsonPath("$.code").value("IN"));
}
```

- Using above two steps include checks for "name" attribute and check if it's value is "India"



```
Finished after 7.425 seconds
Runs: 5/5        Errors: 0        Failures: 0

SpringLearnApiApplicationTests [Runner: JUnit 5] (1.744 s)
    testCountryNotFound() (1.480 s)
    testGetCountryIndia() (0.214 s)
    testGetAllCountries() (0.024 s)
    testGetCountryByCode() (0.009 s)
    contextLoads() (0.012 s)
```

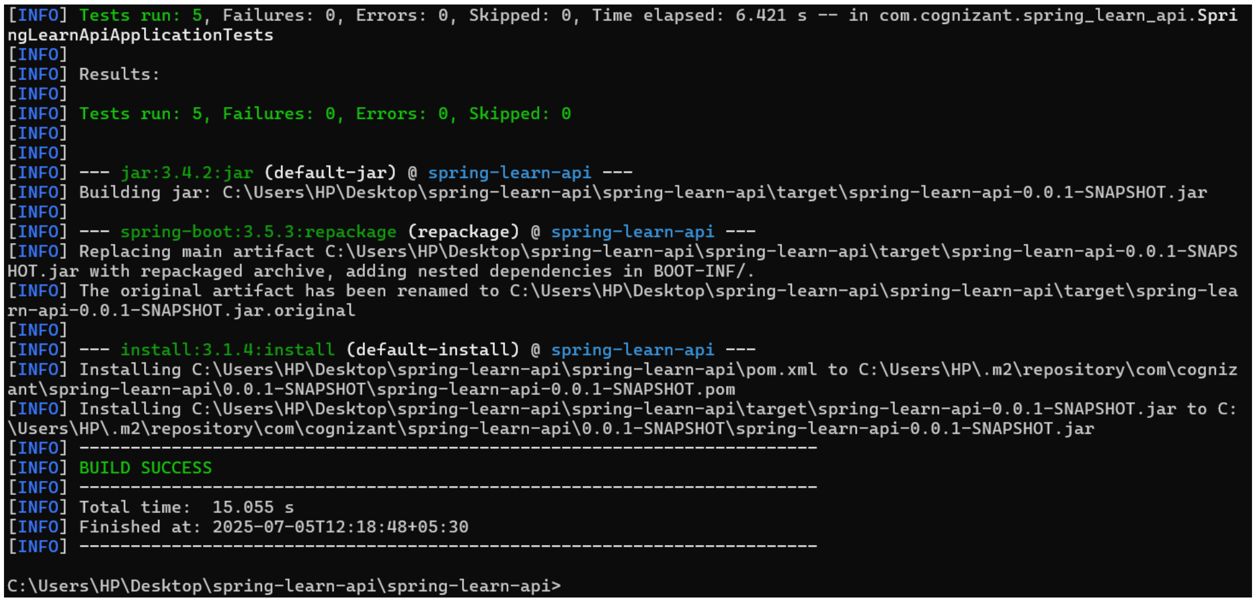
MockMVC - Test get country service for exceptional scenario

Include MockMVC test that checks if correct response is received when there is an error.

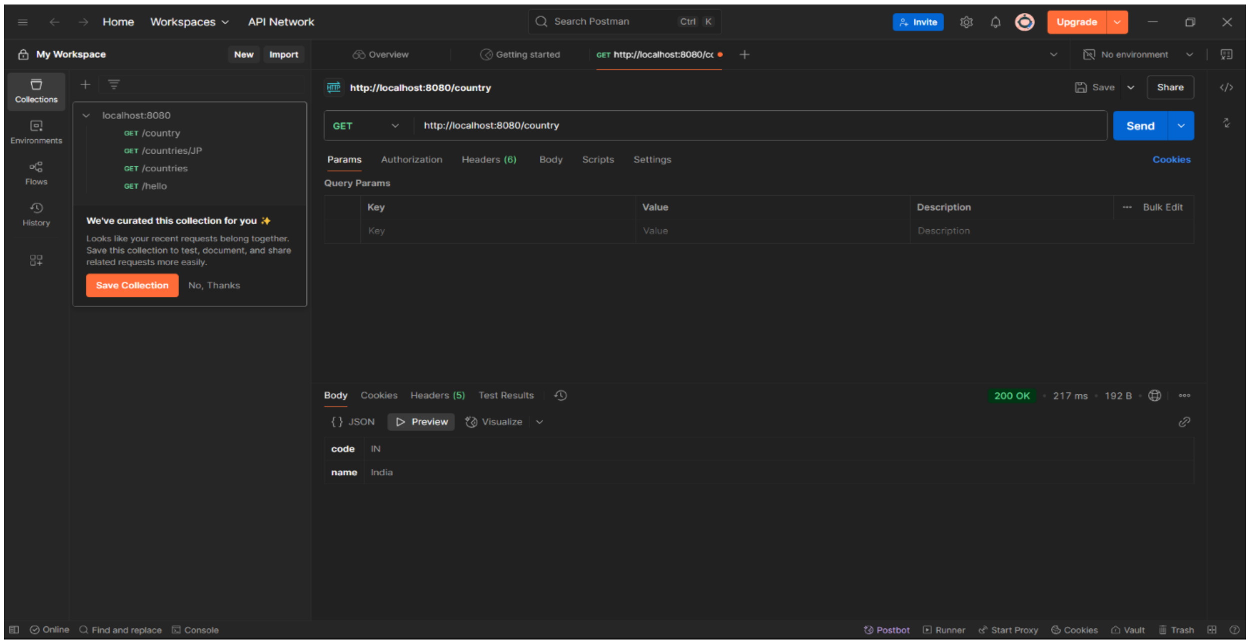
Refer steps below to implement

- Include a new test method testGetCountryException() in SpringLearnApplicationTests.java
- Validate the error response using status(). Refer code below.

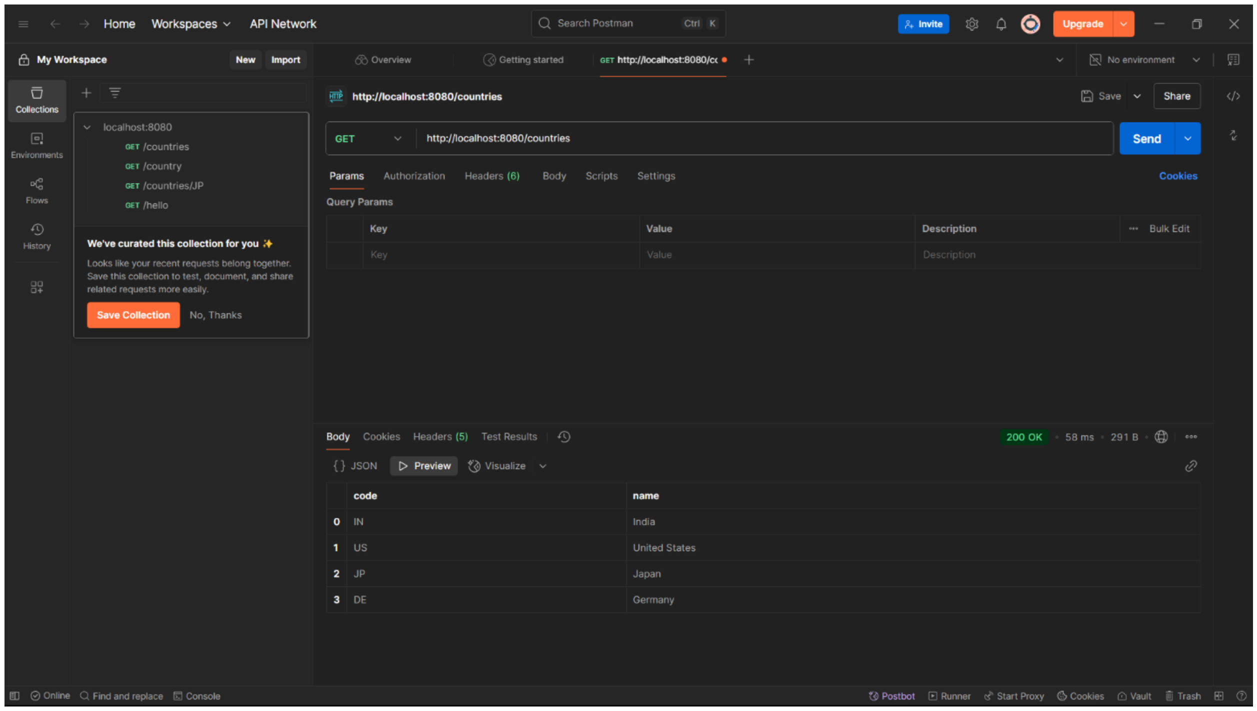
```
        actions.andExpect(status().isBadRequest());
        actions.andExpect(status().reason("Country Not found"));
```



```
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.421 s -- in com.cognizant.spring_learn_api.Spri
ngLearnApiApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ spring-learn-api ---
[INFO] Building jar: C:\Users\HP\Desktop\spring-learn-api\spring-learn-api\target\spring-learn-api-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.5.3:repackage (repackage) @ spring-learn-api ---
[INFO] Replacing main artifact C:\Users\HP\Desktop\spring-learn-api\spring-learn-api\target\spring-learn-api-0.0.1-SNAPS
HOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\HP\Desktop\spring-learn-api\spring-learn-api\target\spring-lea
rn-api-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.4:install (default-install) @ spring-learn-api ---
[INFO] Installing C:\Users\HP\Desktop\spring-learn-api\spring-learn-api\pom.xml to C:\Users\HP\.m2\repository\com\cogniz
ant\spring-learn-api\0.0.1-SNAPSHOT\spring-learn-api-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\HP\Desktop\spring-learn-api\spring-learn-api\target\spring-learn-api-0.0.1-SNAPSHOT.jar to C:
\Users\HP\.m2\repository\com\cognizant\spring-learn-api\0.0.1-SNAPSHOT\spring-learn-api-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  15.055 s
[INFO] Finished at: 2025-07-05T12:18:48+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\HP\Desktop\spring-learn-api\spring-learn-api>
```



```
GET http://localhost:8080/country

200 OK · 217 ms · 192 B

code  IN
name  India
```



localhost:8080
- GET /countries
- GET /country
- GET /countries/JP
- GET /hello

We've curated this collection for you ✨

Looks like your recent requests belong together. Save this collection to test, document, and share related requests more easily.

Save Collection   No, Thanks

http://localhost:8080/countries

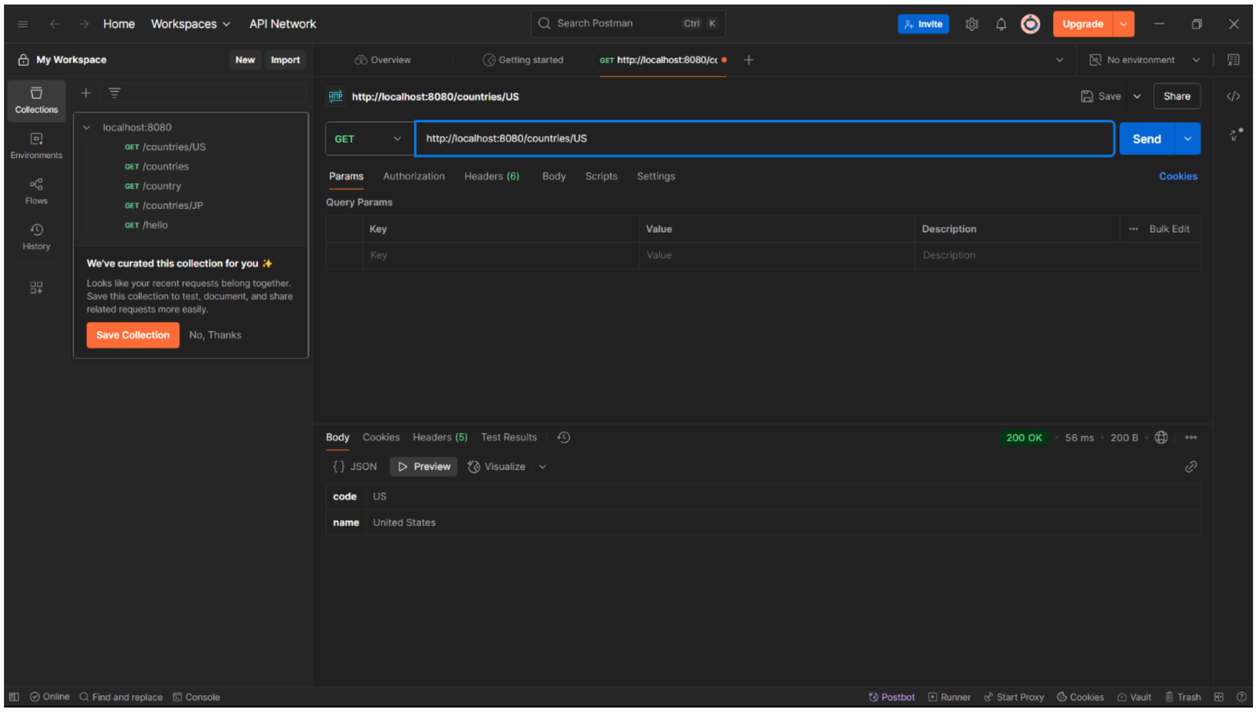
GET http://localhost:8080/countries

Query Params

Key	Value	Description
Key	Value	Description

200 OK · 58 ms · 291 B

	code	name
0	IN	India
1	US	United States
2	JP	Japan
3	DE	Germany



localhost:8080
- GET /countries/US
- GET /countries
- GET /country
- GET /countries/JP
- GET /hello

We've curated this collection for you ✨

Looks like your recent requests belong together. Save this collection to test, document, and share related requests more easily.

Save Collection   No, Thanks

http://localhost:8080/countries/US

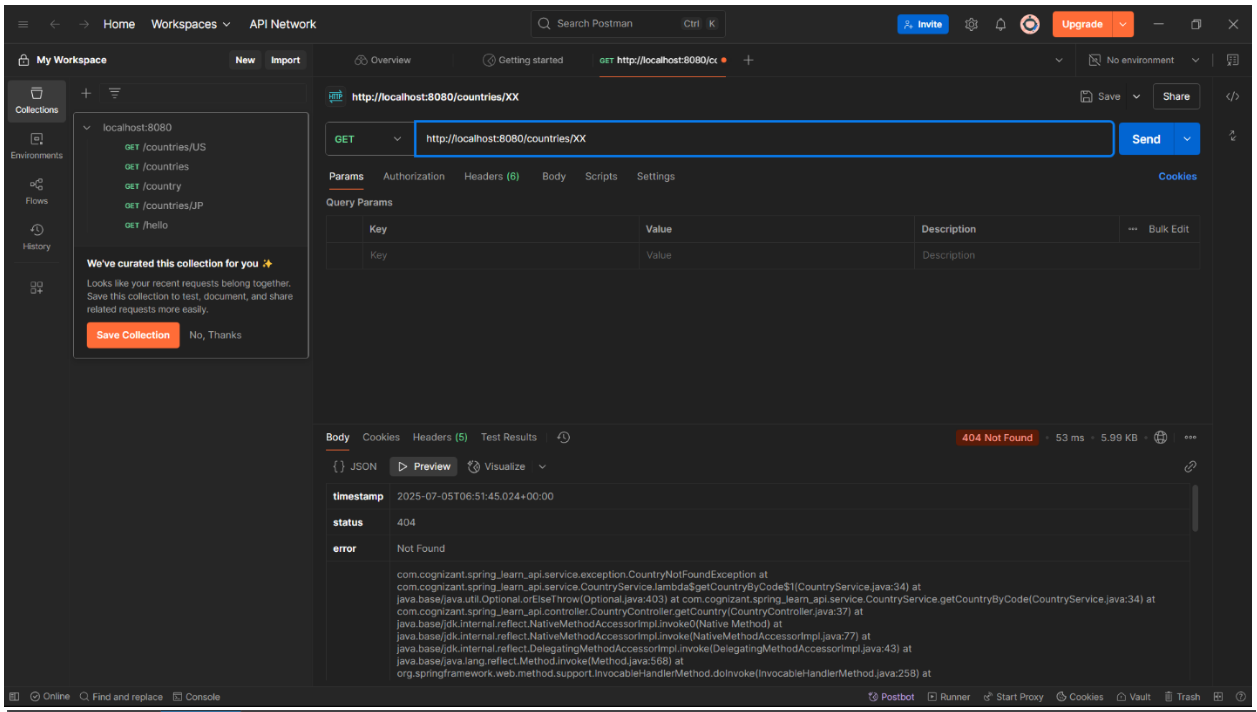
GET http://localhost:8080/countries/US

Query Params

Key	Value	Description
Key	Value	Description

200 OK · 56 ms · 200 B

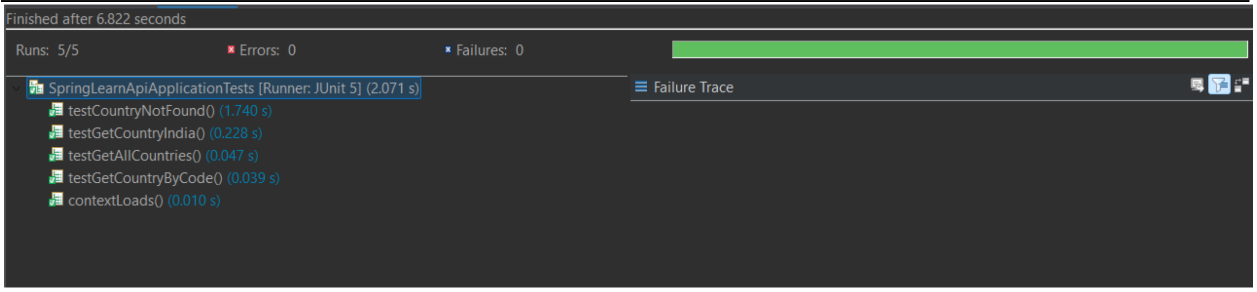
code	US
name	United States


GET http://localhost:8080/countries/XX

404 Not Found · 53 ms · 5.99 KB

timestamp	2025-07-05T06:51:45.024+00:00
status	404
error	Not Found

```
com.cognizant.spring_learn_api.service.exception.CountryNotFoundException at
com.cognizant.spring_learn_api.service.CountryService.lambda$getCountryByCode$1(CountryService.java:34) at
java.base/java.util.Optional.orElseThrow(Optional.java:403) at com.cognizant.spring_learn_api.service.CountryService.getCountryByCode(CountryService.java:34) at
com.cognizant.spring_learn_api.controller.CountryController.getCountry(CountryController.java:37) at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) at
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at
java.base/java.lang.reflect.Method.invoke(Method.java:568) at
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:258) at
```


```
Finished after 6.822 seconds
Runs: 5/5        Errors: 0        Failures: 0

SpringLearnApiApplicationTests [Runner: JUnit 5] (2.071 s)
    testCountryNotFound() (1.740 s)
    testGetCountryIndia() (0.228 s)
    testGetAllCountries() (0.047 s)
    testGetCountryByCode() (0.039 s)
    contextLoads() (0.010 s)
```

Submitted BY:

Name : Lingaraj Nayak

Superset ID : 6387607