**REST - Country Web Service**

**Introduction**

This exercise demonstrates the creation of a simple RESTful web service using Spring Boot. The objective is to expose an endpoint /country that, when accessed via a browser or tool like Postman, returns the details of a country — specifically India — in JSON format.

This hands-on task is designed to help understand how Spring Boot handles REST APIs, bean configuration, and automatic response conversion. It also reinforces how to inspect HTTP interactions using developer tools and API testing platforms.

The implementation involves:

* Defining a Country bean using an external Spring XML configuration file (country.xml).
* Creating a REST controller to retrieve the bean from the context and expose it via an HTTP endpoint.
* Leveraging Spring Boot’s built-in Jackson support to automatically serialize Java objects into JSON format in the response body.

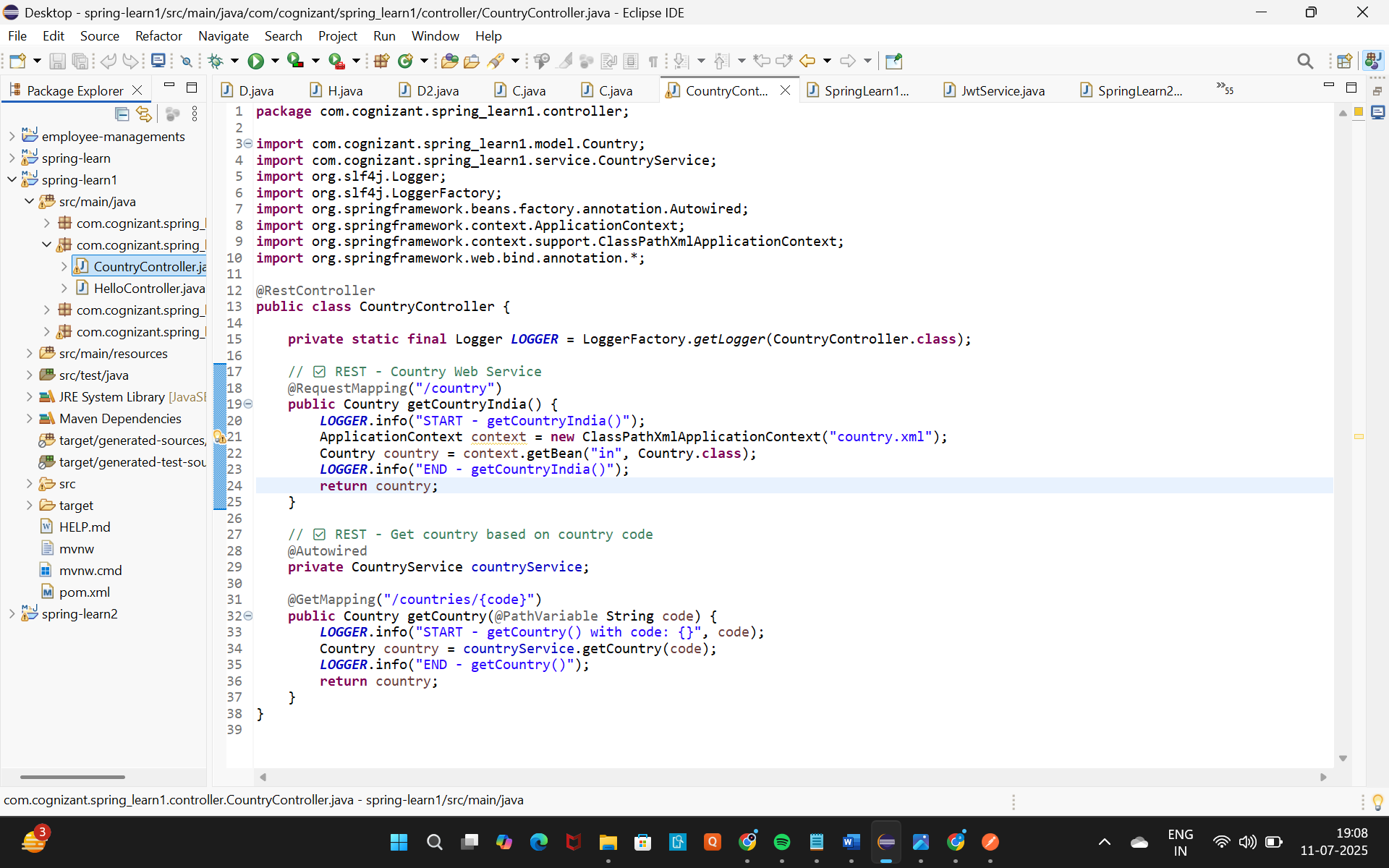
The following sections answer specific SME (Subject Matter Expert) questions, including what happens inside the controller, how the conversion to JSON works, and how to view HTTP header information using both the browser’s developer tools and Postman.

**SME to explain the following aspects:**

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

The controller method getCountryIndia() is mapped to the /country URL using the @RequestMapping annotation. When this URL is accessed via a GET request:

* The Spring application context is loaded from country.xml.
* The country bean is fetched, which contains India's details (code = IN, name = India).
* The method returns this Country object.
* Spring Boot automatically converts the returned object into JSON format in the HTTP response.

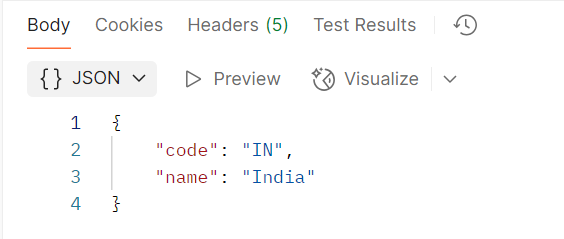


Source: CountryController.java

1. **How is the bean converted into a JSON response?**

Spring Boot includes the Jackson JSON library by default. When the controller returns a Java object (like Country), Spring uses HTTP Message Converters to serialize the object to JSON.

* No additional code is needed.
* The returned JSON is:



Based on the Country class and bean defined in country.xml

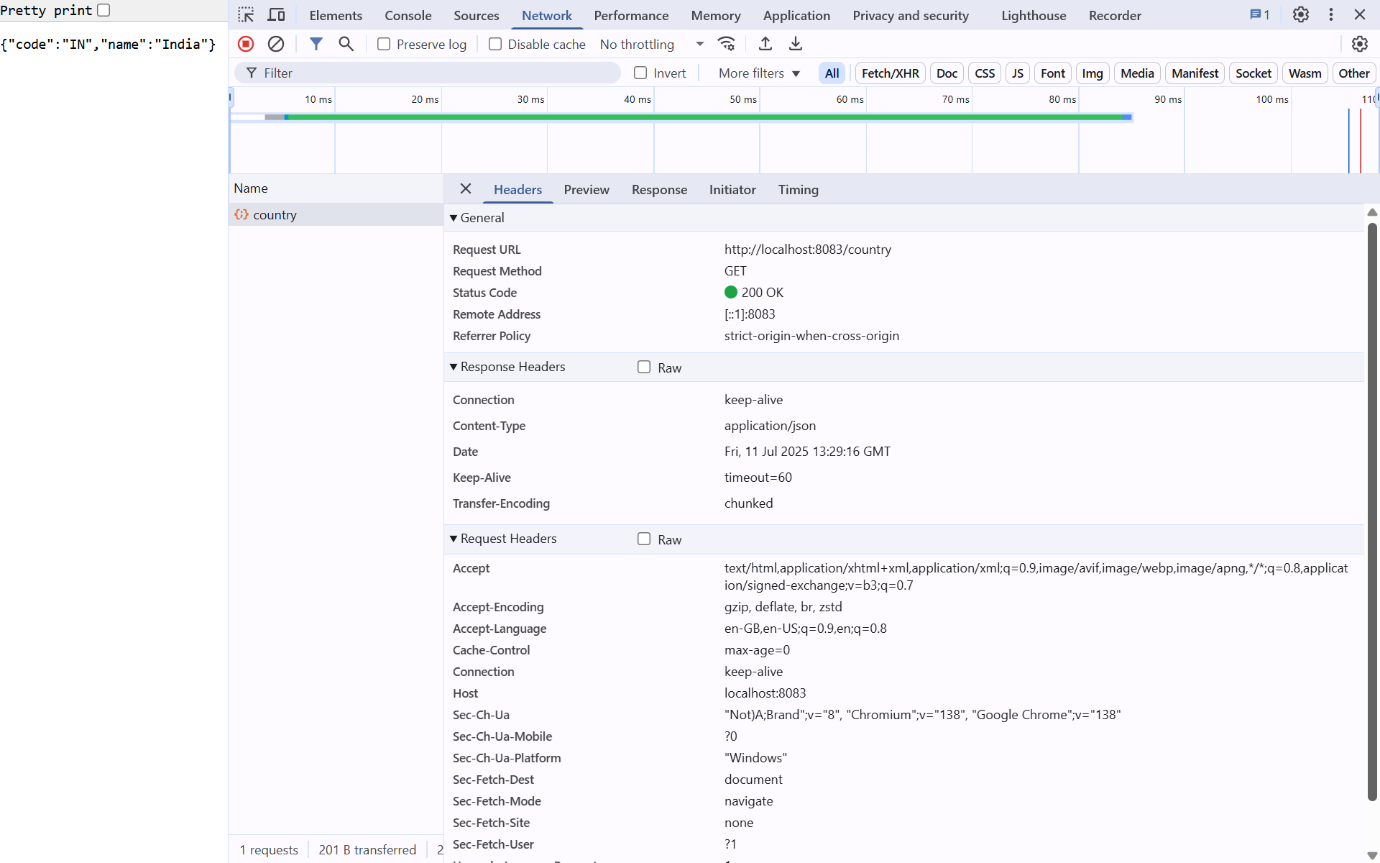
1. **In the browser (Developer Tools → Network tab), where are HTTP header details shown?**

To view HTTP headers in the browser:

1. Open Chrome Developer Tools (F12 or Ctrl + Shift + I).
2. Go to the Network tab.
3. Access http://localhost:8083/country.
4. Click on the /country row.
5. Go to the Headers section to view:

Key Headers Observed:

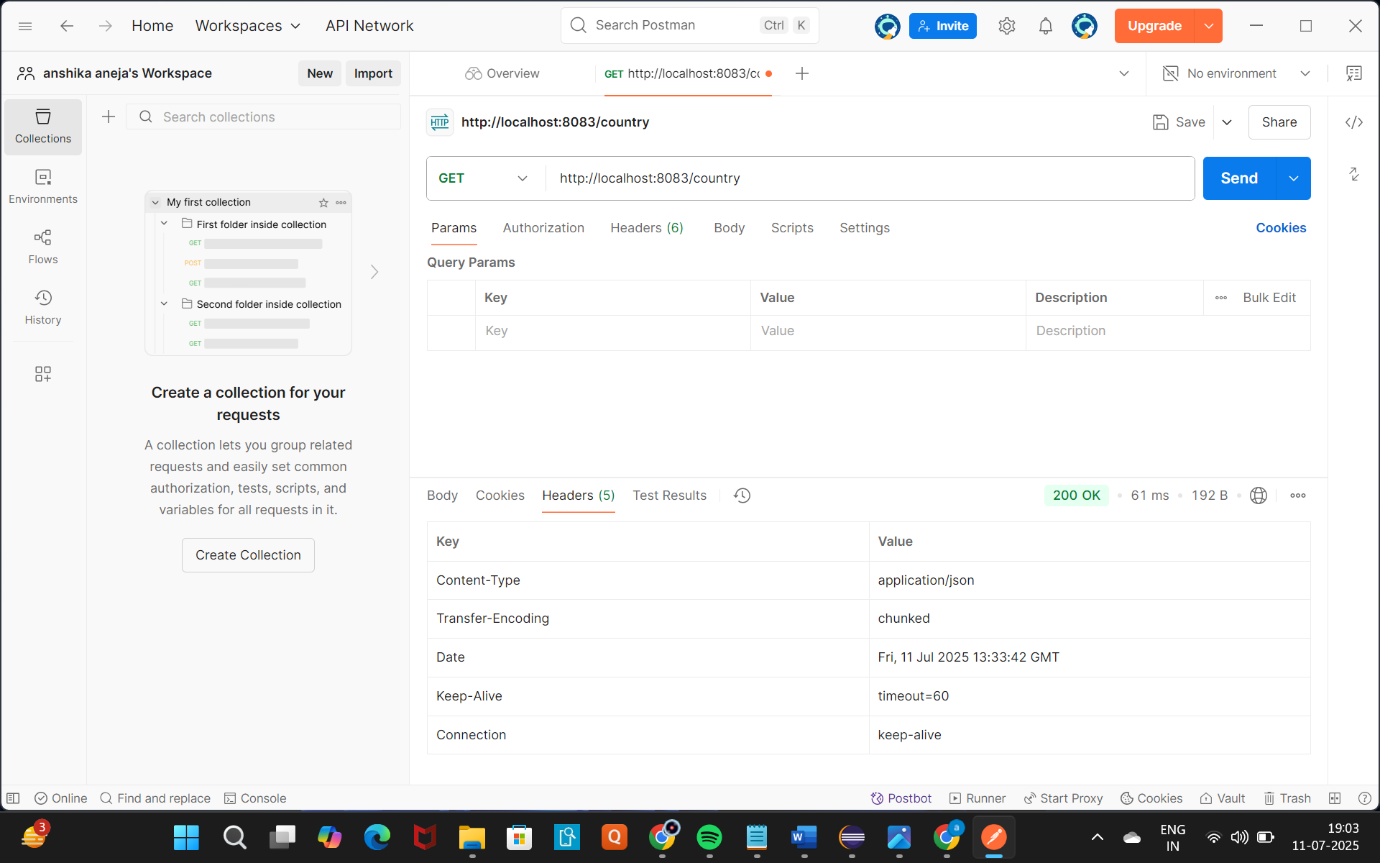
* Request URL: <http://localhost:8083/country>
* Status Code: 200 OK
* Content-Type: application/json
* Request Method: GET



**4. In Postman, click on "Headers" tab to view the HTTP header details received**

Steps:

1. Open Postman and send a GET request to http://localhost:8083/country.
2. After sending, go to the Headers tab in the response panel.
3. You will find details like:
   * Content-Type: application/json
   * Date: (timestamp)
   * Server: (Spring default or Tomcat)
4. Postman – Response Headers Screenshot:



**Conclusion**

This hands-on exercise successfully demonstrated the creation of a RESTful service in Spring Boot that returns JSON-formatted data for a country using a controller method and an XML-defined Spring bean.

Key takeaways include:

* How to use @RequestMapping to expose REST endpoints.
* How Spring Boot automatically serializes Java objects to JSON using Jackson.
* How to inspect HTTP request/response headers using browser developer tools and Postman.

The implementation confirms a clear understanding of basic Spring REST concepts, including bean configuration, response structure, and runtime behavior.