**Hands On Session**

**Spring REST - Country Web Service**

This is a Spring Boot REST service that returns India country details.

**## Project Structure**

```

src/main/java/com/cognizant/springlearn/

├── SpringLearnApplication.java          # Main Spring Boot application

├── controller/

│   └── CountryController.java          # REST controller with /country endpoint

└── model/

    └── Country.java                    # Country model class

src/main/resources/

├── applicationContext.xml              # Spring XML configuration with India bean

└── application.properties              # Application configuration

```

**## Requirements**

- Java 11 or higher

- Maven 3.6 or higher

## How to Run

1. \*\*Build the project:\*\*

   ```bash

   mvn clean package

   ```

2. \*\*Run the application:\*\*

   ```bash

   mvn spring-boot:run

   ```

3. \*\*Access the service:\*\*

   - URL: http://localhost:8083/country

   - Method: GET

**## Expected Response**

```json

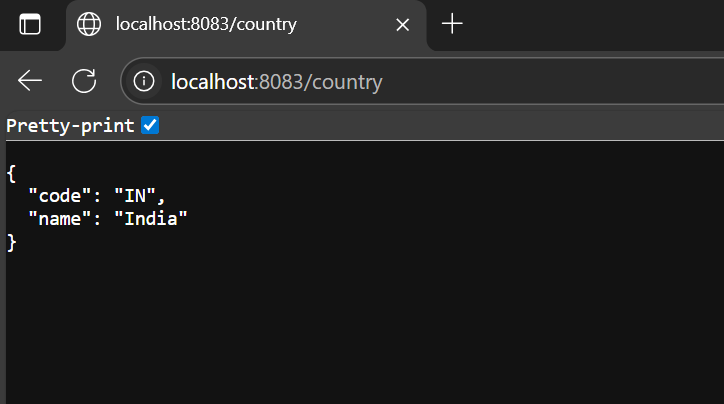
{

  "code": "IN",

  "name": "India"

}

```



**## Testing with Postman**

1. Open Postman

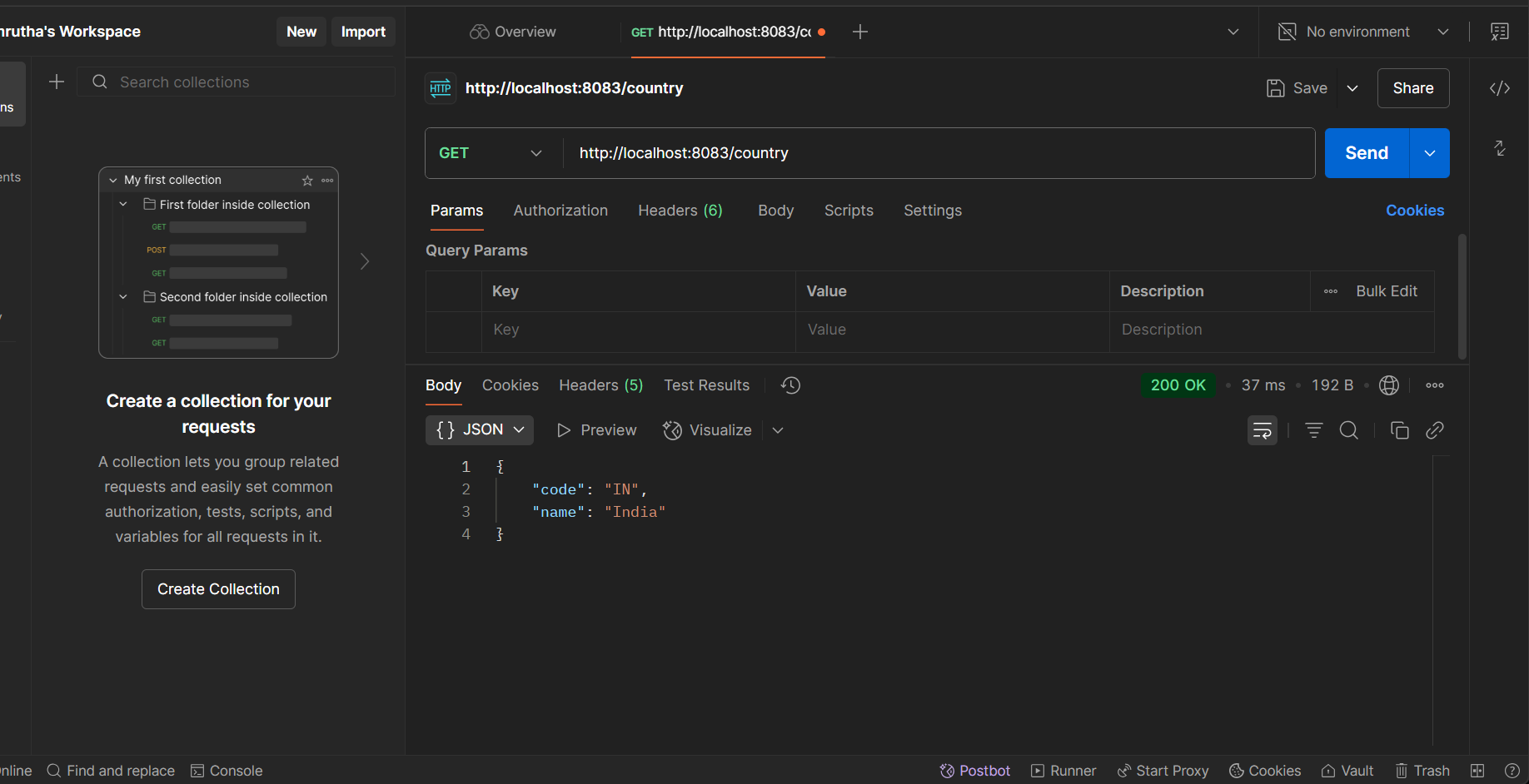
2. Create a new GET request

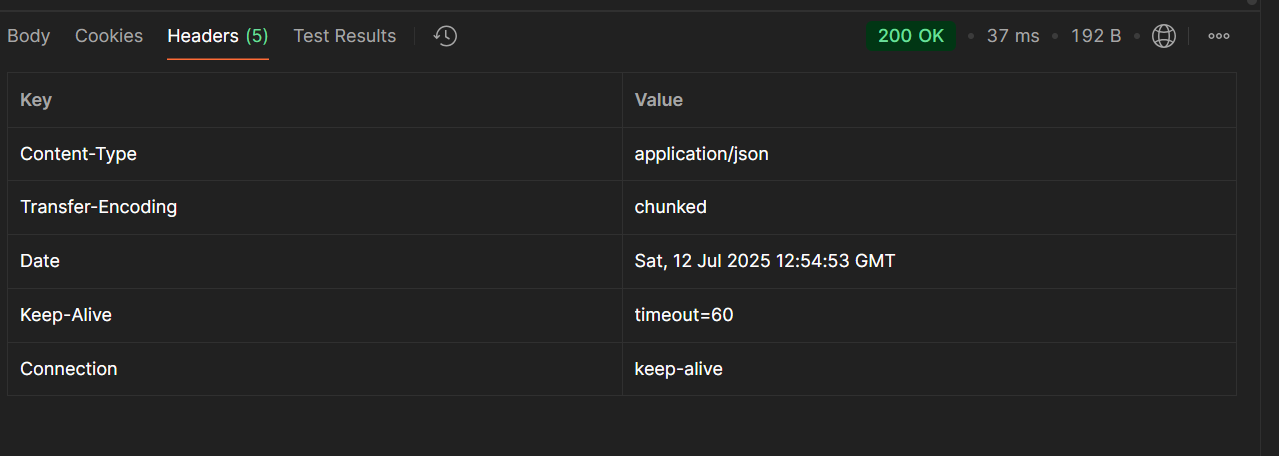
3. Enter URL: `http://localhost:8083/country`

4. Click "Send"

5. View the response in the "Body" tab

6. Click on the "Headers" tab to view HTTP header details





**## Testing with Browser Developer Tools**

1. Open browser and navigate to: `http://localhost:8083/country`

2. Open Developer Tools (F12)

3. Go to the "Network" tab

4. Refresh the page

5. Click on the request to view HTTP header details

**## Technical Details**

**### Controller Method Explanation**

The `getCountryIndia()` method in `CountryController`:

- Uses `@RequestMapping("/country")` to map to the `/country` URL

- Injects `ApplicationContext` to access Spring beans

- Retrieves the "india" bean from the XML configuration

- Returns the Country object, which is automatically converted to JSON

**### JSON Conversion**

Spring Boot automatically converts the returned `Country` object to JSON using:

- Jackson library (included in spring-boot-starter-web)

- Object properties are serialized to JSON key-value pairs

- The `@RestController` annotation enables automatic JSON conversion

**### HTTP Headers**

When testing, you'll see headers like:

- `Content-Type: application/json`

- `Content-Length: [size]`

- `Date: [timestamp]`

- `Server: [server info]`