**1.** What happens in the controller method**?**

**Code:**

@RestController

public class CountryController {

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

@RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.info("START - getCountryIndia()");

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

Country country = (Country) context.getBean("in");

LOGGER.info("END - getCountryIndia()");

return country;

}

}

**Step-by-step Execution:**

1. **Logging Start**:

**LOGGER.info("START - getCountryIndia()");**

Logs the beginning of the method execution using SLF4J Logger.

1. **Loading Spring Application Context**:

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

* 1. This creates a new **Spring application context** by loading the configuration from country.xml, which should be located in the classpath.
  2. country.xml is an **XML configuration file** that defines Spring beans, in this case, probably for the Country class.

1. **Fetching the 'in' Bean**:

**Country country = (Country) context.getBean("in");**

* 1. Retrieves the bean with ID "in" from the Spring context.
  2. Casts it to the Country class.
  3. This bean represents the country **India**, as implied by "in".

1. **Logging End**:

**LOGGER.info("END - getCountryIndia()");**

Logs the end of the method execution.

1. **Returns the Country Object**:

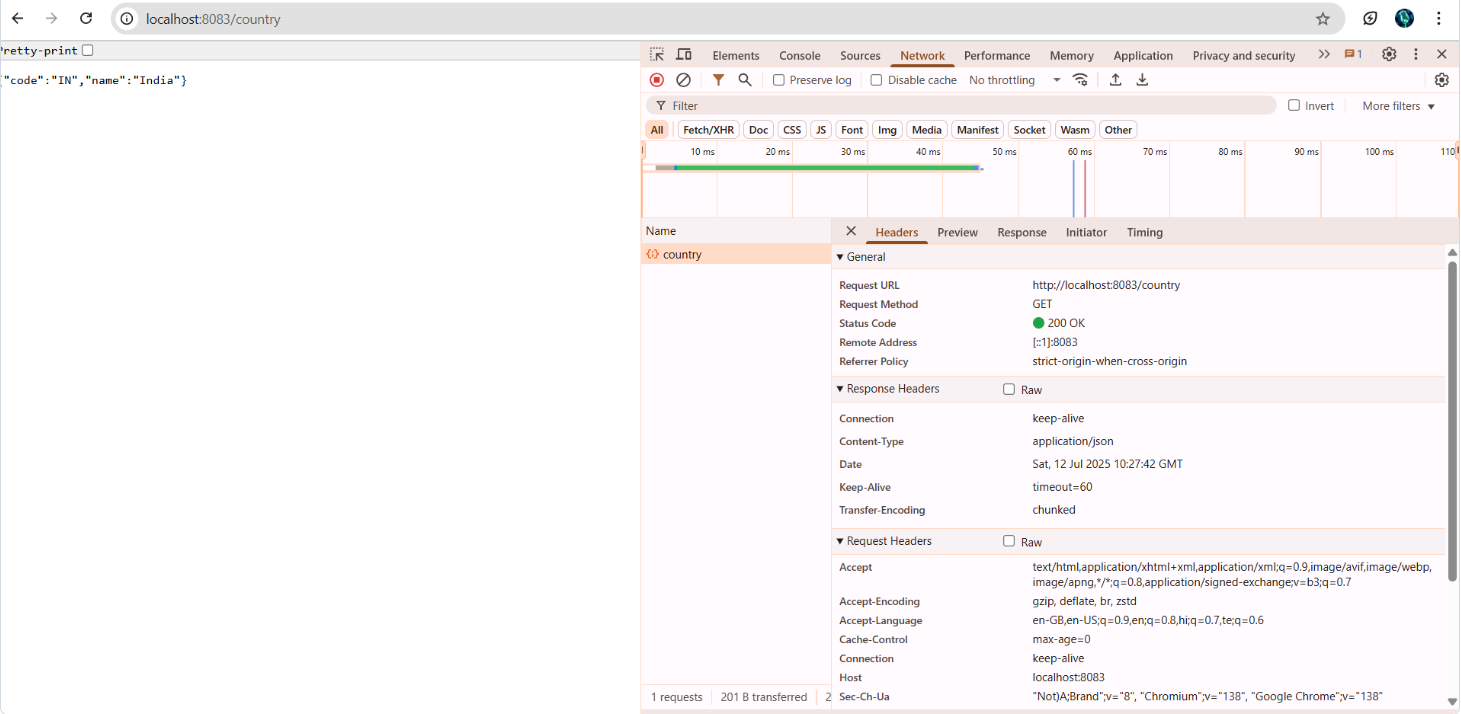
**return country;**

* 1. Sends the Country object as a JSON response (automatically handled by Spring Boot due to @RestController and Jackson).
  2. the body.

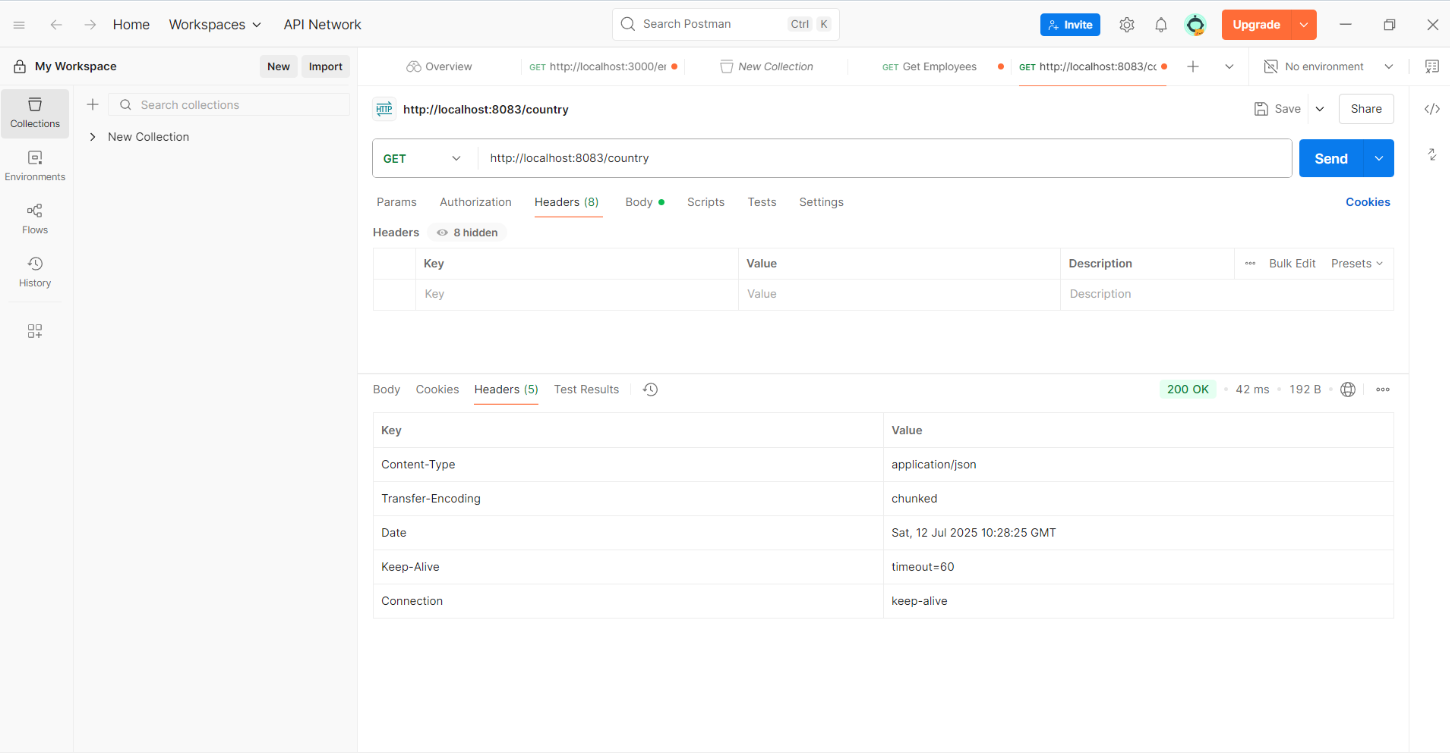
**2.** How the bean is converted into JSON response?

1. The Country object is returned as the body of the response.
2. Because of the @RestController annotation Spring automatically uses **Jackson** to **serialize the Java object into JSON**.
3. Jackson is included by default in spring-boot-starter-web.

## 3.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

****