**1. How does the Spring framework convert the request payload into a Country bean?**

When a POST request with a JSON payload is received, Spring uses Jackson's ObjectMapper to deserialize the JSON into a Java object. The @RequestBody annotation triggers this conversion. Jackson matches JSON keys to Java bean properties by name (case-sensitive) and calls the corresponding setter methods.

Process Flow:

1. Client sends JSON: {"code":"IN","name":"India"}
2. Jackson creates a new Country instance
3. For each JSON key:

* Looks for matching bean property (e.g., code → setCode())
* Calls setter method with the JSON value

4.Populated Country object is passed to controller method

Example:

java

// Controller method

@PostMapping("/countries")

public Country addCountry(@RequestBody Country country) {

System.out.println("Received: " + country); // Log: Received: Country(code=IN, name=India)

return country;

}

**2. How Spring parses JSON using Jackson?**

Spring Boot auto-configures MappingJackson2HttpMessageConverter when Jackson is in the classpath. This converter:

1. Checks Content-Type: application/json header
2. Uses Jackson to map JSON properties to Java fields/setters
3. Handles type conversion (e.g., String → Integer)

Dependency:

xml

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

**3. How JSON attributes map to Java methods?**

Naming Convention:

Jackson converts JSON keys to setter method names using:

1. InitCaps: name → Name
2. Add prefix: Name → setName
3. Call method: Invokes setName("India")

Reflection Process:

java

// Behind the scenes

Country country = new Country();

Method setter = Country.class.getMethod("set" + initCaps(key)); // e.g., setName

setter.invoke(country, value); // country.setName("India")