**Spring Framework: JSON to Java Bean Conversion**

When a client sends a JSON request payload to a Spring REST API endpoint annotated with @RequestBody, Spring performs a series of well-orchestrated steps to convert that payload into a corresponding Java object (in this case, Country). This is part of Spring's **HTTP message conversion** mechanism.

**1. JSON Parsing and Conversion using Jackson**

Spring Boot internally uses the **Jackson library** (a powerful JSON parser and processor) as the default implementation for JSON serialization and deserialization.

* When a POST request is received with a Content-Type: application/json, Spring automatically delegates the payload parsing to **Jackson's ObjectMapper**.
* Jackson reads the JSON structure and attempts to map it to a Java class (e.g., Country).

**2. Property Mapping Using JavaBean Conventions**

Jackson follows **JavaBean naming conventions** for property mapping:

* For each JSON attribute (e.g., "name"), it capitalizes the first letter (Name) and prefixes it with get or set to construct method names like getName() and setName(String name).
* This is how Jackson determines which **getter/setter methods** to invoke via the **Java Reflection API**.

**Example:**

* JSON: { "code": "IN", "name": "India" }
* Jackson constructs:
  + setCode("IN")
  + setName("India")
* These methods are invoked reflectively to populate the fields of the Country object.

**3. Object Instantiation and Injection**

* Jackson **creates an instance of the Country class** using its default no-argument constructor.
* It then **invokes the appropriate setter methods** (setCode, setName) for each matching JSON property.
* After fully populating the Country bean, Spring **injects this object** as an argument to the controller method defined with @RequestBody.

**Controller Example:**

@PostMapping("/countries")

public Country addCountry(@RequestBody Country country) {

// country is fully initialized at this point

}

**4. Controller Method Invocation**

Once the Country bean is created and populated, Spring proceeds to invoke the addCountry() controller method, passing the fully constructed and initialized Country object as an argument.

**Bean Naming Conventions in Spring**

Spring beans (whether POJOs or components) adhere to standard Java naming conventions:

* **Class names** follow **PascalCase**: e.g., Country, UserProfile.
* **Properties (fields)** use **camelCase**: e.g., countryCode, countryName.
* **Getter/Setter methods** follow JavaBeans standards:
  + getFieldName(), setFieldName(value)
  + For boolean properties: isActive(), setActive(boolean)