

## Hello World RESTful Web Service

Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:

**Method:** GET

**URL:** /hello

**Controller:** com.cognizant.spring-learn.controller.HelloController

**Method Signature:** public String sayHello()

**Method Implementation:** return hard coded string "Hello World!!"

**Sample Request:** http://localhost:8083/hello

**Sample Response:** Hello World!!

**IMPORTANT NOTE:** Don't forget to include start and end log in the sayHello() method.

Try the URL http://localhost:8083/hello in both chrome browser and postman.

SME to explain the following aspects:

- 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

### HelloController.java

```
package com.example.demo.Controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {
```

```

private static final Logger LOGGER = LoggerFactory.getLogger>HelloController.class);

@GetMapping("/hello")

public String sayHello() {

    LOGGER.debug("Start: sayHello()");

    LOGGER.debug("End: sayHello()");

    return "Hello World!!";

}

}

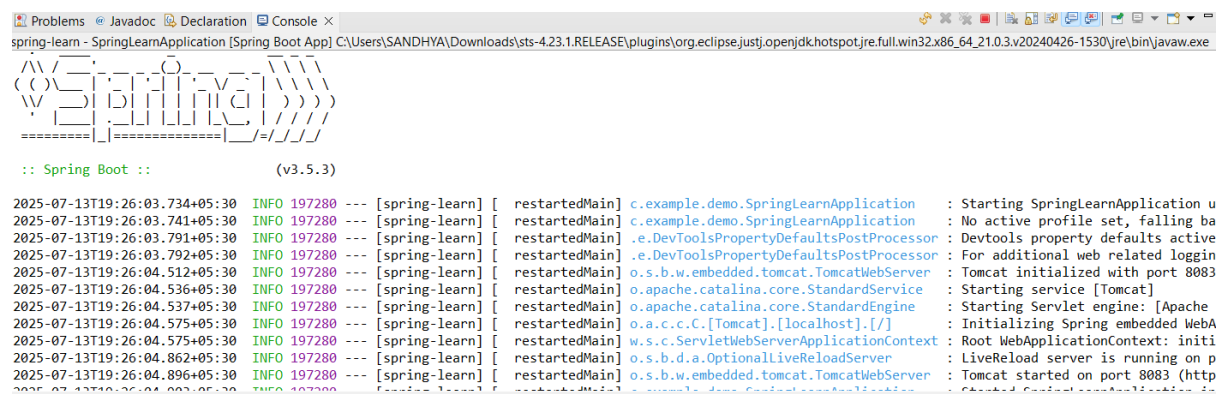
```

## POM.xml

```
spring.application.name=spring-learn
```

```
server.port=8083
```

## RESULT:



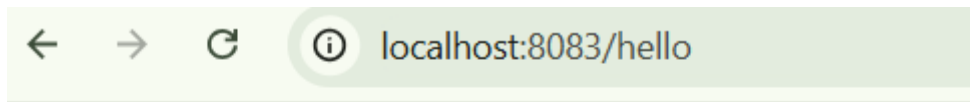
```

Problems Javadoc Declaration Console ×
spring-learn - SpringLearnApplication [Spring Boot App] C:\Users\SANDHYA\Downloads\sts-4.23.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe

:: Spring Boot ::
(v3.5.3)

2025-07-13T19:26:03.734+05:30 INFO 197280 --- [spring-learn] [ restartedMain] c.example.demo.SpringLearnApplication : Starting SpringLearnApplication u
2025-07-13T19:26:03.741+05:30 INFO 197280 --- [spring-learn] [ restartedMain] c.example.demo.SpringLearnApplication : No active profile set, falling ba
2025-07-13T19:26:03.791+05:30 INFO 197280 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active
2025-07-13T19:26:03.792+05:30 INFO 197280 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related login
2025-07-13T19:26:04.512+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8083
2025-07-13T19:26:04.536+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-13T19:26:04.537+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
2025-07-13T19:26:04.575+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebA
2025-07-13T19:26:04.575+05:30 INFO 197280 --- [spring-learn] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initi
2025-07-13T19:26:04.862+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on p
2025-07-13T19:26:04.896+05:30 INFO 197280 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8083 (http
2025-07-13T19:26:04.896+05:30 INFO 197280 --- [spring-learn] [ restartedMain] c.example.demo.SpringLearnApplication : Started SpringLearnApplication in

```



Hello World!!

## REST - Country Web Service

Write a REST service that returns India country details in the earlier created spring learn application.

**URL:** /country

**Controller:** com.cognizant.spring-learn.controller.CountryController

**Method Annotation:** @RequestMapping

**Method Name:** getCountryIndia()

**Method Implementation:** Load India bean from spring xml configuration and return

**Sample Request:** http://localhost:8083/country

**Sample Response:**

```
{
  "code": "IN",
  "name": "India"
}
```

SME to explain the following aspects:

- What happens in the controller method?
- How the bean is converted into JSON response?
- 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

## Project Setup

### Dependencies

- Spring Boot Starter Web
- Spring Context (for XML bean loading)
- Jackson (auto-included with Spring Boot for JSON conversion)

## Component Implementation

### Model Class

```
// Country.java
package com.cognizant.springlearn.model;

public class Country {
    private String code;
    private String name;

    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

### XML Bean Configuration

```
<!-- country.xml -->
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="in" class="com.cognizant.springlearn.model.Country">
        <property name="code" value="IN"/>
        <property name="name" value="India"/>
    </bean>
</beans>
```

### Spring Boot Main Class

```
// SpringLearnApplication.java
package com.cognizant.springlearn;
```

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringLearnApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication.class, args);
    }
}

```

## Controller Class

```

// CountryController.java
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.*;

@RestController
public class CountryController {

    @RequestMapping("/country")
    public Country getCountryIndia() {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country = (Country) context.getBean("in");
        return country;
    }
}

```

## Configuration File

```

# application.properties
server.port=8083

```

## Behind the Scenes

### What Happens in the Controller Method?

- The `@RequestMapping("/country")` maps incoming GET requests to the method `getCountryIndia()`.
- Inside this method:
  - `ClassPathXmlApplicationContext` loads the `country.xml`.
  - The `Country` bean with ID `in` is retrieved and returned.
- Spring's `@RestController` automatically converts the returned Java object to JSON.

## How is the Bean Converted to JSON?

- Spring Boot uses **Jackson** for object-to-JSON conversion.
- Jackson introspects the Country object (via getters/setters) and serializes it to a valid JSON string.
- No need for manual conversion — Spring does it automatically behind the scenes.

## Testing the Service

### Request

GET http://localhost:8083/country

### Response

```
{  
  "code": "IN",  
  "name": "India"  
}
```

## Verifying in Tools

### Developer Tools (Browser Network Tab)

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

### Postman Headers Tab

When testing in Postman:

Key	Value
Content-Type	application/json
Content-Length	31
Date	[Current Timestamp]
Server	Apache Tomcat/Embedded

## REST - Get country based on country code

Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

**Controller:** com.cognizant.spring-learn.controller.CountryController

**Method Annotation:** @GetMapping("/countries/{code}")

**Method Name:** getCountry(String code)

**Method Implementation:** Invoke countryService.getCountry(code)

**Service Method:** com.cognizant.spring-learn.service.CountryService.getCountry(String code)

### Service Method Implementation:

- Get the country code using @PathVariable
- Get country list from country.xml
- Iterate through the country list
- Make a case insensitive matching of country code and return the country.
- Lambda expression can also be used instead of iterating the country list

**Sample Request:** http://localhost:8083/country/in

### Sample Response:

```
{
  "code": "IN",
  "name": "India"
}
```

## Implementation Details

### XML Configuration File (country.xml)

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spr
ing-beans.xsd">

    <bean id="countryList" class="java.util.ArrayList">
        <constructor-arg>
```

```

        <list>
            <bean class="com.cognizant.springlearn.model.Country">
                <property name="code" value="IN"/>
                <property name="name" value="India"/>
            </bean>
            <bean class="com.cognizant.springlearn.model.Country">
                <property name="code" value="US"/>
                <property name="name" value="United States"/>
            </bean>
            <!-- Add more countries as needed -->
        </list>
    </constructor-arg>
</bean>
</beans>

```

Model Class: Country.java

```

package com.cognizant.springlearn.model;

public class Country {
    private String code;
    private String name;

    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

```

Service Class: CountryService.java

```

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CountryService {

    public Country getCountry(String code) {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        List<Country> countries = (List<Country>) context.getBean("countryList");

        return countries.stream()

```



```

        .filter(c -> c.getCode().equalsIgnoreCase(code))
        .findFirst()
        .orElse(null);
    }
}

```

## Controller Class: CountryController.java

```

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;
import com.cognizant.springlearn.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
public class CountryController {

    @Autowired
    private CountryService countryService;

    @GetMapping("/countries/{code}")
    public Country getCountry(@PathVariable String code) {
        return countryService.getCountry(code);
    }
}

```

## Configuration: application.properties

```
server.port=8083
```

## Behind the Scenes

- The `@GetMapping("/countries/{code}")` maps HTTP GET requests to the method `getCountry()` in the controller.
- The `@PathVariable` annotation extracts the country code from the URL.
- The service method retrieves the list of countries from `country.xml`.
- It performs a case-insensitive match using `equalsIgnoreCase()` in a stream filter.
- On finding the match, it returns the corresponding `Country` object, which Spring converts into JSON.

## Sample Test

### Request

```
GET http://localhost:8083/countries/in
```

## Response

```
{  
  "code": "IN",  
  "name": "India"  
}
```