

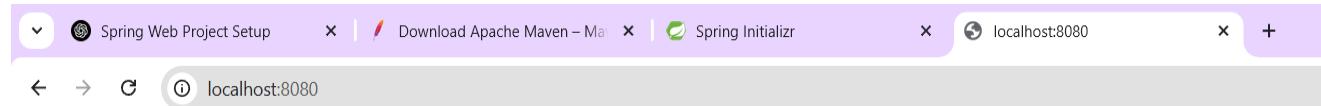
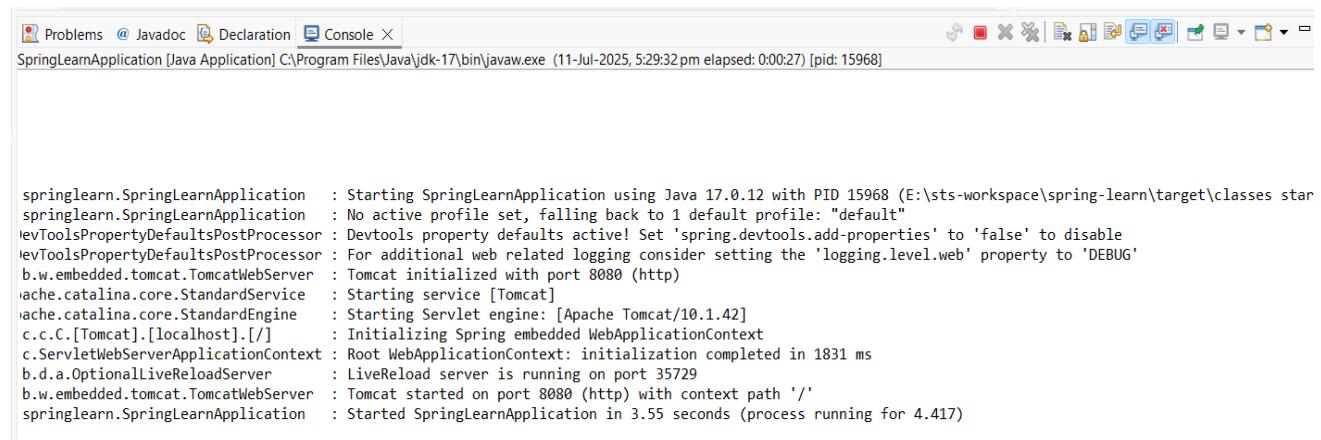
1. SPRING REST HANDSON

Hands on 1

Create a Spring Web Project using Maven

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/vafer/jdependency/2.10
68 kB/s)
[INFO] Replacing main artifact E:\sts-workspace\spring-learn\target\spring-learn-0.0.1-S
ive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to E:\sts-workspace\spring-learn\target\sp
inginal
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 36.135 s
[INFO] Finished at: 2025-07-11T17:23:25+05:30
[INFO] -----
```

E:\sts-workspace\spring-learn>



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

```
Fri Jul 11 17:51:23 IST 2025
There was an unexpected error (type=Not Found, status=404).
No static resource .
org.springframework.web.servlet.resource.NoResourceNotFoundException: No static resource .
    at org.springframework.web.servlet.resource.ResourceHttpRequestHandler.handleRequest(ResourceHttpRequestHandler.java:585)
    at org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter.handle(HttpRequestHandlerAdapter.java:52)
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1089)
    at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)
    at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)
    at jakarta.servlet.http.HttpServlet.service(HttpServletRequest.java:564)
    at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)
    at jakarta.servlet.http.HttpServlet.service(HttpServletRequest.java:658)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:195)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
    at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)
    at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
```

Hands on 4

Spring Core – Load Country from Spring Configuration XML

File name: Country.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Country {
    private static final Logger LOGGER = LoggerFactory.getLogger(Country.class);

    private String code;
    private String name;

    public Country() {
        LOGGER.debug("Inside Country Constructor.");
    }

    public String getCode() {
        LOGGER.debug("Getting country code.");
        return code;
    }

    public void setCode(String code) {
        LOGGER.debug("Setting country code.");
        this.code = code;
    }

    public String getName() {
        LOGGER.debug("Getting country name.");
        return name;
    }

    public void setName(String name) {
        LOGGER.debug("Setting country name.");
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

File name: country.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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="country" class="com.cognizant.springlearn.Country">
```

```
<property name="code" value="IN" />
<property name="name" value="India" />
</bean>

</beans>
```

File name: Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.cognizant</groupId>
  <artifactId>spring-learn</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>spring-learn</name>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>

    <!-- Spring Core and Context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.30</version>
    </dependency>

    <!-- SLF4J API for logging -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>1.7.36</version>
    </dependency>

    <!-- Logback Classic for SLF4J implementation -->
    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>1.2.11</version>
    </dependency>

  </dependencies>
```

```

</project>
File name: SpringLearnApplication.java

package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {
    private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

    public static void main(String[] args) {
        LOGGER.debug("START of main");
        displayCountry();
        LOGGER.debug("END of main");
    }

    public static void displayCountry() {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

        Country country = context.getBean("country", Country.class);
        LOGGER.debug("Country : {}", country.toString());
    }
}

```

OUTPUT

```

<terminated> SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (11-Jul-2025, 8:22:32 pm - 8:22:33 pm elapsed: 0:00:01.299) [pid: 16696]
20:22:33.409 [main] DEBUG c.c.s.SpringLearnApplication - START of main
20:22:33.477 [main] DEBUG o.s.c.s.ClassPathXmlApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@e874448
20:22:33.714 [main] DEBUG o.s.b.f.xml.XmlBeanDefinitionReader - Loaded 1 bean definitions from class path resource [country.xml]
20:22:33.760 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'country'
20:22:33.760 [main] DEBUG com.cognizant.springlearn.Country - Inside Country Constructor.
20:22:33.850 [main] DEBUG com.cognizant.springlearn.Country - Setting country code.
20:22:33.851 [main] DEBUG com.cognizant.springlearn.Country - Setting country name.
20:22:33.871 [main] DEBUG c.c.s.SpringLearnApplication - Country : Country [code=IN, name=India]
20:22:33.873 [main] DEBUG c.c.s.SpringLearnApplication - END of main

```

<bean> Tag

Defines a bean (object) in the Spring container.

```
<bean id="country" class="com.cognizant.springlearn.Country">
```

- id -- unique name for the bean in Spring
- class -- full-qualified class name to be instantiated

<property> Tag

Used to set values of the bean's fields via setter methods.

```
<property name="code" value="IN" />
```

- name matches the setter method name `setCode()`
- value is the string value passed

ApplicationContext

Interface that represents the Spring IoC (Inversion of Control) container.

It is responsible for:

- Loading bean definitions
- Instantiating and wiring beans
- Managing bean life cycle

ClassPathXmlApplicationContext

Implementation of ApplicationContext that loads context from an XML file located in the classpath.

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

This line tells Spring to:

- Look for country.xml in the classpath
- Parse it
- Instantiate all <bean>s defined in it

context.getBean("country", Country.class)

This retrieves the bean named "country" of type Country.

Internally:

1. Spring looks for a bean with id="country"
2. Uses reflection to instantiate the class com.cognizant.springlearn.Country
3. Calls its constructor
4. Invokes setter methods (setCode, setName) with values from XML
5. Returns the fully initialized Country object

2. SPRING REST HANDSON

Hello World RESTful Web Service

File name: HelloController.java

```
package com.cognizant.springlearn.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.info("START");
        String response = "Hello World!!";
        LOGGER.info("END");
        return response;
    }
}
```

```
}
```

File name: SpringLearnApplication.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

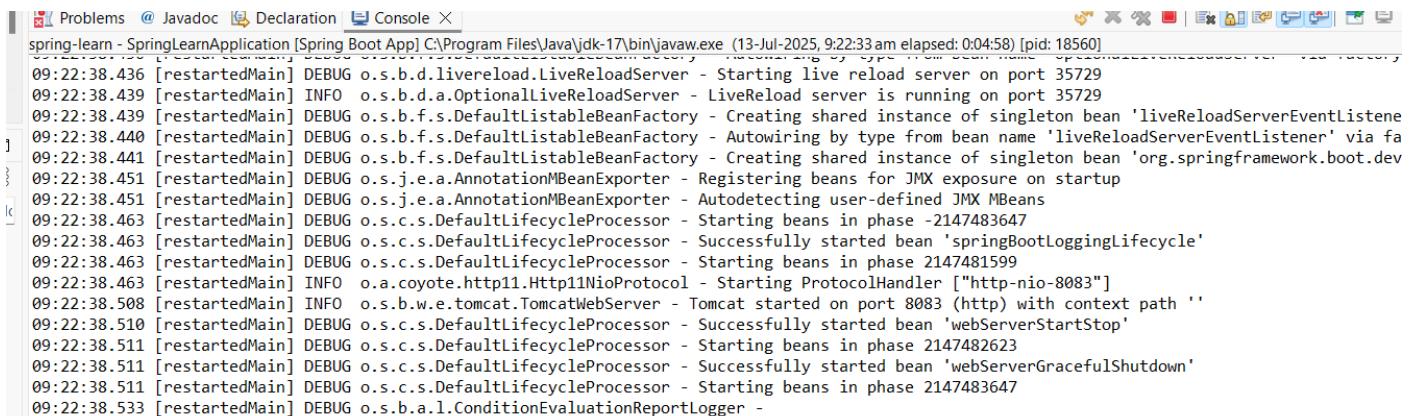
    private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

    public static void main(String[] args) {
        LOGGER.debug("START of main");

        SpringApplication.run(SpringLearnApplication.class, args);

        LOGGER.debug("END of main");
    }
}
```

Output:



```
Problems @ Javadoc Declaration Console
spring-learn - SpringLearnApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (13-Jul-2025, 9:22:33 am elapsed: 0:04:58) [pid: 18560]
09:22:38.436 [restartedMain] DEBUG o.s.b.d.livereload.LiveReloadServer - Starting live reload server on port 35729
09:22:38.439 [restartedMain] INFO o.s.b.d.a.OptionalLiveReloadServer - LiveReload server is running on port 35729
09:22:38.439 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'liveReloadServerEventListener'
09:22:38.440 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Autowiring by type from bean name 'liveReloadServerEventListener' via fa
09:22:38.441 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.boot.dev
09:22:38.451 [restartedMain] DEBUG o.s.j.e.a.AnnotationMBeanExporter - Registering beans for JMX exposure on startup
09:22:38.451 [restartedMain] DEBUG o.s.j.e.a.AnnotationMBeanExporter - Autodetecting user-defined JMX MBeans
09:22:38.463 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase -2147483647
09:22:38.463 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'springBootLoggingLifecycle'
09:22:38.463 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147481599
09:22:38.463 [restartedMain] INFO o.a.coyote.http11.Http11NioProtocol - Starting ProtocolHandler ["http-nio-8083"]
09:22:38.508 [restartedMain] INFO o.s.b.w.e.tomcat.TomcatWebServer - Tomcat started on port 8083 (http) with context path ''
09:22:38.510 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'webServerStartStop'
09:22:38.511 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147482623
09:22:38.511 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'webServerGracefulShutdown'
09:22:38.511 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147483647
09:22:38.533 [restartedMain] DEBUG o.s.b.a.l.ConditionEvaluationReportLogger -
```



localhost:8083/hello

Hello World!!

HTTP <http://localhost:8083/hello>

Save Share ↗

GET <http://localhost:8083/hello>

Send ↘

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description	...	

Body Cookies Headers (5) Test Results | ⏱

200 OK • 190 ms • 177 B • 🌐 | ⚙️

Raw ▾ Preview ⚡ Visualize ▾

```
1 Hello World!!
```

Headers		Preview	Response	Initiator	>>
General					
Request URL	http://localhost:8083/hello				
Request Method	GET				
Status Code	200 OK				
Remote Address	[:1]:8083				
Referrer Policy	strict-origin-when-cross-origin				
Response Headers		<input type="checkbox"/> Raw			
Connection	keep-alive				
Content-Length	13				
Content-Type	text/html;charset=UTF-8				
Date	Sun, 13 Jul 2025 04:00:33 GMT				
Keep-Alive	timeout=60 				
Request Headers		<input type="checkbox"/> Raw			
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7				
Accept-Encoding	gzip, deflate, br, zstd				
Accept-Language	en-US,en;q=0.9				

Body Cookies Headers (5) Test Results 

200 OK • 190 ms • 11

Key	Value
Content-Type	text/plain;charset=UTF-8
Content-Length	13
Date	Sun, 13 Jul 2025 03:53:37 GMT
Keep-Alive	timeout=60
Connection	keep-alive

REST - Country Web Service

File name: SpringLearnApplication.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringLearnApplication {

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

    public static void main(String[] args) {
        LOGGER.debug("START of main");
        SpringApplication.run(SpringLearnApplication.class, args);
        LOGGER.debug("END of main");
    }
}
```

File name: Country.java

```
package com.cognizant.springlearn;

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

    public Country() {
        System.out.println("Inside Country Constructor.");
    }

    public String getCode() {
        return code;
    }
}
```

```

public void setCode(String code) {
    System.out.println("Setting country code.");
    this.code = code;
}

public String getName() {
    return name;
}

public void setName(String name) {
    System.out.println("Setting country name.");
    this.name = name;
}

@Override
public String toString() {
    return "Country [code=" + code + ", name=" + name + "]";
}

```

File name:Country.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<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.Country">
    <property name="code" value="IN"/>
    <property name="name" value="India"/>
</bean>

</beans>

```

File name:CountryController.java

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class CountryController {

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

    @RequestMapping("/country")
    public Country getCountryIndia() {
        LOGGER.info("START");

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

        LOGGER.info("END");
        return country;
    }
}
```

OUTPUT:

```
spring-learn - SpringLearnApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (13-Jul-2025, 9:41:33 am elapsed: 0:00:26) [pid: 27256]
09:41:37.297 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Autowiring by type from bean name 'optionalLiveReloadServer' via factory method
09:41:37.297 [restartedMain] DEBUG o.s.b.d.LiveReloadServer - Starting live reload server on port 35729
09:41:37.299 [restartedMain] DEBUG o.s.b.d.a.OptionalLiveReloadServer - LiveReload server is running on port 35729
09:41:37.299 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'liveReloadServerEventListener'
09:41:37.300 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Autowiring by type from bean name 'liveReloadServerEventListener' via factory method
09:41:37.300 [restartedMain] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.boot.devtools.restart.LifeCycle'
09:41:37.309 [restartedMain] DEBUG o.s.j.e.a.AnnotationMBeanExporter - Registering beans for JMX exposure on startup
09:41:37.309 [restartedMain] DEBUG o.s.j.e.a.AnnotationMBeanExporter - Autodetecting user-defined MBeans
09:41:37.319 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase -2147483647
09:41:37.319 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'springBootLoggingLifecycle'
09:41:37.319 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147481599
09:41:37.319 [restartedMain] INFO o.a.coyote.http11.Http11NioProtocol - Starting ProtocolHandler ["http-nio-8085"]
09:41:37.355 [restartedMain] INFO o.s.b.w.e.tomcat.TomcatWebServer - Tomcat started on port 8085 (http) with context path ''
09:41:37.357 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'webServerStartStop'
09:41:37.358 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147482623
09:41:37.358 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Successfully started bean 'webServerGracefulShutdown'
09:41:37.358 [restartedMain] DEBUG o.s.c.s.DefaultLifecycleProcessor - Starting beans in phase 2147483647
09:41:37.375 [restartedMain] DEBUG o.s.b.a.l.ConditionEvaluationReportLogger -
```

The screenshot shows a browser window with the URL `localhost:8085/country`. Below the address bar, there's a command-line interface with the text `'retty-print`. Underneath, a JSON object is displayed:

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

Below the browser, a terminal window displays a cURL command to make a GET request to the same endpoint:

```
GET http://localhost:8085/country
```

The terminal also shows the JSON response:

```
{}
{"code": "IN",
 "name": "India"} 200 OK
```

At the bottom, there are various status indicators: 390 ms, 192 B, and a globe icon.

Headers		Preview	Response	Initiator	>>				
▼ Response Headers		<input type="checkbox"/> Raw							
<hr/>									
Connection	keep-alive								
Content-Type	application/json								
Date	Sun, 13 Jul 2025 04:12:59 GMT								
Keep-Alive	timeout=60								
Transfer-Encoding	chunked								
Vary	Origin								
Vary	Access-Control-Request-Method								
Vary	Access-Control-Request-Headers								
▼ Request Headers		<input type="checkbox"/> Raw							
<hr/>									
Accept	image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8								
Accept-Encoding	gzip, deflate, br, zstd								
Accept-Language	en-US,en;q=0.9								
Connection	keep-alive								
Host	localhost:8085								
Referer	http://localhost:8085/counter								
Sec-Ch-Ua	"Not)A;Brand";v="8", "Chromium";v="138", "Google Chrome";v="138"								

Body	Cookies	Headers (5)	Test Results	⌚	200 OK	• 390 ms • 192 B • 🌐	...
Key					Value		
Content-Type					application/json		
Transfer-Encoding					chunked		
Date					Sun, 13 Jul 2025 04:12:22 GMT		
Keep-Alive					timeout=60		
Connection					keep-alive		
Response Size		192 B					
Headers		164 B					
Body		28 B					
Request Size		343 B					
Headers		260 B					
Body		83 B					

What happens in the controller method?

1. The controller receives a GET request on /country.
2. It logs START and loads Spring context from country.xml.
3. Retrieves the bean with id in — a Country object representing India.
4. Logs END and returns the bean as the HTTP response.

How is the bean converted into JSON?

- Spring Boot uses Jackson library (auto-included via spring-boot-starter-web) to convert Java objects (POJOs) into JSON format.
- The Country bean has getters, so Jackson serializes the object like this:

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

- This is done automatically due to @RestController, which combines @Controller + @ResponseBody.

REST - Get country based on country code

File name:Country.java

```
package com.cognizant.springlearn;
```

```
public class Country {
```

```
    private String code;
```

```
    private String name;
```

```
    public Country() {}
```

```
    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;
```

```
    }
```

```
}
```

File name: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.Country">
    <property name="code" value="IN"/>
    <property name="name" value="India"/>
</bean>

<bean id="us" class="com.cognizant.springlearn.Country">
    <property name="code" value="US"/>
    <property name="name" value="United States"/>
</bean>

<bean id="countryList" class="java.util.ArrayList">
    <constructor-arg>
        <list>
            <ref bean="in"/>
            <ref bean="us"/>
        </list>
    </constructor-arg>
</bean>
</beans>
```

File name:SpringLearnApplication.java

```
package com.cognizant.springlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```

public class SpringLearnApplication {

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

    public static void main(String[] args) {
        LOGGER.debug("START of main");
        SpringApplication.run(SpringLearnApplication.class, args);
        LOGGER.debug("END of main");
    }
}

```

File name:CountryController.java

```

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;
import com.cognizant.springlearn.service.CountryService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
public class CountryController {

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

    @Autowired
    private CountryService countryService;

    @GetMapping("/countries/{code}")
    public Country getCountry(@PathVariable String code) {
        LOGGER.info("START getCountry with code: {}", code);
        Country result = countryService.getCountry(code);
        LOGGER.info("END");
        return result;
    }
}

```

```
}
```

File name:CountryService.java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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");

        // Find country ignoring case
        return countries.stream()
                .filter(c -> c.getCode().equalsIgnoreCase(code))
                .findFirst()
                .orElse(null); // or throw new CountryNotFoundException(code);
    }
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Method: GET
- URL: http://localhost:8085/countries/in
- Headers: (8)
- Body: (empty)
- Test Results: 200 OK, 70 ms, 192 B
- JSON Response:

```
1  {
2   "code": "IN",
3   "name": "India"
4 }
```

HTTP <http://localhost:8085/countries/us>

Save Share

GET <http://localhost:8085/countries/us> Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description	...	

Body Cookies Headers (5) Test Results 200 OK • 29 ms • 200 B • [...](#)

{ } JSON ▾ Preview Visualize

```
1 {  
2   "code": "US",  
3   "name": "United States"  
4 }
```

localhost:8085/countries/us

Pretty-print

```
{"code": "US", "name": "United States"}
```

localhost:8085/countries/in

Pretty-print

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

What happens in the controller?

- Controller gets the {code} from the URL via @PathVariable
- It calls countryService.getCountry(code)
- The service fetches the countryList from Spring XML
- It performs a case-insensitive match and returns the matching country

How is the bean converted to JSON?

- Spring Boot uses Jackson (auto-configured with spring-boot-starter-web)
- Your Country class is a plain POJO with getters → automatically converted to JSON

Headers		Preview	Response	Initiator	>>
<u>Request Policy</u>					
strict-origin-when-cross-origin					n
<u>Response Headers</u>					
<input type="checkbox"/> Raw					
Connection	keep-alive				
Content-Type	application/json				
Date	Sun, 13 Jul 2025 04:12:59 GMT				
Keep-Alive	timeout=60				
Transfer-Encoding	chunked				
Vary	Origin				
Vary	Access-Control-Request-Method				
Vary	Access-Control-Request-Headers				
<u>Request Headers</u>					
<input type="checkbox"/> Raw					
Accept	image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8				
Accept-Encoding	gzip, deflate, br, zstd				
Accept-Language	en-US,en;q=0.9				
Connection	keep-alive				
Host	localhost:8085				
Referer	http://localhost:8085/count				
Sec-Ch-Ua	"Not)A;Brand";v="8", "Chromium":v="138".				

Body Cookies Headers (5) Test Results ⏱

200 OK • 29 ms • 200 B • |

Key	Value
Content-Type	application/json
Transfer-Encoding	chunked
Date	Sun, 13 Jul 2025 04:32:11 GMT
Keep-Alive	timeout=60
Connection	keep-alive

5. JWT-handson

Create authentication service that returns JWT

Create authentication controller and configure it in SecurityConfig

```
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);
    }
}

package com.cognizant.springlearn.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {

    @Bean
    public InMemoryUserDetailsManager userDetailsService() {
        UserDetails user = User.builder()
            .username("user")
            .password("{noop}pwd") // {noop} = No password encoding
            .roles("USER")
    }
}
```

```
.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

    http.csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/authenticate").permitAll()
            .anyRequest().authenticated()
        )
        .httpBasic();

    return http.build();
}
```

```
package com.cognizant.springlearn.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cognizant.springlearn.util.JwtUtil;

import java.util.Base64;
```

```
import jakarta.servlet.http.HttpServletRequest;
```

```
@RestController

public class AuthenticationController {

    @Autowired
    private JwtUtil jwtUtil;
```

```
@GetMapping("/authenticate")
public ResponseEntity<?> authenticate(HttpServletRequest request) {
    // Extract Authorization header
    String authHeader = request.getHeader("Authorization");

    if (authHeader == null || !authHeader.startsWith("Basic ")) {
        return ResponseEntity.status(401).body("Missing or invalid Authorization header");
    }

    // Decode Base64 credentials
    String base64Credentials = authHeader.substring("Basic ".length());
    String credentials = new String(Base64.getDecoder().decode(base64Credentials));
    String[] values = credentials.split(":", 2);

    String username = values[0];
    String password = values[1];
    System.out.println("🔒 Inside authenticate()");

    // For demo: accept only hardcoded user/pwd
    if ("user".equals(username) && "pwd".equals(password)) {
        String token = jwtUtil.generateToken(username);
        return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");
    } else {
        return ResponseEntity.status(403).body("Invalid credentials");
    }
}

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import org.springframework.stereotype.Component;
```

```

import javax.crypto.SecretKey;
import java.util.Date;

@Component
public class JwtUtil {

    private static final SecretKey SECRET_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);
    private static final long EXPIRATION_TIME = 1000 * 60 * 60; // 1 hour

    public String generateToken(String username) {
        return Jwts.builder()
            .setSubject(username)
            .setIssuedAt(new Date())
            .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
            .signWith(SECRET_KEY)
            .compact();
    }
}

```

OUTPUT

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8090/authenticate` is entered. Below it, a `GET` request is selected. The `Authorization` tab is active, showing `Basic Auth` selected. In the `Username` field, the value `user` is entered. The `Password` field contains `password`, indicated by three dots. The `Send` button is visible at the top right. The bottom section shows the response details: `200 OK`, `419 ms`, `565 B`. The `Body` tab is selected, displaying the JSON response: `{"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUyMzg0MDYzMjleHAi0jE3NTIzODc2NjN9.n0F051VTRt-h-JZikTI2r0Qyb4KhtLE15VSs0vAqe7I"}`.

Read Authorization header and decode the username and password

File name: AuthenticationController.java

```
package com.cognizant.springlearn.controller;

import java.util.Base64;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.web.bind.annotation.*;
import com.cognizant.springlearn.util.JwtUtil;

@RestController
public class AuthenticationController {

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

    private final AuthenticationManager authenticationManager;
    private final JwtUtil jwtUtil;

    public AuthenticationController(AuthenticationManager authenticationManager, JwtUtil jwtUtil) {
        this.authenticationManager = authenticationManager;
        this.jwtUtil = jwtUtil;
    }

    @GetMapping("/authenticate")
    public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.AUTHORIZATION) String authHeader) {
        LOGGER.debug("START of authenticate()");
        return null;
    }
}
```

```

String username = getUser(authHeader);
LOGGER.debug("Extracted Username: {}", username);

try {
    // The actual authentication step (optional since JWT is already returned)
    Authentication auth = new UsernamePasswordAuthenticationToken(username, "pwd");
    authenticationManager.authenticate(auth);
} catch (AuthenticationException e) {
    LOGGER.error("Authentication failed for user: {}", username);
    return ResponseEntity.status(401).build();
}

String token = jwtUtil.generateToken(username);
LOGGER.debug("Generated JWT Token");

LOGGER.debug("END of authenticate()");
return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");
}

// 🔑 This is the method to decode and extract username from Base64 encoded Authorization header
private String getUser(String authHeader) {
    LOGGER.debug("START of getUser()");
    if (authHeader == null || !authHeader.startsWith("Basic ")) {
        LOGGER.error("Invalid Authorization header: {}", authHeader);
        return null;
    }

    String encodedCredentials = authHeader.substring("Basic ".length());
    byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);
    String decodedCredentials = new String(decodedBytes); // will be user:pwd

    LOGGER.debug("Decoded credentials: {}", decodedCredentials);
}

```

```

String username = decodedCredentials.split(":")[0];

LOGGER.debug("Extracted Username from Authorization header: {}", username);
LOGGER.debug("END of getUser()");

return username;
}
}

```

OUTPUT:

```

Problems @ Javadoc Declaration Console X
spring-learn - SpringLearnApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (13-Jul-2025, 10:28:18am elapsed: 0:33:23) [pid: 5172]
11:00:04.763 [http-nio-8090-exec-3] DEBUG o.s.security.web.FilterChainProxy - Securing GET /authenticate
11:00:04.888 [http-nio-8090-exec-3] DEBUG o.s.s.a.d.DaoAuthenticationProvider - Authenticated user
11:00:04.888 [http-nio-8090-exec-3] DEBUG o.s.s.w.a.w.BasicAuthenticationFilter - Set SecurityContextHolder to UsernamePasswordAuthenticationToken [Prin
11:00:04.888 [http-nio-8090-exec-3] DEBUG o.s.security.web.FilterChainProxy - Secured GET /authenticate
11:00:04.889 [http-nio-8090-exec-3] DEBUG o.s.web.servlet.DispatcherServlet - GET "/authenticate", parameters={}
11:00:04.889 [http-nio-8090-exec-3] DEBUG o.s.w.s.m.m.a.RequestMappingHandlerMapping - Mapped to com.cognizant.springlearn.controller.AuthenticationCont
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - START of authenticate()
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - START of getUser()
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - Decoded credentials: user:pwd
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - Extracted Username from Authorization header: user
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - END of getUser()
11:00:04.889 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - Extracted Username: user
11:00:05.009 [http-nio-8090-exec-3] DEBUG o.s.s.a.d.DaoAuthenticationProvider - Authenticated user
11:00:05.011 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - Generated JWT Token
11:00:05.011 [http-nio-8090-exec-3] DEBUG c.c.s.c.AuthenticationController - END of authenticate()
11:00:05.012 [http-nio-8090-exec-3] DEBUG o.s.w.s.m.m.a.HttpEntityMethodProcessor - Using 'text/plain', given ['/*'] and supported [text/plain, /*], appl
11:00:05.012 [http-nio-8090-exec-3] DEBUG o.s.w.s.m.m.a.HttpEntityMethodProcessor - Writing [{"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VylwiawF0Ijo
11:00:05.013 [http-nio-8090-exec-3] DEBUG o.s.web.servlet.DispatcherServlet - Completed 200 OK

```