

Hands on 1

Create a Spring Web Project using Maven

SpringLearnApplication.java:

```
package com.cognizant.spring_learn;
```

```
import org.springframework.boot.SpringApplication;
```

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

```
@SpringBootApplication
```

```
public class SpringLearnApplication {
```

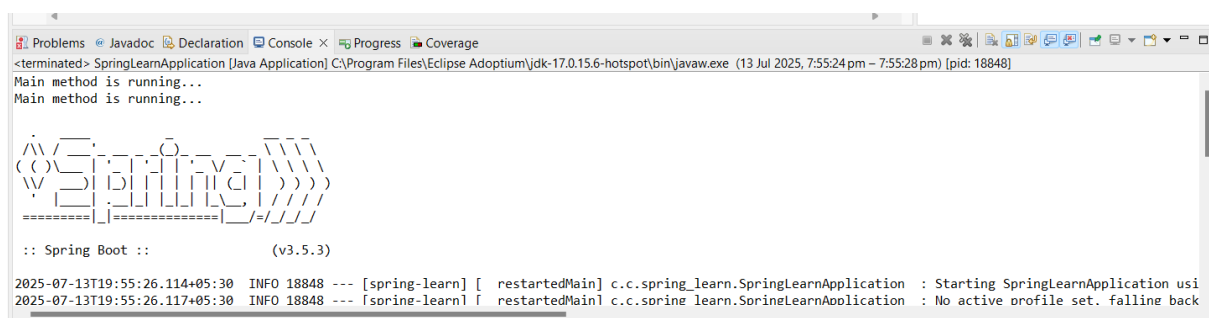
```
    public static void main(String[] args) {
```

```
        System.out.println("Main method is running...");
```

```
        SpringApplication.run(SpringLearnApplication.class, args);
```

```
    }
```

```
}
```



@SpringBootApplication

- Combines:
 - @Configuration

- @EnableAutoConfiguration
- @ComponentScan
- Bootstraps the entire Spring context.

pom.xml

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

Spring Core – Load Country from Spring Configuration XML

An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country.

SpringLearnApplication.java

package com.cognizant.spring_learn;

```

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");

        displayCountry();

        LOGGER.debug("END");

    }


    public static void displayCountry() {

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

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

        LOGGER.debug("Country : {}", country.toString());

        System.out.println("Country: " + country); // Optional

    }

}

Country.java
package com.cognizant.spring_learn;

```

```
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("Inside getCode");
        return code;
    }

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

    public String getName() {
        LOGGER.debug("Inside getName");
        return name;
    }
}
```

```
}
```

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

```
@Override
```

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

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<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>  
    <parent>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-parent</artifactId>  
        <version>3.5.3</version>  
        <relativePath/> <!-- lookup parent from repository -->  
    </parent>  
    <groupId>com.cognizant</groupId>
```

```
<artifactId>spring-learn</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring-learn</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
    <license/>
</licenses>
<developers>
    <developer/>
</developers>
<scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
</scm>
<properties>
    <java.version>17</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

```
<!-- SLF4J simple logger (for console output) -->
```

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.7.36</version>
</dependency>
```

```
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.36</version>
    </dependency>
```

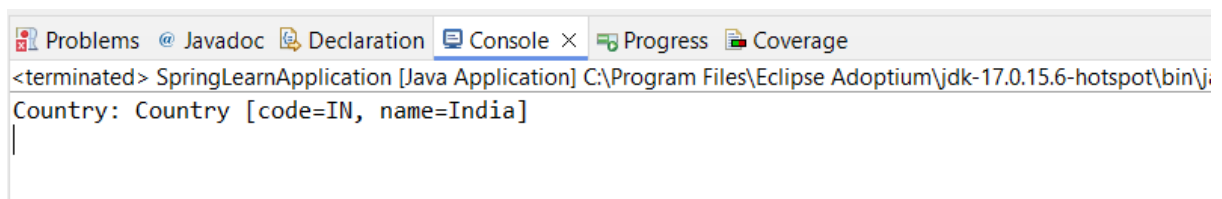
```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
```

```
<version>1.7.36</version>
</dependency>

</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

A screenshot of an IDE's console window. The window has a title bar with tabs for 'Problems', 'Javadoc', 'Declaration', 'Console' (which is active), 'Progress', and 'Coverage'. The console output shows the application has terminated. The text in the console is: '<terminated> SpringLearnApplication [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.15.6-hotspot\bin\j. Country: Country [code=IN, name=India]'. There is a vertical cursor at the end of the line.

```
<terminated> SpringLearnApplication [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.15.6-hotspot\bin\j.
Country: Country [code=IN, name=India]
|
```

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!!

pom.xml

```
<!-- Spring Web for REST Controller -->
```

```
<dependency>
```

```
    <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-web</artifactId>
```

```
</dependency>
```

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 {
```

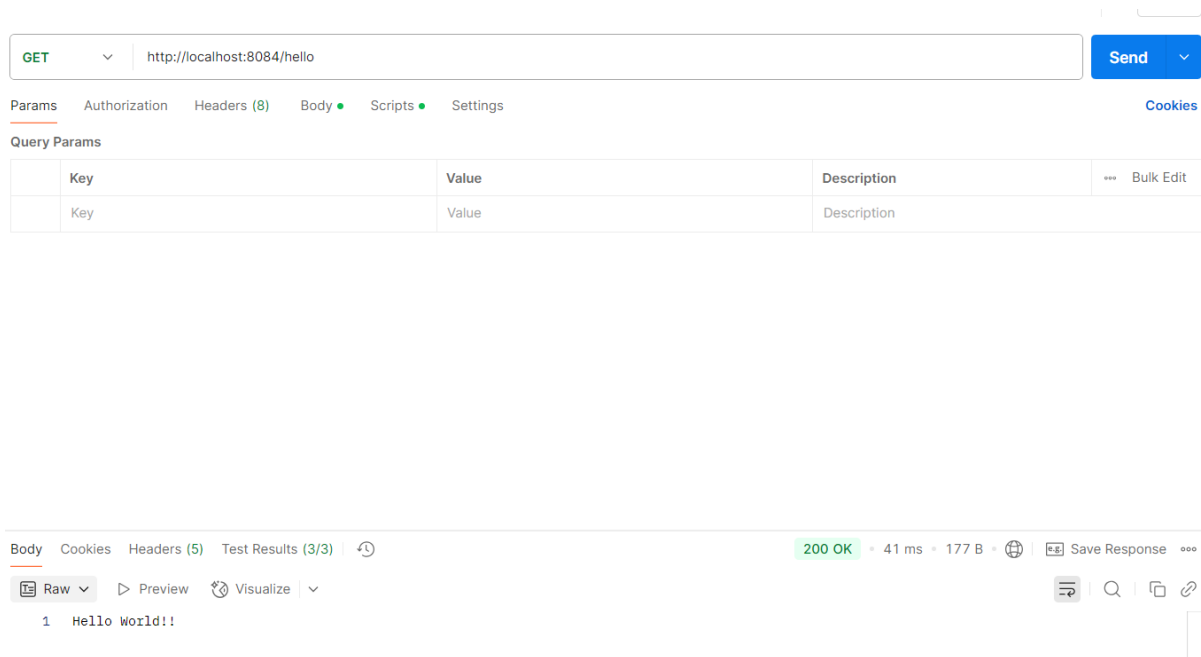
```
    private static final Logger LOGGER =  
    LoggerFactory.getLogger(HelloController.class);
```

```
    @GetMapping("/hello")
```

```
    public String sayHello() {
```

```
        LOGGER.debug("START");  
        String message = "Hello World!!";  
        LOGGER.debug("END");  
        return message;  
    }  
}  
  
SpringLearnApplication.java  
package com.cognizant.springlearn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication(scanBasePackages = "com.cognizant.springlearn")  
public class SpringLearnApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(SpringLearnApplication.class, args);  
    }  
}
```





REST - Country Web Service

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

SpringLearnApplication.java

```
package com.cognizant.spring_learn;
```

```
import org.springframework.boot.SpringApplication;
```

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

```
@SpringBootApplication(scanBasePackages = "com.cognizant.spring_learn")
```

```
public class SpringLearnApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(SpringLearnApplication.class, args);
```

```
    }
```

```
}
```

CountryController.java:

```
package com.cognizant.spring_learn.controller;
```

```
import com.cognizant.spring_learn.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.debug("START");

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

        LOGGER.debug("Country: {}", country);
        LOGGER.debug("END");

        return country;
    }
}

```

Country.java:

```

package com.cognizant.spring_learn;

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

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

    public String getCode() {
        System.out.println("Inside getCode");
        return code;
    }

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

    public String getName() {
        System.out.println("Inside getName");
        return name;
    }
}

```

```

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

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

```

The screenshot shows a REST client interface with a GET request to `http://localhost:8084/country`. The response is a 200 OK status with a 36 ms response time and 192 B of data. The response body is displayed in JSON format:

```

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

```

The interface also shows tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The Body tab is currently selected, showing the JSON response. There are also buttons for Send, Cookies, Bulk Edit, Preview, and Visualize.

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="country" class="com.cognizant.springlearn.Country">
        <property name="code" value="IN" />
        <property name="name" value="India" />
    </bean>

</beans>

```

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.

```
<?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="countryList" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean class="com.cognizant.spring_learn.Country">
                    <property name="code" value="IN"/>
                    <property name="name" value="India"/>
                </bean>
                <bean class="com.cognizant.spring_learn.Country">
                    <property name="code" value="US"/>
                    <property name="name" value="United States"/>
                </bean>
                <bean class="com.cognizant.spring_learn.Country">
                    <property name="code" value="DE"/>
                    <property name="name" value="Germany"/>
                </bean>
                <bean class="com.cognizant.spring_learn.Country">
                    <property name="code" value="JP"/>
                    <property name="name" value="Japan"/>
                </bean>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

```
package com.cognizant.spring_learn.controller;
```

```
import com.cognizant.spring_learn.Country;
```

```
import com.cognizant.spring_learn.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.debug("START");

        Country country = countryService.getCountry(code);

        LOGGER.debug("Country: {}", country);
        LOGGER.debug("END");

        return country;
    }
}

package com.cognizant.spring_learn.service;

import com.cognizant.spring_learn.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> countryList = (List<Country>) context.getBean("countryList");

        // Case-insensitive match using lambda
        return countryList.stream()
            .filter(c -> c.getCode().equalsIgnoreCase(code))
            .findFirst()
            .orElse(null); // or throw custom exception
    }
}

package com.cognizant.spring_learn;

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

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

```

```

public String getCode() {
    System.out.println("Inside getCode");
    return code;
}

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

public String getName() {
    System.out.println("Inside getName");
    return name;
}

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

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

```

The screenshot shows a REST client interface. At the top, a GET request is configured for the URL `http://localhost:8084/countries/in`. Below the URL bar, there are tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The Params tab is active, showing a table with two columns: Key and Value. The table is empty. Below the Params tab, there is a section for Query Params, also with a table that is empty. At the bottom, the Body tab is active, showing the response in JSON format. The response is a 200 OK status with a response time of 37 ms and a body size of 192 B. The JSON response is: `{ "code": "IN", "name": "India" }`.

Key	Value	Description
Key	Value	Description

Key	Value	Description
Key	Value	Description

200 OK • 37 ms • 192 B

```

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

```


Create authentication service that returns JWT

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

```
package com.cognizant.spring_learn.controller;

import com.cognizant.spring_learn.util.JwtUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import jakarta.servlet.http.HttpServletRequest;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

@RestController
public class AuthenticationController {

    @Autowired
    private JwtUtil jwtUtil;

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

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

        try {
            String base64Credentials = authHeader.substring("Basic ".length());
            byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);
            String decoded = new String(decodedBytes);
            String[] parts = decoded.split(":", 2);

            String username = parts[0];
            String password = parts[1];

            // Replace with real authentication
            if ("user".equals(username) && "pwd".equals(password)) {
                String token = jwtUtil.generateToken(username);

                Map<String, String> response = new HashMap<>();
                response.put("token", token);

                return ResponseEntity.ok(response);
            } else {
```

```

        return ResponseEntity.status(401).body("Invalid credentials");
    }

    } catch (Exception e) {
        return ResponseEntity.status(500).body("Authentication failed");
    }
}

package com.cognizant.spring_learn.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;

@Configuration
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .anyRequest().permitAll()
            );

        return http.build();
    }
}

package com.cognizant.spring_learn.util;

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

import java.security.Key;
import java.util.Date;

@Component
public class JwtUtil {

    private static final Key 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(System.currentTimeMillis()))

```

```

        .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
        .signWith(SECRET_KEY)
        .compact();
    }
}

```

```

<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.cognizant</groupId>
  <artifactId>spring-learn</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>spring-learn</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
  </dependencies>
</project>

```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<!-- Spring Web for REST Controller -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
</dependency>

<!-- Spring Security for JWT authentication -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>

<!-- JSON Web Token support -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.5</version>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.5</version>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.5</version>
    <scope>runtime</scope>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

</project>

GET

http://localhost:8084/authenticate

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

Cookies

Auth Type

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [Basic Auth](#) authorization.

Username

user

Password

pwd

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Body

Cookies

Headers (11)

Test Results (3/3)

200 OK

111 ms

475 B

Save Response

JSON

Preview

Visualize

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzU4LCJleHAiOjE3NTI0Mjc5NTh9.s1H3AoyMH1B7C0n68m8f9qM7Qqfsf6TeDCBA1eYPT64"
3 }
```