Hands on 1

Create a Spring Web Project using Maven

Follow steps below to create a project:

1. Go to https://start.spring.io/
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip

   After all the given then, add dependencies and generate the file.

6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line

   I don’t have any proxy id so I have downloaded a binary maven zin for the project.

   Then which we have generated a zip file that I had import to eclipse, the below is the step

8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.

   Already the class SpringLearnApplication is present, I just run that as java application.

10. Run the SpringLearnApplication class.

SME to walk through the following aspects related to the project created:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.

5. Purpose of @SpringBootApplication annotation
6. pom.xml
    1. Walkthrough all the configuration defined in XML file
    2. Open 'Dependency Hierarchy' and show the dependency tree.\

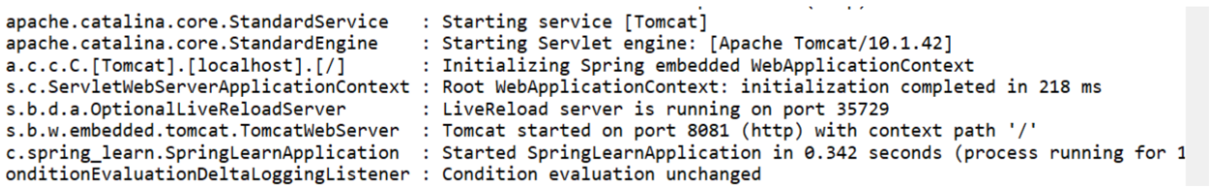
SpringLearnApplication.java:

```
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) {

                SpringApplication.run(SpringLearnApplication.class, args);

        }

}
```

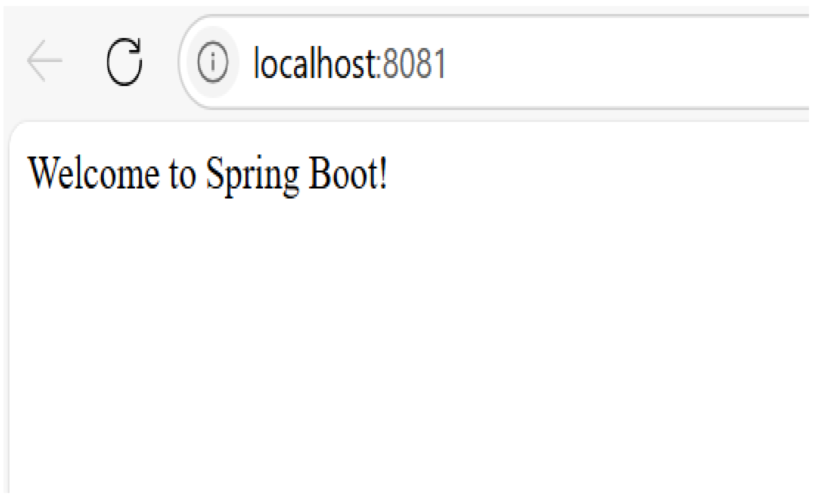
HelloController.java:

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

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

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

@RestController

public class HelloController {

  @GetMapping("/")

  public String home() {

    return "Welcome to Spring Boot!";

  }

}
```

Output:



```
apache.catalina.core.StandardService   : Starting service [Tomcat]
apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 218 ms
s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8081 (http) with context path '/'
c.spring_learn.SpringLearnApplication  : Started SpringLearnApplication in 0.342 seconds (process running for 1
onditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
```


localhost:8081

Welcome to Spring Boot!

Hands on 4

Spring Core – Load Country from Spring Configuration XML

Country.java:

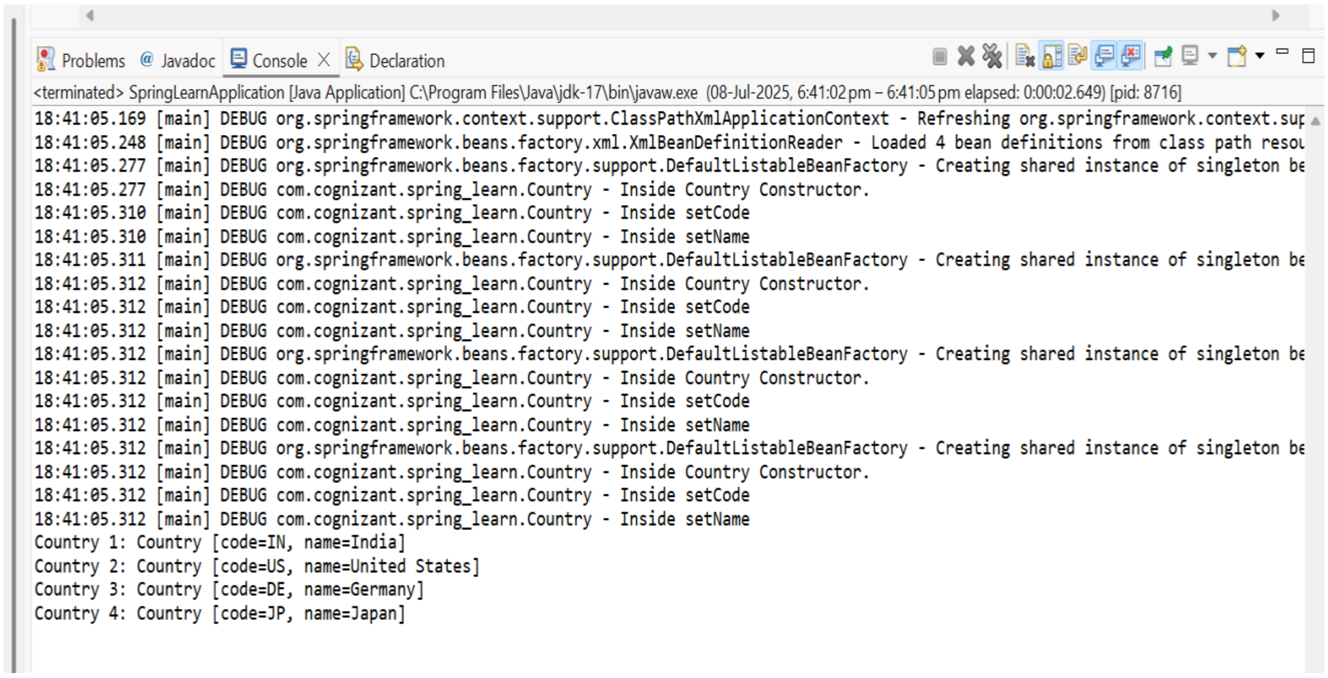
```
package com.cognizant.spring_learn;

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 + "]";
    }
}
```

SpringLearnApplication.java:

```
package com.cognizant.spring_learn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
@SpringBootApplication
public class SpringLearnApplication {
    private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);
    public static void main(String[] args) {
        displayCountry();
    }
    public static void displayCountry() {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country1 = context.getBean("country", Country.class);
        Country country2 = context.getBean("country2", Country.class);
        Country country3 = context.getBean("country3", Country.class);
        Country country4 = context.getBean("country4", Country.class);
        System.out.println("Country 1: " + country1);
        System.out.println("Country 2: " + country2);
        System.out.println("Country 3: " + country3);
        System.out.println("Country 4: " + country4);
    }
}
```

Output:



```
<terminated> SpringLearnApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (08-Jul-2025, 6:41:02 pm – 6:41:05 pm elapsed: 0:00:02.649) [pid: 8716]
18:41:05.169 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Refreshing org.springframework.context.sup
18:41:05.248 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 4 bean definitions from class path resou
18:41:05.277 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton be
18:41:05.277 [main] DEBUG com.cognizant.spring_learn.Country - Inside Country Constructor.
18:41:05.310 [main] DEBUG com.cognizant.spring_learn.Country - Inside setCode
18:41:05.310 [main] DEBUG com.cognizant.spring_learn.Country - Inside setName
18:41:05.311 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton be
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside Country Constructor.
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setCode
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setName
18:41:05.312 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton be
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside Country Constructor.
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setCode
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setName
18:41:05.312 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton be
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside Country Constructor.
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setCode
18:41:05.312 [main] DEBUG com.cognizant.spring_learn.Country - Inside setName
Country 1: Country [code=IN, name=India]
Country 2: Country [code=US, name=United States]
Country 3: Country [code=DE, name=Germany]
Country 4: Country [code=JP, name=Japan]
```

Hello World RESTful Web Service

HelloController.java:

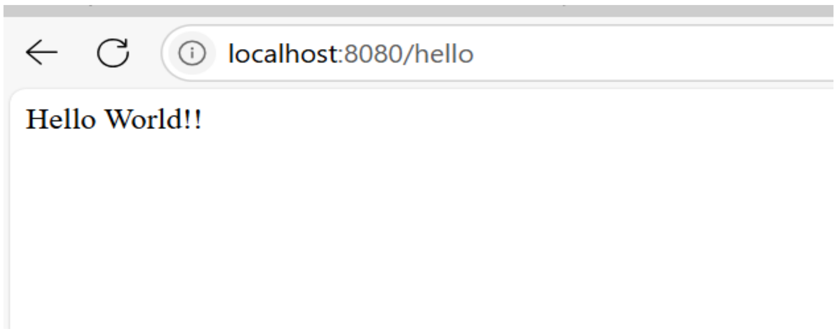
```
package com.cognizant.spring_learn.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
    private static final Logger LOGGER = LoggerFactory.getLogger(HelloController.class);
    @GetMapping("/hello")
    public String sayHello() {
        LOGGER.debug("START sayHello()");
        String message = "Hello World!!";
        LOGGER.debug("END sayHello()");
        return message;
    }
}
```

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

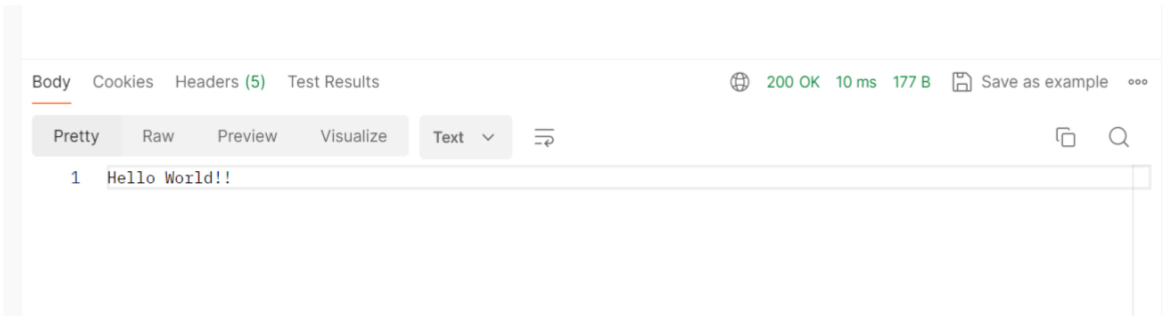
Output:

From the browser,



```
Hello World!!
```

From postman,



```
Hello World!!
```

REST - Country Web Service

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.info("START getCountryIndia");
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country = (Country) context.getBean("country", Country.class);
        LOGGER.info("END getCountryIndia");
        return country;
    }
}
```

SpringLearnApplication.java:

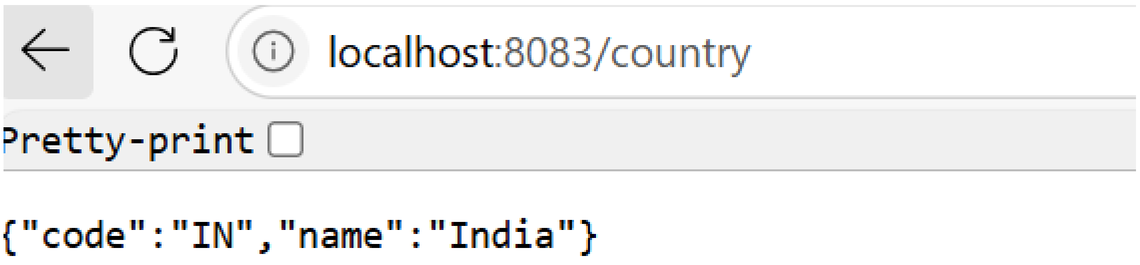
```
package com.cognizant.spring_learn;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringLearnApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringLearnApplication.class, args);
    }
}
```

Country.java:

```
package com.cognizant.spring_learn;
public class Country {
    private String code;
    private String name;
    // Getters and setters
    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;
    }
}
```

Output:

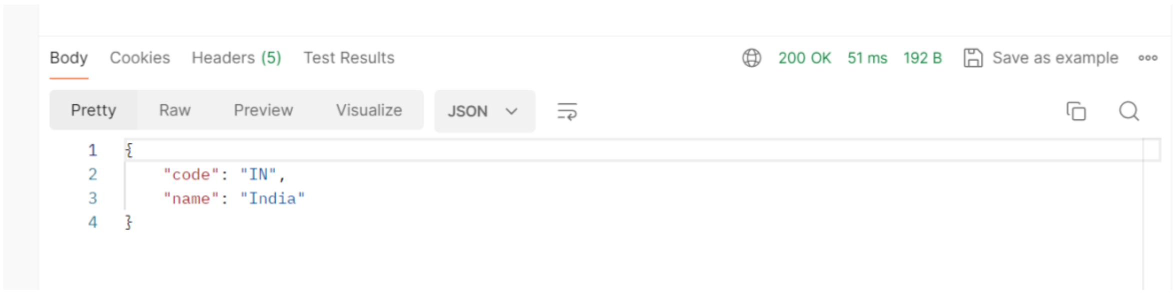
From browser,


localhost:8083/country

Pretty-print

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

From postman,



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

REST - Get country based on country code

Country.java:

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

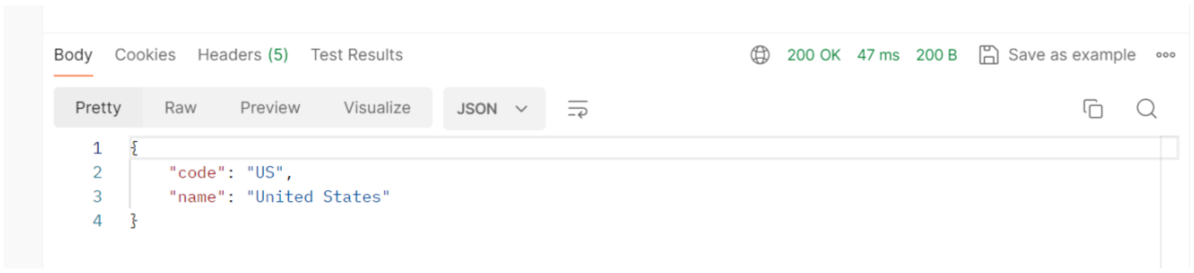
CountryService.java:

```
package com.cognizant.spring_learn.service;
import com.cognizant.spring_learn.Country;
import org.springframework.stereotype.Service;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.List;
@Service
public class CountryService {
    public Country getCountry(String code) {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        List<Country> countryList = context.getBean("countryList", List.class);
        return countryList.stream()
                .filter(country -> country.getCode().equalsIgnoreCase(code))
                .findFirst()
                .orElse(null);
    }
}
```

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

Output:



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

Create authentication service that returns JWT

JwtUtil.java:

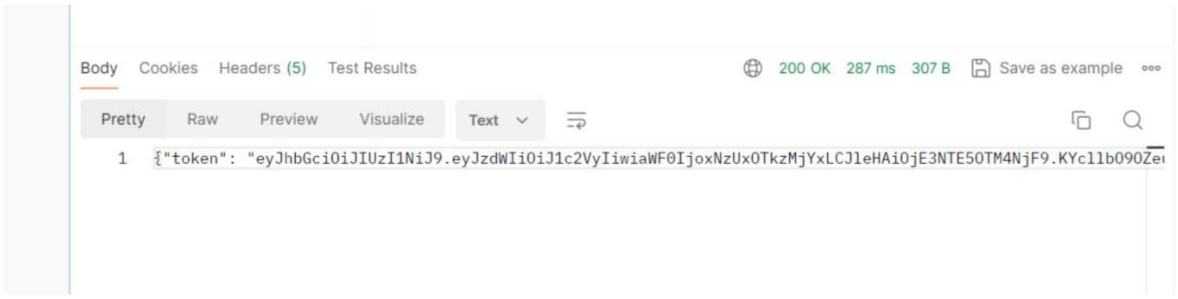
```
package com.cognizant.spring_learn.util;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import java.util.Date;
import java.security.Key;
import org.springframework.stereotype.Component;
@Component
public class JwtUtil {
    private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256); // Generates random key
    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 10)) // 10 min expiry
                .signWith(key)
                .compact();
    }
}
```

AuthenticationController.java:

```
package com.cognizant.spring_learn.controller;
import com.cognizant.spring_learn.util.JwtUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.Base64;
import jakarta.servlet.http.HttpServletRequest;
```

```
@RestController
public class AuthenticationController {
    @Autowired
    private JwtUtil jwtUtil;
    @RequestMapping("/authenticate")
    public String authenticate(HttpServletRequest request) {
        String authHeader = request.getHeader("Authorization");
        if (authHeader != null && authHeader.startsWith("Basic")) {
            // Decode username and password from Basic Auth
            String base64Credentials = authHeader.substring("Basic".length()).trim();
            byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);
            String credentials = new String(decodedBytes);
            String[] values = credentials.split(":", 2);
            String username = values[0];
            String password = values[1];
            // Here you can check against DB or in-memory values
            if ("user".equals(username) && "pwd".equals(password)) {
                String token = jwtUtil.generateToken(username);
                return "{\"token\": \"" + token + "\"}";
            }
        }
        throw new RuntimeException("Invalid credentials");
    }
}
```

Output:



```
{"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUxOTkzMjYxLCJleHAiOjE3NTE5OTM4NjF9.KYcllbO9OZe
```

{

"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUxOTkzMjYxLCJleHAiOjE3NTE5OTM4NjF9.KYcllbO9OZeudvAskGdWyOl3zu2qkZ9CnSDBLkYa00c"

}

This is the token that extracted from the postman.