Single User in Spring Oauth2 to secure REST

Use single user (default user "user"). set password in application.properties

Creating a new Spring Boot project with Initializr

Fill up the basic information and select the following dependencies: Web, Cloud Oauth2, Security, JPA and H2

Take a look at your dependencies in the `pom.xml` file.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-oauth2</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Saying ‘Hello World’ with the RestController

Finally, let’s get our hands dirty. First, we need to provide a public URL that will print “Hello World” when a Get Requests is received. It is extremely easy to create this kind of resource on Spring, all you’ll have to do is annotate a class with `@RestController` and set the URL using `@RequestMapping("/")` annotations. You can even specify the exact type of request that the resource should receive, using `@PostRequest`, `@PutRequest`, `@DeleteRequest` and so on. It is also possible to define the headers, parameters, what kind of data the request consumes and produces. You can take a look at the documentation to get a better understanding.

```
package com.york.demo.controller;

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


@RestController
public class GeneralController {
    @RequestMapping("/")
    public String home() {
        return "Hello World";
    }

    @RequestMapping("/api/user")
    public String apiUser() {
        return "Hello /api/user";
    }

    @RequestMapping("/api/admin")
    public String apiAdmin() {
        return "Hello /api/admin";
    }
}
```

If you run the application now and call the command `curl localhost:8080/`, you’ll receive a **401 Status, “Full authentication is required to access this resource”**. Even though the resource is already available, it is still closed to external access. Out of the box, the `spring-boot-starter-security` dependency will turn “on” some security configuration:

- It automatically creates a user, called “user” and with a random defined password.
- Ignored (insecure) paths for common static resource locations (`/css/**`, `/js/**`, `/images/**`, `/webjars/**` and `**/favicon.ico`)
- HTTP Basic security for all other endpoints
- Common low-level features (HSTS, XSS, CSRF, caching) provided by Spring Security are on by default

All those options can be turned “off” using the `@EnableWebSecurity` annotation and providing an `@Bean` of type `WebSecurityConfigurerAdapter` that will provide many customization options. So, to access our `home()` endpoint, we’ll need to authenticate a valid user. Firlet’slets define a static password for the user. On the `application.properties` file add:

```
#Spring Security will create default user. Here is the password for the default user
```

```
# curl user:password@localhost:8080/api/user
# curl user:password@localhost:8080/api/admin
# curl user:password@localhost:8080/
security.user.password=password123!
```

Now run the application again and call the endpoint with the username and password:

```
curl user:password123!@localhost:8080/api/user
```

```
Base 64 encode "user:password123!" = "dXNlcjpwYXNzd29yZDEyMyE="
```

```
curl -H "AUthorization: Basic dXNlcjpwYXNzd29yZDEyMyE=" localhost:8080/api/user
```

Finally, we'll receive the classic "Hello World"!