Unierzitet u Sarajevu
Elektrotehnički fakultet
Računarstvo i informatika

Dokumentacija za zadatak 2

Predmet: Praktikum - Napredne web tehnologije

Predmetni nastavnik:
Doc. dr Vensada Okanović

Student:
Amar Panjeta

Sarajevo, mart 2017.

POJEDINAČNA DOKUMENTACIJA ZA ZADATAK

1. U sklopu ovog zadatka su implementirane custom REST metode za dva kontrolera unutar modula UsersModule i CompilerModule.

Unutar UsersModula u kontroleru UsersController su dodane metode *register i login*, koje ce biti prikazane na sljedećim slikama:

```
@RequestMapping("/register")
public void register(@RequestBody UserRegisterBody user) throws ServletException{

    if(user.email==null || user.email.isEmpty()||user.username==null || user.username.isEmpty()||
            user.password==null || user.password.isEmpty()){
        throw new ServletException("Polja nisu popunjena!");
    }

    int numberOfUsers=ur.userexists(user.username, user.email);
    if(numberOfUsers>0) throw new ServletException("Username ili email su zauzeti!");

    RegisteredUser newUser= new RegisteredUser();
    newUser.setUsername(user.username);
    newUser.setEmail(user.email);
    // OVDJE TREBA HASHIRATI PASSWORD
    newUser.setPassword(HashService.hashPassword(user.password));
    newUser.setVerified(false);
    ur.save(newUser);

}
```

Slika 1 - Register metoda

```
@RequestMapping("/login")
public String login(@RequestBody UserLoginBody login) throws ServletException{

    RegisteredUser user=ur.findUserByUsername(login.username);

    if(!HashService.checkPassword(login.password, user.getPassword())) throw new ServletException("Netacna pristupna sifra");
    else return "some token";
}
```

Slika 2 - Login metoda

Ove metode primaju standardizirane ulaze, pa su za te svrhe napravljene dvije reprezentacijske klase koje primaju metode. One su prikazane na slici ispod:

```
@SuppressWarnings("unused")
private static class UserRegisterBody{
    public String username;
    public String password;
    public String email;
}

@SuppressWarnings("unused")
private static class UserLoginBody{
    public String username;
    public String password;
}
```

Slika 3 - Reprezentacijske klase

Također, za svrhe ovih metoda je implementiran servis koji služi za heširanje i ispitivanje heširane šifre.

```
public class HashService {
    private static int workload = 12;

    public static String hashPassword(String rawPassword) {
        String salt = BCrypt.gensalt(workload);
        String hashed_password = BCrypt.hashpw(rawPassword, salt);
        return(hashed_password);
    }


    public static boolean checkPassword(String rawPassword, String hashedPassword) {
        boolean password_verified = false;
        if(null == hashedPassword || !hashedPassword.startsWith("$2a$"))
            throw new java.lang.IllegalArgumentException("Invalid hash provided for comparison");
        password_verified = BCrypt.checkpw(rawPassword, hashedPassword);
        return(password_verified);
    }

}
```

Slika 4 -Servis za heširanje

Unutar CompilerModule u kontroleru CodeController su dodane metode *gccVersion i activeThreads*, koje služe sa provjeru gcc verzije na serveru i broja aktivnih thread-ova, respektivno. Metode su prikazane na sljedećoj slici:

```java
@RequestMapping("gccversion")
public String gccVersion() throws ServletException{
    Runtime r=Runtime.getRuntime();
    ProcessBuilder pb=new ProcessBuilder("g++","--version");
    String result;
    try {
        Process p = pb.start();
        p.waitFor();
        InputStream is=p.getInputStream();
        Scanner s=new Scanner(is).useDelimiter("\\A");
        result=s.hasNext()? s.next():"";

    } catch (Exception e) {
        throw new ServletException("Greska!");
    }
    return result;
}

@RequestMapping("threads")
public String activeThreads(){
    return "Broj trenutno aktivnih tredova unutar JVMa:"+Integer.toString(Thread.activeCount());
}
```

Slika 5 - gccVersion i activeThreads metode

2. Za centralizovanu konfiguraciju je prvo kreiran novi modul za konfiguracijski server, ConfigurationServerModule. Ovaj modul je koristi sljedeće postavke za application.properties:

```
server.port=8888

#spring.cloud.config.server.git.uri=${HOME}/Desktop/gitrepo
spring.cloud.config.server.git.uri=https://github.com/AmarPanjeta/NWTCentralizedConf.git
#security.basic.enabled=false
security.user.name=root
security.user.password=s3cr3t
security.ignored=/encrypt,/decrypt
encrypt.key-store.location=classpath:/config-server.jks
encrypt.key-store.password=my-s70r3-s3cr3t
encrypt.key-store.alias=config-server-key
encrypt.key-store.secret=my-k34-s3cr3t
```

Slika 6 - application.properties modula

Može se primjetiti da je modul podešen da posmatra udaljeni repozitorij na github. Ovaj repozitorij je napravljen za postavljanje konfiguracijskih fajlova za sve ostale module. Kako bi klijenti pristupili ovom modulu, potrebno je da koriste username i password root i s3cr3t. Također, podešeno je korištenje asimetričnog ključa za enkripciju i dekripciju podataka.

Na klijentima UsersModule i CompilerModule su napravljeni sljedeći bootstrap.properties fajlovi, koji određuju postavke za pristup konfiguracijskom serveru od strane klijenata:

```
spring.application.name=users-client

spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=root
spring.cloud.config.password=s3cr3t
management.security.enabled=false
```

Slika 7 - UsersModule bootstrap.properties

```
spring.application.name=compiler-client

spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=root
spring.cloud.config.password=s3cr3t
management.security.enabled=false
```

Slika 8 - CompilerModule bootstrap.properties

Za ove module su napravljene odvojene konfiguracije *users-client.properties i compiler-client.properties*, koje su postavljene na repozitorij koji posmatra konfiguracijski server.

users-client.properties

```
server.port=8081

spring.jpa.database=POSTGRESQL
spring.datasource.platform=postgres
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.database.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/nwtusersmodule
spring.datasource.username=postgres
spring.datasource.password={cipher}AgBdiJMbK8a8aHgzkxkL6n+Y4wF/VgmQQxQxeAgKRtG03vbWR9ZmkPZdRBzsGy3TDSd66/
ry80mvD4RT9ndmSoBMIIlvHwmb8u8FbSaf9Gj7/23eUiB6griErJXILAjtBu8O7poOS01SegwJNFizs/
Sf2lAsX2oglym7k6OEK7SYHLOyufrLAI5tq1lEAdTquJB13ny96dBAXYGeyrGSD8u/dEt3FTW+nN7YXKAUyiqSboAoGgA7GczI2kF9UMLdxOgxOwJP7uvh//
vgMbEALsfbWmZvtHBkY3SGtmZp6+kMDI0q2iPHBQFf987KePigfzxCb6D21X7vaMe02Q/TI4kykA4ofhmsjrlwbbAtTjAU10JyzoAfntYQs8SZuAGboG650gdIUaiofovVMtaT2Gd/
kZltdhAi4qwHdc90sqypaHJsgIrTAUBNRscqL+DPhQ+/h4OlPqL0fmjzjSlWYFEo6neWBJjJJicS+mUNpjvpOFEUFCIPXObn4ecZeq7B3OwNn4oyYN3vdALE1EBkyUBueli/
QnkkQKx4AtSUL2AUCp9W9XHU7ttP/259YnvhL5yFcPqK2CTHw0hofEkGf+bWDw3YRN0cjDTq7cCj/1AGs9bF8NuLsC/ZfoAVP8zB2MlfpVjsOVN0Yh0eSEwR
+Mn4Y78Dqb6tFdQgMIHg0sGlajOAdwC9LVWSqrfnUbmeh7B+RotL8agDt4AvqQMhQtYq
```

Slika 9 - users-client.properties

compiler-client.properties

```
server.port=8099
```

Slika 10 - users-client.properties