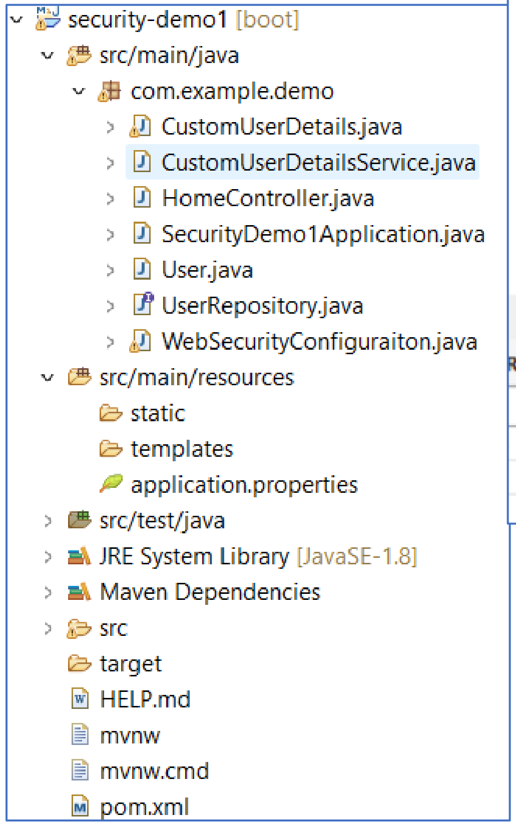
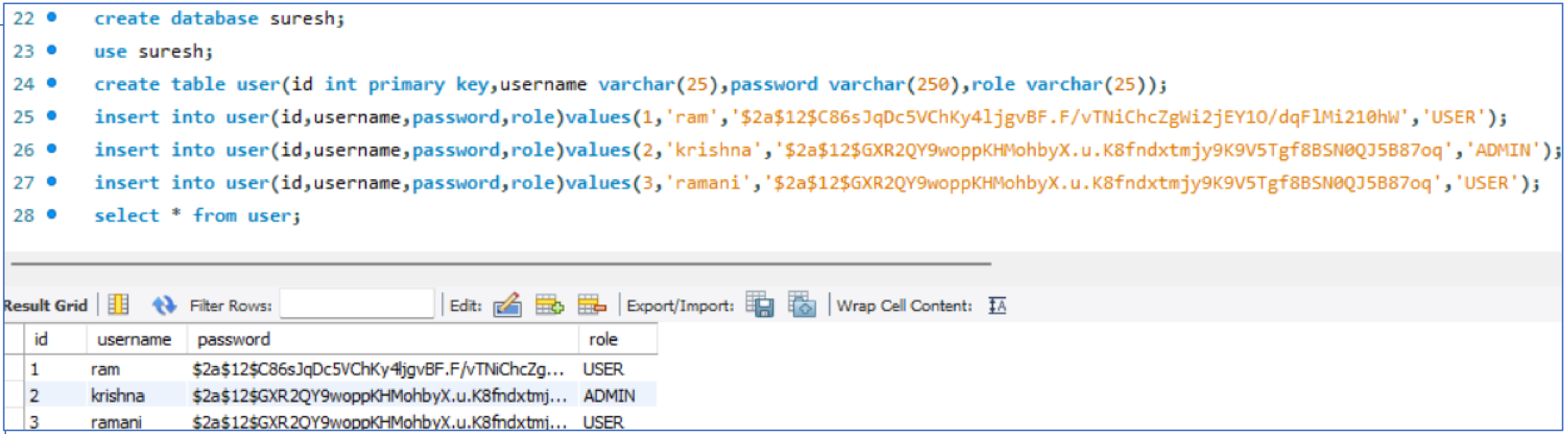
Spring Security

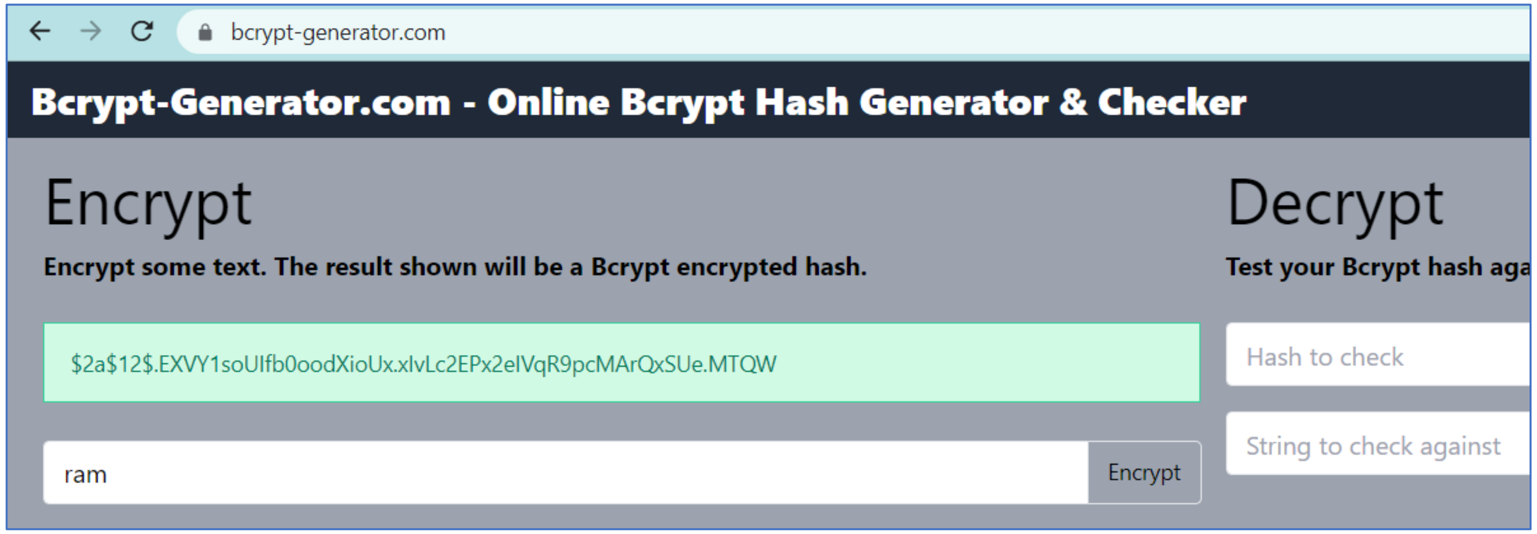


```
security-demo1 [boot]
  src/main/java
    com.example.demo
      CustomUserDetails.java
      CustomUserDetailsService.java
      HomeController.java
      SecurityDemo1Application.java
      User.java
      UserRepository.java
      WebSecurityConfiguraiton.java
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```



```sql
create database suresh;
use suresh;
create table user(id int primary key,username varchar(25),password varchar(250),role varchar(25));
insert into user(id,username,password,role)values(1,'ram','$2a$12$C86sJqDc5VChKy4ljgvBF.F/vTNiChcZgWi2jEY1O/dqFlMi210hW','USER');
insert into user(id,username,password,role)values(2,'krishna','$2a$12$GXR2QY9woppKHMohbyX.u.K8fndxtmjy9K9V5Tgf8BSN0QJ5B87oq','ADMIN');
insert into user(id,username,password,role)values(3,'ramani','$2a$12$GXR2QY9woppKHMohbyX.u.K8fndxtmjy9K9V5Tgf8BSN0QJ5B87oq','USER');
select * from user;
```

id	username	password	role
1	ram	$2a$12$C86sJqDc5VChKy4ljgvBF.F/vTNiChcZg...	USER
2	krishna	$2a$12$GXR2QY9woppKHMohbyX.u.K8fndxtmj...	ADMIN
3	ramani	$2a$12$GXR2QY9woppKHMohbyX.u.K8fndxtmj...	USER



bcrypt-generator.com

Bcrypt-Generator.com - Online Bcrypt Hash Generator & Checker

Encrypt

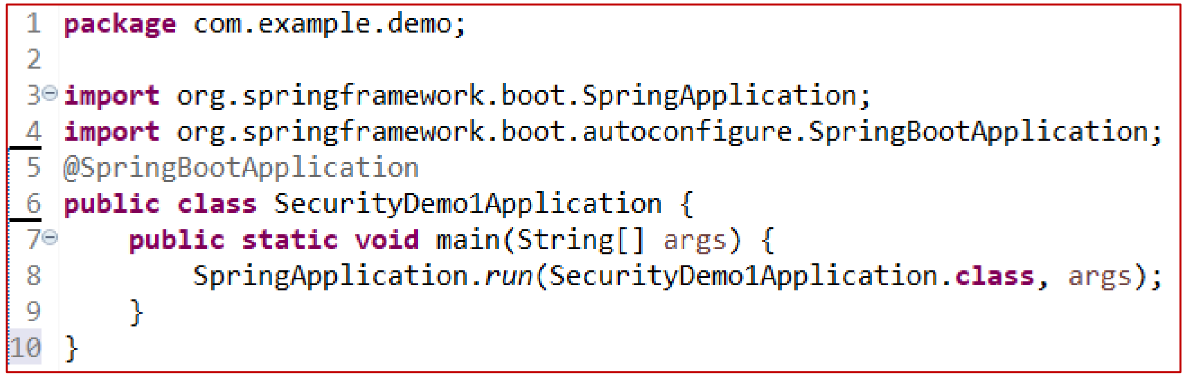
Encrypt some text. The result shown will be a Bcrypt encrypted hash.

$2a$12$.EXVY1soUIfb0oodXioUx.xlvLc2EPx2elVqR9pcMArQxSUe.MTQW

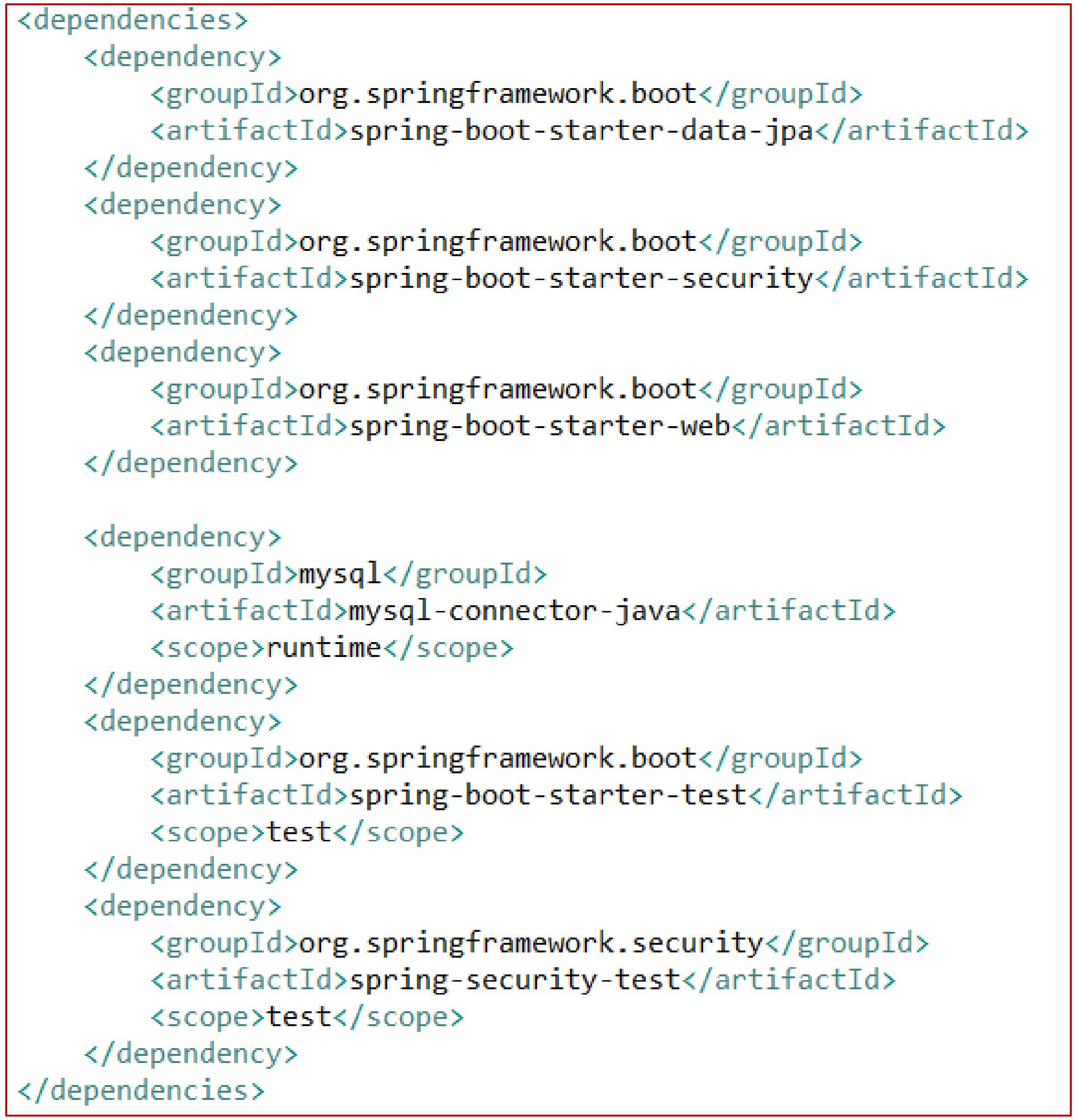
ram

Decrypt

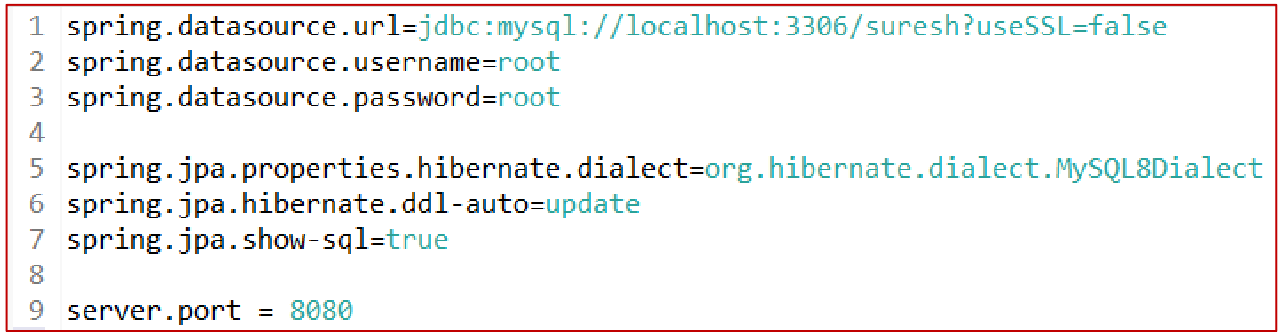
Test your Bcrypt hash aga



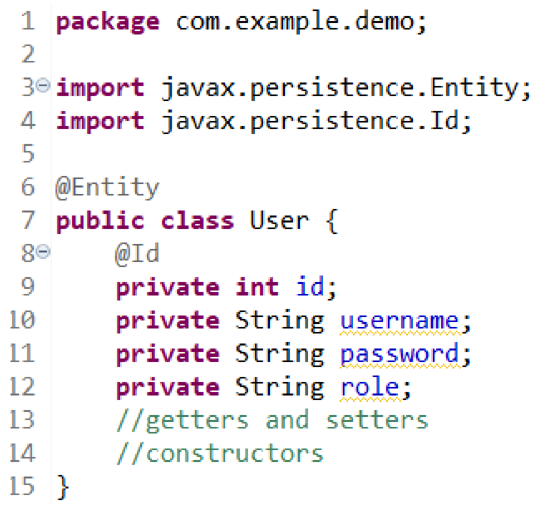
```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SecurityDemo1Application {
    public static void main(String[] args) {
        SpringApplication.run(SecurityDemo1Application.class, args);
    }
}
```



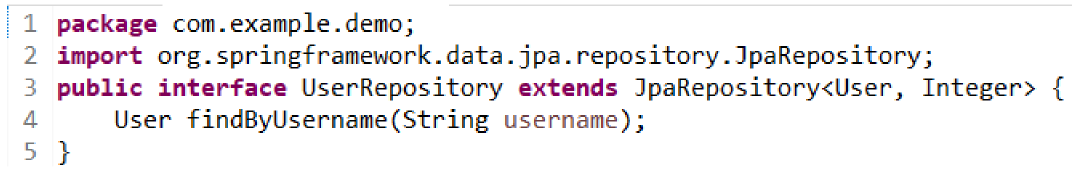
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>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

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```



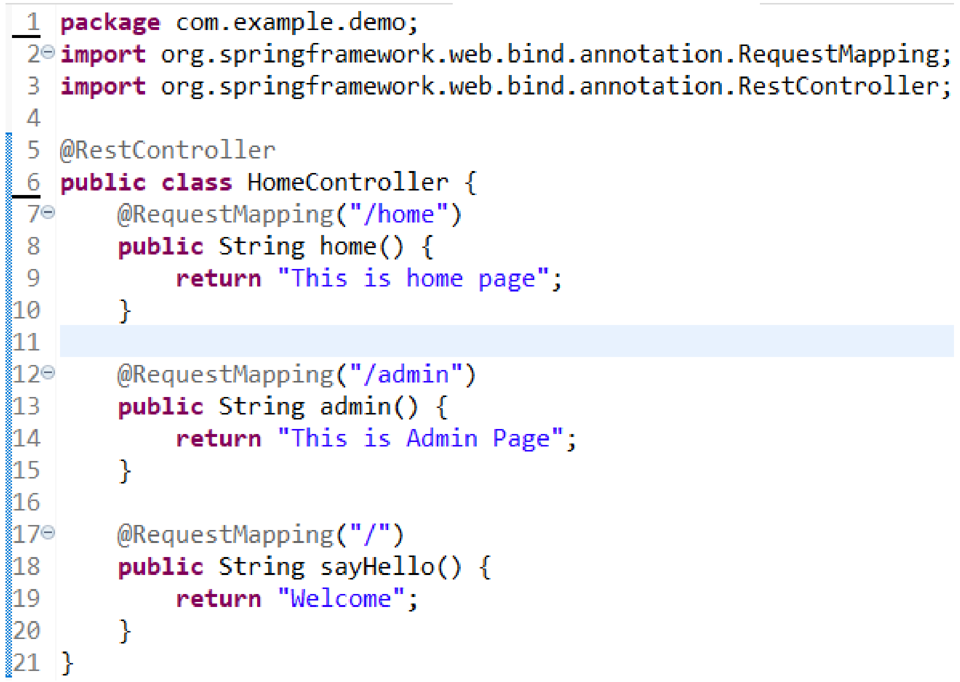
```properties
spring.datasource.url=jdbc:mysql://localhost:3306/suresh?useSSL=false
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

server.port = 8080
```



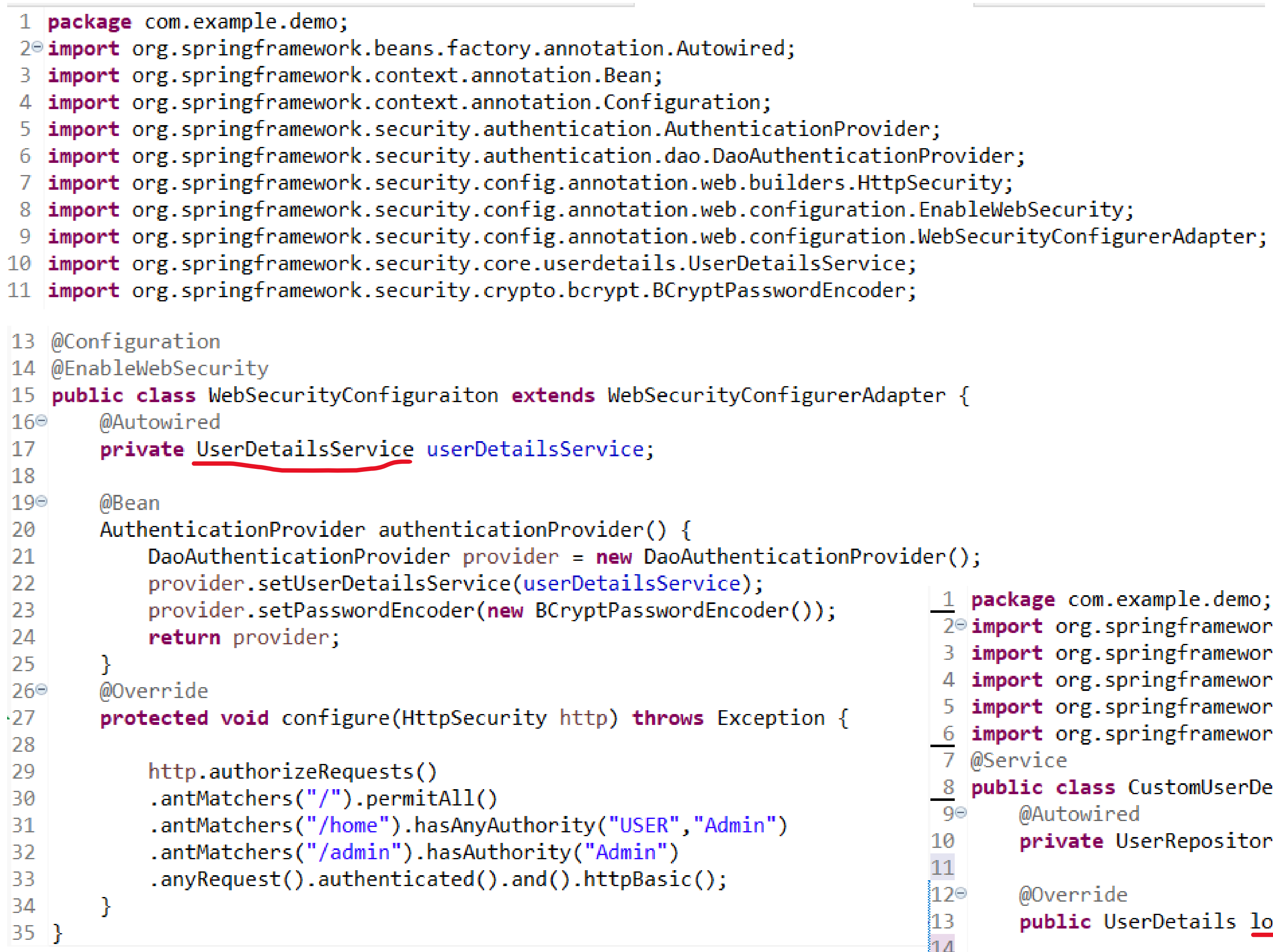
```java
package com.example.demo;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class User {
    @Id
    private int id;
    private String username;
    private String password;
    private String role;
    //getters and setters
    //constructors
}
```



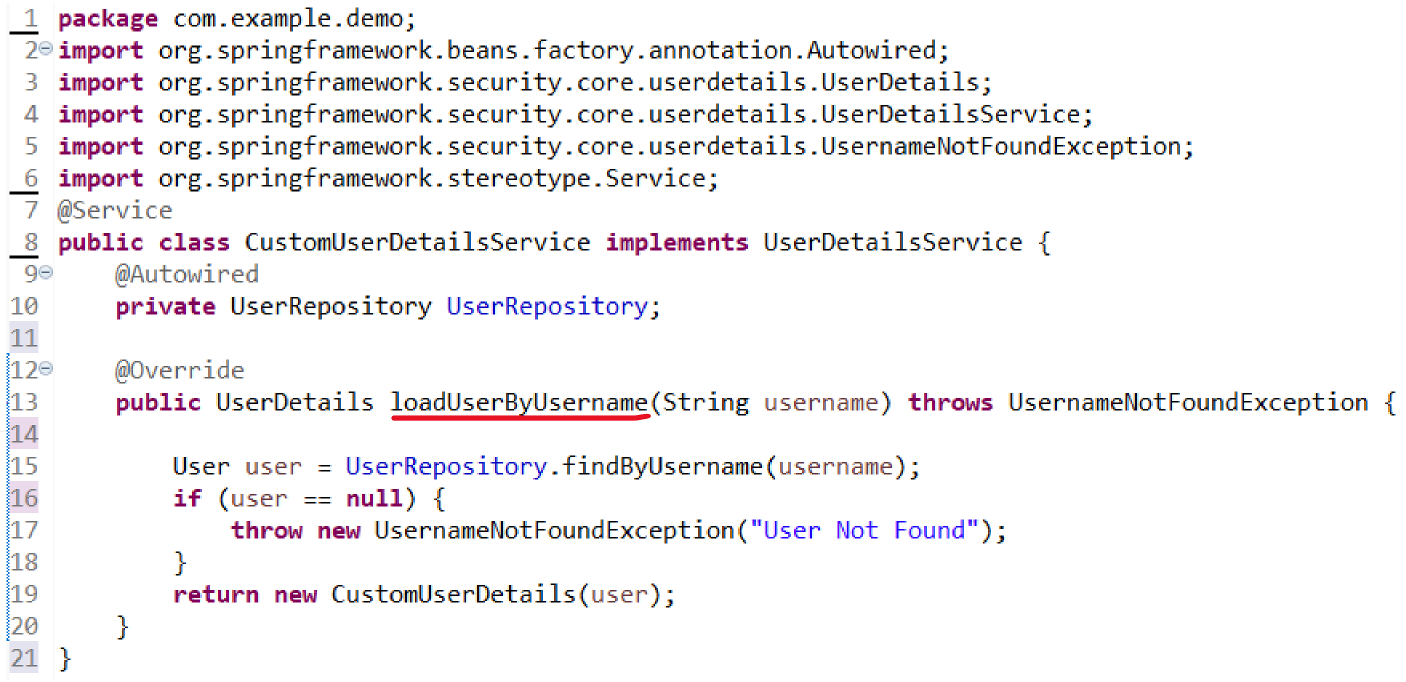
```java
package com.example.demo;
import org.springframework.data.jpa.repository.JpaRepository;
public interface UserRepository extends JpaRepository<User, Integer> {
    User findByUsername(String username);
}
```



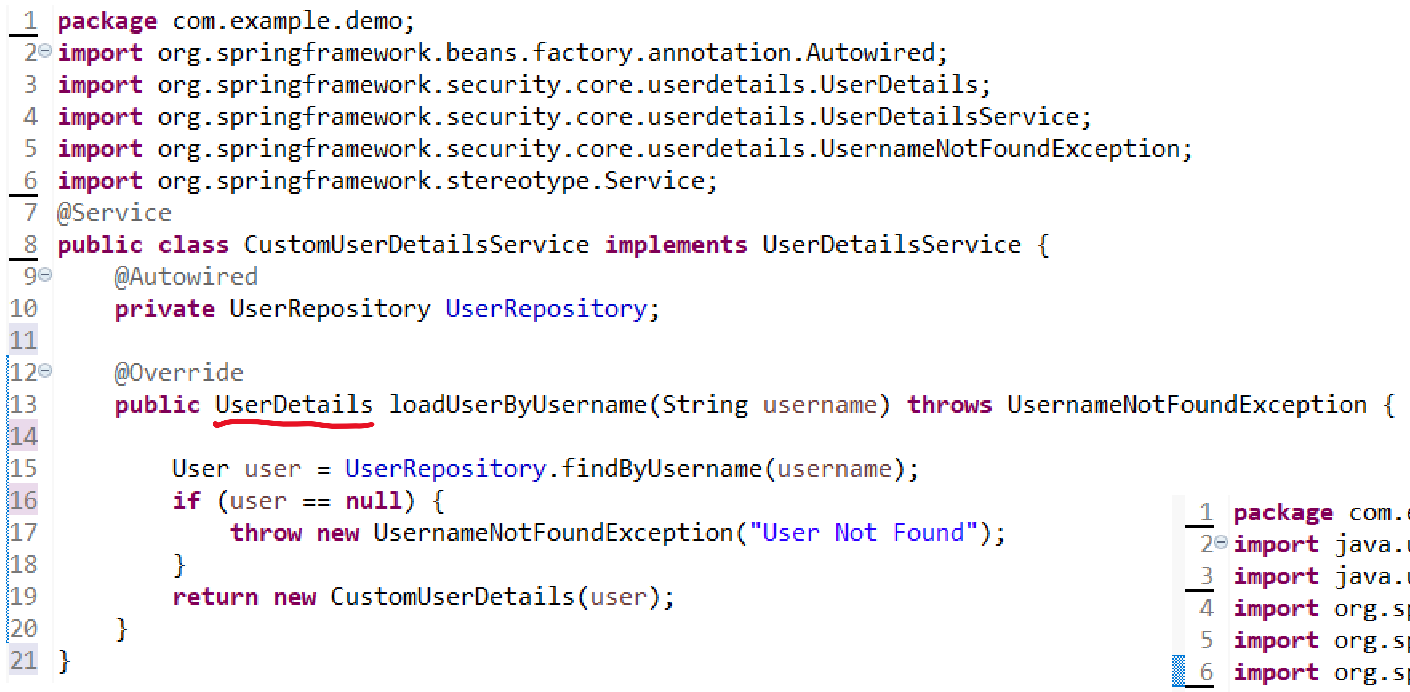
```java
package com.example.demo;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {
    @RequestMapping("/home")
    public String home() {
        return "This is home page";
    }

    @RequestMapping("/admin")
    public String admin() {
        return "This is Admin Page";
    }

    @RequestMapping("/")
    public String sayHello() {
        return "Welcome";
    }
}
```



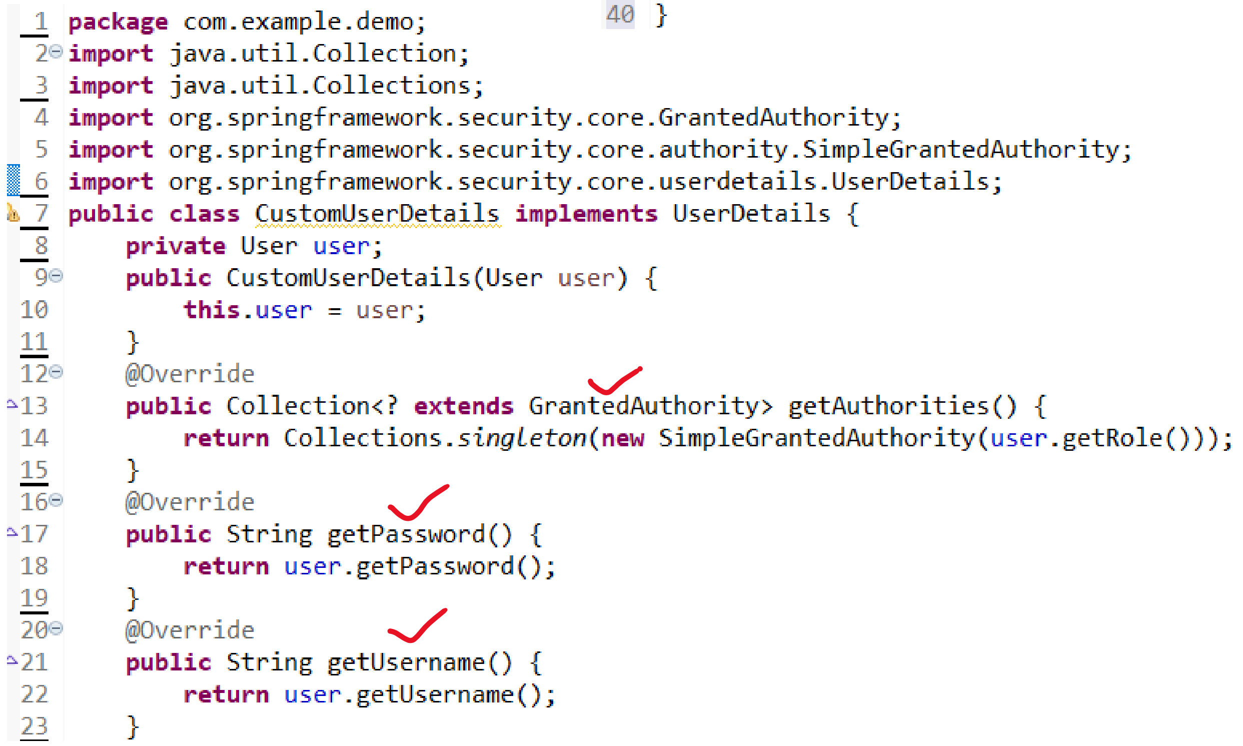
```java
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity
public class WebSecurityConfiguraiton extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
        provider.setUserDetailsService(userDetailsService);
        provider.setPasswordEncoder(new BCryptPasswordEncoder());
        return provider;
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
        .antMatchers("/").permitAll()
        .antMatchers("/home").hasAnyAuthority("USER","Admin")
        .antMatchers("/admin").hasAuthority("Admin")
        .anyRequest().authenticated().and().httpBasic();
    }
}
```



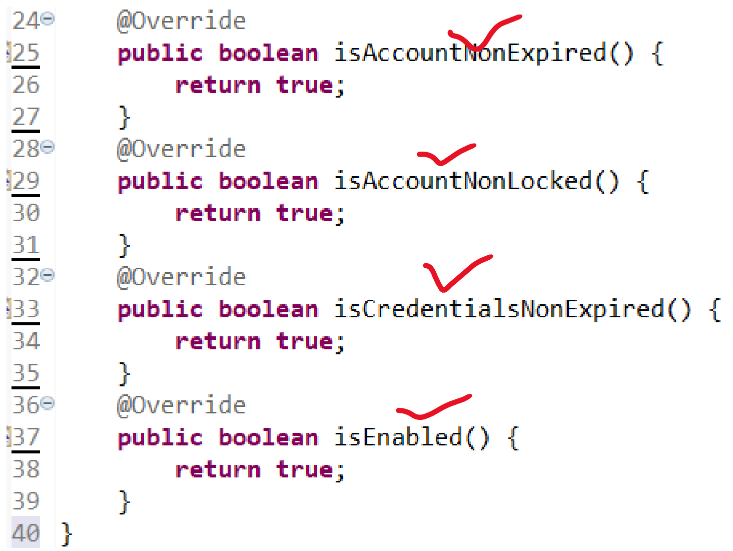
```java
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository UserRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        User user = UserRepository.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException("User Not Found");
        }
        return new CustomUserDetails(user);
    }
}
```



```java
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository UserRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        User user = UserRepository.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException("User Not Found");
        }
        return new CustomUserDetails(user);
    }
}
```



```java
package com.example.demo;
import java.util.Collection;
import java.util.Collections;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
public class CustomUserDetails implements UserDetails {
    private User user;
    public CustomUserDetails(User user) {
        this.user = user;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return Collections.singleton(new SimpleGrantedAuthority(user.getRole()));
    }
    @Override
    public String getPassword() {
        return user.getPassword();
    }
    @Override
    public String getUsername() {
        return user.getUsername();
    }
```



```java
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return true;
    }
}
```