## [**Implementation to Make Registration API RESTful in Spring Security**](about:blank)

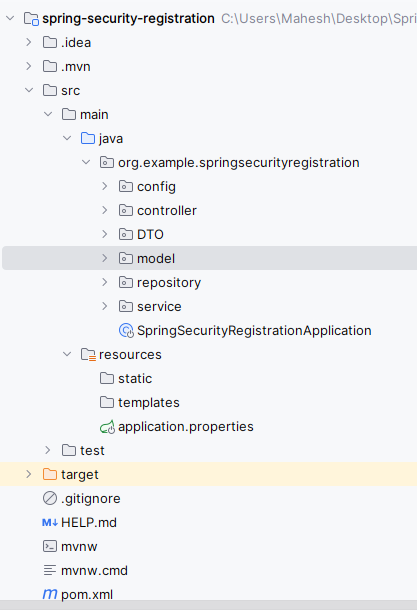
[Below are the step-by-step implementation to make Registration API RESTful in spring security.](about:blank)

[**Step 1: C**reate a Spring project using Spring Initializr, add the following dependencies when generating the project:](about:blank)

[**Dependencies:**](about:blank)

* [Spring Web](about:blank)
* [Spring Security](about:blank)
* [Spring data JPA](about:blank)
* [MySQL Driver](about:blank)
* [Spring Dev Tools](about:blank)
* [Lombok](about:blank)

[Once the Spring project is created, the file structure typically resembles the following:](about:blank)

[](about:blank)

[**Step 2**: Open the **application.properties** file and add the following code to configure the server port and MySQL database:](about:blank)

[spring.application.name=spring-security-registration](about:blank)

[# DataSource configuration](about:blank)

[spring.datasource.url=jdbc:mysql://localhost:3306/example](about:blank)

[spring.datasource.username=root](about:blank)

[spring.datasource.password=](about:blank)

[spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver](about:blank)

[# Hibernate configuration](about:blank)

[spring.jpa.hibernate.ddl-auto=update](about:blank)

[spring.jpa.show-sql=true](about:blank)

[**Step 3:** Create a new package named “model”. Inside this package, create a new Java class named “User”.](about:blank)

[Go to **src > org.example.springsecurityregistration > model > User** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.model**;](about:blank)

[**import** **jakarta.persistence.Entity**;](about:blank)

[**import** **jakarta.persistence.GeneratedValue**;](about:blank)

[**import** **jakarta.persistence.GenerationType**;](about:blank)

[**import** **jakarta.persistence.Id**;](about:blank)

[**import** **lombok.AllArgsConstructor**;](about:blank)

[**import** **lombok.Data**;](about:blank)

[**import** **lombok.NoArgsConstructor**;](about:blank)

[@Entity](about:blank)

[@Data](about:blank)

[@AllArgsConstructor](about:blank)

[@NoArgsConstructor](about:blank)

[**public** **class** **User** {](about:blank)

[@Id](about:blank)

[@GeneratedValue(strategy = GenerationType.IDENTITY)](about:blank)

[**private** Long id;](about:blank)

[**private** String username;](about:blank)

[**private** String email;](about:blank)

[**private** String password;](about:blank)

[}](about:blank)

[**Step 4**: Create a new package named “repository”. Inside this package, create a new Java interface named “UserRepository”.](about:blank)

[Go to **src > org.example.springsecurityregistration > repository > UserRepository** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.repository**;](about:blank)

[**import** **org.example.springsecurityregistration.model.User**;](about:blank)

[**import** **org.springframework.data.jpa.repository.JpaRepository**;](about:blank)

[**import** **org.springframework.stereotype.Repository**;](about:blank)

[@Repository](about:blank)

[**public** **interface** **UserRepository** **extends** JpaRepository<User,Long> {](about:blank)

[User findByUsername(String username);](about:blank)

[}](about:blank)

[**Step 5**: Create a new package named “dto”. Inside this package, create a new Java class named “RegistrationRequest”.](about:blank)

[Go to **src > org.example.springsecurityregistration > dto > RegistrationRequest** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.DTO**;](about:blank)

[**import** **lombok.AllArgsConstructor**;](about:blank)

[**import** **lombok.Data**;](about:blank)

[**import** **lombok.NoArgsConstructor**;](about:blank)

[@Data](about:blank)

[@AllArgsConstructor](about:blank)

[@NoArgsConstructor](about:blank)

[**public** **class** **RegistrationRequest** {](about:blank)

[**private** String username;](about:blank)

[**private** String email;](about:blank)

[**private** String password;](about:blank)

[*// Constructors, getters, and setters*](about:blank)

[}](about:blank)

[**Step 6**: Create a new package named “config”. Inside this package, create a new Java class named “SecurityConfig”.](about:blank)

[Go to **src > org.example.springsecurityregistration > config > SecurityConfig** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.config**;](about:blank)

[**import** **org.springframework.beans.factory.annotation.Autowired**;](about:blank)

[**import** **org.springframework.context.annotation.Bean**;](about:blank)

[**import** **org.springframework.context.annotation.Configuration**;](about:blank)

[**import** **org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder**;](about:blank)

[**import** **org.springframework.security.config.annotation.web.builders.HttpSecurity**;](about:blank)

[**import** **org.springframework.security.config.annotation.web.configuration.EnableWebSecurity**;](about:blank)

[**import** **org.springframework.security.core.userdetails.UserDetailsService**;](about:blank)

[**import** **org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder**;](about:blank)

[**import** **org.springframework.security.crypto.password.PasswordEncoder**;](about:blank)

[**import** **org.springframework.security.web.SecurityFilterChain**;](about:blank)

[@Configuration](about:blank)

[@EnableWebSecurity](about:blank)

[**public** **class** **SecurityConfig** {](about:blank)

[@Autowired](about:blank)

[**private** UserDetailsService userDetailsService;](about:blank)

[**protected** void configure(AuthenticationManagerBuilder auth) **throws** Exception {](about:blank)

[auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());](about:blank)

[}](about:blank)

[@Bean](about:blank)

[**protected** SecurityFilterChain securityFilterChain(HttpSecurity http) **throws** Exception {](about:blank)

[http](about:blank)

[.authorizeRequests()](about:blank)

[.requestMatchers("/api/register").permitAll() *// Allow registration endpoint without authentication*](about:blank)

[.anyRequest().authenticated()](about:blank)

[.and()](about:blank)

[.formLogin()](about:blank)

[.and()](about:blank)

[.httpBasic()](about:blank)

[.and()](about:blank)

[.csrf().disable();](about:blank)

[**return** http.build();](about:blank)

[}](about:blank)

[@Bean](about:blank)

[**public** PasswordEncoder passwordEncoder() {](about:blank)

[**return** **new** BCryptPasswordEncoder();](about:blank)

[}](about:blank)

[}](about:blank)

[**Step 7**: Create a new package named “service”. Inside this package, create a new Java class named “UserService”.](about:blank)

[Go to **src > org.example.springsecurityregistration > service > UserService** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.service**;](about:blank)

[**import** **org.example.springsecurityregistration.DTO.RegistrationRequest**;](about:blank)

[**import** **org.example.springsecurityregistration.model.User**;](about:blank)

[**import** **org.example.springsecurityregistration.repository.UserRepository**;](about:blank)

[**import** **org.springframework.beans.factory.annotation.Autowired**;](about:blank)

[**import** **org.springframework.stereotype.Service**;](about:blank)

[@Service](about:blank)

[**public** **class** **UserService** {](about:blank)

[@Autowired](about:blank)

[**private** UserRepository userRepository;](about:blank)

[**public** void registerUser(RegistrationRequest request) {](about:blank)

[User user = **new** User();](about:blank)

[user.setUsername(request.getUsername());](about:blank)

[user.setEmail(request.getEmail());](about:blank)

[user.setPassword(request.getPassword());](about:blank)

[userRepository.save(user);](about:blank)

[}](about:blank)

[}](about:blank)

[**Step 8**: Create a new package named “service”. Inside this package, create a new Java class named “UserDetailsServiceImpl”.](about:blank)

[Go to **src > org.example.springsecurityregistration > service > UserDetailsServiceImpl** and put the below code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration.service**;](about:blank)

[**import** **org.example.springsecurityregistration.model.User**;](about:blank)

[**import** **org.example.springsecurityregistration.repository.UserRepository**;](about:blank)

[**import** **org.springframework.beans.factory.annotation.Autowired**;](about:blank)

[**import** **org.springframework.security.core.userdetails.UserDetails**;](about:blank)

[**import** **org.springframework.security.core.userdetails.UserDetailsService**;](about:blank)

[**import** **org.springframework.security.core.userdetails.UsernameNotFoundException**;](about:blank)

[**import** **org.springframework.stereotype.Service**;](about:blank)

[**import** **java.util.Collections**;](about:blank)

[@Service](about:blank)

[**public** **class** **UserDetailsServiceImpl** **implements** UserDetailsService {](about:blank)

[@Autowired](about:blank)

[**private** UserRepository userRepository;](about:blank)

[@Override](about:blank)

[**public** UserDetails loadUserByUsername(String username) **throws** UsernameNotFoundException {](about:blank)

[User user = userRepository.findByUsername(username);](about:blank)

[**if** (user == **null**) {](about:blank)

[**throw** **new** UsernameNotFoundException("User not found with username: " + username);](about:blank)

[}](about:blank)

[**return** **new** org.springframework.security.core.userdetails.User(](about:blank)

[user.getUsername(),](about:blank)

[user.getPassword(),](about:blank)

[Collections.emptyList()](about:blank)

[);](about:blank)

[}](about:blank)

[}](about:blank)

[**Step 9**: Open the main class and insert the following code.](about:blank)

[Java](about:blank)

[**package** **org.example.springsecurityregistration**;](about:blank)

[**import** **org.springframework.boot.SpringApplication**;](about:blank)

[**import** **org.springframework.boot.autoconfigure.SpringBootApplication**;](about:blank)

[@SpringBootApplication](about:blank)

[**public** **class** **SpringSecurityRegistrationApplication** {](about:blank)

[**public** **static** void main(String[] args) {](about:blank)

[SpringApplication.run(SpringSecurityRegistrationApplication.class, args);](about:blank)

[}](about:blank)

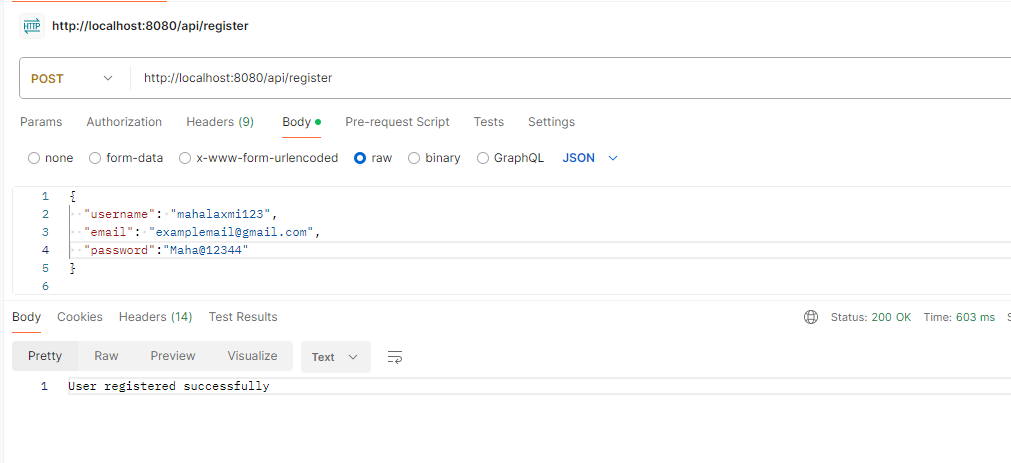
[}](about:blank)

[**Step 10**: Once the Spring project is completed and runs as a Spring application successfully, it will start at port 8080.](about:blank)

[  
**Registration Restful API:**](about:blank)

[**POST** http://localhost:8080/api/register](about:blank)

#### [**Output:**](about:blank)

[****](about:blank)

This example demonstrates how to integrate Spring Security into a Spring Boot application to secure the registration endpoint. Users need to access other endpoints while the registration endpoint remains accessible without authentication.