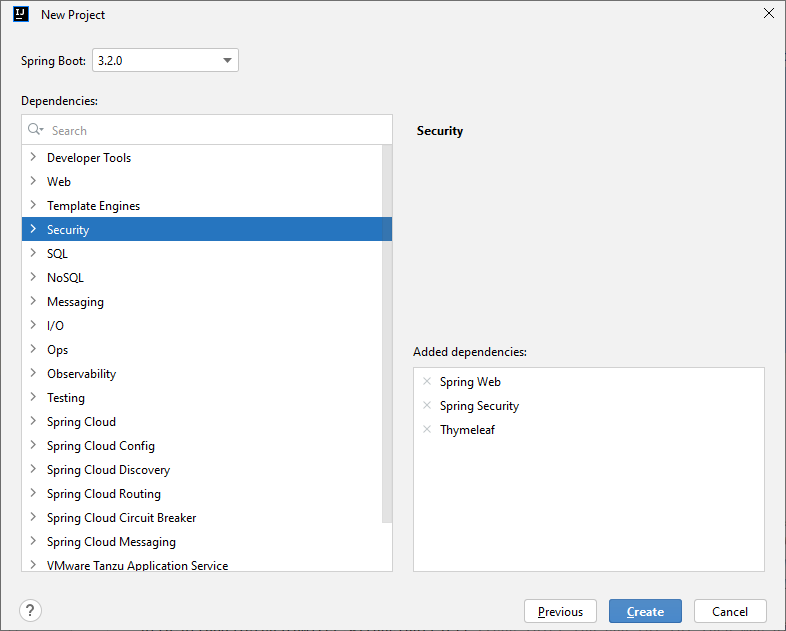
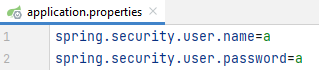
Hướng dẫn spring security với database

Tạo một Springboot project với các dependencies như hình



Khởi động ứng dụng, mở browser rồi nhập vào địa chỉ <http://localhost:8080> 🡪 cửa sổ logon sẽ hiện ra bắt chúng ta đăng nhập.Tên người dùng mặc định là user, mật khẩu là chuỗi mật khẩu sẽ xuất hiện trong cửa sổ console của ứng dụng spring.

Nếu muốn dùng user/password của riêng mình thì thêm 2 dòng sau vào file application.properties



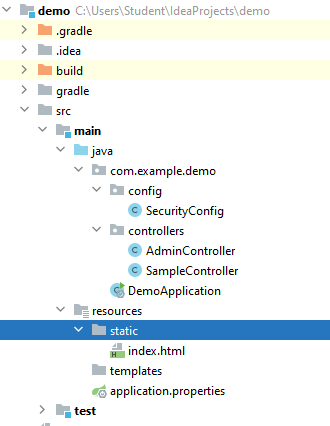
Trường hợp muốn dùng mật khẩu mã hóa thì thêm tiền tố {bcrypt} vào trước password đã mã hóa. Ví dụ với chữ a, thì password sẽ là:

$2a$12$WI9UDC3JMmvdsQFWvoH5d.618kbP.80IFityRT17ABZYgbCNvXD22

Bạn có thể tự viết đoạn code để mã hóa hoặc vào trang <https://bcrypt-generator.com/> để sinh mã.

# In Memory Authentication

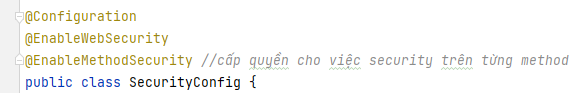
Cấu trúc project như hình. Thêm vào các gói config để chứa lớp cấu hình, gói controllers chứa các controller demo. Trong thư mục static của resources, tạo một file index.html với nội dung tùy ý.



Trong lớp SecurityConfig, ta cấu hình như code sau

|  |
| --- |
| package com.example.demo.config;  import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.security.config.Customizer; import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder; import org.springframework.security.config.annotation.web.builders.HttpSecurity; import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity; import org.springframework.security.core.userdetails.User; import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.security.web.SecurityFilterChain;  @Configuration @EnableWebSecurity public class SecurityConfig {   @Autowired  public void globalConfig(AuthenticationManagerBuilder auth, PasswordEncoder encoder)throws Exception{  auth.inMemoryAuthentication()  .withUser(User.*withUsername*("admin")  .password(encoder.encode("admin"))  .roles("ADMIN")  .build())  .withUser(User.*withUsername*("teo")  .password(encoder.encode("teo"))  .roles("TEO")  .build())  .withUser(User.*withUsername*("ty")  .password(encoder.encode("ty"))  .roles("USER")  .build())  ;  }   @Bean  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {   http.authorizeHttpRequests(auth->auth  .requestMatchers("/","/home","/index").permitAll()*//nhung links nay khong can authenticate* .requestMatchers("/api/\*\*").hasAnyRole("ADMIN","USER","TEO")*//nhung uri bat dau bang /api can phai dang nhap voi cac role admin/user/teo* .requestMatchers(("/admin/\*\*")).hasRole("ADMIN")*//uri bat dau bang /admin thi phai dang nhap voi quyen admin* .anyRequest().authenticated()*//cac uri khac can dang nhap duoi bat ky role nao* );  http.httpBasic(Customizer.*withDefaults*());*//cac thiet lap con lai thi theo mac dinh* return http.build();  }   @Bean  public PasswordEncoder passwordEncoder(){  return new BCryptPasswordEncoder();  } } |

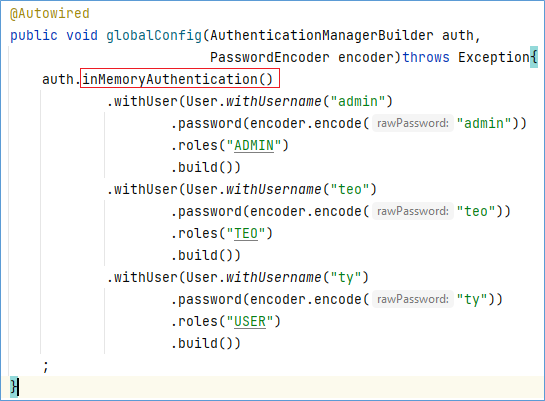
Giải thích



Bạn cần bật security của web. Trong trường hợp muốn cấp quyền chi tiết hơn trên từng method, bạn cần bật chế độ MethodSecurity.

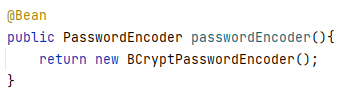


Bean này cấu hình việc phân quyền. Bạn nên nhóm các công việc với nhau để dễ phân quyền. Nội dung giải thích như trong ghi chú.



Phương thức này xác lập các users cũng như role (vai trò) của họ trong ứng dụng. Ở đây chúng ta thêm các users cố định trong memory.

Trong trường hợp này có 3 user thuộc 3 roles khác nhau. Mật khẩu được mã hóa bởi thuật giải bcrypt. Bạn cần thêm bean encoder cho việc mã hóa mật khẩu này:

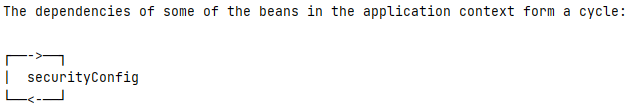


Các controllers dành cho các mục đích thử nghiệm các role khác nhau.

|  |
| --- |
| @RestController @RequestMapping("/admin") public class AdminController {  @GetMapping()  public String admin(){  return "this is admin area";  } } |
| @RestController @RequestMapping("/api") public class SampleController {   @GetMapping  public String hello(){  return "hello world";  } } |

**Chú ý:**

Khi chạy ứng dụng sẽ gặp lỗi



Trường hợp này bạn thêm dòng config sau vào application.properties



Thực thi ứng dụng và vào các paths khác nhau trên trình duyệt để thử. Chú ý nên mở cửa sổ ẩn danh trên các trình duyệt khác nhau.

# In JDBC Authentication

Với việc lưu trữ user vào database, bạn cần thêm các dependencies cho việc truy xuất đến csdl đích. Ví dụ này dùng H2 relational database

|  |
| --- |
| dependencies **{** implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  implementation 'org.springframework.boot:spring-boot-starter-security'  implementation 'org.springframework.boot:spring-boot-starter-web'   implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity6'   compileOnly 'org.projectlombok:lombok'  annotationProcessor 'org.projectlombok:lombok'   runtimeOnly 'com.h2database:h2'    testImplementation 'org.springframework.boot:spring-boot-starter-test'  testImplementation 'org.springframework.security:spring-security-test'   **}** |

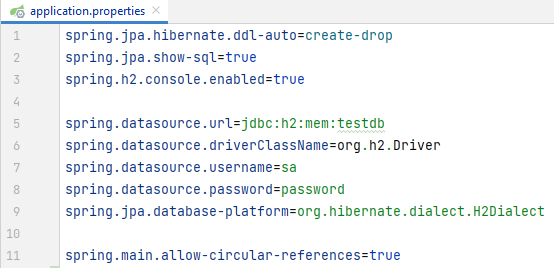
Với MariaDb, bạn thêm dependencies

|  |
| --- |
| runtimeOnly 'org.mariadb.jdbc:mariadb-java-client' |

Cấu hình giống như phần trước. Tuy nhiên trong configureGlobal method, thay vì dùng memory, ta dùng jdbc



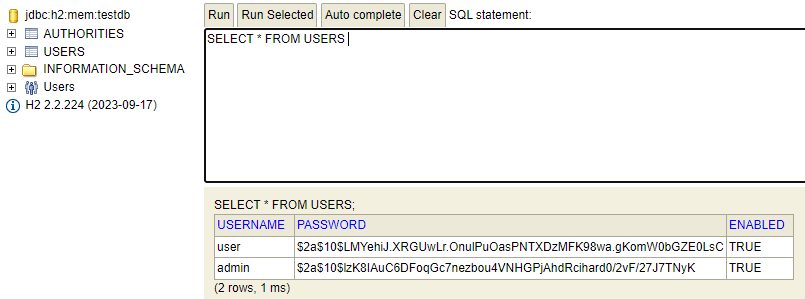
Phần cấu hình database, ta thêm vào các thuộc tính như sau:

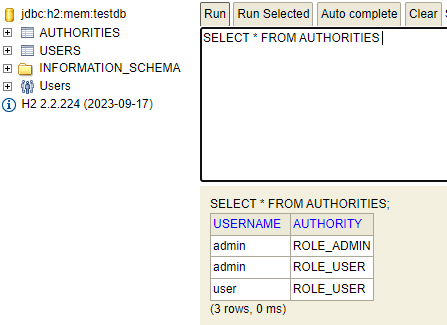


Để có thể cấp quyền cho h2-console, ta cấu hình filterChain lại như sau



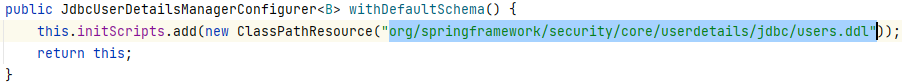
Thực thi ứng dụng, mở trình duyệt, vào <http://localhost:8080/h2-console> . Sau khi đăng nhập ta có 2 bảng users và authorities với các mẫu tin như sau:





Giải thích

Với , Sring sẽ lấy một database mặt định với các bảng và câu truy vấn được định nghĩa sẵn.



Chúng ta có thể xem cụ thể trong lớp JdbcUserDetailsManager của Spring



## Sử dụng swagger tron ứng dụng

Thêm Swagger dependency trong build.gradle/pom.xm

|  |
| --- |
| implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.2.0' |

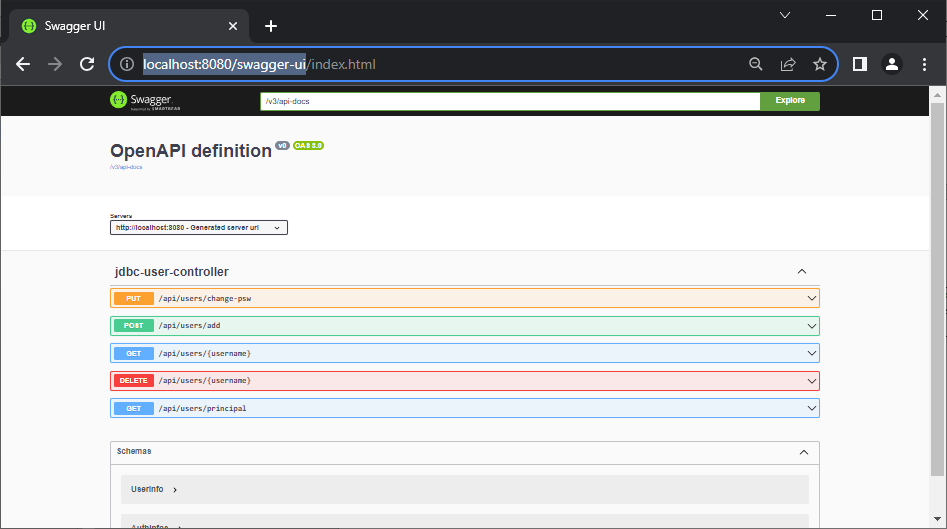
Trong application.properties, thêm cấu hình



Thêm matcher trong filterChain

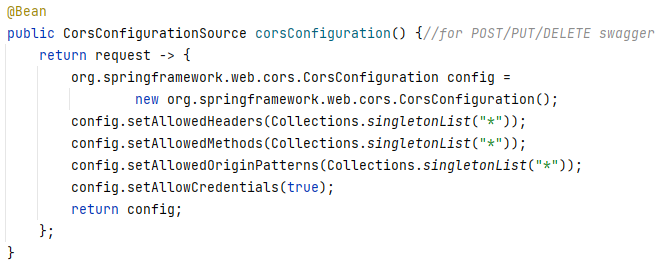
|  |
| --- |
| .requestMatchers("/api/v1/auth/\*\*", "/v2/api-docs/\*\*", "/v3/api-docs/\*\*",  "/swagger-resources/\*\*", "/swagger-ui/\*\*", "/webjars/\*\*").permitAll() |

Mở trình duyệt, vào địa chỉ <http://localhost:8080/swagger-ui> ta có



## Cập nhật thông tin đăng nhập

Để cập nhật (add/update/delete) thông tin đăng nhập, ta cần phải cấu hình CORS như sau:



FilterChain cũng phải cập nhật lại như sau



Tạo một service như sau

|  |
| --- |
| package com.wordpress.vovanhai.services;  import lombok.RequiredArgsConstructor; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.provisioning.JdbcUserDetailsManager; import org.springframework.stereotype.Service;  import javax.sql.DataSource;  @Service @RequiredArgsConstructor public class JdbcUserService {  private DataSource dataSource;  private JdbcUserDetailsManager manager;   @Autowired  public JdbcUserService(DataSource dataSource) {  this.dataSource = dataSource;  manager = new JdbcUserDetailsManager(dataSource);  }   public UserDetails addUser(UserDetails userDetails) {  manager.createUser(userDetails);  return userDetails;  }   public UserDetails changePassword(String username, String newPass) {  UserDetails us = manager.loadUserByUsername(username);  manager.changePassword("noop", newPass);  return (us);  }   public UserDetails deleteUser(String username) {  UserDetails us = manager.loadUserByUsername(username);  manager.deleteUser(username);  return us;  }   public UserDetails getByName(String username) {  return manager.loadUserByUsername(username);  } } |

Controller cho việc quản lý user

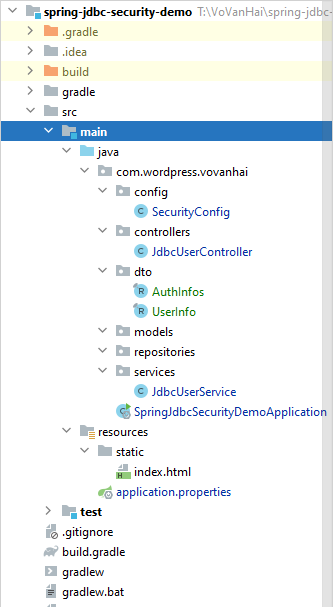
|  |
| --- |
| package com.wordpress.vovanhai.controllers;  import com.wordpress.vovanhai.dto.AuthInfos; import com.wordpress.vovanhai.dto.UserInfo; import com.wordpress.vovanhai.services.JdbcUserService; import lombok.extern.slf4j.Slf4j; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.security.access.annotation.Secured; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.security.core.Authentication; import org.springframework.security.core.userdetails.User; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.web.bind.annotation.\*;  import java.security.Principal;  @RestController @RequestMapping("/api/users") @Slf4j public class JdbcUserController {  @Autowired  private JdbcUserService jdbcUserService;  @Autowired  private PasswordEncoder encoder;   @PostMapping("/add")  @PreAuthorize("hasRole('ROLE\_ADMIN')")  public ResponseEntity<?> add(@RequestBody UserInfo userInfo) {  UserDetails userDetails = User  .*withUsername*(userInfo.userName())  .password(encoder.encode(userInfo.password()))  .roles(userInfo.authorities())  .build();  return ResponseEntity.*ok*(jdbcUserService.addUser(userDetails));  }   @PutMapping("/change-psw")  @PreAuthorize("hasAnyRole('ROLE\_ADMIN', 'ROLE\_USER')") *// @PreAuthorize("#username == authentication.principal.username")* public ResponseEntity<?> changePsw(Principal principal, @RequestBody String newPass) {  return ResponseEntity.*ok*(jdbcUserService.changePassword(  principal.getName(),  encoder.encode(newPass))  );  }   @DeleteMapping("/{username}")  @Secured("ROLE\_ADMIN")  public ResponseEntity<?> del(@PathVariable("username") String username) {  *log*.info("\*\*\*\*\*\*deleting user {}", username);  UserDetails s = jdbcUserService.getByName(username);  if (s == null)  return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body("Credential " + username + " not found");  return ResponseEntity.*ok*(jdbcUserService.deleteUser(username));  }   @GetMapping("/{username}")  @PreAuthorize("hasAnyRole('ROLE\_ADMIN','ROLE\_USER')")  public ResponseEntity<?> getByName(@PathVariable("username") String username) {  UserDetails s = jdbcUserService.getByName(username);  if (s == null)  return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body("Credential " + username + " not found");  return ResponseEntity.*ok*(s);  }   @GetMapping("/principal")  @PreAuthorize("hasAnyRole('ROLE\_ADMIN','ROLE\_USER')")  public ResponseEntity<AuthInfos> retrievePrincipal(Principal principal, Authentication auth) {  return ResponseEntity.*ok*(new AuthInfos(principal, auth));  } } |

Việc cấp quyền trong trường hợp này là cho từng method.

Ngoài ra, ta cần hai đối tượng dto để truyền thông tin như sau

|  |
| --- |
| package com.wordpress.vovanhai.dto;  import org.springframework.security.core.Authentication; import org.springframework.util.StringUtils;  import java.security.Principal;  public record AuthInfos(Principal principal, Authentication auth){  @Override  public String toString() {  return "AuthInfos{" +  "principal=" + principal.getName() +  ", auth=" + StringUtils.*collectionToDelimitedString*(auth.getAuthorities(), ",")+  '}';  } } |
| package com.wordpress.vovanhai.dto;  public record UserInfo(String userName, String password, boolean enable, String [] authorities){  } |

Cấu trúc chương trình



Có thể tiến hành chạy và thử nghiệm với swagger hoặc POSTMAN/SOAPUI