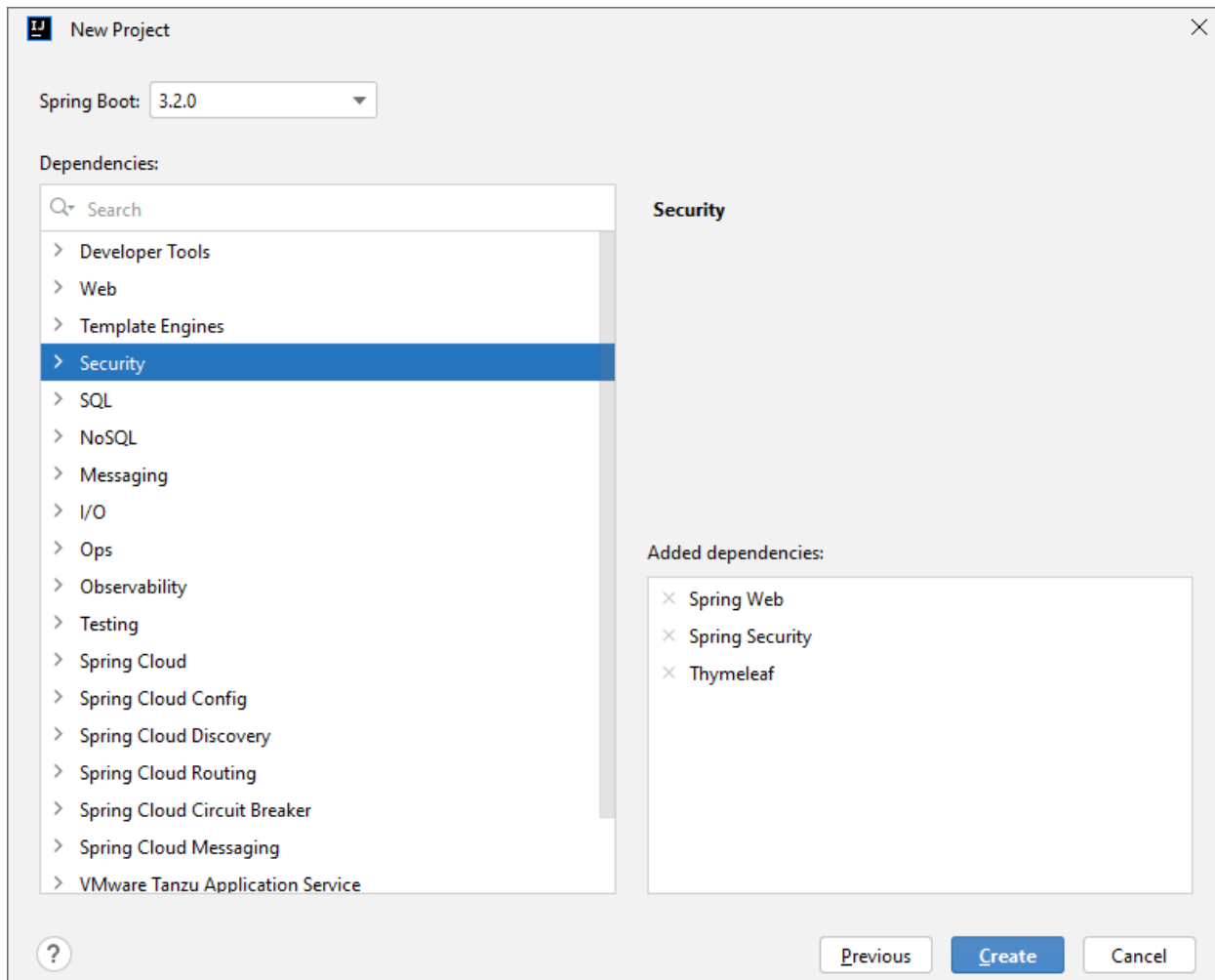


# Hướng dẫn spring security với database

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 login 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

```
application.properties
1 spring.security.user.name=a
2 spring.security.user.password=a
```

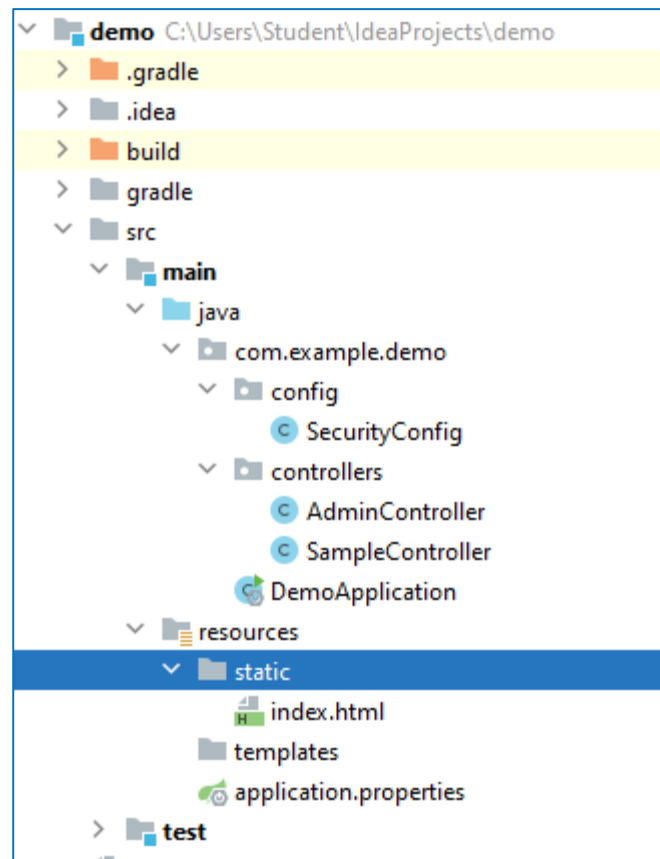
Trường hợp muốn dùng mật khẩu mã hóa thì thêm tiền tố **{bcrypt}** vào trước password đã mã hóa. Ví dụ với chữ a, thì password sẽ là:

```
$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.Authentication
ManagerBuilder;
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
        .requestMatchers("/api/**").hasAnyRole("ADMIN", "USER", "TEO") //nhung
        .requestMatchers("/admin/**").hasRole("ADMIN") //uri bat dau bang
        .anyRequest().authenticated() //cac uri khac can dang nhap duoi bat ky
    );
    http.httpBasic(Customizer.withDefaults()); //cac thiet lap con lai thi theo
    return http.build();
}

@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
}

```

## Giải thích

```

@Configuration
@EnableWebSecurity
@EnableMethodSecurity //cấp quyền cho việc security trên từng method
public class SecurityConfig {

```

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
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 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.

```

@Autowired
public void globalConfig(AuthenticationManagerBuilder auth,
    PasswordEncoder encoder) throws Exception {
    auth.inMemoryAuthentication()
        .withUser(User.withUsername("admin")
            .password(encoder.encode(rawPassword: "admin"))
            .roles("ADMIN")
            .build())
        .withUser(User.withUsername("teo")
            .password(encoder.encode(rawPassword: "teo"))
            .roles("TEO")
            .build())
        .withUser(User.withUsername("ty")
            .password(encoder.encode(rawPassword: "ty"))
            .roles("USER")
            .build())
    ;
}

```

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:

```
@Bean
public PasswordEncoder passwordEncoder(){
    return new BCryptPasswordEncoder();
}
```

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";
    }
}
```

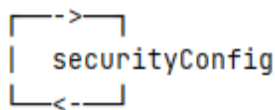
```
@RestController
@RequestMapping("/api")
public class SampleController {

    @GetMapping
    public String hello(){
        return "hello world";
    }
}
```

## Chú ý:

Khi chạy ứng dụng sẽ gặp lỗi

The dependencies of some of the beans in the application context form a cycle:



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

```
spring.main.allow-circular-references=true
```

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

```

@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth,
                           PasswordEncoder encoder, DataSource dataSource) throws Exception {
    auth.jdbcAuthentication()
        .dataSource(dataSource)
        .withDefaultSchema()//không cần cấu hình database
        .withUser(User.withUsername("user")
            .password(encoder.encode(rawPassword: "user"))
            .roles("USER"))
        .withUser(
            User.withUsername("admin")
                .password(encoder.encode(rawPassword: "admin"))
                .roles("ADMIN", "USER"))
    ;
}

```

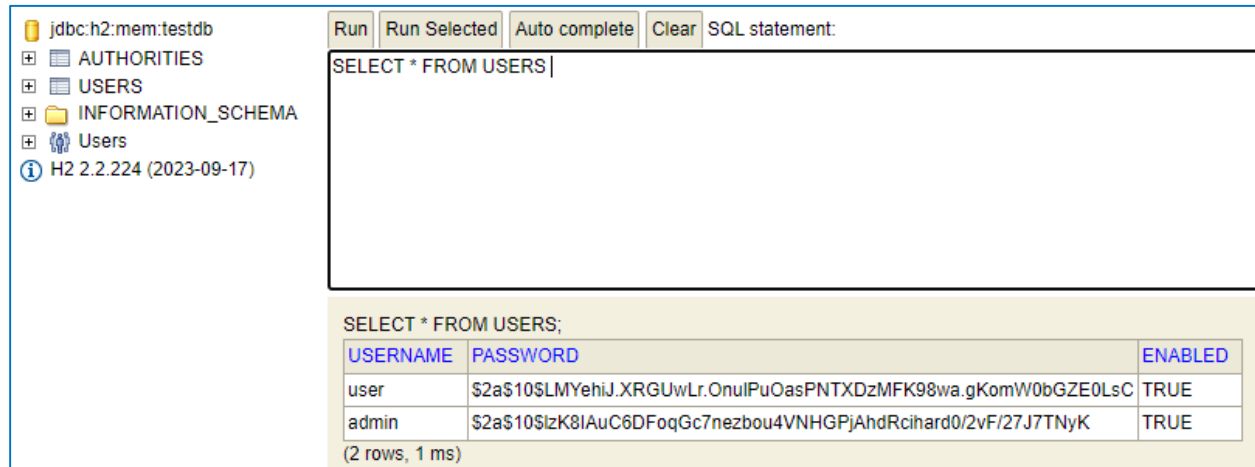
Phần cấu hình database, ta thêm vào các thuộc tính như sau:

```

application.properties
1  spring.jpa.hibernate.ddl-auto=create-drop
2  spring.jpa.show-sql=true
3  spring.h2.console.enabled=true
4
5  spring.datasource.url=jdbc:h2:mem:testdb
6  spring.datasource.driverClassName=org.h2.Driver
7  spring.datasource.username=sa
8  spring.datasource.password=password
9  spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
10
11 spring.main.allow-circular-references=true

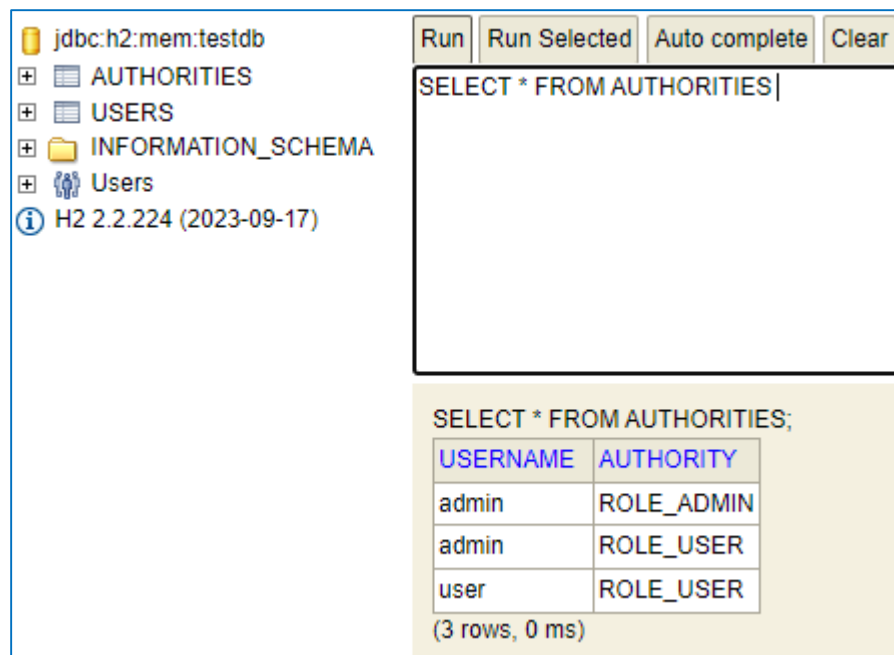
```

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:



The screenshot shows the H2 console interface. On the left, a tree view displays the database structure: jdbc:h2:mem:testdb, INFORMATION\_SCHEMA, and a Users table. The main area contains the SQL statement 'SELECT \* FROM USERS;' and its results. The results are displayed in a table with three columns: USERNAME, PASSWORD, and ENABLED. There are two rows: 'user' and 'admin'. The status is '(2 rows, 1 ms)'.

USERNAME	PASSWORD	ENABLED
user	\$2a\$10\$LMYehiJ.XRGUwLr.OnuIPuOasPNTXDzMFk98wa.gKomW0bGZE0LsC	TRUE
admin	\$2a\$10\$lzK8IAuC6DFoqGc7nezbou4VNHGPjAhdRcihard0/2vF/27J7TNyK	TRUE



The screenshot shows the H2 console interface. On the left, a tree view displays the database structure: jdbc:h2:mem:testdb, INFORMATION\_SCHEMA, and a Users table. The main area contains the SQL statement 'SELECT \* FROM AUTHORITIES;' and its results. The results are displayed in a table with two columns: USERNAME and AUTHORITY. There are three rows: 'admin' with 'ROLE\_ADMIN', 'admin' with 'ROLE\_USER', and 'user' with 'ROLE\_USER'. The status is '(3 rows, 0 ms)'.

USERNAME	AUTHORITY
admin	ROLE_ADMIN
admin	ROLE_USER
user	ROLE_USER

Giải thích

```
auth.jdbcAuthentication()  
    .dataSource(dataSource)  
    .withDefaultSchema()
```

Với `.withDefaultSchema()`, Spring sẽ lấy một database mặc định với các bảng và câu truy vấn được định nghĩa sẵn.

```
public JdbcUserDetailsManagerConfigurer<B> withDefaultSchema() {  
    this.initScripts.add(new ClassPathResource("org/springframework/security/core/userdetails/jdbc/users.ddl"));  
    return this;  
}
```

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

```
public class JdbcUserDetailsManager extends JdbcDaoImpl implements UserDetailsManager, GroupManager {

    public static final String DEF_CREATE_USER_SQL = "insert into users (username, password, enabled) values (?, ?, ?)";

    public static final String DEF_DELETE_USER_SQL = "delete from users where username = ?";

    public static final String DEF_UPDATE_USER_SQL = "update users set password = ?, enabled = ? where username = ?";

    public static final String DEF_INSERT_AUTHORITY_SQL = "insert into authorities (username, authority) values (?, ?)";

    public static final String DEF_DELETE_USER_AUTHORITIES_SQL = "delete from authorities where username = ?";

    public static final String DEF_USER_EXISTS_SQL = "select username from users where username = ?";
```