FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SPRING BOOT SECURITY 2

GIẢNG VIÊN: NGUYỄN NGHIỆM

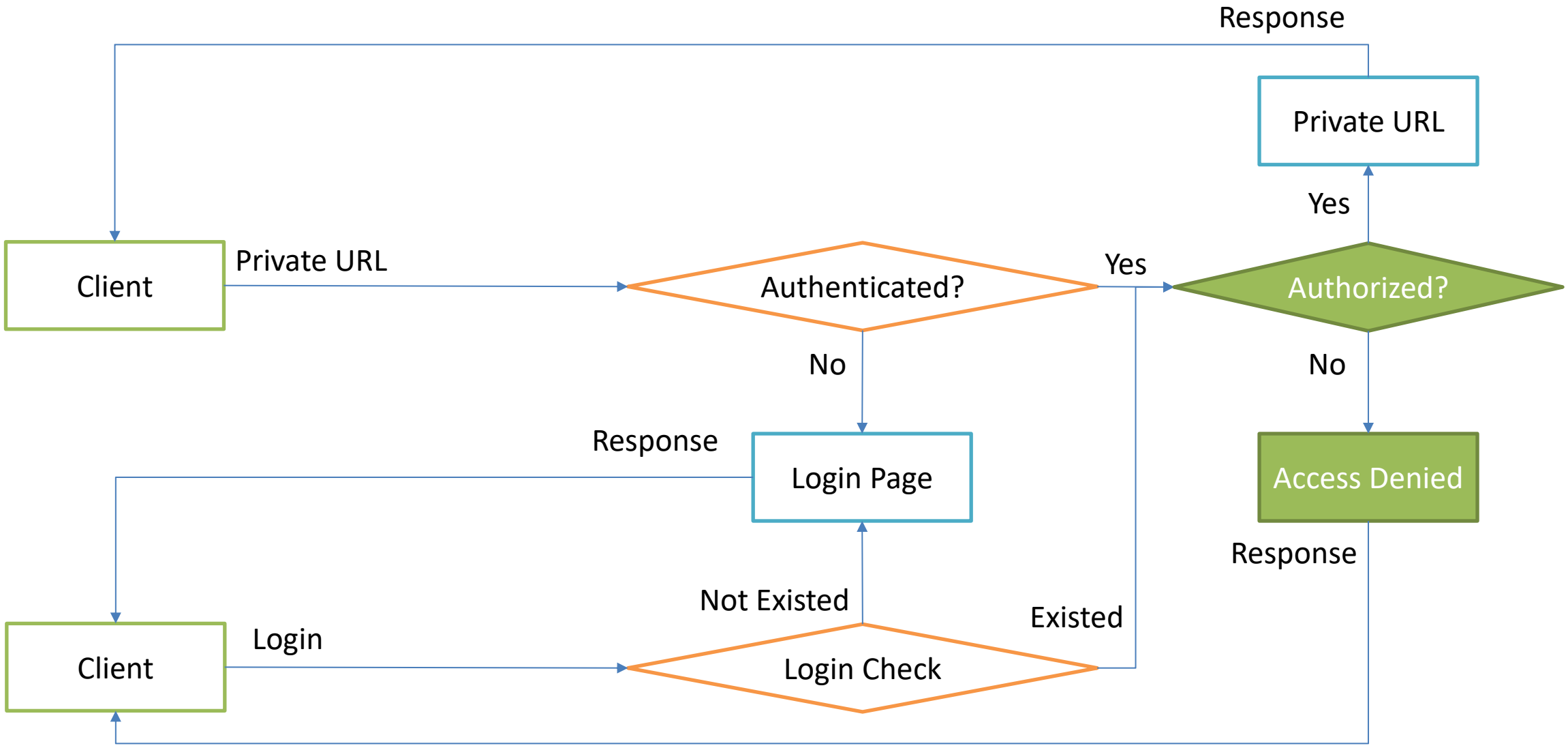
www.poly.edu.vn

AGENDA

- Giới thiệu Authorization
- UserDetailsService
  - InMemoryUserDetailsManager
  - JdbcUserDetailsManager
- Xây dựng UserDetailsService tùy biến

AUTHORIZATION

Authentication & Authorization Flow



Authorization Case Study

Controller

- ✓ @/
- ✓ @/poly/url0
- ✓ @/poly/url1
- ✓ @/poly/url2
- ✓ @/poly/url3
- ✓ @/poly/url4

Authorizing

Url	Role
@/poly/url1	Authenticated
@/poly/url2	USER
@/poly/url3	ADMIN
@/poly/url4	ADMIN và USER

User Source

Users

Username	Password	Enabled
user@gmail.com	123	True
admin@gmail.com	123	True
both@gmail.com	123	True

Authorities

Username	Authority
user@gmail.com	ROLE_USER
admin@gmail.com	ROLE_ADMIN
both@gmail.com	ROLE_USER
both@gmail.com	ROLE_ADMIN

InMemoryUserDetailsManager

- Sử dụng phương thức roles(String...roles) để gán vai trò cho user
- Mỗi user có thể có nhiều hoặc không có vai trò

```
@Bean
public UserDetailsService userDetailsService(PasswordEncoder pe) {
        String password = pe.encode("123");
        UserDetails user1 = User.withUsername("user@gmail.com")
                                .password(password).roles("USER").build();
        UserDetails user2 = User.withUsername("admin@gmail.com")
                                .password(password).roles("ADMIN").build();
        UserDetails user3 = User.withUsername("both@gmail.com")
                                .password(password).roles("USER", "ADMIN").build();
        return new InMemoryUserDetailsManager(user1, user2, user3);
}
```

AUTHORIZING

- Quyền truy xuất đến tài nguyên được phân cho vai trò nghĩa là những user có vai trò đó đều được phép truy xuất
- Sử dụng các phương thức
  - authenticated(): chỉ yêu cầu đăng nhập (không quan tâm đến vai trò)
  - hasRole(role): yêu cầu có vai trò
  - hasAnyRole: yêu cầu có nhiều vai trò
  - permitAll(): không bảo vệ

```
// Phân quyền sử dụng
http.authorizeHttpRequests(req -> {
        req.requestMatchers("/poly/url1").authenticated();
        req.requestMatchers("/poly/url2").hasRole("USER");
        req.requestMatchers("/poly/url3").hasRole("ADMIN");
        req.requestMatchers("/poly/url4").hasAnyRole("ADMIN", "USER");
        req.anyRequest().permitAll();
});
```

PHÂN BIỆT ROLE VÀ AUTHORITY

- Theo quy ước của Spring Security thì "USER" được hiểu là role và "ROLE_USER" được hiểu là authority.
- Ví dụ: 2 dòng mã lệnh sau đây là tương đương

```
UserDetails user = User.withUsername("user@gmail.com")
        .password(password).roles("USER").build();

UserDetails user = User.withUsername("user@gmail.com")
        .password(password).authorities("ROLE_USER").build();
```

PHÂN BIỆT ROLE VÀ AUTHORITY

- 2 đoạn mã lệnh sau đây là tương đương

```
http.authorizeHttpRequests(req -> {
        req.requestMatchers("/poly/url1").authenticated();
        req.requestMatchers("/poly/url2").hasAuthority("ROLE_USER");
        req.requestMatchers("/poly/url3").hasAuthority("ROLE_ADMIN");
        req.requestMatchers("/poly/url4").hasAnyAuthority("ROLE_ADMIN", "ROLE_USER");
        req.anyRequest().permitAll();
});
```

Tương đương

```
http.authorizeHttpRequests(req -> {
        req.requestMatchers("/poly/url1").authenticated();
        req.requestMatchers("/poly/url2").hasRole("USER");
        req.requestMatchers("/poly/url3").hasRole("ADMIN");
        req.requestMatchers("/poly/url4").hasAnyRole("ADMIN", "USER");
        req.anyRequest().permitAll();
});
```

ĐỌC VAI TRÒ TỪ ĐỐI TƯỢNG AUTHENTICATION

- 2 đoạn mã sau đây dung để bóc tách authority và role của người sử dụng đã đăng nhập

```
Authentication auth = SecurityContextHolder.getContext().getAuthentication();
// Đọc lấy authority
List<String> authorities = auth.getAuthorities().stream()
                                .map(au -> au.getAuthority()).toList();

// Đọc lấy role (chỉ lấy chuỗi con từ ký tự thứ 6 về cuối chuỗi)
List<String> roles = auth.getAuthorities().stream()
                         .map(au -> au.getAuthority().substring(5)).toList();
```

XỬ LÝ TỪ CHỐI TRUY XUẤT

- Cấu hình xử lý truy xuất tài nguyên chưa được phân quyền vai trò phù hợp.

```
http.exceptionHandling(denied -> denied.accessDeniedPage("/access-denied.html"));
```

- Trên đây yêu cầu hệ thống chuyển về 1 trang web tĩnh để hiển thị thông báo cho người sử dụng biết. Bạn có thể chuyển về 1 địa chỉ động (controller) để xử lý theo ý riêng của mình.

JdbcUserDetailsManager

CSDL SECURITY – THEO LƯỢC ĐỒ MẶC ĐỊNH

```
CREATE TABLE Users
(
   Username VARCHAR(50) NOT NULL,
   Password VARCHAR(500) NOT NULL,
   Enabled BIT NOT NULL,
   PRIMARY KEY(Username)
)

CREATE TABLE Authorities
(
   Id BIGINT NOT NULL identity(1, 1),
   Username VARCHAR(50) NOT NULL,
   Authority VARCHAR(50) NOT NULL,
   PRIMARY KEY(Id),
   UNIQUE(Username, Authority),
   FOREIGN KEY(Username) REFERENCES Users(Username)
      ON DELETE CASCADE ON UPDATE CASCADE
)
```

- Users lưu trữ thông tin người sử dụng
- Authorities lưu trữ thông tin phân vai trò
- 2 bảng này có cấu trúc dữ liệu tùy ý nhưng phải có những thành phần màu xanh.

DỮ LIỆU MẪU

- Nhập dữ liệu mẫu cho các bảng Users và Authorities theo đúng như Case Study:

```
INSERT INTO Users(Username, Password, Enabled) VALUES('user@gmail.com', '{noop}123', 1)
INSERT INTO Users(Username, Password, Enabled) VALUES('admin@gmail.com', '{noop}123', 1)
INSERT INTO Users(Username, Password, Enabled) VALUES('both@gmail.com', '{noop}123', 1)

INSERT INTO Authorities(Username, Authority) VALUES('user@gmail.com', 'ROLE_USER')
INSERT INTO Authorities(Username, Authority) VALUES('admin@gmail.com', 'ROLE_ADMIN')
INSERT INTO Authorities(Username, Authority) VALUES('both@gmail.com', 'ROLE_USER')
INSERT INTO Authorities(Username, Authority) VALUES('both@gmail.com', 'ROLE_ADMIN')
```

KHAI BÁO DATA SOURCE

- Khai thông tin kết nối CSDL vào file application.properties và thư viện cần thiết vào pom.xml

```
spring.datasource.url=jdbc:sqlserver://localhost;database=<db>;encrypt=true;trustServerCertificate=true;
spring.datasource.username=sa
spring.datasource.password=123
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto = none
```

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <scope>runtime</scope>
</dependency>
```

Cấu hình bean UserDetailsService

- Cấu hình như một trong 2 tình huống sau đây để sử dụng User Source từ CSDL

```
@Bean // CSDL đúng lược đồ mặc định
public UserDetailsService userDetailsService(DataSource dataSource) {
        return new JdbcUserDetailsManager(dataSource);
}

@Bean // CSDL không đúng lược đồ mặc định
public UserDetailsService userDetailsService(DataSource dataSource) {
        String userSql = "SELECT Username, Password, Enabled FROM Users WHERE Username=?";
        String roleSql = "SELECT Username, Authority FROM Authorities WHERE Username=?";
        JdbcUserDetailsManager manager = new JdbcUserDetailsManager(dataSource);
        manager.setUsersByUsernameQuery(userSql);
        manager.setAuthoritiesByUsernameQuery(roleSql);
        return manager;
}
```

CẤU HÌNH BEAN USERDETAILSSERVICE

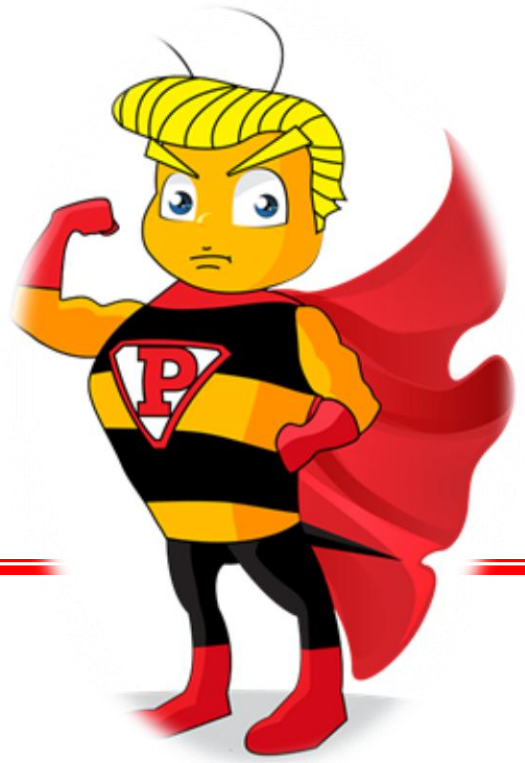
- Nếu bạn sử dụng CSDL không đúng lược đồ mặc định được quy định bởi Spring Security thì bạn phải viết mã theo tình huống 2, tức là bạn phải được ra 2 câu lệnh SQL để truy vấn user và authority theo username
  - User Query:

    SELECT Username ***AS Username***, Password ***AS Password***, Enabled ***AS Enabled***

    FROM Users WHERE Username=?
  - Authority Query:

    SELECT Username ***AS Username***, Authority ***AS Authority***

    FROM Authorities WHERE Username=?



TÙY BIẾN USERDETAILSMANAGER

SECURITY VỚI CSDL TÙY BIẾN

Users

Username	Password	Enabled
user@gmail.com	{noop}123	1
admin@gmail.com	{noop}123	1
both@gmail.com	{noop}123	1

Roles

Id	Name
ROLE_USER	Nhân viên
ROLE_ADMIN	Quản lý

UserRoles

Id	Username	RoleId
1	user@gmail.com	ROLE_USER
2	admin@gmail.com	ROLE_ADMIN
3	both@gmail.com	ROLE_USER
4	both@gmail.com	ROLE_ADMIN

ENTITY CLASS

```java
@Data
@Entity
@Table(name = "J6users")
public class User {
    @Id
    String username;
    String password;
    boolean enabled;
    @OneToMany(mappedBy = "user", fetch = FetchType.EAGER)
    List<UserRole> userRoles;
}
```

```java
@Data
@Entity
@Table(name = "J6userroles")
public class UserRole {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    @ManyToOne
    @JoinColumn(name = "username")
    User user;
    @ManyToOne
    @JoinColumn(name = "roleid")
    Role role;
}
```

```java
@Data
@Entity
@Table(name = "J6roles")
public class Role {
    @Id
    String id;
    String name;
    @OneToMany(mappedBy = "role")
    List<UserRole> userRoles;
}
```

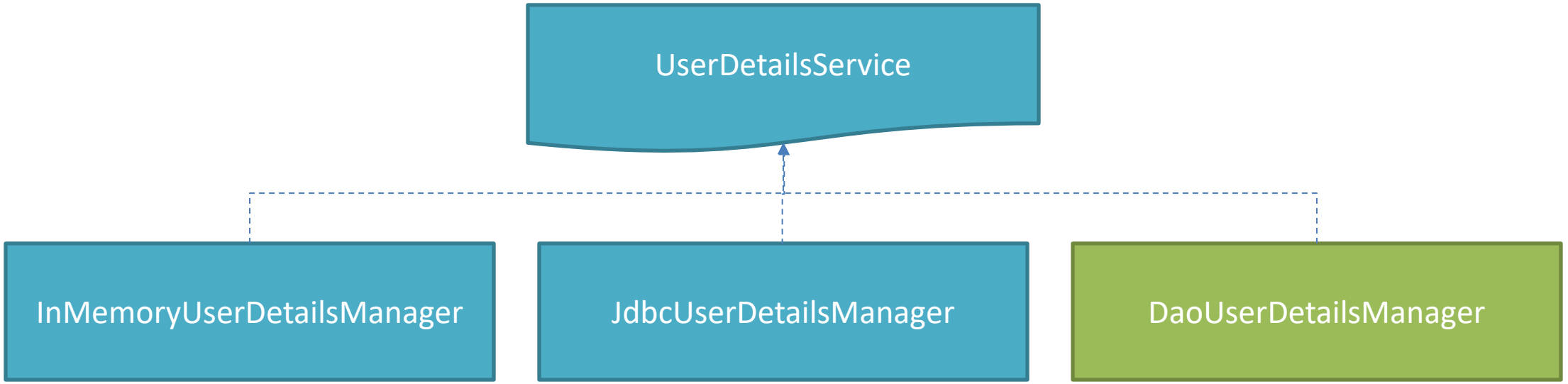
DAO - Data Access Object

```
public interface UserDAO extends JpaRepository<User, String> {}

public interface RoleDAO extends JpaRepository<Role, String> {}

public interface UserRoleDAO extends JpaRepository<UserRole, Long> {}
```

PHÂN CẤP THỪA KẾ USERDETAILSSERVICE



- InMemoryUserDetailsManager và JdbcUserDetailsManager do Spring Security cung cấp sẵn rất thuận tiện trong việc triển khai security.
- Một số tình huống đòi hỏi sự tùy biến cao thì chúng ta phải xây dựng DaoUserDetailsManager cho riêng mình và thực thi theo interface UserDetailsService

XÂY DỰNG DAOUSERDETAILSMANAGER

```
public class DaoUserDetailsManager implements UserDetailsService {
    @Autowired
    UserDAO dao;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        poly.demo.entity.User user = dao.findById(username).get();
        String password = user.getPassword();
        String[] roles = user.getUserRoles().stream()
                .map(ur -> ur.getRole().getId().substring(5))
                .toList().toArray(String[]::new);
        return User.withUsername(username).password(password).roles(roles).build();
    }
}
```

CẤU HÌNH USERDETAILSSERVICE

- Khai báo bean UserDetailsService với DaoUserDetailsManager thay cho JdbcUserDetailsManager

```
@Bean
public UserDetailsService userDetailsService() {
        return new DaoUserDetailsManager();
}
```

SUMMARY

- ☑ Giới thiệu Authorization
- ☑ UserDetailsService
  - ☑ InMemoryUserDetailsManager
  - ☑ JdbcUserDetailsManager
- ☑ Xây dựng UserDetailsService tùy biến


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn