통합 구현(Spring, Django)

Spring Security 적용하기

한국기술교육대학교
온라인평생교육원

학습내용

- Spring Security 알아보기
- 회원가입
- 로그인

학습목표

- Spring Security에 대해 설명할 수 있다.
- 회원가입 기능을 구현할 수 있다.
- 로그인 기능을 구현할 수 있다.

Spring Security 알아보기

1 Spring Security란

《1》 Spring Security

1. 스프링 프레임워크를 통해 각종 로그인 처리, 권한 처리 등을 수행
2. 기본적으로 자원에 대한 접근을 가로채(Intercept) 권한을 체크
3. 간단한 설정으로 보안 수준을 유지 가능

《2》 Spring Security의 인증

Credential Based Authentication : 권한을 부여받는데 필요한 사용자명과 비밀번호를 입력하여 그 비밀번호가 저장된 비밀번호와 일치하는지 확인

일반적으로 Spring Security에서는 아이디를 Principal, 비밀번호를 Credential이라고 부름

《3》 Spring Security의 권한 부여

Granted Authority	로그인 등을 통해 적절한 인증이 성공했다면 미리 설정한 권한이 부여됨
Intercept	DispatcherServlet보다 먼저 요청을 가로채서 권한에 따른 Resource 접근 필터링을 함

Spring Security 알아보기

2 환경 설정하기

/src/main/webapp/WEB-INF/web.xml

```
<!--SpringSecurityFilterChain-->
  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

<!--dispatcher-servlet.xml-->
  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
```

Spring Security 알아보기

2 환경 설정하기

/src/main/resources/security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:sec="http://www.springframework.org/schema/security"
      xsi:schemaLocation="
   http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
   http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.2.xsd">

   <sec:global-method-security secured-annotations="enabled" pre-post-
annotations="enabled" />
   <sec:http use-expressions="true">
      <sec:form-login login-page="/user/signin" default-target-
url="/user/signinSuccess" authentication-failure-url="/user/signinFailed"/>

 <sec:logout logout-url="/user/signout" logout-success-url="/user/signin" />
      <sec:intercept-url pattern="/user/onlyUserByXml"
access="hasRole('ROLE_USER')" />
      <sec:intercept-url pattern="/user/onlyAdminByXml"
access="hasRole('ROLE_ADMIN')" />
      <sec:intercept-url pattern="/**" access="permitAll" />
   </sec:http>
   <bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"
/>
   <sec:authentication-manager>
      <sec:authentication-provider user-service-ref="userService">
         <sec:password-encoder ref="passwordEncoder" />
      </sec:authentication-provider>
   </sec:authentication-manager>
</beans>
```

Spring Security 알아보기

2 환경 설정하기

security.xml 주요 설정 분석

1 로그인 주소, 로그인 성공했을 때 이동할 주소, 실패했을 때 이동할 주소에 대한 설정

```
<sec:form-login login-page="/user/signin" default-target-
url="/user/signinSuccess" authentication-failure-url="/user/signinFailed"/>
```

2 로그아웃 주소, 로그아웃 성공했을 때 이동할 주소에 대한 설정

```
<sec:logout logout-url="/user/signout" logout-success-url="/user/signin" />
```

3 암호화된 패스워드를 읽을 수 있는 인코더(Md5, Sha256, BCrypt 등) 설정

```
<bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />
```

회원가입

1 UserDetails와 GrantedAuthority

1 UserDetails

Spring Security기반 회원 객체를 구현하기 위해서는 UserDetails Interface를 Implement 해야 하며, 이 때 다음과 같은 함수를 구현해야 함

Modifier and Type	Method and Description
Collection<? Extends GrantedAuthority>	getAutoritis() Return theauthorities granted to the user.
String	Getpassword() Returns the password used to authenticate the user.
String	Getpassword() Returns the user nae used to authenticate the user.
boolean	isAccountNonExpired() Indicates whether the user's account has expired.
boolean	isAccountNonlocked() Indicates whether the user is locked or unlocked.
boolean	isCredentialNonExpired() Indicates whether the user's credentials (password) has expired.
boolean	isenabled() Indicates whether the user is enabled or disabled.

회원가입

1 UserDetails와 GrantedAuthority

2 GrantedAuthority

회원에 대한 권한 객체를 구현하기 위해서는 Granted Authority Interface를 Implement해야 하며, 이 때 다음과 같은 함수를 구현해야 함

Modifier and Type	Method and Description
String	getAutority() If the GrantedAutority can be reperesented as a String and that String is sufficient in precision to be relied upon for an access control decision by AccessDecisionManager(or delegate), this method should return such a String.

회원가입

2 회원가입 서비스 구현

⟨1⟩ UserService에서 회원가입 함수 구현

```
public Boolean signup(User user) {
    if(user.getEmail() == null || user.getPassword() ==  null)
        return false;

    user.setPassword(passwordEncoder.encode(user.getPassword()));

    // 사용자의 실제 비밀번호를 암호화함
    userMapper.insert(user);

    Authority authority = new Authority();
    authority.setUserId(user.getId());
    authority.setRole("ROLE_USER");
    authorityMapper.insert(authority);

    System.out.println("user signup :" + new Date());
    return true;
}
```

⟨2⟩ 비밀번호 암호화

회원가입 시 비밀번호는 평문으로 서버에 전송됨

이 때, 평문으로 비밀번호를 데이터베이스에 저장 시, 보안에 심각한 위험이 생길 수 있음

비밀번호는 SHA256, Bcrypt 등 최신 암호화 기술로 암호화해서 새로 저장

이때, 인코딩은 단방향 암호화 방식을 쓰며, 암호화된 비밀번호를 다시 해석할 수 없이 올바른 비밀번호를 입력했을 때만 매칭

회원가입

2 회원가입 서비스 구현

3 권한 부여

1. Spring Security에서 권한은 기본적으로 “ROLE_권한이름” 형식으로 관리함

2. 이 때, ROLE_USER는 기본적인 유저 권한을 나타내며, 필요에 따라 ROLE_ADMIN, ROLE_MANAGER 등의 관리자 권한을 부여할 수 있음

로그인

1 로그인 폼 구성

⟨1⟩ 기본 설정

로그인 페이지에서 유저네임은 j_username, 비밀번호는 j_password,
폼 액션대상은 j_spring_security로 지정하면 Spring Security를 바탕으로 로그인을 시도함

```
<form action="j_spring_security_check" method="post">
    <input type="text" placeholder="email" name="j_username"/>
    <input type="password" placeholder="password"
name="j_password"/>
    <input type="submit" value="Signin"/>
</form>
```

⟨2⟩ 로그인 후 처리

security.xml에 로그인 주소, 로그인 성공했을 때 이동할 주소, 실패했을 때 이동할 주소에 대한 설정

```
<sec:form-login login-page="/user/signin" default-target-
url="/user/signinSuccess" authentication-failure-url="/user/signinFailed"/>
```

로그인 페이지 URL	/user/signin
로그인 성공 후 이동 URL	/user/signinSuccess
로그인 실패 후 이동 URL	/user/signinFailed

로그인

2 데이터베이스를 바탕으로 한 로그인

〈1〉 UserDetailsService

유저의 로그인을 서비스로 구현하기 위해서는 UserDetailsService Interface를 Implement 해야 하며, 이 때 다음과 같은 함수를 구현해야 함

All Method	Instance Method	Abstract Method
Modifier and Type	Method and Description	
UserDetails	loadUserByUsername(String username) Locates the user based on the username.	

〈2〉 loadUserByUsername 함수 구현

1. UserService에서 데이터베이스를 통해 유저 객체를 가져오는 함수를 구현함
2. 이 때, 유저가 존재한다면 해당 유저의 권한 또한 가져와서 유저 객체에 삽입함
3. 로그인 성공 시 유저 객체를 리턴함

로그인

2 데이터베이스를 바탕으로 한 로그인

〈2〉 loadUserByUsername 함수 구현

```
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userMapper.findByEmail(username);
        if (user == null) {
            throw new UsernameNotFoundException("Invalid
username/password.");
        }
        List<Authority> authorities = authorityMapper.findByUserId(user.getId());
        user.setAuthorities(authorities);
        System.out.println("user = " + user);
        return user;
    }
```

〈3〉 로그인 성공/실패에 대한 URL 구성 예시

앞서 Security.xml에 설정한 URL을 Cotroller에 구성

```
@RequestMapping(value="/signinSuccess")
    @ResponseBody
    public String signinSuccess() {
        System.out.println("signin Success");
        return "signinSuccess";
    }

    @RequestMapping(value="/signinFailed")
    @ResponseBody
    public String signinFailed() {
        System.out.println("signin Failed");
        return "signinFailed";
    }
```

학습정리

1. Spring Security 알아보기

- Spring Security
  - 스프링 프레임워크를 통해 각종 로그인 처리, 권한 처리 등을 수행
  - 기본적으로 자원에 대한 접근을 가로채(Intercept) 권한을 체크
  - 간단한 설정으로 보안 수준을 유지 가능
- 환경 설정하기
  - /src/main/webapp/WEB-INF/web.xml
  - /src/main/resources/security.xml
  - security.xml 주요 설정 분석

학습정리

2. 회원가입

- UserDetails와 GrantedAuthority
  - UserDetails : Spring Security기반 회원 객체를 구현하기 위해서는 UserDetails Interface를 Implement해야 함
  - GrantedAuthority : 회원에 대한 권한 객체를 구현하기 위해서는 Granted Authority Interface를 Implement해야 함
- 회원가입 서비스 구현
  - UserService에서 회원가입 함수 구현
  - 비밀번호 암호화
    - 회원가입 시 비밀번호는 평문으로 서버에 전송됨
    - 비밀번호는 SHA256, Bcrypt 등 최신 암호화 기술로 암호화해서 새로 저장해야 함
    - 이 때, 인코딩은 단방향 암호화 방식을 쓰며, 암호화된 비밀번호를 다시 해석할 수 없이 올바른 비밀번호를 입력했을 때만 매칭이 됨
  - 권한 부여
    - Spring Security에서 권한은 기본적으로 “ROLE_권한이름” 형식으로 관리함

학습정리

3. 로그인

- 로그인 폼 구성
  - 로그인 페이지에서 유저네임은 j_username, 비밀번호는 j_password, 폼 액션대상은 j_spring_security로 지정하면 Spring Security를 바탕으로 로그인을 시도함
  - Security.xml에 로그인 주소, 로그인 성공했을 때 이동할 주소, 실패했을 때 이동할 주소에 대한 설정

- 데이터베이스를 바탕으로 한 로그인
  - UserDetailsService : 유저의 로그인을 서비스로 구현하기 위해서는 UserDetailsService Interface를 Implement해야 함
  - loadUserByUsername 함수 구현 : UserService에서 데이터베이스를 통해 유저 객체를 가져오는 함수를 구현하고, 이 때, 유저가 존재한다면 해당 유저의 권한 또한 가져와서 유저 객체에 삽입. 로그인 성공 시 유저 객체를 리턴함