Spring Security

- 웹 시큐리티
- 메소드 시큐리티
- 다양한 인증 방법 지원
    - LDAP, 폼 인증, Basic 인증, OAuth, ...

Security 적용 전

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>Welcome!</h1>
<a href="/hello">Hello</a>
<a href="/my">my</a>
</body>
</html>
```

my.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>My</h1>
</body>
</html>
```

hello.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>My</h1>
</body>
</html>
```

HomeController.java

```
package com.megait.security;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {


    @RequestMapping("/my")
    public String my(){
        return "my";
    }

    @RequestMapping("/hello")
    public String hello(){
        return "hello";
    }
}
```

결과 Test 해보기

`@WebMvcTest` 추가

```
package com.megait.security;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.BootstrapWith;
import org.springframework.test.web.servlet.MockMvc;

import static
org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
```

```
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;


@WebMvcTest(HomeController.class)
class SecurityApplicationTests {

    @Autowired
    MockMvc mockMvc;

    @Test
    void helloTest() throws Exception{
        mockMvc.perform(get("/hello"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(view().name("hello"));
    }

    @Test
    void myTest() throws Exception{
        mockMvc.perform(get("/my"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(view().name("my"));
    }

}
```

Security 적용

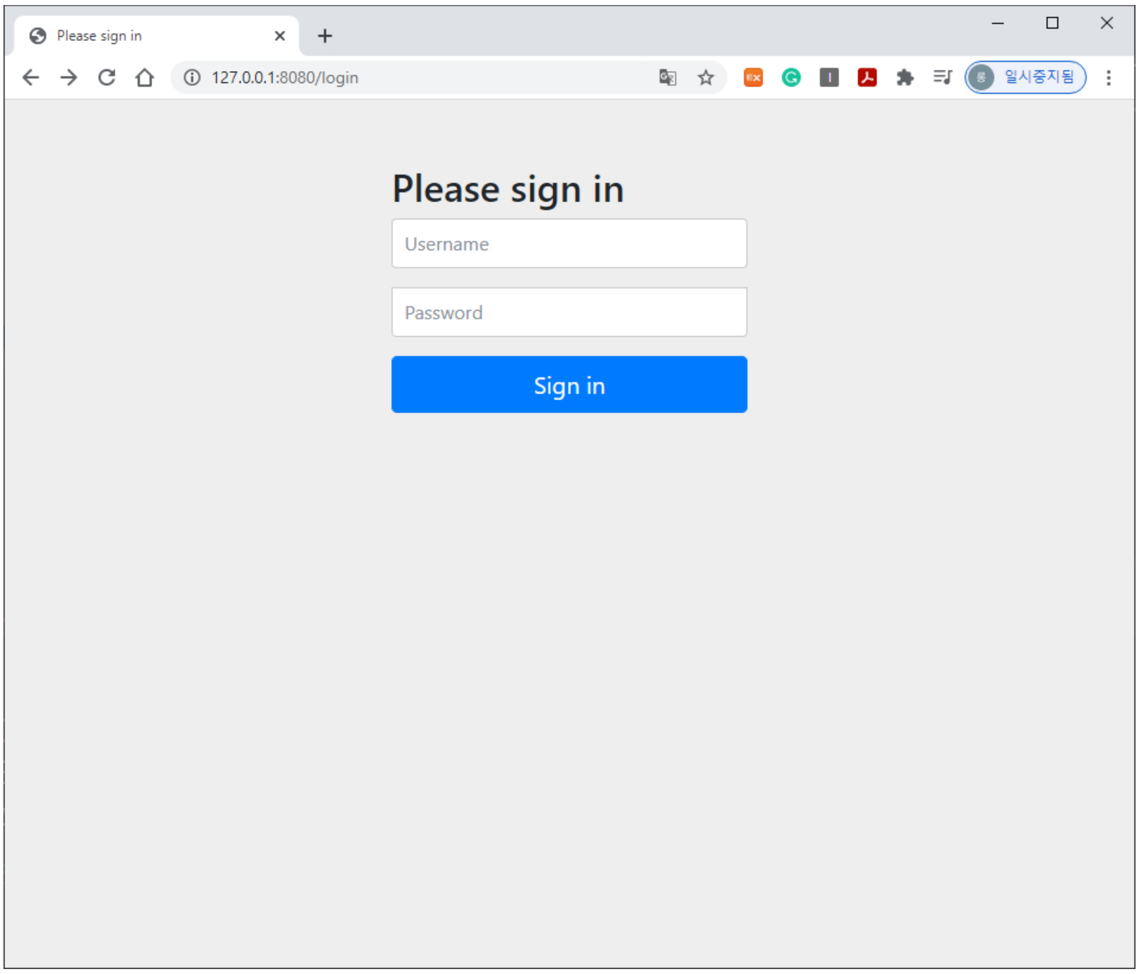
pom.xml 에 다음 의존성 추가

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Security 가 적용되면 모든 테스트가 실패한다. (Error message = Unauthorized)

대신 index 를 요청하면 로그인하라는 페이지로 리다이렉트한다.

결과



ID: user

PW : 콘솔 확인

(Using generated security password: b4ec145b-0730-4384-9de3-5f403c7a7209)

- spring.security.user.name
- spring.security.user.password

으로도 default 유저, 패스워드 속성 지정 가능

Spring Security 자동 설정

- `SecurityAutoConfiguration` (Spring Framework로 부터 대부분의 자동 설정을 받는다.)
  - `DefaultAuthenticationEventPublisher`
  - `SpringBootWebSecurityConfiguration.getHttp()`

    -
      ```
      @ConditionalOnMissingBean(WebSecurityConfigurerAdapter.class)
      ```

      `WebSecurityConfigurerAdapter` 빈이 없으면 적용된다.
- `UserDetailsServiceAutoConfiguration` (기본 비밀번호를 생성한다.)

-
  ```
  @ConditionalOnMissingBean(
  value = {
    AuthenticationManager.class,
    AuthenticationProvider.class,
    UserDetailsService.class }
    ...)
  ```

  위에 선언된 빈이 모두 없으면 적용된다. (하나라도 있으면 그게 적용됨)

참고) Mocking 인증

테스트 단위에서 인증을 Mocking 하고 싶다면?

pom.xml에 의존성 추가

```xml
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <version>${spring-security.version}</version>
    <scope>test</scope>
</dependency>
```

test

```java
package com.megait.security;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.web.servlet.MockMvc;

import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static
org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;


@WebMvcTest(HomeController.class)
class SecurityApplicationTests {

    @Autowired
    MockMvc mockMvc;

    @Test
    @WithMockUser // 이 부분!!
    void helloTest() throws Exception{
        mockMvc.perform(get("/hello"))
                .andDo(print())
```

```
                .andExpect(status().isOk()) // 이 부분!!!
                .andExpect(view().name("hello"));
    }

    @Test
    void myTestWihoutUser() throws Exception{ // 메서드 수정
        mockMvc.perform(get("/my"))
                .andDo(print())
                .andExpect(status().isUnauthorized())  // 이 부분!!!
                .andExpect(view().name("my"));
    }
}
```

Spring Security 커스터마이징

Account (회원 엔티티) 생성

Account

```
package com.megait.security.account;


import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Account {

    @Id @GeneratedValue
    private Long id;

    private String username;

    private String password;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

AccountService

`implements UserDetailsService` <-- 아주 중요!

```
package com.megait.security.account;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.Collection;
import java.util.Optional;

@Service
public class AccountService implements UserDetailsService {

    @Autowired
    private AccountRepository accountRepository;

    public Account createAccount(String username, String password){
        Account account = new Account();
        account.setUsername(username);
        account.setPassword(password);

        return accountRepository.save(account);
    }

    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {
        Optional<Account> byUsername = accountRepository.findByUsername(s);
        Account account = byUsername.orElseThrow(()->new
UsernameNotFoundException(s));
        return new User(account.getUsername(), account.getPassword(),
authorities() );
    }

    private Collection<? extends GrantedAuthority> authorities() {
        return Arrays.asList(new SimpleGrantedAuthority("ROLE_USER"));
    }
}
```

Optional 이란? http://www.tcpschool.com/java/java_stream_optional

AccountRepository

```
package com.megait.security.account;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface AccountRepository extends JpaRepository<Account, Long> {

    Optional<Account> findByUsername(String s);
}
```

웹 시큐리티 Configuration

`@Configuration`, `extends WebSecurityConfigurerAdapter`

```
package com.megait.security.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;

@Configuration
public class MySecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/", "/hello").permitAll()
                .anyRequest().authenticated()
                .and()
            .formLogin()
                .and()
            .httpBasic();
    }
}
```

password encoding을 안하면 예외발생

PasswordEncoder 설정

MySecurityConfig

```
package com.megait.security.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.crypto.factory.PasswordEncoderFactories;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
public class MySecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/", "/hello").permitAll()
                .anyRequest().authenticated()
                .and()
            .formLogin()
                .and()
            .httpBasic();
    }

    // 이 부분 추가
    @Bean
    public PasswordEncoder passwordEncoder(){
        return PasswordEncoderFactories.createDelegatingPasswordEncoder();
    }
}
```

AccountService

```
package com.megait.security.account;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.Collection;
```

```java
import java.util.Optional;

@Service
public class AccountService implements UserDetailsService {

    @Autowired
    private AccountRepository accountRepository;

    @Autowired // 이 부분 추가!
    private PasswordEncoder passwordEncoder;

    public Account createAccount(String username, String password){
        Account account = new Account();
        account.setUsername(username);
        account.setPassword(passwordEncoder.encode(password)); // 이 부분 수정!

        return accountRepository.save(account);
    }

    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {
        Optional<Account> byUsername = accountRepository.findByUsername(s);
        Account account = byUsername.orElseThrow(()->new
UsernameNotFoundException(s));
        return new User(account.getUsername(), account.getPassword(),
authorities() );
    }

    private Collection<? extends GrantedAuthority> authorities() {
        return Arrays.asList(new SimpleGrantedAuthority("ROLE_USER"));
    }
}
```