

Spring Security

실행환경

springframework 5.3.20

- spring-context
- spring-webmvc
- spring-test

springsecurity 5.7.1

- spring-security-core
- spring-security-web
- spring-security-config
- spring-security-taglibs

slf4j 2.0.7

ojdbc10 19.21.0.0

mybatis 3.5.9

mybatis-spring 3.0.3

lombok 1.18.30

HikariCP 4.0.3

jstl 1.2

junit 4.12

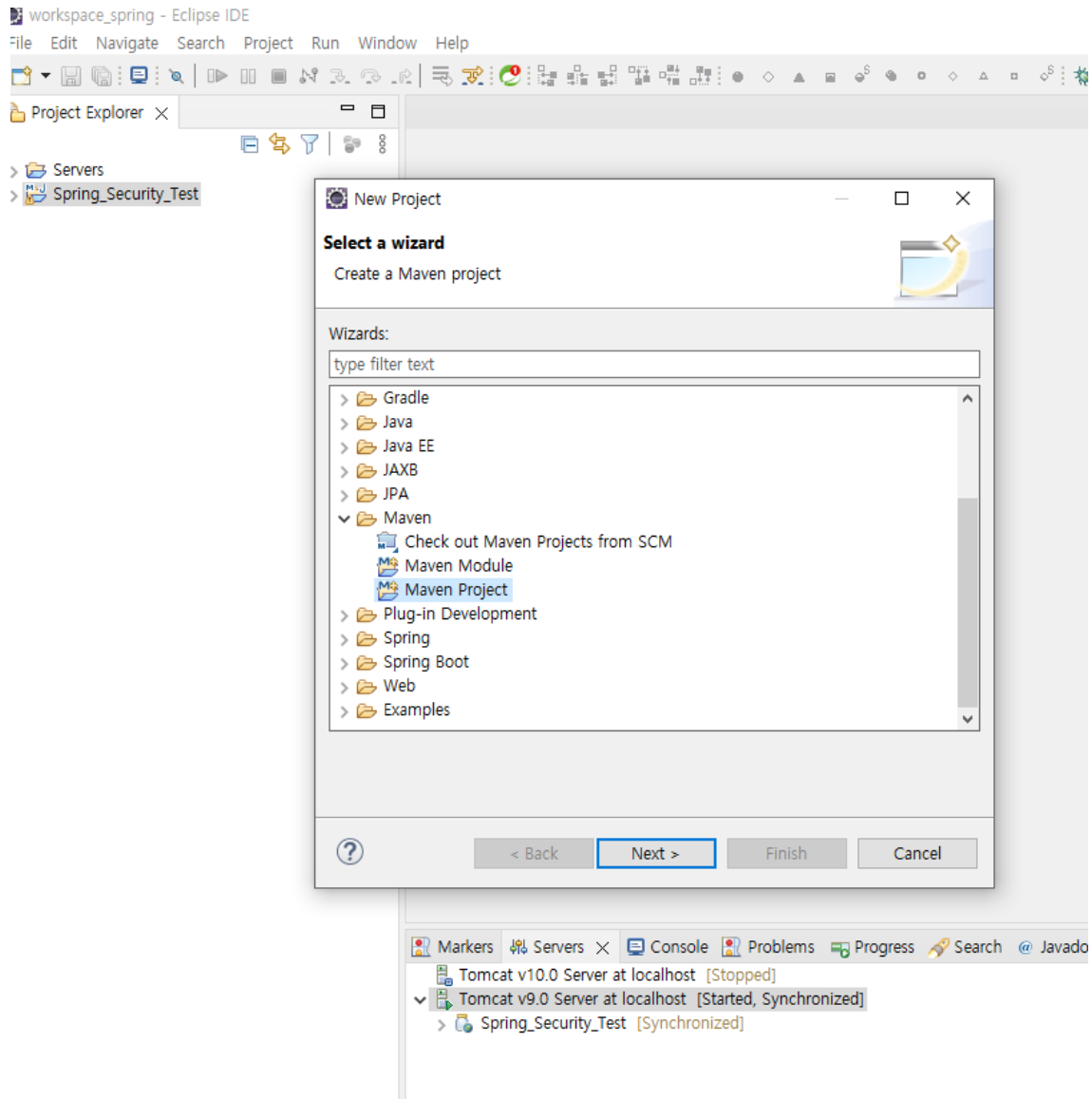
eclipse 2022-03(4.23)

Oracle Database 21c Express Edition for Windows x64

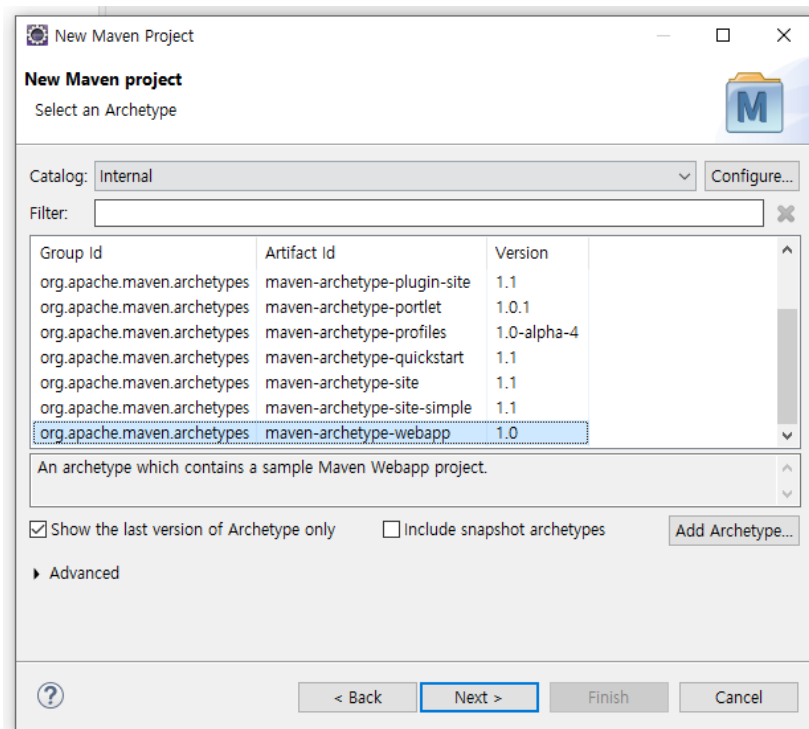
참고사이트 : <https://m.blog.naver.com/mangocomputer/223354717695>

프로젝트 생성

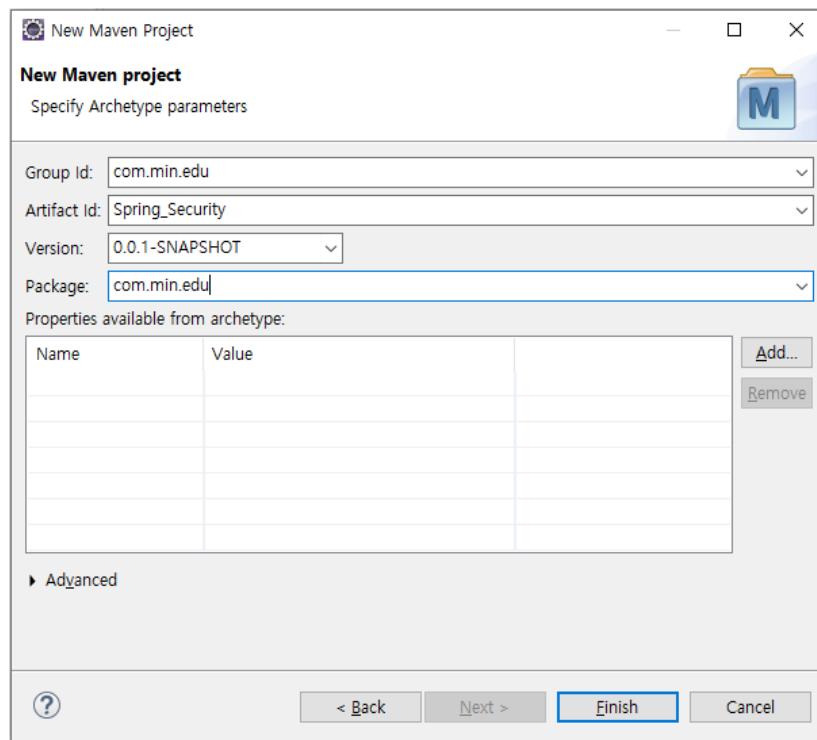
1. Maven Project 생성



2. webapp 선택



3. Spring_Security



New Maven Project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value

▶ Advanced

< Back Next > **Finish** Cancel

전자정부프레임워크 버전 공지

<https://www.egovframe.go.kr/home/ntt/nttRead.do?pagerOffset=10&searchKey=&searchValue=&menuNo=74&bbsId=6&nttId=1868>

4. pom.xml설정

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>com.min.edu</groupId>
```

```
    <artifactId>Spring_Security</artifactId>
```

```
    <packaging>war</packaging>
```

```
    <version>0.0.1-SNAPSHOT</version>
```

```
    <name>Spring_Security Maven Webapp</name>
```

```
    <url>http://maven.apache.org</url>
```

```
    <properties>
```

```
        <org.springframework>5.3.20</org.springframework>
```

```
        <org.springframework.security>5.7.1</org.springframework.security>
```

```
        <org.slf4j-version>2.0.7</org.slf4j-version>
```

```
    </properties>
```

```
    <dependencies>
```

```
        <!-- IoC 컨테이너 제공/DI지원/이벤트 처리/AOP지원/Bean Lifecycle 관리/환경 설정 관리 -->
```

```
        <dependency>
```

```
            <groupId>org.springframework</groupId>
```

```
            <artifactId>spring-context</artifactId>
```

```
            <version>${org.springframework}</version>
```

```
        </dependency>
```

```
        <!-- Spring 프레임워크에서 웹 애플리케이션과 관련된 기능을 제공하는 모듈의 의존성 -->
```

```
        <dependency>
```

```
            <groupId>org.springframework</groupId>
```

```
            <artifactId>spring-webmvc</artifactId>
```

```

        <version>${org.springframework}</version>
    </dependency>

    <!-- Spring 애플리케이션의 테스트를 지원하는 모듈의 의존성으로, 테스트 스코프로 설정되어 있음
-->

    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-test</artifactId>

        <version>${org.springframework}</version>

        <scope>test</scope>

    </dependency>

    <!-- Spring Security 사용을 위한 dependency 등록 Start -->

    <!-- 인증 및 액세스 제어 기능이 포함 -->

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-core</artifactId>

        <version>${org.springframework.security}</version>

    </dependency>

    <!-- Servlet 환경에서 URL 액세스 제어를 가능하게 하는 필터 및 관련 웹 Security 인프라가 포함 --
>

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-web</artifactId>

        <version>${org.springframework.security}</version>

    </dependency>

    <!-- XML 또는 JavaConfig를 사용하여 인증, 권한부여 등 설정을 지정하는데 사용 -->

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-config</artifactId>

        <version>${org.springframework.security}</version>

    </dependency>

```

<!-- jsp에서 taglib를 사용하기 위한 추가 -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-taglibs</artifactId>

<version>\${org.springframework.security}</version>

</dependency>

<!-- Spring Security 사용을 위한 dependency 등록 End -->

<!-- 로그처리를 자동으로 해주는 라이브러리 log4j보다 향상된 기능을 제공, slf4j는 log4j를
재정의한 구현체 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>\${org.slf4j-version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>\${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>\${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.15</version>


```

        <exclusions>
            <exclusion>
                <groupId>javax.mail</groupId>
                <artifactId>mail</artifactId>
            </exclusion>
            <exclusion>
                <groupId>javax.jms</groupId>
                <artifactId>jms</artifactId>
            </exclusion>
            <exclusion>
                <groupId>com.sun.jdmk</groupId>
                <artifactId>jmxtools</artifactId>
            </exclusion>
            <exclusion>
                <groupId>com.sun.jmx</groupId>
                <artifactId>jmxri</artifactId>
            </exclusion>
        </exclusions>
        <scope>runtime</scope>
    </dependency>

    <!-- oracle jdbc를 활용하기 위한 모듈 -->
    <dependency>
        <groupId>com.oracle.database.jdbc</groupId>
        <artifactId>ojdbc10</artifactId>
        <version>19.21.0.0</version>
    </dependency>

    <!-- MyBatis를 Spring 프레임워크와 통합하는 모듈의 의존성. -->
    <dependency>
        <groupId>org.mybatis</groupId>

```

```
        <artifactId>mybatis</artifactId>

        <version>3.5.9</version>

</dependency>
```

```
<!-- MyBatis를 Spring 프레임워크와 통합하는 모듈 -->
```

```
<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-orm</artifactId>

    <version>${org.springframework}</version>

</dependency>
```

```
<!--MyBatis를 Spring 프레임워크와 통합하는 모듈의 의존성.-->
```

```
<dependency>

    <groupId>org.mybatis</groupId>

    <artifactId>mybatis-spring</artifactId>

    <version>3.0.3</version>

</dependency>
```

```
<dependency>

    <groupId>org.projectlombok</groupId>

    <artifactId>lombok</artifactId>

    <version>1.18.30</version>

    <scope>provided</scope>

</dependency>
```

```
<!-- Java Standard Tag Library 자바의 문법은 JSP파일에서 쉽게 Tag를 통해서 사용할 수 있도록
```

```
해줌 -->
```

```
<dependency>

    <groupId>javax.servlet</groupId>
```

```

        <artifactId>jstl</artifactId>

        <version>1.2</version>

    </dependency>

    <dependency>

        <groupId>com.zaxxer</groupId>

        <artifactId>HikariCP</artifactId>

        <version>4.0.3</version>

    </dependency>

    <dependency>

        <groupId>junit</groupId>

        <artifactId>junit</artifactId>

        <version>4.12</version>

        <scope>test</scope>

    </dependency>
</dependencies>

<build>

    <finalName>Spring_Security</finalName>

    <plugins>

        <plugin>

            <groupId>org.apache.maven.plugins</groupId>

            <artifactId>maven-compiler-plugin</artifactId>

            <version>3.10.1</version>

            <configuration>

                <source>17</source>

                <target>17</target>

            </configuration>

        </plugin>

        <plugin>

```

```
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.3.2</version>
    </plugin>
</plugins>

</build>

</project>
```

5. web.xml 설정

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

    <!-- 첫페이지 설정 -->

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

</web-app>
```

6. index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Hello</title>

</head>

<body>

    <div>

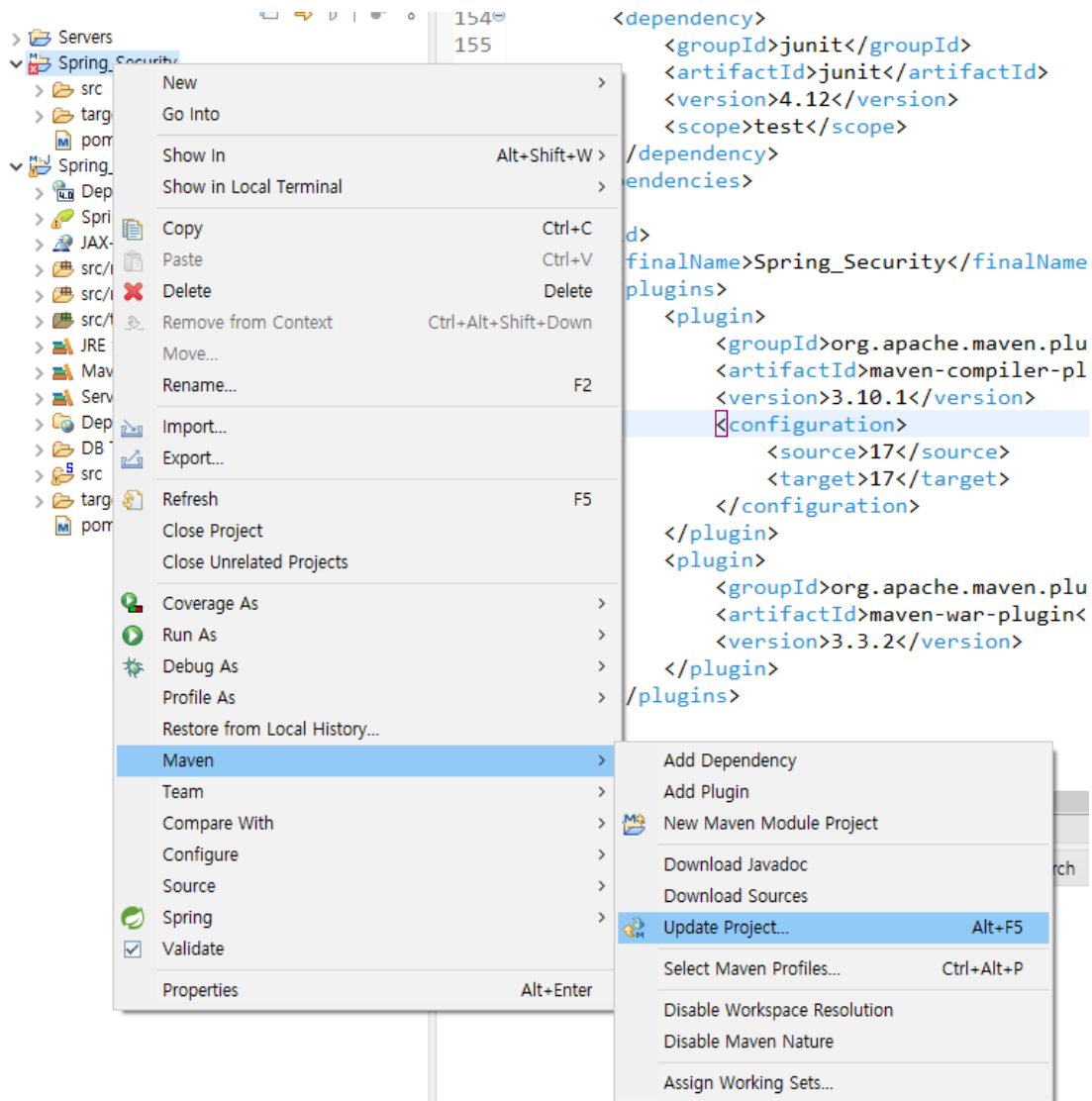
        Hello!!

    </div>

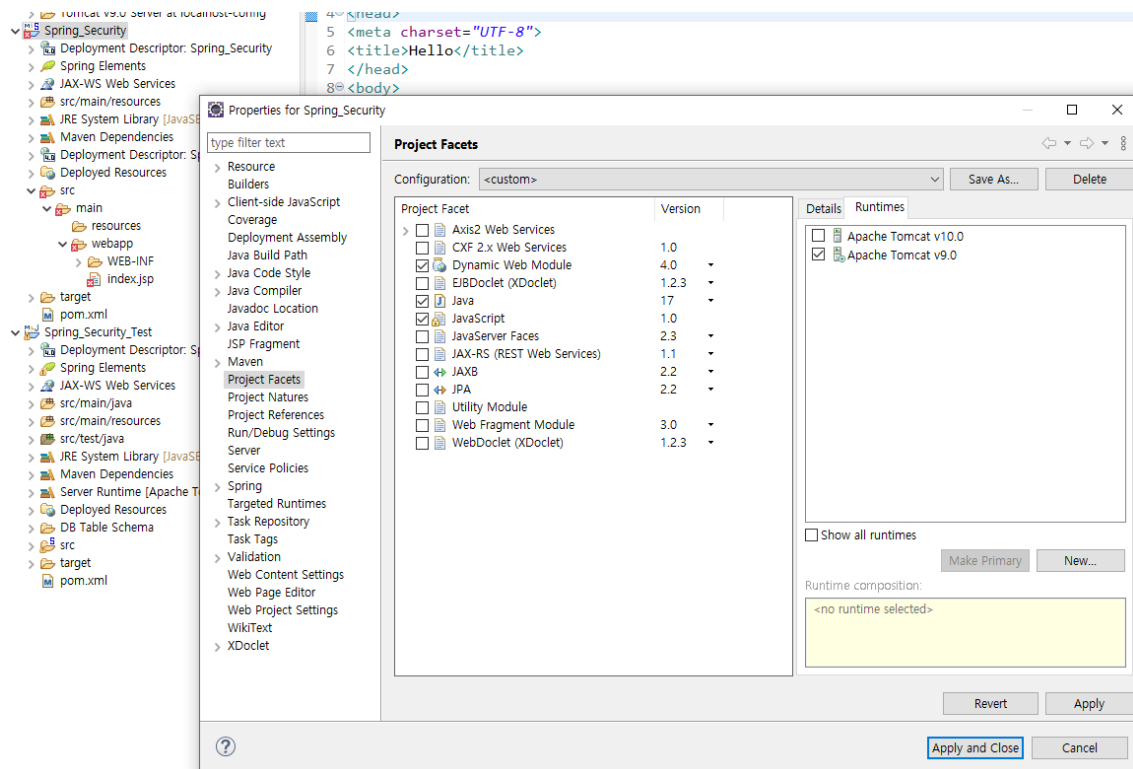
</body>

</html>
```

7. Maven Update



8. Project Facets 설정



http://localhost:8080/Spring_Security/

9. 회원 테이블 생성

CREATE TABLE MEMBER_T (

 ID VARCHAR2(20) NOT NULL ENABLE,

 NAME VARCHAR2(50) NOT NULL ENABLE,

 PASSWORD VARCHAR2(100) NOT NULL ENABLE,

 AUTH VARCHAR2(20) DEFAULT ('USER') NOT NULL ENABLE,

 JOINDATE DATE DEFAULT (SYSDATE) NOT NULL ENABLE

);

ALTER TABLE MEMBER_T ADD CONSTRAINT PK_MEMBER_T_ID PRIMARY KEY (ID);

10. 회원가입 처리를 위한 설정

10-1. /resources/properties/db.properties 생성 (DB연결 설정 처리)

```
driver=oracle.jdbc.driver.OracleDriver  
  
url=jdbc:oracle:thin:@localhost:1521:xe  
  
user=GD  
  
pwd=GD
```

10-2. /resources/sqls/UserMapper.xml Mapper 생성

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  
  
<mapper namespace="com.min.edu.dao.UserInfoDaoImpl">  
  
    <!-- 로그인 처리 시 사용 -->  
  
    <select id="getUserInfo" parameterType="String" resultType="UserInfoVo">  
  
        SELECT  
  
            ID ,NAME ,PASSWORD, AUTH ,JOINDATE  
  
        FROM MEMBER_T  
  
        WHERE ID = #{id}  
  
    </select>  
  
  
    <!-- 회원리스트 -->  
  
    <select id="getUserInfoAllList" resultType="UserInfoVo">  
  
        SELECT  
  
            ID ,NAME , AUTH ,JOINDATE  
  
        FROM MEMBER_T  
  
        ORDER BY JOINDATE DESC  
  
    </select>  
  
  
    <!-- 회원가입 처리 시 사용 -->
```

```

<insert id="insertUserInfo" parameterType="UserInfoVo">

    INSERT INTO MEMBER_T (ID ,NAME ,PASSWORD, AUTH ,JOINDATE)

    VALUES(#{id}, #{name}, #{password}, 'ROLE_USER', CURRENT_TIMESTAMP)

</insert>

<!-- 자동 회원가입 처리 시 사용 -->

<insert id="insertAutoUserInfo" parameterType="UserInfoVo">

    INSERT INTO MEMBER_T (ID ,NAME ,PASSWORD, AUTH ,JOINDATE)

    VALUES(#{id}, #{name}, #{password}, #{auth}, CURRENT_TIMESTAMP)

</insert>

</mapper>

```

10-3. /resources/config/Config.xml 생성

```

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-
config.dtd" >

<configuration>

    <typeAliases>

        <typeAlias type="java.lang.String" alias="String" />

        <typeAlias type="java.lang.Integer" alias="Integer" />

        <typeAlias type="com.min.edu.vo.UserInfoVo" alias="UserInfoVo" />

    </typeAliases>

    <mappers>

        <mapper resource="sqls/UserMapper.xml" />

    </mappers>

</configuration>

```

10-4. src/main/java Package 생성

com.min.edu.cmmn

com.min.edu.ctrl

com.min.edu.dao

com.min.edu.security

com.min.edu.service

com.min.edu.vo

10-5. com.min.edu.cmmn

10-5-1. ErrorController.java 파일 생성 (권한접근오류 공통페이지 처리)

```
package com.min.edu.cmmn;

import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import lombok.extern.slf4j.Slf4j;

@Controller
@Slf4j

public class ErrorController {

    @GetMapping("/accessError.do")

    public String accessDenied(Authentication auth, Model model) {

        log.info("권한없음 : {}", auth);

        model.addAttribute("msg","Access Denied");
    }
}
```

```
        return "accessError";
    }

}
```

10-6. com.min.edu.vo

10-6-1. UserInfoVo.java 파일 생성 (회원정보 Vo처리)

```
package com.min.edu.vo;
```

```
import java.util.Date;
```

```
import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;
```

```
@Data
```

```
@Getter
```

```
@Setter
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
public class UserInfoVo {

    private String id;

    private String name;

    private String password;

    private String auth;

    private Date joindate;

}
```

10-7. com.min.edu.dao

10-7-1. IUserInfoDao.java 파일 생성

```
package com.min.edu.dao;
```

```
import java.util.List;
```

```
import com.min.edu.vo.UserInfoVo;
```

```
public interface IUserInfoDao {
```

```
    public UserInfoVo getUserInfo(String id);
```

```
    public int insertUserInfo(UserInfoVo vo);
```

```
    public int insertAutoUserInfo(UserInfoVo vo);
```

```
public List<UserInfoVo> getUserInfoAllList();
```

```
}
```


10-7-2. UserInfoDaoImpl.java 파일 생성

```
package com.min.edu.dao;

import java.util.List;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.min.edu.vo.UserInfoVo;

@Repository

public class UserInfoDaoImpl implements IUserInfoDao {

    @Autowired

    private SqlSessionTemplate sessionTemplate;

    private final String NS = "com.min.edu.dao.UserInfoDaoImpl.";

    @Override

    public UserInfoVo getUserInfo(String id) {

        return sessionTemplate.selectOne(NS+"getUserInfo", id);

    }

}
```

```
@Override
```

```
public int insertUserInfo(UserInfoVo vo) {  
  
    return sessionTemplate.insert(NS+"insertUserInfo", vo);  
  
}
```

```
@Override
```

```
public int insertAutoUserInfo(UserInfoVo vo) {  
  
    return sessionTemplate.insert(NS+"insertAutoUserInfo", vo);  
  
}
```

```
@Override
```

```
public List<UserInfoVo> getUserInfoAllList() {  
  
    return sessionTemplate.selectList(NS+"getUserInfoAllList");  
  
}
```

```
}
```

10-8. com.min.edu.service

10-8-1. luserInfoService.java 파일 생성

```
package com.min.edu.service;
```

```
import java.util.List;
```

```
import com.min.edu.vo.UserInfoVo;
```

```
public interface IUserInfoService {
```

```
    public UserInfoVo getUserInfo(String id);
```

```
    public int insertUserInfo(UserInfoVo vo);
```

```
    public int insertAutoUserInfo(UserInfoVo vo);
```

```
    public List<UserInfoVo> getUserInfoAllList();
```

```
}
```

10-8-2. UserInfoServiceImpl.java 파일 생성

```
package com.min.edu.service;
```

```
import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Service;
```

```
import com.min.edu.dao.IUserInfoDao;
```

```
import com.min.edu.vo.UserInfoVo;
```

```
import lombok.extern.slf4j.Slf4j;
```

```
@Service
```

```
@Slf4j
```

```
public class UserInfoServiceImpl implements IUserInfoService {
```

@Autowired

private IUserInfoDao dao;

@Override

```
public UserInfoVo getUserInfo(String id) {  
  
    log.info("UserInfoServiceImpl getUserInfo");  
  
    return dao.getUserInfo(id);  
}
```

@Override

```
public int insertUserInfo(UserInfoVo vo) {  
  
    return dao.insertUserInfo(vo);  
}
```

@Override

```
public int insertAutoUserInfo(UserInfoVo vo) {  
  
    return dao.insertAutoUserInfo(vo);  
}
```

@Override

```
public List<UserInfoVo> getUserInfoAllList() {  
  
    return dao.getUserInfoAllList();  
}
```

```
}
```

10-9. com.min.edu.ctrl

10-9-1. MemberColtroller.java 파일 생성 (회원가입 및 로그인 처리)

```
package com.min.edu.ctrl;
```

```
import java.io.IOException;
```

```
import java.io.UnsupportedEncodingException;
```

```
import java.util.List;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.security.crypto.password.PasswordEncoder;
```

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import com.min.edu.service.UserInfoServiceImpl;
```

```
import com.min.edu.vo.UserInfoVo;
```

```
import lombok.extern.slf4j.Slf4j;
```

@Controller

@Slf4j

public class MemberColtroller {

 @Autowired

 private UserInfoServiceImpl service;

 @Autowired

 private PasswordEncoder passwordEncoder;

 //회원가입 폼 이동

 @GetMapping("/joinForm.do")

 public String joinForm() {

 log.info("CommonColtroller joinForm.do");

 return "joinForm";

 }

 //회원가입 처리

 @PostMapping("/join.do")

 public String join(UserInfoVo vo) throws UnsupportedEncodingException {

 log.info("CommonColtroller 회원가입 정보 : {}", vo);

 // 비밀번호 BCryptPasswordEncoder 처리

```

        String encodePassword = passwordEncoder.encode(vo.getPassword());

        vo.setPassword(encodePassword);

        int n = service.insertUserInfo(vo);

        log.info("CommonColtroller 회원가입 처리 수 : {}", n);

        return "redirect:./index.jsp";
    }

```

//회원가입 처리(자동생성용)

@GetMapping("/autoJoin.do")

public void autoJoin(HttpServletResponse response) throws IOException {

```

        log.info("CommonColtroller 자동 회원가입 처리");

```

```

        String id = "test";

```

```

        String name = "테스트";

```

```

        String pw = "1111";

```

```

        UserInfoVo vo = new UserInfoVo();

```

```

        int n = 0;

```

```

        for(int i=1;i<=10;i++) {

```

```

            log.info("i : {}", i);

```

```

            // 비밀번호 BCryptPasswordEncoder 처리

```

```

            String encodePassword = passwordEncoder.encode(pw);

```

```

            vo.setId(id+""+i);

```



```

vo.setName(name+""+i);

vo.setPassword(encodePassword);

if(i>=10) {

    vo.setAuth("ROLE_SYSADMIN");

}else if (i>=7) {

    vo.setAuth("ROLE_ADMIN");

}else {

    vo.setAuth("ROLE_USER");

}

StringBuffer sb = new StringBuffer();

sb.append("<script>");

try {

    n += service.insertAutoUserInfo(vo);

    log.info("CommonColtroller 자동 회원가입 처리 수 : {}", n);

    sb.append("alert('자동생성이 완료 되었습니다.');

```

```
}
```

```
@GetMapping("/getUserInfoAllList.do")
```

```
public String getUserInfoAllList(Model model) {
```

```
    log.info("CommonColtroller 멤버 전체 리스트");
```

```
    List<UserInfoVo> lists = service.getUserInfoAllList();
```

```
    log.info("CommonColtroller 멤버 전체 리스트 조회 : {}", lists);
```

```
    model.addAttribute("lists", lists);
```

```
    return "getUserInfoAllList";
```

```
}
```

```
//로그인 폼 이동
```

```
@GetMapping("/loginForm.do")
```

```
public String loginForm(String error, String logout, Model model) {
```

```
    log.info("error : {}", error);
```

```
    log.info("logout : {}", logout);
```

```
    if (error != null) {
```

```
        model.addAttribute("error", "로그인 오류! 계정을 확인하세요.");
```

```
    }
```

```
    if (logout != null) {
```

```
        model.addAttribute("logout", "로그아웃!!");
```

```
}
```

```
return "loginForm";
```

```
}
```

```
}
```

10-9-2. SampleController.java 파일 생성 (로그인 후 권한확인 페이지)

```
package com.min.edu.ctrl;
```

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import lombok.extern.slf4j.Slf4j;
```

```
@Controller
```

```
@Slf4j
```

```
public class SampleController {
```

```
    @GetMapping("/all.do")
```

```
    public String doAll() {
```

```
        log.info("SampleController All 모두접근 가능");
```

```
        return "all";
```

```
    }
```

```
    //회원권한 페이지 접근
```

```
    @GetMapping("/member.do")
```

```
    public String doMember() {
```

```
        log.info("SampleController Member 로그인 회원만 접근 가능");
```

```
        return "member";
```

```
    }
```

```
    //관리자 권한 페이지 접근
```

```
    @GetMapping("/admin.do")
```

```
    public String doAdmin() {
```

```
        log.info("SampleController Admin만 접근 가능");
```

```
        return "admin";
```

```
    }
```

```
    @GetMapping("/adminManager1.do")
```

```
    public String doAdminManager1() {
```

```
        log.info("SampleController Admin만 접근 가능_Manager1");

        return "adminManager1";
    }

    @GetMapping("/adminManager2.do")

    public String doAdminManager2() {

        log.info("SampleController Admin만 접근 가능_Manager2");

        return "adminManager2";
    }

    //시스템관리자 권한 페이지 접근

    @GetMapping("/system.do")

    public String doSystem() {

        log.info("SampleController System Admin만 접근 가능");

        return "system";
    }
}
```

10-10. src/main/webapp/WEB-INF/web.xml 설정

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
```

```
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
```

```
<!-- 첫페이지 설정 -->
```

```
<welcome-file-list>
```

```
    <welcome-file>index.jsp</welcome-file>
```

```
</welcome-file-list>
```

```
<!--
```

```
/WEB-INF/spring/appServlet/security-context.xml : spring security를 위한 xml 설정파일
```

```
-->
```

```
<context-param>
```

```
    <param-name>contextConfigLocation</param-name>
```

```
    <param-value>
```

```
        /WEB-INF/spring/appServlet/application-context.xml
```

```
    <!-- /WEB-INF/spring/appServlet/security-context.xml -->
```

```
    </param-value>
```

```
</context-param>
```

```
<listener>
```

```
    <listener-
```

```
class>org.springframework.web.context.ContextLoaderListener</listener-class>
```


</listener>

<servlet>

<servlet-name>springDispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springDispatcherServlet</servlet-name>

<url-pattern>*.do</url-pattern>

</servlet-mapping>

<!-- Spring으로 전달되는 요청을 Spring Filter에 의해서 Request, Response를 모두 UTF-8로 강제 인코딩 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

```
        <param-name>encoding</param-name>

        <param-value>UTF-8</param-value>

    </init-param>

    <init-param>

        <param-name>forceEncoding</param-name>

        <param-value>true</param-value>

    </init-param>

</filter>

<filter-mapping>

    <filter-name>encodingFilter</filter-name>

    <url-pattern>/*</url-pattern>

</filter-mapping>

<!-- Spring Security 사용을 위해 filter 추가 Start -->

<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>

<!-- Spring Security 사용을 위해 filter 추가 End -->

</web-app>
```

10-11. /WEB-INF/spring/appServlet/application-context.xml 설정

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

      xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="propertyPlaceholderConfigurer"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

        <property name="location" value="classpath:properties/db.properties" />

    </bean>

    <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource">

        <property name="driverClassName" value="${driver}" />

        <property name="jdbcUrl" value="${url}" />

        <property name="username" value="${user}" />

        <property name="password" value="${pwd}" />

    </bean>

    <bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">

        <property name="dataSource" ref="dataSource" />

        <property name="configLocation" value="classpath:config/Config.xml" />
```

```
</bean>
```

```
<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
```

```
    <constructor-arg ref="sqlSessionFactoryBean" />
```

```
</bean>
```

```
</beans>
```

10-12. /WEB-INF/spring/appServlet/servlet-context.xml 설정

```
<?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:context="http://www.springframework.org/schema/context"
```

```
    xmlns:mvc="http://www.springframework.org/schema/mvc"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
```

<http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd>

<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring-beans.xsd>

<http://www.springframework.org/schema/context>

[http://www.springframework.org/schema/context/spring-context-4.3.xsd">](http://www.springframework.org/schema/context/spring-context-4.3.xsd)

```
<context:component-scan base-package="com.min.edu" />
```

```
<mvc:annotation-driven />
```

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
```

```
    <property name="prefix" value="/WEB-INF/views/" />
```

```
    <property name="suffix" value=".jsp" />
```

```
</bean>
```

```
</beans>
```

10-13. /WEB-INF/spring/appServlet/security-context.xml 설정

```
<?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:security="http://www.springframework.org/schema/security"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-5.7.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="com.min.edu" />
```

```

<!-- Spring Security API제공 암호화 -->

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

<!-- security:http : http 요청처리를 위한 보안구성 정의 auto-config="true" : Spring
Security 보안설정을 자동으로 구성 적용처리 -->

<security:http auto-config="true" use-expressions="true">

    <!-- csrf 사용여부(사이트간 위변조 방지를 위한 처리 - true : 미사용) -->

    <security:csrf disabled="false" />

</security:http>

```

<!--

ProviderManager 인터페이스를 가지고 있으며, ID,Password 검증하는 실질적인 클래스

1. ID검증: UserDetailsService에서 DB에서 사용자를 조회. 사용자가 없을 경우
UserNotFoundException 예외를 발생시킨다.

성공하면 UserDetails 객체를 반환한다.

2. Password 검증: 반환 받은 UserDetails 객체에 저장된 Password와
Authentication에 저장된 Password(로그인시 입력한 Password)를
matches 메서드를 이용하여 비교. 일치하지 않는다면 BadCredentialException
예외를 발생시킨다.

3. 추가 검증 까지 완료하면 Authentication(유저 정보, 권한 정보)를
AuthenticationManager에게 전달.

AuthenticationManager는 Filter에게 전달하고, Filter는 이 정보를 전역적으로 사용할 수 있게 SecurityContext에 전달한다.

-->

<security:authentication-manager>

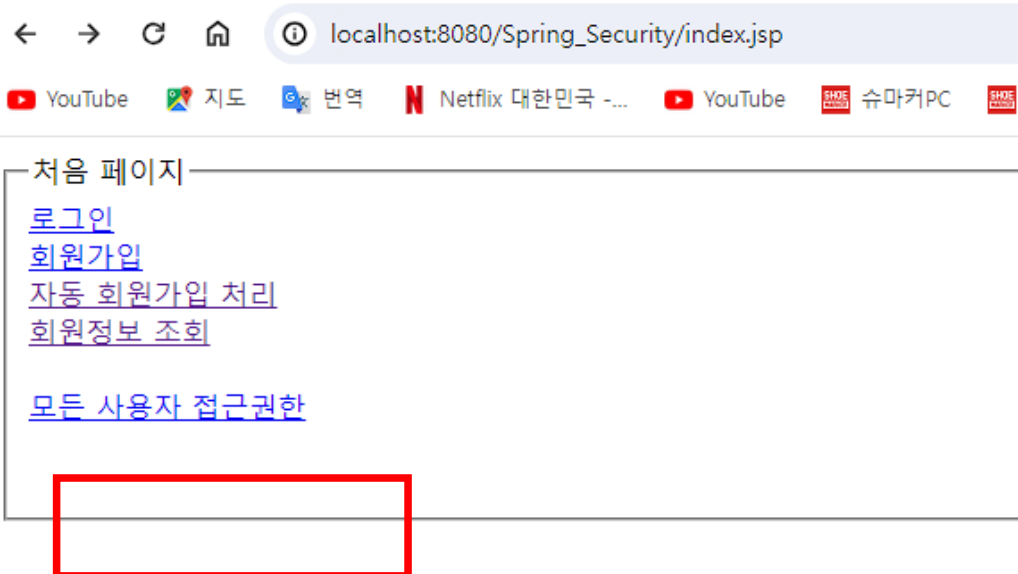
</security:authentication-manager>

</beans>

10-14. http://localhost:8080/Spring_Security/index.jsp 접속

10-15. 자동 회원가입 처리

 각 권한별 사용자 자동으로 생성처리 후 회원정보 페이지를 통한 회원가입 확인



📄 회원정보 조회를 통한 확인

회원리스트 조회

[메인화면](#)

아이디	성명	권한	가입일
test3	테스트3	ROLE_USER	2024-02-19
test4	테스트4	ROLE_USER	2024-02-19
test5	테스트5	ROLE_USER	2024-02-19
test6	테스트6	ROLE_USER	2024-02-19
test2	테스트2	ROLE_USER	2024-02-19
test8	테스트8	ROLE_ADMIN	2024-02-19
test9	테스트9	ROLE_ADMIN	2024-02-19
test10	테스트10	ROLE_SYSADMIN	2024-02-19
test1	테스트1	ROLE_USER	2024-02-19
test7	테스트7	ROLE_ADMIN	2024-02-19

11. 로그인 설정을 위한 Package(com.min.edu.security)

11-1. CustomAccessDeniedHandler.java(권한 제한 페이지 접근시 로직처리 구현)

```
package com.min.edu.security;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.web.access.AccessDeniedHandler;

import lombok.extern.slf4j.Slf4j;

@Slf4j

public class CustomAccessDeniedHandler implements AccessDeniedHandler {

    @Override
    public void handle(HttpServletRequest request, HttpServletResponse response,
        AccessDeniedException accessDeniedException) throws IOException, ServletException {

        log.info("Access Denied Handler");

        log.info("Redirect...");
```

```
        response.sendRedirect("./accessError.do");
    }

}
```

11-2. CustomLoginSuccessHandler.java(로그인 성공 시 로직처리 구현)

```
package com.min.edu.security;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import org.springframework.security.core.Authentication;
```

```
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import lombok.extern.slf4j.Slf4j;
```

```
@Slf4j
```

```
public class CustomLoginSuccessHandler implements AuthenticationSuccessHandler {
```

```
    @Override
```

```
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse  
response,
```

```
Authentication authentication) throws IOException, ServletException {
```

```
    log.info("login Success");
```

```
    List<String> roleNames = new ArrayList<String>();
```

```
    /*
```

```
    Spring Security의 Authentication 객체에서 사용자의 권한(authority) 정보를  
추출하여 roleNames 리스트에 추가
```

```
    authentication.getAuthorities()는 현재 사용자의 권한을 나타내는 컬렉션을 반환
```

```
ex) // ["ROLE_USER"],["ROLE_ADMIN"]
```

```
*/

authentication.getAuthorities().forEach(authority -> {

    roleNames.add(authority.getAuthority());

});

log.info("ROLE NAME : {}", roleNames);

// 시스템 관리자 로그인 시

if(roleNames.contains("SYSADMIN")) {

    response.sendRedirect("./system.do");

    return;

}

// 관리자 로그인 시

if(roleNames.contains("ADMIN")) {

    response.sendRedirect("./admin.do");

    return;

}

// 사용자 로그인 시

if(roleNames.contains("USER")) {

    response.sendRedirect("./member.do");

    return;

}
```

```
        response.sendRedirect("./index.jsp");  
    }  
  
}
```

11-3. LoginService.java(로그인 처리 서비스 구현)

```
package com.min.edu.security;
```

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.authority.AuthorityUtils;

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 com.min.edu.dao.IUserInfoDao;

import com.min.edu.vo.UserInfoVo;
```

```
import lombok.extern.slf4j.Slf4j;
```

```
/*
```

UserDetailsService를 구현한 LoginService 클래스.

사용자가 입력한 사용자 아이디를 기반으로 사용자 정보를 데이터베이스에서 검색하고, 해당 사용자의 권한 정보를 로드하여 UserDetails 객체로 반환처리.

```
*/
```

```
@Slf4j
```

```
public class LoginService implements UserDetailsService {
```

```
    @Autowired
```

```
    private IUserInfoDao dao;
```

```
    /*
```

loadUserByUsername(String userId)은 UserDetailsService 인터페이스의 추상메서드로

사용자의 아이디를 입력받아 사용자의 상세정보를 로드

```
    */

    public UserDetails loadUserByUsername(String userId) throws
UsernameNotFoundException {

        log.info("LoginService loadUserByUsername : {}", userId);

        log.info("LoginService repository : {}", dao);

        UserInfoVo userInfoVo = dao.getUserInfo(userId);    //userId로 상세정보 조회

        log.info("LoginService userInfoVo : {}", userInfoVo);

        if(userInfoVo != null) {    //정보가 있다면...

            /*

            ROLE 정보가 여러개 있다면 Collection 객체로 전달한다.

            Collection<SimpleGrantedAuthority> roles = new

ArrayList<SimpleGrantedAuthority>();

            roles.add(new SimpleGrantedAuthority(userInfoVo.getAuth()));

            return new User(userId, userInfoVo.getPassword(), roles);

            */

            return new User(userId, userInfoVo.getPassword(),

AuthorityUtils.createAuthorityList(userInfoVo.getAuth()));

        }else {

            return null;

        }

    }

}
```


11-4. xml 수정(/WEB-INF/spring/appServlet/security-context.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:security="http://www.springframework.org/schema/security"

        xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

        xsi:schemaLocation="http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-5.7.xsd

        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd

        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd

        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">


        <!-- loginService 사용자 정보 DAO 처리 시 필요함 -->

        <context:component-scan base-package="com.min.edu" />


        <!-- 접근권한 미만 처리로직 Handler -->

        <bean id="customAccessDenied"
```

```

class="com.min.edu.security.CustomAccessDeniedHandler"/>

<!-- 로그인 성공 시 처리로직 Handler -->

<bean id="customLoginSuccess"

class="com.min.edu.security.CustomLoginSuccessHandler" />

<!-- Spring Security API제공 암호화 -->

<bean id="bcryptPasswordEncoder"

class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />

<!-- 로그인 처리 서비스 -->

<bean id="loginService" class="com.min.edu.security.LoginService" />

<!--

security:http : http 요청처리를 위한 보안구성 정의

auto-config="true" : Spring Security 보안설정을 자동으로 구성 적용처리

-->

<security:http auto-config="true" use-expressions="true">

    <!-- 접근제한 설정 : URL 단위로 매핑 -->

    <!-- 모든 사용자 접근 권한 -->

    <security:intercept-url pattern="/all.do" access="permitAll" />

    <!-- 일반회원 접근 권한 -->

    <security:intercept-url pattern="/member.do"

access="hasAnyRole('USER','ADMIN','SYSADMIN')" />

    <!-- 관리자 접근 권한 -->

    <security:intercept-url pattern="/admin.do"

access="hasAnyRole('ADMIN','SYSADMIN')" />

    <!-- 파일 특정명칭으로 관리자 권한부여 가능 -->

```


<!-- csrf 사용여부(사이트간 위변조 방지를 위한 처리 - true : 미사용) -->

<security:csrf disabled="false" />

<!--

key : token 생성용 키값

token-validity-seconds : token의 유효 시간(10일)

remember-me-parameter : 로그인 화면에서 전달되는 remember-me 파라미터

-->

<security:remember-me key="HCM"

token-validity-seconds="864000"

remember-me-

parameter="remember-me"/>

</security:http>

<security:authentication-manager>

<!-- loginService 인증 프로세스로 위임 -->

<security:authentication-provider user-service-ref="loginService">

<!-- Spring Security 암호화 bcryptPasswordEncoder 참조 -->

<security:password-encoder ref="bcryptPasswordEncoder" />

</security:authentication-provider>

</security:authentication-manager>

</beans>

12. 관리자 권한으로 로그인 시(test7)

☞ 메뉴에서 모든접근페이지 및 회원, 관리자 권한 페이지 메뉴만 노출된다.

principal : org.springframework.security.core.userdetails.User [Username

principal.username : test7

principal.authorities : [ROLE_ADMIN]

처음 페이지

[모든 사용자 접근권한](#)

[회원 권한](#)

[관리자 권한](#)

[관리자 권한1](#)

[관리자 권한2](#)

로그아웃

-http://localhost:8080/Spring_Security/system.do 페이지로 강제 접근 시 권한오류 페이지로

이동한다.(com.min.edu.security.CustomAccessDeniedHandler 자동 이동처리)

권한없음

Access Denied

[메인화면](#)

13. 로그아웃 처리 설정 (/WEB-INF/spring/appServlet/security-context.xml)

```
<security:http auto-config="true" use-expressions="true">
```

.....

```
<!--
```

logout 처리

logout-url : 로그아웃 처리 URL

logout-success-url : 로그아웃 후 이동 Url

invalidate-session: 로그아웃 시 session을 무효화할 지 선택

```
-->
```

```
<security:logout logout-url="/logout.do"
```

```
logout-success-url="/index.jsp"
```

```
invalidate-session="true" />
```

.....

```
</security:http>
```

- 로그아웃 처리 시 자동 로그인 기능이 설정되어 있다면 해당 쿠키정보까지 자동으로 모두 삭제된다.

14. 자동로그인 처리(remember-me)

14-1. xml 추가(/WEB-INF/spring/appServlet/security-context.xml)

```
<security:http auto-config="true" use-expressions="true">
```

.....

```
<!--
```

```
key : token 생성용 키값
```

```
token-validity-seconds : token의 유효 시간(10일)
```

```
remember-me-parameter : 로그인 화면에서 전달되는 remember-me 파라미터
```

```
-->
```

```
<security:remember-me key="HCM"
```

```
token-validity-seconds="864000"
```

```
remember-me-parameter="remember-me"/>
```

.....

```
</security:http>
```

14-2. html 추가(/webapp/WEB-INF/views/loginForm.jsp)

.....

```
<form method="post" action="/login">
```

```
<div>
```

```
아이디 : <input type="text" name="username"
```

```
required="required">
```

```
</div>
```

```
<div>
```

```
비밀번호 : <input type="password" name="password"
```

```
required="required">
```

```
</div>
```

```
<div>
```

```
<!-- 로그인 유지를 위한 처리, 로그아웃 시 유지는 삭제됨 -->
```

```
<label for="remember">로그인 유지</label>
```

```
<input type="checkbox" id="remember" name="remember-
```

```
me">
```

```
</div>
```

```
<div>
```

```
<input type="submit" value="로그인">
```

```
<input type="button" value="메인화면"
```

```
onclick="location.href='./index.jsp';">
```

```
.....
```