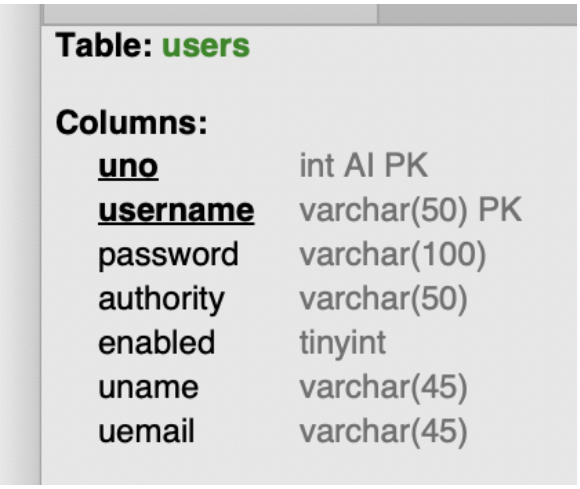
Spring Security 1

Table



pom.xml

```
<!-- <https://mvnrepository.com/artifact/org.springframework.security/spring-security-
web> -->
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-web</artifactId>
      <version>5.4.6</version>
    </dependency>
    <!-- <https://mvnrepository.com/artifact/org.springframework.security/spring-secur
ity-config> -->
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-config</artifactId>
      <version>5.4.6</version>
    </dependency>
    <!-- <https://mvnrepository.com/artifact/org.springframework.security/spring-secur
ity-taglibs> -->
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-taglibs</artifactId>
      <version>5.4.6</version>
    </dependency>
```

Security-context.xml

로그인시 입력한 아이디username 암호 password 의 일치 여부를 따로 클래스로 만들어 구현하는 방법도 있으나 아래와 같이 비교 인증 절차도 스프링 시큐리티에 일임하는 것이 편하다.

```
<?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>"
  xsi:schemaLocation="<http://www.springframework.org/schema/security> <http://www.spr
ingframework.org/schema/security/spring-security.xsd>
    <http://www.springframework.org/schema/beans> <http://www.springframework.org/sche
ma/beans/spring-beans.xsd>">


  <!--  각각의  interceptr-url, form-login,logout 은  내부적으로 Filter를 만들어 사용한다.  그
래서 web.xml에서 이 모든걸 엮어줄 FilterChaing을 따로 설정해준다. -->
  <!-- web.xml에서 사용하는 FilterChain의 대한 설정부분이다. -->
  <security:http use-expressions="true">
    <security:intercept-url pattern="/cars/add/**" access="hasAuthority('USER_MANAGE
R')" />
    <security:intercept-url pattern="/**" access="permitAll" />

             <security:form-login login-page="/login"
                    default-target-url="/cars"
                    authentication-failure-url="/loginfailed"
                    username-parameter="username"
                    password-parameter="password"/>
        <security:csrf />
        <security:logout logout-success-url="/logout"/>

        </security:http>
  <!-- form-login은 기본 로그인 폼 양식을 보여준다.logout은 로그아웃처리를.. -->


  <!-- 암호화를 위한 passwordEncoder -->
<bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCr
yptPasswordEncoder"></bean>


<!-- DB연동은 data-source만 지정해주면 된다, 테이블이름은 정확히.   users 랑 authorities -->
  <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider>
    <security:password-encoder hash="bcrypt"/>
     <security:jdbc-user-service data-source-ref="dataSource"
     users-by-username-query="SELECT username, password, enabled FROM users WHERE user
name=?"
     authorities-by-username-query="SELECT username, authority FROM users WHERE userna
me=?"
     />
    </security:authentication-provider>
  </security:authentication-manager>

</beans>
```

User.java

```
package com.carshop.users;

public class User {

  private int uno, enabled;

  private String username, password, authority, uname, uemail;

  public int getUno() {
    return uno;
  }

  public void setUno(int uno) {
    this.uno = uno;
  }

  public int getEnabled() {
    return enabled;
  }

  public void setEnabled(int enabled) {
    this.enabled = enabled;
  }

  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;
  }

  public String getAuthority() {
    return authority;
  }

  public void setAuthority(String authority) {
    this.authority = authority;
  }

  public String getUname() {
    return uname;
  }

  public void setUname(String uname) {
    this.uname = uname;
  }
```

```
  public String getUemail() {
    return uemail;
  }

  public void setUemail(String uemail) {
    this.uemail = uemail;
  }

  public User() {

  }

  public User(int uno, int enabled, String username, String password, String authorit
y, String uname, String uemail) {

    this.uno = uno;
    this.enabled = enabled;
    this.username = username;
    this.password = password;
    this.authority = authority;
    this.uname = uname;
    this.uemail = uemail;
  }




}
```

C R U D

DB/xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE  mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "<http://mybatis.org/dt
d/mybatis-3-mapper.dtd>">
<mapper namespace="users">

  <insert id="insert"
    parameterType="com.carshop.users.User" useGeneratedKeys="true"
    keyProperty="username">
    <![CDATA[
      INSERT INTO users
      (username, password, authority, enabled, uname, uemail)
      VALUES
      (#{username}, #{password}, "USER", 1, #{uname}, #{uemail})
    ]]>

  </insert>

  </mapper>
```

Repository

```
package com.carshop.users;

public interface UserRepository {

  void setNewUser(User user);
}

package com.carshop.users;

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

@Repository
public class UserRepositoryImpl implements UserRepository {

  @Autowired
  SqlSessionTemplate sqlSessionTemplate;

  @Override
  public void setNewUser(User user) {
    this.sqlSessionTemplate.insert("users.insert", user);

  }

}
```

Service

```
package com.carshop.users;

import org.springframework.stereotype.Service;

public interface UserService {

  void setNewUser(User user);

}

package com.carshop.users;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl implements UserService {

  @Autowired
  UserRepository userRepository;
```

```
  @Override
  public void setNewUser(User user) {

    userRepository.setNewUser(user);

  }

}
```

Controller

```
package com.carshop.users;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@RequestMapping("users")
@Controller
public class UsersController {

  @Autowired
  UserService userService;

  @GetMapping("/join")
  public String joinForm(@ModelAttribute("NewUser") User user) {
    return "users/joinform";
  }

  @PostMapping("/join")
  public String submitForm(@ModelAttribute("NewUser") com.carshop.users.User user) {
    userService.setNewUser(user);
    return "redirect:/login";
  }

}
```

View/jsp

joinform.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c"%>
<%@ taglib uri="<http://www.springframework.org/tags/form>" prefix="form"%>
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="UTF-8">
<title>회원 등록</title>
</head>
<body>


  <form:form modelAttribute="NewUser"
         action="./join?${_csrf.parameterName}=${_csrf.token}"
             class="form-horizontal"
             method = "post">
  <fieldset>
  id : <form:input path="username" class="form-control"/>
  pw : <form:input path="password" type="password" class="form-control"/>
  name : <form:input path="uname" class="form-control"/>
  mail : <form:input path="uemail" class="form-control"/>
  <input type="submit" class="btn btn-primary" value="등록"/>


  </fieldset>
  </form:form>

</body>
</html>
```

Controller 에 암호화 처리 부분

```
@Autowired
  BCryptPasswordEncoder bcryptPasswordEncoder;

@PostMapping("/join")
  public String submitForm(@ModelAttribute("NewUser") User user) {

    //스프링은 반드시 password를 암호화 하여 저장해야만 로그인을 할수 있는게 기본이다.

    String encodedPassword = bcryptPasswordEncoder.encode(user.getPassword());
    user.setPassword(encodedPassword);
    userService.setNewUser(user);
    return "redirect:/login";
  }
```

C R U D

DB/xml

```
<select id="select_list"
  resultType="com.carshop.users.User">
  <![CDATA[
    SELECT * FROM users ORDER BY uno DESC
```

```
  ]]>
</select>
```

Repository

```
List<User> getAllUserList();
-------------------------------------------------------------------------------
@Override
public List<User> getAllUserList() {
  return this.sqlSessionTemplate.selectList("users.select_list");
}
```

Service

```
List<User> getAllUserList();
-------------------------------------------------------------------------------
@Override
public List<User> getAllUserList() {
  return this.userRepository.getAllUserList();
}
```

Controller

```
@GetMapping("/list")
public String manageProduct(Model model) {
    List<User> list = userService.getAllUserList();
    model.addAttribute("userList", list);

    return "users/list";
}
```

View/jsp

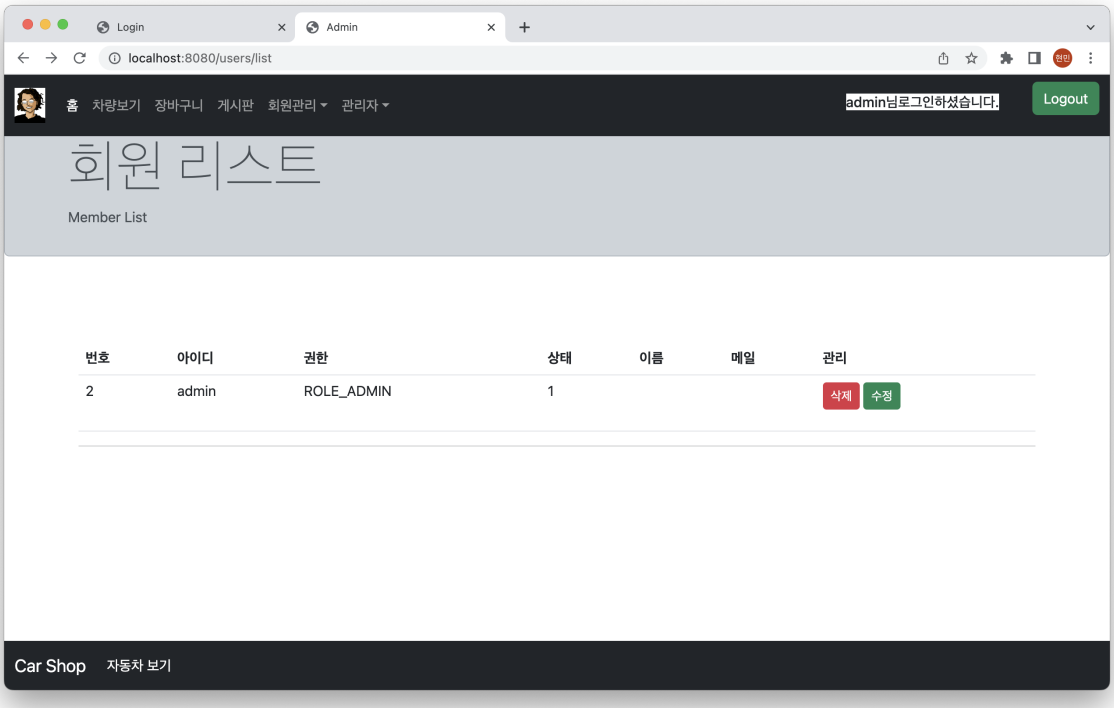
list.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<html>
<head>
<title>User List</title>
<script src="https://code.jquery.com/jquery-3.6.3.min.js"
  integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU="
  crossorigin="anonymous"></script>
<script>
```

```
    function removeCar(cid) {
        $.ajax({
            type:"POST",
            url:"/users/remove",
            data:{cid: cid },
            beforeSend : function(xhr)
            {   /*데이터를 전송하기 전에 헤더에 csrf값을 설정한다*/
                xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
            },
            success: function(result) {
                alert("회원이 삭제되었습니다.")
            },
            error:function (request, status, error) {
                alert(request.status + " " + request.responseText);
            }
        })

        window.location.reload();
    }
</script>
</head>
<body>
  <div class="container">
    <div class="container">
      <div style="padding-top: 50px">
        <table class="table table-hover">
          <tr>
            <th>번호</th>
            <th>아이디</th>
            <th>권한</th>
            <th>상태</th>
            <th>이름</th>
            <th>이메일</th>
          </tr>
          <form:form name="removeForm" method="put">
            <c:forEach items="${userList}" var="user">
              <tr>
                <td>${user.uno}</td>
                <td>${user.username}</td>
                <td>${user.authority}</td>
                <td>${user.enabled}</td>
                <td>${user.uname}</td>
                <td>${user.uemail}</td>
                <td>
                  <p>
                    <a href="javascript:removeUser('${user.username}')"
                      class="btn btn-danger btn-sm">삭제</a> <a
                      href="<c:url value="/users/update?id=${user.username}"/>"
                      class="btn btn-success btn-sm">수정</a>
                </td>
              </tr>
            </c:forEach>
          </form:form>
        </table>
      </div>
      <hr>
    </div>
  </div>
```

```
</body>
</html>
```



C R U D

DB/xml

```
<update id="updateAuth"
  parameterType="hashMap">
  <![CDATA[
    UPDATE users
    SET authority = #{authority}
    WHERE username = #{username}
  ]]>
</update>
```

Repository

```
void updateAuth(Map<String, Object> auth);
-----------------------------------------------------------------------------
@Override
public void updateAuth(Map<String, Object> auth) {
```

```
  this.sqlSessionTemplate.update("users.updateAuth", auth);
}
```

Service

```
void updateAuth(Map<String, Object> auth);
-----------------------------------------------------------------------------
@Override
public void updateAuth(Map<String, Object> auth) {
  this.userRepository.updateAuth(auth);
}
```

Controller

```
@PostMapping("/list")
public void updateAuth(@RequestParam Map<String, Object> auth) {// 여러개 바뀜
    userService.updateAuth(auth);
}
```

View/jsp

```
<td><select
  onchange="updateAuth('${user.username }', this)"
  class="form-select form-select-sm"
  aria-label=".form-select-sm example">
    <option selected>${user.authority}</option>
    <option value="ROLE_USER">ROLE_USER</option>
    <option value="ROLE_MANAGER">ROLE_MANAGER</option>
    <option value="ROLE_ADMIN">ROLE_ADMIN</option>
</select></td>
-----------------------------------------------------------------------------
<script>
function updateAuth(username, e) {
  $.ajax({
    type : "POST",
    url : "/users/list",
    data : {
      username : username,
      authority : e.value
    },
    beforeSend : function(xhr) { /*데이터를 전송하기 전에 헤더에 csrf값을 설정한다*/
      xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
    },
    success : function(result) {
      alert("고객 정보가 수정되었습니다.")
    },
    error : function(request, status, error) {
      alert(request.status + " " + request.responseText);
```

```
    }
  })

  window.location.reload();
}
</script>
```

C R U D

DB/xml

```
<delete id="delete"
  parameterType="String">
  <![CDATA[
    DELETE FROM users
    WHERE username = #{username}
  ]]>
</delete>
```

Repository

```
void removeUser(String username);
-----------------------------------------------------------------------------
@Override
public void removeUser(String username) {
  this.sqlSessionTemplate.selectList("users.delete", username);
}
```

Service

```
void removeUser(String username);
-----------------------------------------------------------------------------
@Override
public void removeUser(String username) {
  this.userRepository.removeUser(username);
}
```

Controller

```
@ResponseBody
@RequestMapping("/remove")
public void removeUser(@RequestParam("username") String username) {
```

```
  userService.removeUser(username);
}
```

View/jsp

list.jsp

```
<script>
  function removeUser(username) {
      $.ajax({
          type:"POST",
          url:"/users/remove",
          data:{username: username },
          beforeSend : function(xhr)
          {   /*데이터를 전송하기 전에 헤더에 csrf값을 설정한다*/
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
          success: function(result) {
              alert("고객 정보가 삭제되었습니다.")
          },
          error:function (request, status, error) {
              alert(request.status + " " + request.responseText);
          }
      })

      window.location.reload();
  }
</script>
```