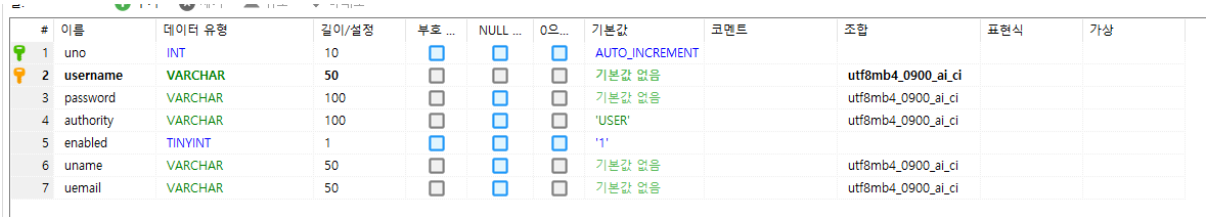
230320 Spring Security

Spring Security

Setting



#	이름	데이터 유형	길이/설정	부호 ...	NULL ...	0으...	기본값	코멘트	조합	표현식	가상
1	uno	INT	10	☐	☐	☐	AUTO_INCREMENT				
2	username	VARCHAR	50	☐	☐	☐	기본값 없음		utf8mb4_0900_ai_ci		
3	password	VARCHAR	100	☐	☐	☐	기본값 없음		utf8mb4_0900_ai_ci		
4	authority	VARCHAR	100	☐	☐	☐	'USER'		utf8mb4_0900_ai_ci		
5	enabled	TINYINT	1	☐	☐	☐	'1'				
6	uname	VARCHAR	50	☐	☐	☐	기본값 없음		utf8mb4_0900_ai_ci		
7	uemail	VARCHAR	50	☐	☐	☐	기본값 없음		utf8mb4_0900_ai_ci		

- 테이블에 하나로 구성하는 것을 추천하며 반드시 username, password, authority, enabled를 같이 넣어준다.
- username과 uname에 차이점은 스프링 시큐리티를 사용하기위해 username을 사용하고
  uname인 진짜 user에 name이다.

Security-context.xml

▼ 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"
  xsi:schemaLocation="http://www.springframework.org/schema/security http://www.sp
ringframework.org/schema/security/spring-security.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/sch
ema/beans/spring-beans.xsd">



  <!--  각각의  interceptr-url, form-login,logout 은  내부적으로 Filter를 만들어 사용한다.
그래서 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_MANA
GER')" />
```

```
    <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.bcryp
t.BCryptPasswordEncoder"></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
 username=?"
     authorities-by-username-query="SELECT username, authority FROM users WHERE us
ername=?"
     />
    </security:authentication-provider>
  </security:authentication-manager>


</beans>
```

DTO

▼ User

```
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(int uno, int enabled, String username, String password, String autho
rity, String uname, String uemail) {
    super();
    this.uno = uno;
    this.enabled = enabled;
    this.username = username;
    this.password = password;
    this.authority = authority;
    this.uname = uname;
    this.uemail = uemail;
  }

  public User() { }




}
```

암호화

```
    <!-- 암호화를 위한 passwordEncoder -->
<bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"></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 username=?"
         authorities-by-username-query="SELECT username, authority FROM users WHERE username=?"
         />
        </security:authentication-provider>
    </security:authentication-manager>
```

C R U D

▼ Create

DB/xml

```
<?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="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

```
<%@ 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>
```

C R U D

Read

DB/xml

```
<select id="select_list" resultType="com.carshop.users.User" >

  <![CDATA[
    SELECT * FROM users ORDER BY uno DESC

  ]]>

</select>
```

Repository

```
package com.carshop.users;

import java.util.List;

public interface UserRepository {

  void setNewUser(User user);

  List<User> getAllUserList();
}
```

```
package com.carshop.users;

import java.util.List;

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);

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

  }
}
```

Service

```
package com.carshop.users;

import java.util.List;

import org.springframework.stereotype.Service;

public interface UserService {

  void setNewUser(User user);

  List<User> getAllUserList();
}
```

```
package com.carshop.users;

import java.util.List;

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);
  }

  public List<User> getAllUserList() {
    return userRepository.getAllUserList();
  }
}
```

Controller

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

      return "users/list";
  }
```

View/jsp

```

```

```
<li class="nav-item dropdown"><a
    class="nav-link dropdown-toggle" href="#" role="button"
    data-bs-toggle="dropdown" aria-expanded="false"> 관리자</a>

    <ul class="dropdown-menu">


        <li><a class="dropdown-item" href="/cars/add">제품등록</a></li>
        <li><a class="dropdown-item" href="/cars/product">제품관리</a></li>

        <li><a class="dropdown-item" href="/users/list">회원관리</a></li>


    </ul>
```

```
<definition name="users/list" extends="base-Template">
    <put-attribute name="title" value="Members" />
    <put-attribute name="heading" value="회원 관리" />
    <put-attribute name="subheading" value="Members" />
    <put-attribute name="content"
        value="/WEB-INF/views/users/list.jsp" />
</definition>
```

C R U D

DB/xml

Repository

Service

Controller

View/jsp

C R U D

▼ Delete

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.delete("users.delete", username);

  }
```

Service

```
void removeUser(String username);




public void removeUser(String username) {
    userRepository.removeUser(username);
  }
```

Controller

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

  }
```

View/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>Car Detail</title>

<script src="https://code.jquery.com/jquery-3.6.3.min.js"
  integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU="
  crossorigin="anonymous"></script>

<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>
</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>메일</th>
            <th>관리</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>
```

권한 변경 ajax

view

```
<td>
  <select onchange="updateAuth('${user.username}', this)" class="form-select form-sele
ct-sm" aria-label=".form-select-sm example">
      <option selected>${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>
```

```
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("권한 정보 변경이 완료되었습니다.")
      },
      error : function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }
    })

    window.location.reload();
  }
```

controller

```
@RequestMapping(value = "/list", method = RequestMethod.POST)
  public void updateAuth(@RequestParam Map<String, Object> auth) {
    //System.out.println(auth);

      this.userService.updateAuth(auth);


  }
```

service

```
void updateAuth(Map<String, Object> auth);




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

Repository

```
void updateAuth(Map<String, Object> auth)
```

```
public void updateAuth(Map<String, Object> auth) {
    this.sqlSessionTemplate.update("users.updateAuth", auth);

  }
```

xml

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

ajax시 success에서 window.location.assign 을 넣어주면 원할하게 작동한다 .!

```
success: function(result) {
    alert("권한 변경이 완료되었습니다.");
    window.location.assign('/users/list');
},
```