Spring Day 15

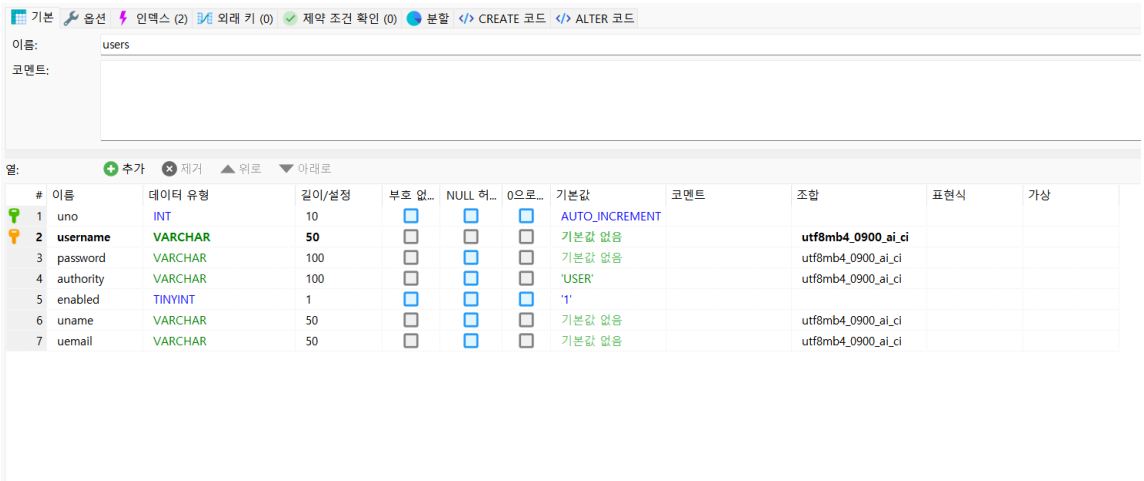
▼ Spring

Spring Security

▼ 환경설정 업데이트

> 💡 **반드시 username, password, authority, enabled 필드가 있어야 한다.**

통합 테이블 설정(통합 & 비통합은 선택사항이다.)



#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	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		

```
CREATE TABLE `users` (
  `uno` INT(10) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `password` VARCHAR(100) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `authority` VARCHAR(100) NOT NULL DEFAULT 'USER' COLLATE 'utf8mb4_0900_ai_ci',
  `enabled` TINYINT(1) NOT NULL DEFAULT '1',
  `uname` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `uemail` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  PRIMARY KEY (`username`) USING BTREE,
  INDEX `uno` (`uno`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
AUTO_INCREMENT=4
;
```

security-context.xml 설정

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

```
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:security="http://www.springframework.org/schema/security"
  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
  http://www.springframework.org/schema/security
  http://www.springframework.org/schema/security/spring-security.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">
<!--  <http auto-config="true" use-expressions="true"> 기본 로그인 창 사용시 -->
  <security:intercept-url pattern="/cars/add/**" access="hasAuthority('ROLE_ADMIN')" /> <!-- ADMIN만 허용 -->
  <security:intercept-url pattern="/manager" access="hasRole('ROLE_MANAGER')" /> <!-- MANAGER만 허용 -->
  <security:intercept-url pattern="/member" access="isAuthenticated()" /> <!-- 로그인한 사람만 허용 -->
  <security:intercept-url pattern="/**" access="permitAll" /> <!-- 전체 허용 -->

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

  <security:csrf/>

  <security:logout logout-success-url="/cars/logout"/>

  </security:http>

<!--  <authentication-manager> -->
<!--    <authentication-provider> -->
<!--    <user-service> -->
<!--  <user name="admin" password="{noop}admin" authorities="ROLE_ADMIN, ROLE_USER" /> -->
<!--  <user name="manager" password="{noop}manager" authorities="ROLE_MANAGER, ROLE_USER" /> -->
<!--  <user name="guest" password="{noop}guest" authorities="ROLE_USER" /> -->
<!--    </user-service> -->
<!--    </authentication-provider> -->
<!--  </authentication-manager> -->

  <!-- 암호화를 위한 passwordEncoder -->
<bean id="bcryptpasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"></bean>

  <!-- DB연동은 data-source만 지정해주면 된다 -->
<!--  로그인시 입력한 아이디(username), 암호(password)의 일치 여부를 따로 클래스로 만들어 구현하는 방법도 있으나 -->
<!--  아래와 같이 비교 인증 절차를 Spring Security 에 일임하는 것이 편하다. -->
  <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:password-encoder ref="bcryptpasswordEncoder"></security:password-encoder>
    </security:authentication-provider>
  </security:authentication-manager>



  <!-- 이것만 있으면  JDBC 코드 (Connection, Statement,ResultSet)로   DB연결 가능 -->
<!--  이미 root-context.xml에서 연결시켜놓았기 때문에 주석처리해도 된다. -->
<!--  <bean id="dataSource" -->
<!--    class="org.apache.commons.dbcp2.BasicDataSource" -->
<!--    destroy-method="close"> -->
<!--    <property name="driverClassName" -->
<!--      value="com.mysql.cj.jdbc.Driver" />  -->
<!--    <property name="url" value="jdbc:mysql://localhost:3306/difbfl4751?characterEncoding=utf8" /> -->
<!--    <property name="username" value="difbfl4751" /> -->
<!--    <property name="password" value="kdy3529216" /> -->
<!--    <property name="defaultAutoCommit" value="true" /> -->
<!--  </bean> -->

</beans>
```

User.java(DTO)

```
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 authority, 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;
  }



}
```

▼ CRUD

user_SQL.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="user">

<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>
```

UserRepository.java

```
package com.carshop.users;

public interface UserRepository {

  void setNewUser (User user);

}
```

UserRepositoryImpl.java

```
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("user.insert", user);
  }

}
```

UserService.java

```
package com.carshop.users;

public interface UserService {

  void setNewUser(User user);

}
```

UserServiceImpl.java

```
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);

  }

}
```

UsersController.java

```
package com.carshop.users;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
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;

  @Autowired
  BCryptPasswordEncoder bCryptPasswordEncoder;

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

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

    //Spring은 기본적으로 password를 암호화하여 DB에 등록해야만 로그인을 할 수 있도록 설정되어 있다.

    String encodedPassword = bCryptPasswordEncoder.encode(user.getPassword());
    user.setPassword(encodedPassword);
    userService.setNewUser(user);
    return "login";
  }

}
```

joinform.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<head>
  <title>자동차 등록</title>
  <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
      <meta charset="utf-8">

</head>
<body class="text-center">
<div class="alert alert-dark" role="alert">
<div class="container"><h1>차량 등록</h1>

      <form:form modelAttribute="NewUser" action="./join?${_csrf.parameterName}=${_csrf.token}" class="form-horizontal" enctype
        <fieldset>
        <legend>
        ${addTitle }
        </legend>
          ID : <form:input path="username" class="form-control" />
          password : <form:input path="password" type="password" class="form-control" />
```

```
          name : <form:input path="uname" class="form-control" />
          email : <form:input path="uemail" class="form-control" />

        <input type="submit" class="btn btn-primary" value="등록">


        </fieldset>
      </form:form>




    </div>
</div>




</body>
</html>
```

▼ C R U D

user_SQL.xml

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

UserRepository.java

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

UserRepositoryImpl.java

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

UserService.java

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

UserServiceImpl.java

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

UsersController.java

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

list.jsp

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


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>list</title>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU=" cross



</head>
<body>



  <div class="my-5">
    <div class="alert alert-dark">
      <div class="container">
        <h1>회원목록</h1>
      </div>
    </div>
  </div>

  <div class="container">
    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
    </div>
    <div style="padding-top: 50px">
      <table class="table table-hover">
        <tr>
          <th>No.</th>
          <th>아이디</th>
          <th>권한</th>
          <th>상태</th>
          <th>이메일</th>
          <th>이름</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.uemail}</td>
              <td>${user.uname}</td>
              <td>
              <p><a href="javascript:removeUser('${user.username}')"
                class="btn btn-danger btn-sm">삭제</a>
              <a href="/users/update?uno=${user.uno}"
                class="btn btn-primary btn-sm">수정</a></td>
            </tr>
          </c:forEach>
        </form:form>
        <tr>
          <th></th>
          <th></th>
          <th></th>
        </tr>
      </table>
```

```
      <a href="<c:url value="/cars" />" class="btn btn-primary">
        &laquo; 쇼핑 계속하기</a>
    </div>
    <hr>

  </div>
</body>


<script>


  function removeUser(user) {
    $.ajax({
      type:"POST",
      url:"/users/removeUser",
      data:{username: user},
      beforeSend : function(xhr)
          {
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
      success:function(result) {
        window.location.reload();
      },
      error:function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }

    })

  }

  </script>
</html>
```

▼ C R U D

user_SQL.xml

```
<update id="update_Auth" parameterType="HashMap">

UPDATE users SET authority=#{authority} WHERE username = #{username}

</update>
```

UserRepository.java

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

UserRepositoryImpl.java

```
@Override
  public void updateAuth(Map<String, Object> auth) {
    this.sqlSessionTemplate.delete("user.update_Auth", auth);

  }
```

UserService.java

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

UserServiceImpl.java

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

  }
```

UsersController.java

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

list.jsp

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


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>list</title>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU=" cross



</head>
<body>




  <div class="my-5">
    <div class="alert alert-dark">
      <div class="container">
        <h1>회원목록</h1>
      </div>
    </div>
  </div>

  <div class="container">
    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
    </div>
    <div style="padding-top: 50px">
      <table class="table table-hover">
        <tr>
          <th>No.</th>
          <th>아이디</th>
          <th style="width: 200px;">권한</th>
          <th>상태</th>
          <th>이메일</th>
          <th>이름</th>
          <th>관리</th>
        </tr>
        <form:form name="removeForm" method="put">
          <c:forEach items="${userList}" var="user">
            <tr>
              <td >${user.uno}</td>
              <td>${user.username}</td>
              <td>
              <select onchange="updateAuth('${user.username}', this)" class="form-select form-select" aria-label=".form-select-
                <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>
```

```
              <td>${user.enabled}</td>
              <td>${user.uemail}</td>
              <td>${user.uname}</td>
              <td>
              <p><a href="javascript:removeUser('${user.username}')"
                class="btn btn-danger btn-sm">삭제</a>
              <a href="/users/update?uno=${user.uno}"
                class="btn btn-primary btn-sm">수정</a></td>
            </tr>
          </c:forEach>
        </form:form>
        <tr>
          <th></th>
          <th></th>
          <th></th>
        </tr>
      </table>

      <a href="<c:url value="/cars" />" class="btn btn-primary">
        &laquo; 쇼핑 계속하기</a>
    </div>
    <hr>

  </div>
</body>


<script>


  function updateAuth(username, event) {
    $.ajax({
      type:"POST",
      url:"/users/list",
      data:{username: username,
          authority : event.value
      },
      beforeSend : function(xhr)
          {
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
      success:function(result) {
        window.location.reload();
      },
      error:function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }

    })

  }

  function removeUser(user) {
    $.ajax({
      type:"POST",
      url:"/users/removeUser",
      data:{username: user},
      beforeSend : function(xhr)
          {
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
      success:function(result) {
        window.location.reload();
      },
      error:function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }

    })

  }

  </script>
</html>
```

▼ C R U D

user_SQL.xml

```
<delete id="delete" parameterType="String">

    DELETE FROM users WHERE username = #{username}

</delete>
```

UserRepository.java

```
void deleteUser(String username);
```

UserRepositoryImpl.java

```
@Override
  public void deleteUser(String username) {
    this.sqlSessionTemplate.delete("user.delete", username);

  }
```

UserService.java

```
void deleteUser(String username);
```

UserServiceImpl.java

```
@Override
  public void deleteUser(String username) {
    userRepository.deleteUser(username);
  }
```

UsersController.java

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

list.jsp

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


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>list</title>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU=" cross



</head>
<body>
```

```
  <div class="my-5">
    <div class="alert alert-dark">
      <div class="container">
        <h1>회원목록</h1>
      </div>
    </div>
  </div>

  <div class="container">
    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
    </div>
    <div style="padding-top: 50px">
      <table class="table table-hover">
        <tr>
          <th>No.</th>
          <th>아이디</th>
          <th style="width: 200px;">권한</th>
          <th>상태</th>
          <th>이메일</th>
          <th>이름</th>
          <th>관리</th>
        </tr>
        <form:form name="removeForm" method="put">
          <c:forEach items="${userList}" var="user">
            <tr>
              <td >${user.uno}</td>
              <td>${user.username}</td>
              <td>
              <select onchange="updateAuth('${user.username}', this)" class="form-select form-select" aria-label=".form-select-
                <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>

              <td>${user.enabled}</td>
              <td>${user.uemail}</td>
              <td>${user.uname}</td>
              <td>
              <p><a href="javascript:removeUser('${user.username}')"
                class="btn btn-danger btn-sm">삭제</a>
              <a href="/users/update?uno=${user.uno}"
                class="btn btn-primary btn-sm">수정</a></td>
            </tr>
          </c:forEach>
        </form:form>
        <tr>
          <th></th>
          <th></th>
          <th></th>
        </tr>
      </table>

      <a href="<c:url value="/cars" />" class="btn btn-primary">
        &laquo; 쇼핑 계속하기</a>
    </div>
    <hr>

  </div>
</body>


<script>


  function updateAuth(username, event) {
    $.ajax({
      type:"POST",
      url:"/users/list",
      data:{username: username,
          authority : event.value
      },
      beforeSend : function(xhr)
          {
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
      success:function(result) {
        window.location.reload();
      },
      error:function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }
```

```
    })

  }

  function removeUser(user) {
    $.ajax({
      type:"POST",
      url:"/users/removeUser",
      data:{username: user},
      beforeSend : function(xhr)
          {
              xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
          },
      success:function(result) {
        window.location.reload();
      },
      error:function(request, status, error) {
        alert(request.status + " " + request.responseText);
      }

    })

  }

  </script>
</html>
```

▼ Python

Machine Learning

Machine Learning

```
# Machine Learning

# 데이터를 모델에 투입하면 데이터와 정답 사이의 관계를 찾아 모델을 만든다.

# 새로운 데이터를 입력했을 때 모델이 파악한 관계식을 적용하여 결과를 예측할 수 있게 된다.

# 정답이 없을 경우에는 데이터 속에 숨어 있는 패턴이나 규칙을 스스로 찾는다.

# (★중요)컴퓨터가 스스로 학습하여 문제를 해결하는 과정을 말한다.
```

```
# Supervised Learning(지도학습) VS Unsupervised Learning(비지도학습)

# 지도학습은 학습 과정에서 정답이 주어진다. 즉 문제와 답을 모두 알고 있는 상태에서 관계식을 찾아내는 머신러닝 알고리즘을 말한다.

# 비지도학습은 정답이 없이 문제만 주어진다. 따라서 정답 자체만 예측하는 것이 목표가 아니라 주어진 데이터에서 패턴 또는 규칙을 찾아내는 것이 목표가 된다.
```

```
# Machine Learning Process

# 문제점 파악 –> 데이터 탐색 –> ★데이터 전처리★ –> 모델 학습 –> 예측

# 문제점 파악 : 데이터 분석의 목표 설정, 분석 방법과 일정 수립.
# 데이터 탐색 : 데이터를 읽어서 이상유무[결측치(missing value), 이상치(outlier)] 확인.
# ★데이터 전처리★ : 모델 학습이 가능한 형태로 데이터를 정리하는 단계.
#                    데이터를 병합하고 이전 단계에서 확인한 결측치, 이상치 등을 처리한다.
#                    새로운 속성을 만들어서 추가하거나 불필요한 속성은 제거한다.
# 모델 학습 : 학습에 필요한 훈련용 데이터(ex.20%)와 평가를 위한 검증용 데이터(ex.80%)를 구분한다.
# 예측 : 테스트 데이터를 모델에 입력하여 예측 후 성능을 평가한다.
#        결과가 좋지 않을 경우에 다른 알고리즘을 사용하거나 파라미터[하이퍼파라미터(HyperParameter)] 값을 조정하여 다시 예측하고 평가한다.
```

```
# 문제점 파악 –> 데이터 탐색 –> ★데이터 전처리★ –> 모델 학습 –> 예측
```

```
# 1. 문제점 파악

x = [-3,31,-11,4,0,22,-2,-5,-25,-14]
y = [-2,32,-10,5,1,23,-1,-4,-24,-13]

# 각 리스트 x와 y는 각각 10개의 숫자를 원소로 갖는다.
# 잘 살펴보면 x와 y는 일차함수 관계가 성립한다.
# 즉 x값에 1을 더한 값이 y가 된다.
```

```
# 2. 데이터 탐색

import matplotlib.pyplot as plt

plt.plot(x,y)
```

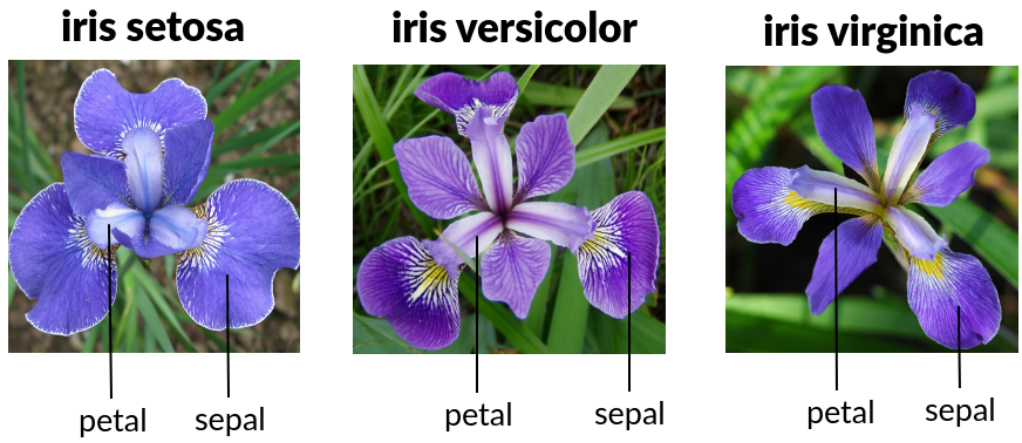
```
# 3. 데이터 전처리

import pandas as pd

x = [-3,31,-11,4,0,22,-2,-5,-25,-14]
y = [-2,32,-10,5,1,23,-1,-4,-24,-13]

df = pd.DataFrame({'X':x, 'Y':y})
df

df.shape

type(df)

train_features = ['X']
target_cols = ['Y']

X_train = df.loc[:, train_features]
y_train = df.loc[:, target_cols]
```

```
# 4. 모델 학습

from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_train, y_train)

import numpy as np

# X_test = np.arange(11, 16, 1).reshape(-1,1)
X_test = np.arange(111, 116, 1).reshape(-1,1)
X_test

y_pred = lr.predict(X_test)
y_pred
```

붓꽃(IRIS) 분류

```
# 붓꽃(IRIS) 분류

import pandas as pd
import numpy as np
from sklearn import datasets
iris = datasets.load_iris()

# 사이킷 런(sklearn)에는 학습용으로 붓꽃 데이터셋이 딕셔너리로 내장되어 있다.

iris.keys()

print(iris['DESCR'])

# sepal length(꽃받침 길이)
# sepal width(꽃받침 너비)
# petal length(꽃잎 길이)
# petal width(꽃잎 너비)

# 위의 4가지 값을 이용하여 3가지 범주로 데이터를 분류하게 된다.
```

```
# - Iris-Setosa
# - Iris-Versicolour
# - Iris-Virginica
# 3가지 품종 중에서 하나를 선택하는 다중 분류 문제이다.
```

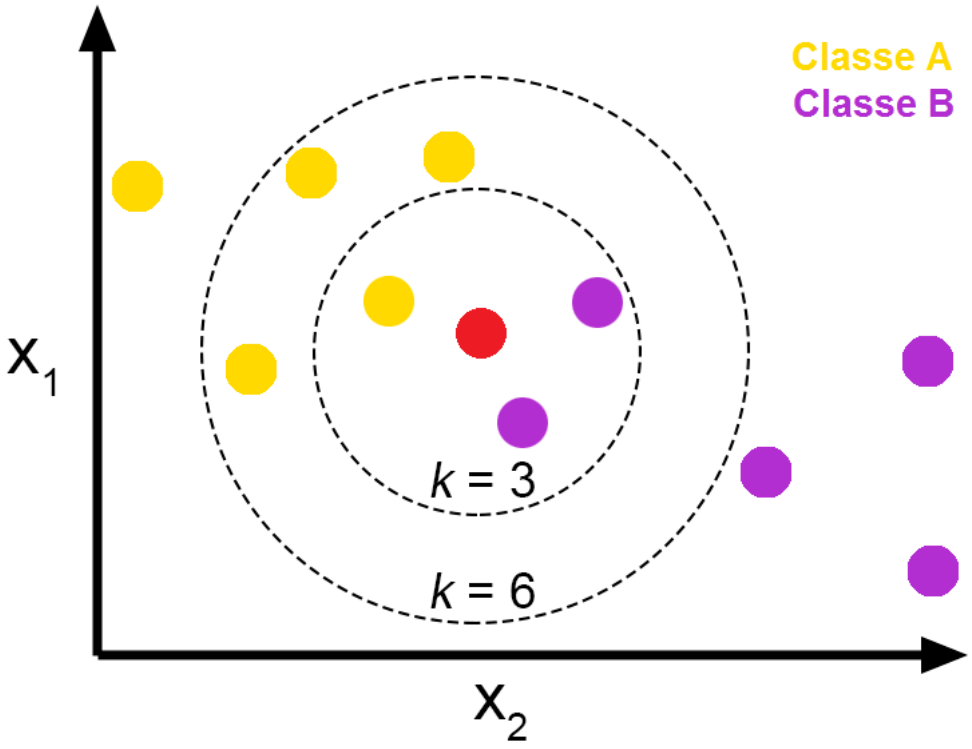



```
# 문제지
iris['data']

#답안지
iris['target']

# 데이터 프레임 전환

df = pd.DataFrame(iris['data'], columns=iris['feature_names'])
df

# 열 이름 바꾸기

df.columns = ['sepal_length','sepal_width','petal_length','petal_width']

df['Target'] = iris['target']
df

# 데이터 탐색 : 데이터를 읽어서 이상유무[결측치(missing value), 이상치(outlier)] 확인.

# 기본 정보보기
df.info()

# 요약 통계 정보
df.describe()

# 결측치 확인
df.isnull().sum()

# 중복 데이터 확인
df.duplicated().sum()

df.loc[df.duplicated(), :]

df.loc[(df.sepal_length==5.8)&(df.petal_width==1.9), :]

df = df.drop_duplicates()
df

# 중복 데이터 확인
df.duplicated().sum()

# 상관계수
df.corr()

import seaborn as sns

sns.heatmap(data=df.corr(), annot=True)

plt.hist(x="petal_width", data=df)

plt.hist(x='sepal_width', data=df)
```

```
sns.displot(x='sepal_width', kind='kde', data=df)

sns.displot(x='petal_width', kind='kde', data=df)

sns.displot(x='sepal_length', kind='kde', data=df)

sns.displot(x='sepal_length', hue='Target',kind='kde', data=df)

for col in ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']:
    sns.displot(x=col, hue='Target', kind='kde', data=df)

sns.pairplot(df, hue="Target", diag_kind="kind")

from sklearn.model_selection import train_test_split

X_data = df.loc[:,'sepal_length':'petal_width']
Y_data = df.loc[:,'Target']

X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    random_state=20)
```

KNN(K Nearest Neighbors)

```
# KNN(K 근접 이웃)

# K Nearest Neighbors
# 새로운 데이터가 들어왔을 때 기존 데이터의 그룹 중 어떤 그룹에 속하는지 분류하는 방법.
# 예측 하려는 데이터 X를 좌표평면상에 위치시켜 기존 데이터와 가장 비슷한 요소를 찾는다.
# K값에 따라 모델의 예측력이 달라지므로 적절할 K를 찾아야한다.

# 장점 : 수식에 관한 설명이 필요 없을 정도로 직관적이고 간단하다.
# 단점 : 데이터가 많아질수록 상당히 느려진다. 또한 이상치에 취약하다.
# 적용 : 주로 분류에 사용이 되며 작은 데이터 셋(대략 10,000개 정도)에 적합하다.
```




```
from sklearn.neighbors import KNeighborsClassifier

# 객체 생성
knn = KNeighborsClassifier(n_neighbors=11)

# 학습
```

```
knn.fit(X_train, Y_train)

# 예측
y_knn_pred = knn.predict(X_test)

# 평가
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, y_knn_pred)

# [shuffle=True]
# [n_neighbors=7]인 경우,  정확도 0.9666666666666667
# [n_neighbors=21]인 경우, 정확도 0.9333333333333333
# [n_neighbors=3]인 경우,  정확도 0.9333333333333333
# [n_neighbors=11]인 경우, 정확도 100%

# [shuffle=False]
# [n_neighbors=11]인 경우, 0.7333333333333333

# 예측
knn.predict([[1.9,3.0,5.1,1.1]])
```

SVM(Support Vector Machine)

```
# SVM

# Support Vector Machine
# 각 열들이 고유의 축을 갖는 벡터 공간을 이룬다고 가정한다.
# 모든 데이터를 벡터 공간에 점으로 표시하고 각 데이터가 속하는 목표별로 군집을 이룬다고 가정한다.
# 각 군집까지의 거리를 최대한 멀리 유지하는 선을 찾는다.
# 그 구분선을 이용하여 각 군집을 서로 확연하게 구분할 수 있다.
```

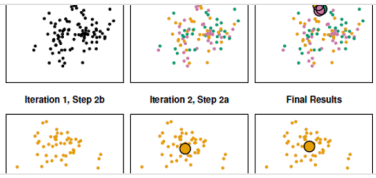
```
from sklearn.svm import SVC

# 객체 생성
# svc = SVC(kernel='rbf')
svc = SVC(kernel='linear')

# 학습
svc.fit(X_train, Y_train)

# 예측
y_knn_pred = svc.predict(X_test)

# 평가
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, y_knn_pred)
```

KNN(K Nearest Neighbors)

KNN(K Neighbor Nearest)이란?

KNN (K-Nearest Neighbor) In [1]: #주피터 노트북 블로그게시용 함수 from IPython.core.display import display, HTML display(HTML("")) KNN (K -Nearest Neighbor) 개요¶KNN은 K - 최근접 이웃법으로 분류 문제에 사용하는 알고리즘입니다.새로운 데이터가 들어왔을 때 기존 데이터의 그룹 중 어떤 그룹에 속하는지 분류하

https://smecsm.tistory.com/53

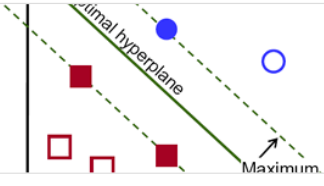


SVM(Support Vector Machine)

[OpenCV 머신러닝] 서포트 벡터 머신 알고리즘이란?

서포트 벡터 머신(SVM, Support Vectir Machine) 알고리즘 SVM 알고리즘은 머신러닝 알고리즘에서 가장 유명하고 많이 사용되고 있습니다. SVM 알고리즘은 기본적으로 두 개의 그룹(데이터)을 분리하는 방법으로 데이터들과 거리가 가장 먼 초평면(hyperplane)을 선택하여 분리하는 방법입니다. 데이터를 분리하기 위해 직선이 필요합

https://deep-learning-study.tistory.com/288

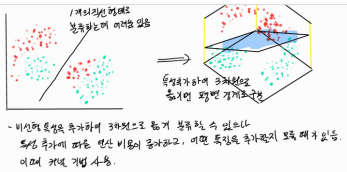


Kernel SVM

[Machine Learning] Kernel SVM(Support Vector Machine)

1. 커널 SVM이란 - 앞서 SVM 포스트에서는 선형 SVM을 중심으로 다뤘다. 그러나 실제로 선형 SVM으로 분류하기 어려운 데이터 형태들도 있다. 커널 기법의 기본적인 아이디어는 데이터를 높은 차원으로 이동시켜 고차원 공간에서 데이터를 분류하고자 함이다. 2. 듀얼 형태 변형 - 지난 SVM 포스팅에서 듀얼 형태를 다룬 적 있다. 듀얼 형

https://icefree.tistory.com/entry/Machine-Learning-Kernel-SVMSupport-Vector-Machine



프로그래밍에서의 Vector

[인공지능 기초] Vector 란?

Vector에 대해 알아보자. 1. Vector 란? 숫자를 원소로 가지는 리스트(list) 또는 배열(array) 을 말한다. import numpy as np x1 = [1,2,3] # 리스트 x2 = np.array([1,2,3]) # 배열 공간에서의 한 점을 말하며, 벡터는 원점으로부터 상대적 위치를 표현한다. 2. 스칼라곱 이란? 벡터에 숫자를 곱하면 길이가 변하게 되는데 이를 스칼라곱이라

https://upgrade-j.tistory.com/entry/인공지능-Vector-란