
데이터 입출력 구현

MEMBER 테이블 CRUD 프로그램 작업

김이화

1. Spring 기반 프로젝트 작업 – 초기 설정

1. pom.xml 버전 변경

- JAVA 버전 1.8
- SPRING 버전 4.3.28
- properties - Project Facets – Java 버전 1.8 변경

2. src - main - webapp - WEB-INF

- class 폴더 삭제
- spring 폴더 삭제
- views - home.jsp 삭제
- web.xml 내용 삭제

3. src/main/java/패키지

- HomeController.java 삭제

4. src/main/resources

- spring-context.xml 파일 생성
- aop, beans, c, context, mvc, p 네임스페이스 지정

5. 필요한 라이브러리 추가

- pom.xml - dependency 추가



```
<!-- 추가 필수 라이브러리 -->
<dependency>
  <groupId>org.aspectj</groupId>
  <artifactId>aspectjweaver</artifactId>
  <version>1.9.6</version>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.49</version>
</dependency>

<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.6</version>
</dependency>

<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.6</version>
</dependency>

<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.8.0</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
```

2. 화면 구현 - List

1. src - main - webapp - WEB-INF - views

- list.jsp 파일 생성
- modify.jsp 파일 생성
- register.jsp 파일 생성

2. list.jsp 코드는 다음과 같이 작성한다.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>list</title>
</head>
<body>
  <h3>Member 목록</h3>
  <a href="/membercrud/register">등록하기</a>
  <table border="1">
    <tr>
      <th>아이디</th>
      <th>이름</th>
      <th>휴대폰</th>
      <th>직급</th>
      <th>부서</th>
      <th>입사일</th>
      <th>관리</th>
    </tr>
    <tr>
      <td>uid</td>
      <td>name</td>
      <td>hp</td>
      <td>pos</td>
      <td>dep</td>
      <td>rdate</td>
      <td>
        <a href="/membercrud/modify">수정</a>
        <a href="/membercrud/delete">삭제</a>
      </td>
    </tr>
  </table>
</body>
</html>
```

기능 구현 전 임시로 작성된 테이블 내용

3. 결과 화면

Member 목록						
등록하기						
아이디	이름	휴대폰	직급	부서	입사일	관리
uid	name	hp	pos	dep	rdate	수정 삭제

2. 화면 구현 – Register

1. register.jsp의 body 영역 코드는 다음과 같이 작성한다.

```
<h3>Member 등록</h3>
<a href="/membercrud/list">목록으로</a>
<form action="/membercrud/register" method="post">
  <table border="1">
    <tr>
      <td>아이디</td>
      <td><input type="text" name="uid"/></td>
    </tr>
    <tr>
      <td>이름</td>
      <td><input type="text" name="name"/></td>
    </tr>
    <tr>
      <td>휴대폰</td>
      <td><input type="text" name="hp"/></td>
    </tr>
    <tr>
      <td>직급</td>
      <td><select name="pos">
        <option>사원</option>
        <option>대리</option>
        <option>과장</option>
        <option>차장</option>
        <option>부장</option>
      </select>
    </td>
    <td>부서</td>
    <td><select name="dep">
      <option value="101">영업1부</option>
      <option value="102">영업2부</option>
      <option value="103">영업3부</option>
      <option value="104">영업지원부</option>
      <option value="105">인사부</option>
    </select>
    </td>
    <td colspan="2" align="right"><input type="submit" value="등록"/></td>
  </tr>
</table>
</form>
```

2. 결과 화면

Member 등록	
목록으로	
아이디	<input type="text"/>
이름	<input type="text"/>
휴대폰	<input type="text"/>
직급	<div>사원 ▼</div>
부서	<div>영업1부 ▼</div>
<input type="submit" value="등록"/>	

직급(pos)과 부서(dep)는 셀렉트박스
값을 선택하여 입력하도록 코딩한다.

2. 화면 구현 – Modify

1. modify.jsp의 body 영역 코드는 다음과 같이 작성한다.

```
<h3>Member 수정</h3>
<a href="/membercrud/list">목록으로</a>
<form action="/membercrud/modify" method="post">
  <table border="1">
    <tr>
      <td>아이디</td>
      <td><input type="text" name="uid" readonly value="uid"/></td>
    </tr>
    <tr>
      <td>이름</td>
      <td><input type="text" name="name" value="name"/></td>
    </tr>
    <tr>
      <td>휴대폰</td>
      <td><input type="text" name="hp" value="hp"/></td>
    </tr>
    <tr>
      <td>직급</td>
      <td><select name="pos">
        <option>사원</option>
        <option>대리</option>
        <option>과장</option>
        <option>자장</option>
        <option>부장</option>
      </select>
    </td>
    <td>부서</td>
    <td><select name="dep">
      <option value="101">영업1부</option>
      <option value="102">영업2부</option>
      <option value="103">영업3부</option>
      <option value="104">영업지원부</option>
      <option value="105">인사부</option>
    </select>
    </td>
    <td colspan="2" align="right"><input type="submit" value="수정"/></td>
  </tr>
</table>
</form>
```

uid는 수정이 불가하도록 readonly 속성 추가

기능 구현 전 임시로 작성된 value 내용

2. 결과 화면

Member 수정

[목록으로](#)

아이디	uid
이름	name
휴대폰	hp
직급	사원 ▼
부서	영업1부 ▼
<input type="button" value="수정"/>	

3. Mybatis ORM 프레임워크 사용

1. src/main/resources

- spring-context.xml 필요한 설정 추가

작성하는 프로젝트의 패키지 이름을 입력한다.

```
<!-- 어노테이션 기반 컴포넌트 스캔 -->
<context:component-scan base-package="kr.co.membercrud" />

<!-- 뷰 처리를 위한 ViewResolver 설정 -->
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
</bean>

<!-- Spring MVC(Web)에서 사용되는 Annotation 활성화를 위해 설정 -->
<mvc:annotation-driven />

<!-- 데이터베이스 설정 -->
<bean id="basicDataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://192.168.10.114:3306/keh" />
    <property name="username" value="keh" />
    <property name="password" value="1234" />
</bean>

<!-- JDBC 설정 -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="basicDataSource" />
</bean>

<!-- Mybatis 설정 -->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="basicDataSource" />
    <property name="configLocation" value="classpath:mybatis-context.xml" />
</bean>

<!-- Mybatis-Spring 설정 -->
<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg ref="sqlSessionFactoryBean" />
</bean>
```

실제로 사용할 데이터베이스 주소, 아이디, 비밀번호를 입력한다.

3. Mybatis ORM 프레임워크 사용

1. src/main/resources

- mapper 패키지를 생성하고 그 안에 member.xml 작성

```
<?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="mapper.member">
```

```
<insert id="INSERT_MEMBER">
  INSERT INTO `MEMBER` VALUES (#{uid}, #{name}, #{hp}, #{pos}, #{dep}, NOW());
</insert>
```

매핑된 INSERT 구문

```
<select id="SELECT_MEMBERS" resultType="kr.co.membercrud.MemberVo">
  SELECT * FROM `MEMBER`;
</select>
```

매핑된 SELECT 구문 (목록 출력용)

```
<select id="SELECT_MEMBER" resultType="kr.co.membercrud.MemberVo">
  SELECT * FROM `MEMBER` WHERE uid=#{uid};
</select>
```

매핑된 SELECT 구문 (수정/삭제용)

```
<update id="UPDATE_MEMBER">
  UPDATE `MEMBER` SET
    `name`=#{name},
    `hp`=#{hp},
    `pos`=#{pos},
    `dep`=#{dep}
  WHERE
    `uid`=#{uid};
</update>
```

매핑된 UPDATE 구문

```
<delete id="DELETE_MEMBER">
  DELETE FROM `MEMBER` WHERE uid=#{uid};
</delete>
```

매핑된 DELETE 구문

```
</mapper>
```

3. Mybatis ORM 프레임워크 사용

1. src/main/resources

- mybatis-context.xml 작성

```
<?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>
  <mappers>
    <mapper resource="./mapper/member.xml"/>
  </mappers>
</configuration>
```

2. src/main/java/패키지

- MemberVo.java 작성

변수 선언 후 **Getter, Setter** 생성

```
private String uid;
private String name;
private String hp;
private String pos;
private int dep;
private String rdate;
```

```
public String getUid() {
    return uid;
}
public void setUid(String uid) {
    this.uid = uid;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getHp() {
    return hp;
}
public void setHp(String hp) {
    this.hp = hp;
}
public String getPos() {
    return pos;
}
public void setPos(String pos) {
    this.pos = pos;
}
public int getDep() {
    return dep;
}
public void setDep(int dep) {
    this.dep = dep;
}
public String getRdate() {
    return rdate;
}
```

변수 rdate는 데이터베이스에서 SQL 명령어로 생성된 값을 가져만 올 것이기 때문에 Setter 없이 Getter만 생성한다.

3. Mybatis ORM 프레임워크 사용

2. src/main/java/패키지

- MemberDao.java 작성

```
package kr.co.membercrud;  
  
import java.util.List;  
  
import javax.inject.Inject;  
  
import org.mybatis.spring.SqlSessionTemplate;  
import org.springframework.stereotype.Repository;  
  
import kr.co.membercrud.MemberVo;
```

@Repository 해당 클래스를 DAO 객체로 생성
`public class MemberDao {`

@Inject 타입이 일치하는 객체를 자동으로 주입
`private SqlSessionTemplate mybatis;`

```
public void insertMember(MemberVo vo) {  
    mybatis.insert("mapper.member.INSERT_MEMBER", vo);  
}  
  
public MemberVo selectMember(String uid) {  
    return mybatis.selectOne("mapper.member.SELECT_MEMBER", uid);  
}  
  
public List<MemberVo> selectMembers() {  
    return mybatis.selectList("mapper.member.SELECT_MEMBERS");  
}  
  
public void updateMember(MemberVo vo) {  
    mybatis.update("mapper.member.UPDATE_MEMBER", vo);  
}  
  
public void deleteMember(MemberVo vo) {  
    mybatis.delete("mapper.member.DELETE_MEMBER", vo);  
}  
}
```

3. Mybatis ORM 프레임워크 사용

2. src/main/java/패키지

- MemberService.java 작성

```
package kr.co.membercrud;

import java.util.List;

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

import kr.co.membercrud.MemberDao;
import kr.co.membercrud.MemberVo;
```

@Service 해당 클래스를 Service 객체로 생성

```
public class MemberService {
```

@Autowired 타입이 일치하는 객체를 자동으로 주입

```
    private MemberDao dao;

    public void insertMember(MemberVo vo) {
        dao.insertMember(vo);
    }

    public MemberVo selectMember(String uid) {
        return dao.selectMember(uid);
    }

    public List<MemberVo> selectMembers() {
        return dao.selectMembers();
    }

    public void updateMember(MemberVo vo) {
        dao.updateMember(vo);
    }

    public void deleteMember(MemberVo vo) {
        dao.deleteMember(vo);
    }
}
```

3. Mybatis ORM 프레임워크 사용

2. src/main/java/패키지

- MemberController.java 작성

```
package kr.co.membercrud;
```

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

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

```
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 kr.co.membercrud.MemberService;
```

```
import kr.co.membercrud.MemberVo;
```

@Controller 해당 클래스를 Controller 객체로 생성

```
public class MemberController {
```

@Autowired 타입이 일치하는 객체를 자동으로 주입

```
private MemberService service;
```

@GetMapping("/register") HTTP GET 요청 처리

```
public String register() {
```

```
    return "/register";
```

```
}
```

@PostMapping("/register") HTTP POST 요청 처리

```
public String register(MemberVo vo) {
```

```
    service.insertMember(vo);
```

```
    return "redirect:/list";
```

```
}
```

@GetMapping(value={"/", "/list"}) /list 주소 뿐 아니라 기본 주소도 list 페이지로 연결

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

```
    List<MemberVo> members = service.selectMembers();
```

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

```
    return "/list";
```

```
}
```

```
@GetMapping("/modify")
```

```
public String modify(String uid, Model model) {
```

```
    MemberVo vo = service.selectMember(uid);
```

```
    model.addAttribute(vo);
```

```
    return "/modify";
```

```
}
```

```
@PostMapping("/modify")
```

```
public String modify(MemberVo vo) {
```

```
    service.updateMember(vo);
```

```
    return "redirect:/list";
```

```
}
```

```
@GetMapping("/delete")
```

```
public String delete(MemberVo vo) {
```

```
    service.deleteMember(vo);
```

```
    return "redirect:/list";
```

```
}
```

```
}
```

3. Mybatis ORM 프레임워크 사용

3. src - main - webapp - WEB-INF

- web.xml 필요한 설정 추가

```
<!-- Spring Front-Controller(Dispatcher) 등록 -->
<servlet>
    <servlet-name>DispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring-context.xml</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>DispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- 전송 데이터 문자 인코딩 설정 -->
<filter>
    <filter-name>characterEncodingFilter</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>
</filter>
<filter-mapping>
    <filter-name>characterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

4. 기능 구현 - List

1. src - main - webapp - WEB-INF - views

- list.jsp 코드를 다음과 같이 수정한다.

```
<c:forEach var="member" items="${members}">
  <c:set var="rdateSubstr" value="${member.rdate}"/>
  <tr>
    <td>${member.uid}</td>
    <td>${member.name}</td>
    <td>${member.hp}</td>
    <td>${member.pos}</td>
    <td>${member.dep}</td>
    <td>${fn:substring(rdateSubstr,2,10)}</td>
  </tr>
  <tr>
    <td>
      <a href="/membercrud/modify?uid=${member.uid}">수정</a>
      <a href="/membercrud/delete?uid=${member.uid}">삭제</a>
    </td>
  </tr>
</c:forEach>
```

- <c:forEach> 태그로 리스트화된 Member 테이블의 내용을 출력한다.

- <fn:substring> 태그로 입사일을 YY-MM-DD 형태로 출력

- 수정/삭제 페이지로 이동할 때 해당 멤버를 식별하기 위해 uid 값을 가져간다.

2. 결과 화면

Member 목록

[등록하기](#)

아이디	이름	휴대폰	직급	부서	입사일	관리
a101	박혁거세	010-1234-1001	부장	101	21-03-25	수정 삭제
a102	김유신	010-1234-1002	차장	101	21-03-24	수정 삭제
a103	김춘추	010-1234-1003	사원	101	21-03-25	수정 삭제
a104	장보고	010-1234-1004	대리	102	21-03-24	수정 삭제
a105	강감찬	010-1234-1005	과장	102	21-03-24	수정 삭제
a106	이성계	010-1234-1006	차장	103	21-03-24	수정 삭제
a107	정철	010-1234-1007	차장	103	21-03-24	수정 삭제
a108	이순신	010-1234-1008	부장	104	21-03-24	수정 삭제
a109	허균	010-1234-1009	부장	104	21-03-24	수정 삭제
a110	정약용	010-1234-1010	사원	105	21-03-24	수정 삭제
a111	박지원	010-1234-1011	사원	105	21-03-24	수정 삭제
t000	김하나	000-0000-0000	대리	102	21-04-14	수정 삭제
t001	최두리	000-0000-0001	사원	101	21-04-14	수정 삭제

4. 기능 구현 – Register

1. 화면 구현에서 작성한 코드에서 수정할 사항이 없으므로 설명을 생략한다.

4. 기능 구현 - Modify

1. src - main - webapp - WEB-INF - views

- modify.jsp 코드를 다음과 같이 수정한다.

```
<tr>
  <td>아이디</td>
  <td><input type="text" name="uid" readonly value="${memberVo.uid}"/></td>
</tr>
<tr>
  <td>이름</td>
  <td><input type="text" name="name" value="${memberVo.name}"/></td>
</tr>
<tr>
  <td>휴대폰</td>
  <td><input type="text" name="hp" value="${memberVo.hp}"/></td>
</tr>
```

수정하려는 멤버의 아이디, 이름, 휴대폰 정보가
입력폼에 출력되어 있도록 memberVo에서 값을
가져온다.

2. 결과 화면

Member 수정

목록으로

아이디	a101
이름	박혁거세
휴대폰	010-1234-1001
직급	사원 ▼
부서	영업1부 ▼
<input type="button" value="수정"/>	