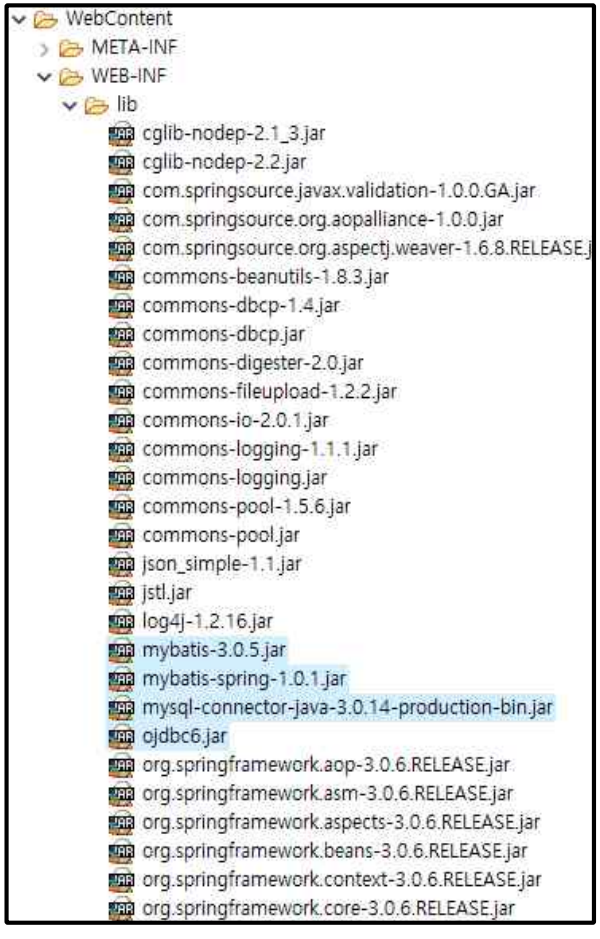
24장 스프링과 마이바티스 연동하기

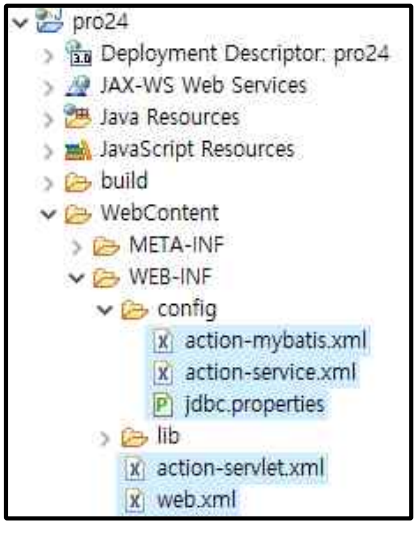
24.1 스프링-마이바티스 연동 XML 파일 설정

1. 새 프로젝트 pro24를 만들고 이 책에서 제공하는 스프링 라이브러리를 복사해 lib 폴더에 붙여 넣습니다.



24.1 스프링-마이바티스 연동 XML 파일 설정

2. 스프링에서 사용할 빈을 생성하는 데 필요한 XML 파일을 설정하겠습니다. 먼저 XML 관련 파일들을 다음과 같이 준비합니다.



24.1 스프링-마이바티스 연동 XML 파일 설정

3. web.xml을 다음과 같이 작성하여 애플리케이션 실행 시 여러 설정 파일들을 /WEB-INF/config 폴더에서 읽어 들이도록 합니다.

코드 24-1 pro24/WebContent/WEB-INF/web.xml

```
<web-app ...>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/config/action-mybatis.xml
      /WEB-INF/config/action-service.xml
    </param-value>
  </context-param>
...
</web-app>
```

마이바티스 설정 파일을 읽어 들입니다.

24.1 스프링-마이바티스 연동 XML 파일 설정

4. action-servlet.xml에서는 뷰 관련 빈과 각 URL 요청명에 대해 호출할 메서드들을 설정합니다.

코드 24-2 pro24/WebContent/WEB-INF/action-servlet.xml

```
...
<beans>
  <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/views/member/" />
    <property name="suffix" value=".jsp"/>
  </bean>
  <bean id="memberController"
        class="com.spring.member.controller.MemberControllerImpl">
    <property name="methodNameResolver">
      <ref local="memberMethodNameResolver"/>
    </property>
    <property name="memberService" ref="memberService"/>
  </bean>
```

JSP 파일의 위치를 지정합니다.

24.1 스프링-마이바티스 연동 XML 파일 설정

```
<bean id="memberMethodNameResolver"
                    class="org.springframework.web.servlet.mvc.multiaction.
                                                    PropertiesMethodNameResolver">
  <property name="mappings">
    <props>
      <prop key="/member/listMembers.do">listMembers</prop>
      <prop key="/member/addMember.do">addMember</prop>
      <prop key="/member/removeMember.do">removeMember</prop>
      <prop key="/member/memberForm.do">form</prop>
    </props>
  </property>
</bean>

<bean id="memberUrlMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="/member/*.do">memberController</prop>
    </props>
  </property>
</bean>
</beans>
```

24.1 스프링-마이바티스 연동 XML 파일 설정

5. action-mybatis.xml을 다음과 같이 작성합니다.
(jdbc.properties 파일은 22장의 jdbc.properties 파일을 복사해 붙여 넣은 것입니다.)

코드 24-3 pro24/WebContent/WEB-INF/config/action-mybatis.xml

```
...
<bean id="propertyPlaceholderConfigurer"
          class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
  <property name="locations">
    <value>/WEB-INF/config/JDBC.properties</value>
  </property>
</bean>
```

PropertyPlaceholderConfigurer 클래스를 이용해 데이터베이스 설정 관련 정보를 jdbc.properties 파일에서 읽어 들입니다.

```
<bean id="dataSource" class="org.apache.ibatis.datasource.pooled.PooledDataSource">
  <property name="driver" value="${jdbc.driverClassName}" />
  <property name="url" value="${jdbc.url}" />
  <property name="username" value="${jdbc.username}" />
  <property name="password" value="${jdbc.password}" />
</bean>
```

마이바티스에서 제공하는 PooledDataSource 클래스를 이용해서 dataSource 빈을 생성합니다.

24.1 스프링-마이바티스 연동 XML 파일 설정

SqlSessionFactoryBean 클래스를 이용해 dataSource 속성에 dataSource 빈을 설정합니다.

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name='configLocation" value="classpath:mybatis /model/modelConfig.xml"  />
  <property name="mapperLocations" value="classpath:mybatis/mappers/*.xml" />
</bean>
```

configLoation 속성에 modelConfig.xml을 설정합니다.

mapperLocations 속성에 mybatis/mappers 패키지의 모든 매퍼 파일들을 읽어 들여와 설정합니다.

```
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg index="0" ref="sqlSessionFactory"></constructor-arg>
</bean>
```

SqlSessionTemplate 클래스를 이용해 sqlSession 빈을 생성합니다.

```
<bean id="memberDAO" class="com.spring.member.dao.MemberDAOImpl">
  <property name="sqlSession" ref="sqlSession"></property>
</bean>
</beans>
```

sqlSession 빈을 memberDAO 빈 속성에 주입합니다.

24.1 스프링-마이바티스 연동 XML 파일 설정

6. MemberDAO 빈을 사용할 memberService 빈을 설정합니다.

코드 24-4 pro24/WebContent/WEB-INF/config/action-service.xml

```
...
  <bean id="memberService" class="com.spring.member.service.MemberServiceImpl">
    <property name="memberDAO" ref="memberDAO"/>
  </bean>
</beans>
```

memberDAO 빈을 memberService 빈의 속성에 주입합니다.

24.2 마이바티스 관련 XML 파일 설정하기

1. src 패키지 아래에 mybatis.mappers 패키지를 만들고 23장에서 실습한 member.xml을 복사해 붙여 넣습니다.

24.2 마이바티스 관련 XML 파일 설정하기

2. 회원 관련 기능 SQL문이 있는 매퍼 파일인 member.xml입니다. 23장에서 실습한 member.xml을 그대로 복사해 붙여 넣습니다.

코드 24-5 pro24/src/mybatis/mappers/member.xml

```
...
<mapper namespace="mapper.member">
  <resultMap id="memResult" type="memberVO">
    <result property="id" column="id" />
    <result property="pwd" column="pwd" />
    <result property="name" column="name" />
    <result property="email" column="email" />
    <result property="joinDate" column="joinDate" />
  </resultMap>

  <select id="selectAllMemberList" resultMap="memResult">
    <![CDATA[
      select * from t_member  order by joinDate desc
    ]]>
  </select>

  <insert id="insertMember"  parameterType="memberVO">
    <![CDATA[
      insert into t_member(id,pwd, name, email)
      values(#{id}, #{pwd}, #{name}, #{email})
    ]]>
  </insert>
  ...
```

24.2 마이바티스 관련 XML 파일 설정하기

3. modelConfig.xml에서는 <typeAlias> 태그를 이용해 매퍼 파일에서 긴 이름의 클래스를 별칭으로 사용할 수 있게 설정합니다.

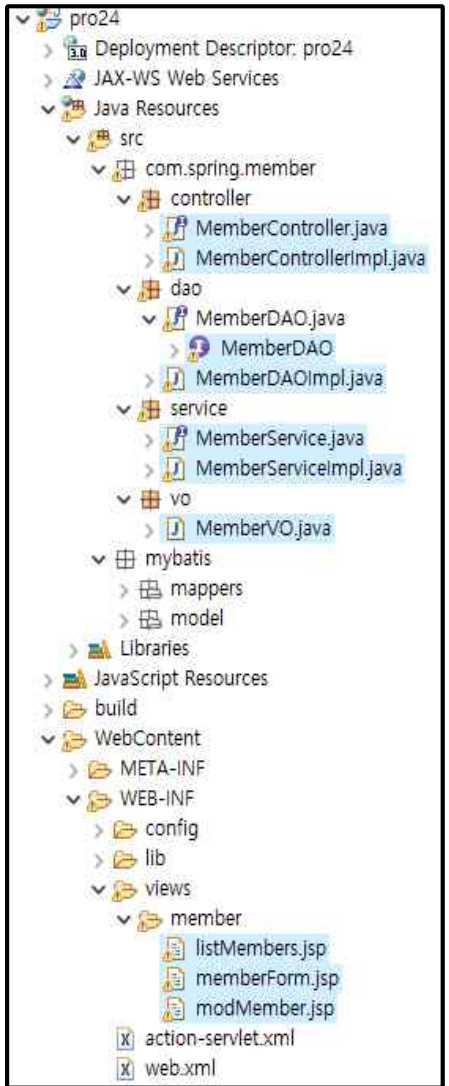
코드 24-6 pro24/src/mybatis/model/modelConfig.xml

```
...
<configuration>
  <typeAliases>
    <typeAlias type="com.spring.member.vo.MemberVO" alias="memberVO" />
  </typeAliases>
</configuration>
```

〈typeAliases〉 태그로 마이바티스에서 데이터 전달에 사용될 memberVO 빈을 설정합니다.

24.3 자바 클래스와 JSP 파일 구현하기

1. 자바 클래스는 다음과 같이 com.spring.member 패키지 아래에 다시 패키지별로 위치시킵니다. 또 JSP 파일은 WEB-INF/views/member 폴더에 위치하도록 준비합니다.



24.3 자바 클래스와 JSP 파일 구현하기

2. 컨트롤러 역할을 하는 MemberControllerImpl 클래스에서는 memberService 속성에 빈을 주입하기 위해 setter를 구현합니다.

코드 24-7 pro24/src/com/spring/member/controller/MemberControllerImpl.java

```
package com.spring.member.controller;
 ...
public class MemberControllerImpl extends MultiActionController implements
MemberController {
  private MemberService memberService;
  public void setMemberService(MemberServiceImpl memberService) {
    this.memberService = memberService;
  }

  @Override
  public ModelAndView listMembers(HttpServletRequest request,
                                  HttpServletResponse response) throws Exception {
    String viewName = getViewName(request);
    List membersList = memberService.listMembers();
    ModelAndView mav = new ModelAndView(viewName);
    mav.addObject("membersList", membersList);
    return mav;
  }
```

memberService 빈을 주입하기 위해 setter를 구현합니다.

조회한 회원 정보를 ModelAndView의 addObject() 메서드를 이용해 바인딩합니다.

24.3 자바 클래스와 JSP 파일 구현하기

```
@Override
public ModelAndView addMember(HttpServletRequest request,
                              HttpServletResponse response) throws Exception {
  request.setCharacterEncoding("utf-8");
  MemberVO memberVO = new MemberVO();
  /*
  String id=request.getParameter("id");
  String pwd=request.getParameter("pwd");
  String name=request.getParameter("name");
  String email = request.getParameter("email");
  memberVO.setId(id);
  memberVO.setPwd(pwd);
  memberVO.setName(name);
  memberVO.setEmail(email);
  */
  bind(request, memberVO);
  int result = 0;
```

회원 가입창에서 전송된 회원 정보를 bind() 메서드를 이용해 memberVO 해당 속성에 자동으로 설정합니다.

24.3 자바 클래스와 JSP 파일 구현하기

```
    result = memberService.addMember(memberVO);
    ModelAndView mav = new ModelAndView("redirect:/member/listMembers.do");
    return mav;
  }

  @Override
  public ModelAndView removeMember(HttpServletRequest request,
                                   HttpServletResponse response) throws Exception{
    request.setCharacterEncoding("utf-8");
    String id=request.getParameter("id");
    memberService.removeMember(id);
    ModelAndView mav = new ModelAndView("redirect:/member/listMembers.do");
    return mav;
  }

  public ModelAndView form(HttpServletRequest request, HttpServletResponse response)
  throws Exception {
    String viewName = getViewName(request);
    ModelAndView mav = new ModelAndView();
    mav.setViewName(viewName);
    return mav;
  }
}
```

회원 정보 추가 후 ModelAndView 클래스의 redirect 속성을 이용해 /member/listMembers.do로 리다이렉트합니다.

회원 정보를 삭제하고 회원 목록창으로 리다이렉트합니다.

데이터베이스 연동 작업이 없는 입력창 요청 시 뷰이름만 ModelAndView로 반환합니다.

24.3 자바 클래스와 JSP 파일 구현하기

3. 속성 memberDAO에 memberDAO 빈을 주입하기 위해 setter를 구현합니다.

코드 24-8 pro24/src/com/spring/member/service/MemberServiceImpl.java

```
package com.spring.member.service;
...
public class MemberServiceImpl  implements MemberService{
  private MemberDAO memberDAO;
  public void setMemberDAO(MemberDAO memberDAO){
    this.memberDAO = memberDAO;
  }

  @Override
  public List listMembers() throws DataAccessException {
    List membersList = null;
    membersList = memberDAO.selectAllMemberList();
    return membersList;
  }

  @Override
  public int addMember(MemberVO memberVO) throws DataAccessException {
    return memberDAO.insertMember(memberVO);
  }

  @Override
  public int removeMember(String id) throws DataAccessException {
    return memberDAO.deleteMember(id);
  }
}
```

속성 memberDAO에 memberDAO 빈을 주입하기 위해 setter를 구현합니다.

24.3 자바 클래스와 JSP 파일 구현하기

4. 설정 파일에서 만든 sqlSession 빈을 속성 sqlSession에 주입하기 위해 setter를 구현합니다. sqlSession 빈의 메서드들을 이용해 매퍼 파일에 정의된 SQL문을 사용합니다.

코드 24-9 pro24/src/com/spring/member/dao/MemberDAOImpl.java

```
package com.spring.member.dao;
...
public class MemberDAOImpl implements MemberDAO {
  private SqlSession sqlSession;
  public void setSqlSession(SqlSession sqlSession) {
    this.sqlSession = sqlSession;
  }

  @Override
  public List selectAllMemberList() throws DataAccessException {
    List<MemberVO> membersList = null;
    membersList = sqlSession.selectList("mapper.member.selectAllMemberList");
    return membersList;
  }

  @Override
  public int insertMember(MemberVO memberVO) throws DataAccessException {
    int result = sqlSession.insert("mapper.member.insertMember", memberVO);
    return result;
  }
```

속성 sqlSession에 sqlSession 빈을 주입하기 위해 setter를 구현합니다.

주입된 sqlSession 빈으로 selectList() 메서드를 호출하면서 SQL문의 id를 전달합니다.

주입된 sqlSession 빈으로 insert() 메서드를 호출하면서 SQL문의 id와 memberVO를 전달합니다.

24.3 자바 클래스와 JSP 파일 구현하기

```
  @Override
  public int deleteMember(String id) throws DataAccessException {
    int result = sqlSession.delete("mapper.member.deleteMember", id);
    return result;
  }
}
```

주입된 sqlSession 빈으로 delete() 메서드를 호출하면서 SQL문의 id와 회원 ID를 전달합니다.

24.3 자바 클래스와 JSP 파일 구현하기

5. listMembers.jsp를 다음과 같이 작성합니다. 회원 목록창에서 회원 정보를 표시하면서 삭제하기 링크도 추가합니다.

코드 24-10 pro24/WebContent/WEB-INF/view/member/listMembers.jsp

```
...
<c:forEach var="member" items="${membersList}" >
   <tr align=center>
      <td>${member.id}</td>
      <td>${member.pwd}</td>
      <td>${member.name}</td>
      <td>${member.email }</td>
      <td>${member.joinDate }</td>
      <td><a href="${contextPath}/member/removeMember.do?id=${member.id}">삭제하기</a></td>
   </tr>
</c:forEach>
  </table>
  <a href="${contextPath}/member/memberForm.do">
    <h1 style="text-align:center">회원가입</h1>
  </a>
  ...
```

삭제하기 클릭 시 /member/removeMember.do로 요청합니다.

회원가입 클릭 시 /member/memberForm.do로 요청합니다.

24.3 자바 클래스와 JSP 파일 구현하기

6. 회원 가입창에서 회원 정보를 입력한 후 action 값을 /member/addMember.do 서블릿으로 전송합니다.

코드 24-11 pro24/WebContent/WEB-INF/view/member/memberForm.jsp

```
...
<form method="post" action="${contextPath}/member/addMember.do">
<h1  class="text_center">회원 가입창</h1>
<table  align="center">
  <tr>
    <td width="200"><p align="right">사용자 아이디</td>
    <td width="400"><input type="text" name="id"></td>
  </tr>
  ...
  ...
  <tr>
    <td width="200"><p> </p></td>
    <td width="400"><
      input type="submit" value="가입하기"><input type="reset" value="다시입력">
    </td>
</tr>
...
```

회원 가입창에서 **가입하기** 클릭 시 /member/addMember.do로 요청합니다.

24.3 자바 클래스와 JSP 파일 구현하기

7. http://localhost:8080/pro24/member/listMembers.do로 요청하면 다음과 같이 회원 정보를 표시합니다. 하단의 회원가입을 클릭합니다.

아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
park	1234	박찬호	park@test.com	2018-09-04	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입

24.3 자바 클래스와 JSP 파일 구현하기

8. 김동준이라는 새 회원 정보를 입력하고 가입하기를 클릭합니다.

24.3 자바 클래스와 JSP 파일 구현하기

9. 그러면 컨트롤러에 http://localhost:8080/pro24/member/addMember.do로 요청하여 회원을 추가하고 다시 회원 목록을 표시합니다. 이번에는 삭제하기를 클릭해 앞에서 추가한 '김동준' 회원을 삭제해 볼까요?

아이디	비밀번호	이름	이메일	가입일	삭제
djkim	1234	김동준	djkim@test.com	2018-10-01	삭제하기
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
jspark	1234	박지성	jspark@test.com	2018-09-29	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
park	1234	박찬호	park@test.com	2018-09-04	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입

24.3 자바 클래스와 JSP 파일 구현하기

10. 삭제하기를 클릭하면 서버의 http://localhost:8080/pro24/member/removeMember.do로 요청합니다. 다음과 같이 '김동준' 회원 정보가 삭제된 것을 볼 수 있습니다.

아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
park	1234	박찬호	park@test.com	2018-09-04	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입