

23장 마이바티스 프레임워크 사용하기

23.1 마이바티스란?

23.2 마이바티스 설치하기

23.3 마이바티스 이용해 회원 기능 실습하기

23.4 마이바티스 이용해 회원 정보 CRUD 실습하기

23.5 마이바티스의 동적 SQL문 사용하기

23.1 마이바티스란?

마이바티스 등장 배경

기존 JDBC에서 SQL문 실행 방식

코드 23-1 AdminDAO.java

...

```
public void addGoods(GoodsVO goodsVO) throws SQLException {  
    Connection con = dataFactory.getConnection();  
    Statement stmt = con.createStatement();  
    String query = "insert into t_Goods_info (goods_id,"+  
        "goods_sort,"+  
        "goods_title,"+  
        "goods_writer,"+  
        "goods_publisher,"+  
        "goods_price,"+  
        "goods_sales_price,"+  
        "goods_point,"+  
        "goods_published_date,"+  
        "goods_total_page,"+  
        "goods_isbn,"+  
        "goods_delivery_price,"+
```

23.1 마이바티스란?

```
"goods_delivery_date,"+
"goods_type,"+
"goods_writer_intro,"+
"goods_intro,"+
"goods_publisher_comment,"+
"goods_recommendation,"+
"goods_contents_order");
query+=" values('"+
    goodsV0.getGoods_id()+"', '"+
    goodsV0.getGoods_sort()+"', '"+
    goodsV0.getGoods_title()+"', '"+
    goodsV0.getGoods_writer()+"', '"+
    goodsV0.getGoods_publisher()+"', '"+
    goodsV0.getGoods_price()+"', '"+
    goodsV0.getGoods_sales_price()+"', '"+
    goodsV0.getGoods_point()+"', '"+
    goodsV0.getGoods_published_date()+"', '"+
    goodsV0.getGoods_page_total()+"', '"+
    goodsV0.getGoods_isbn()+"', '"+
    goodsV0.getGoods_delivery_price()+"', '"+
    goodsV0.getGoods_delivery_date()+"', '"+
    goodsV0.getGoods_type()+"', '"+
    goodsV0.getGoods_writer_intro()+"', '"+
    goodsV0.getGoods_intro()+"', '"+
```

23.1 마이바티스란?

기존의 JDBC의 문제점

- connection → Statement 객체 생성 → SQL문 전송 → 결과 반환 → close 과정으로 작업함
- SQL문이 프로그래밍 코드에 섞여 코드를 복잡해서 사용 및 유지 보수가 어려워짐



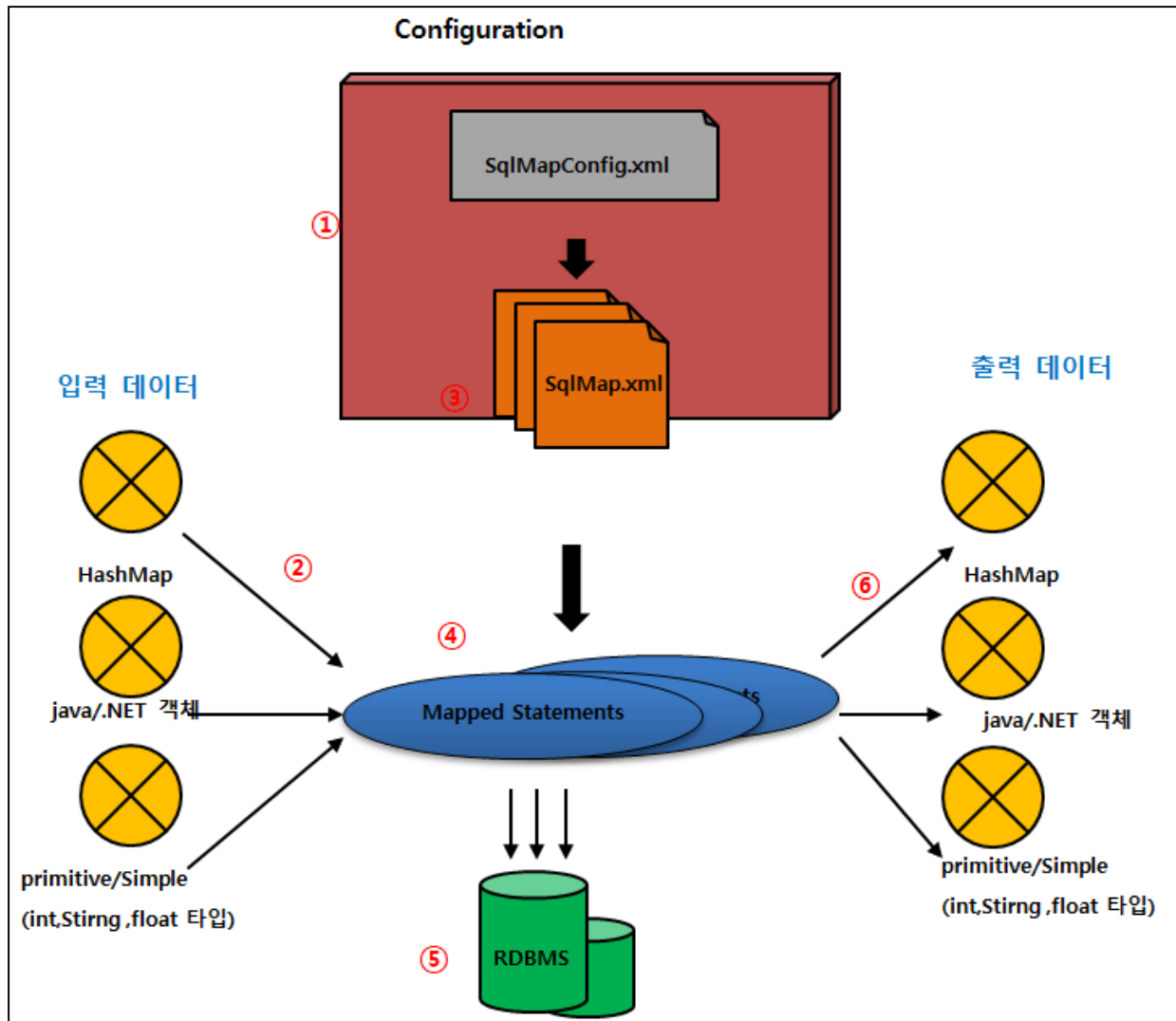
- 마이바티스 프레임워크를 도입해서 SQL문의 가독성을 높여서 사용이 편리하게 만듦
- 코드와 SQL문을 분리해서 사용 및 유지 보수를 편리하게 함

마이바티스 프레임워크의 특징

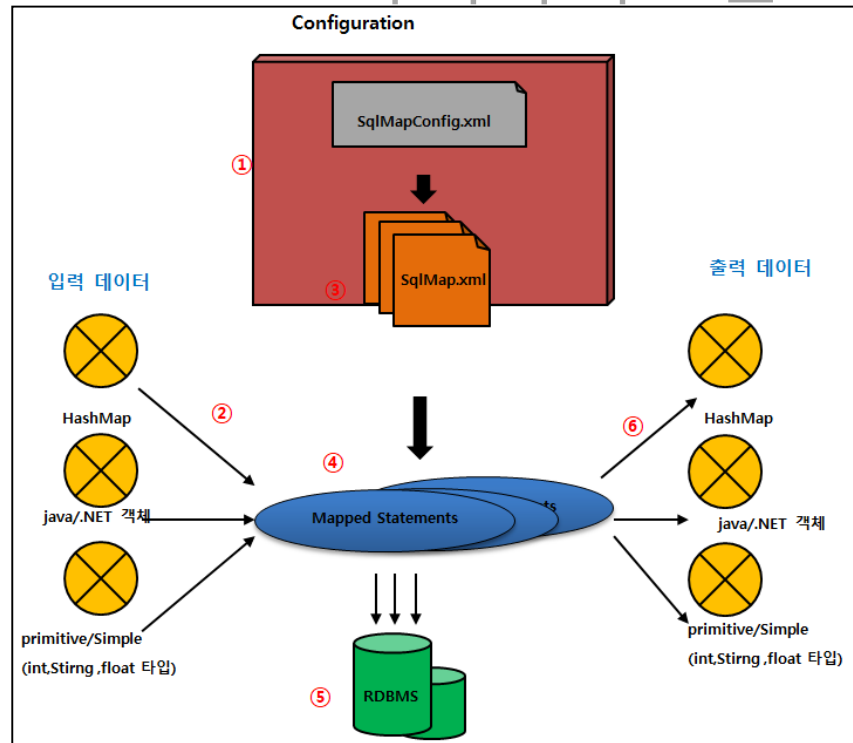
- SQL 실행 결과를 자바 빈즈 또는 Map 객체에 매핑해 주는 Persistence 솔루션으로 관리함
즉, SQL을 소스 코드가 아닌 XML로 분리함
- SQL문과 프로그래밍 코드를 분리해서 구현함
- 데이터소스(DataSource) 기능과 트랜잭션 처리 기능을 제공

23.1 마이바티스란?

퍼시스턴스 프레임워크로 사용되는 마이바티스 구조



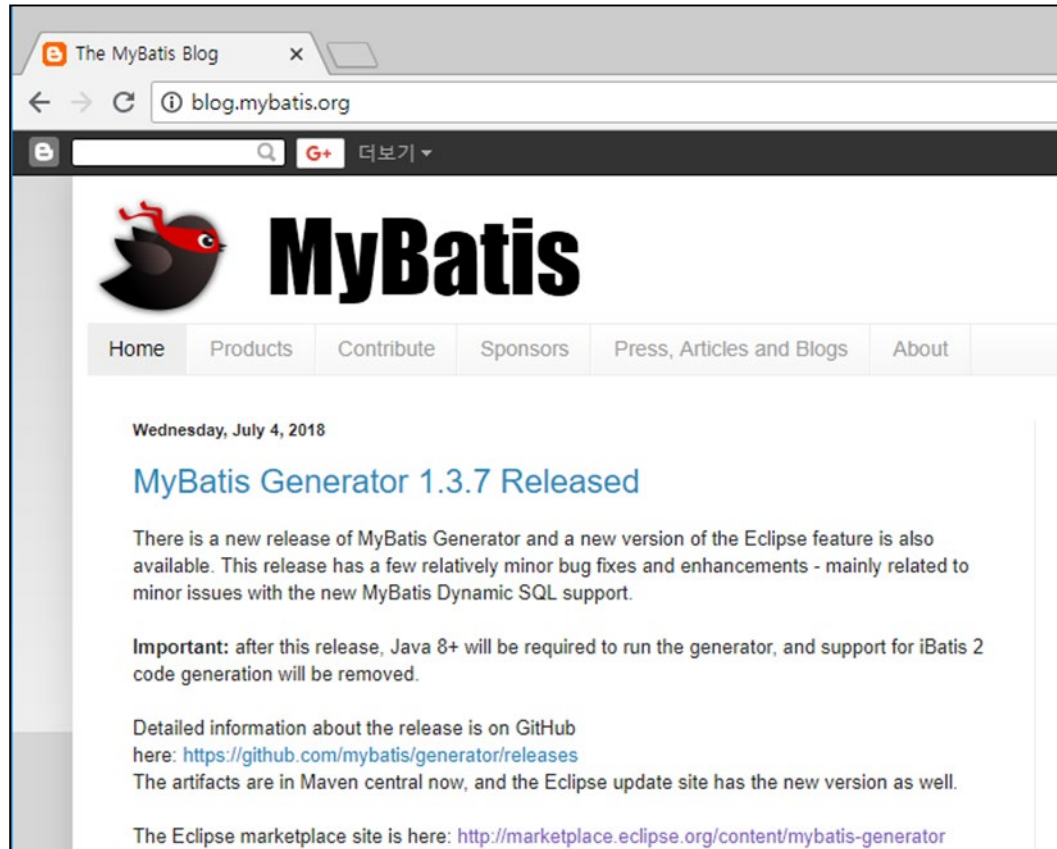
23.1 마이바티스란?



- ① SqlMapConfig.xml에 각 기능별로 실행할 SQL문을 SqlMap.xml에 미리 작성한 후 등록
- ② 애플리케이션에서 데이터베이스와 연동하는 데 필요한 데이터를 각각의 매개변수에 저장한 후 마이바티스에 전달
- ③ 애플리케이션에서 요청한 SQL문을 SqlMap.xml에서 선택
- ④ 전달한 매개변수와 선택한 SQL문을 결합
- ⑤ 매개변수와 결합된 SQL문을 DBMS에서 실행
- ⑥ DBMS에서 반환된 데이터를 애플리케이션에서 제공하는 적당한 매개변수에 저장한 후 반환

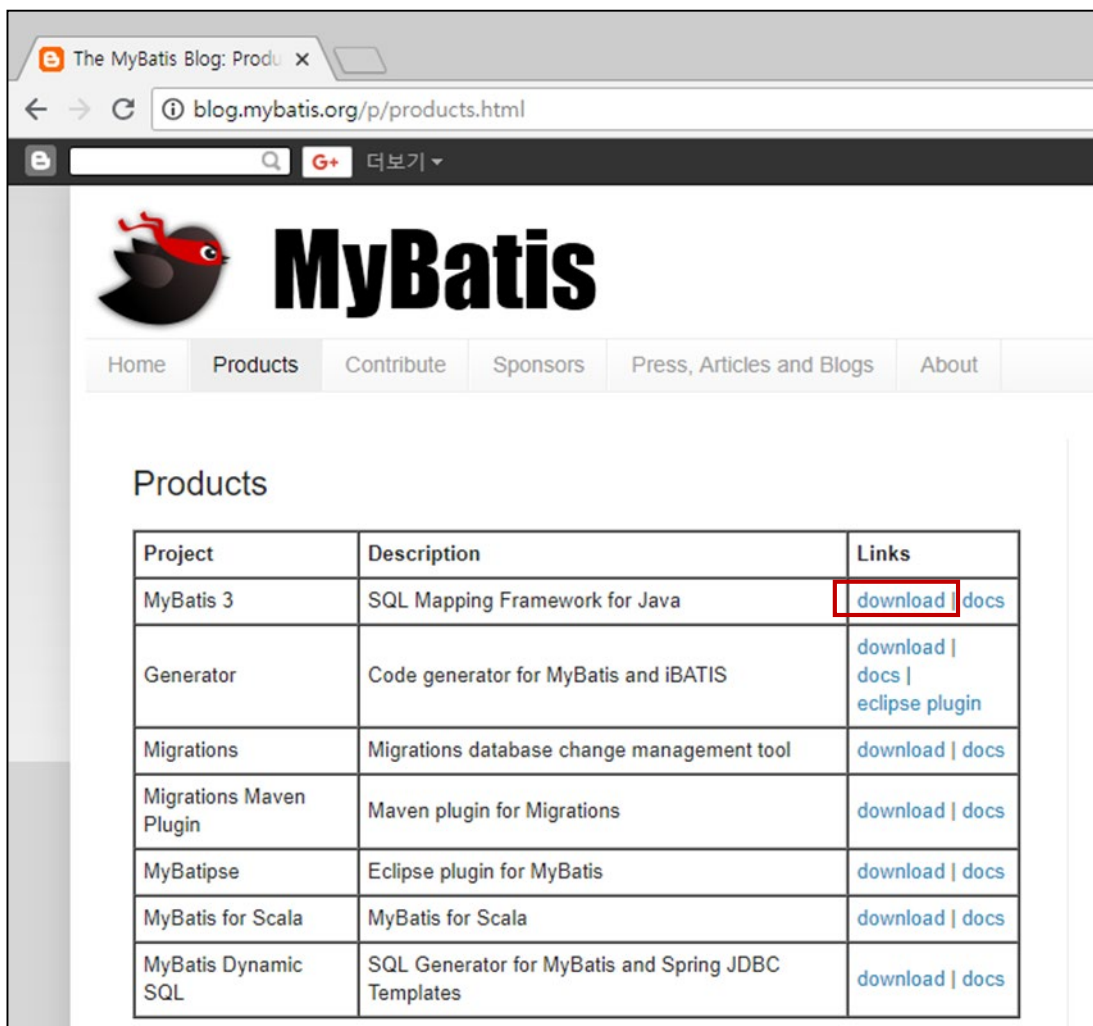
23.2 마이바티스 설치하기

1. <http://www.mybatis.org>에 접속한 후 Products 탭을 클릭합니다.



23.2 마이바티스 설치하기

2. MyBatis3 옆에 있는 download 링크를 클릭합니다.

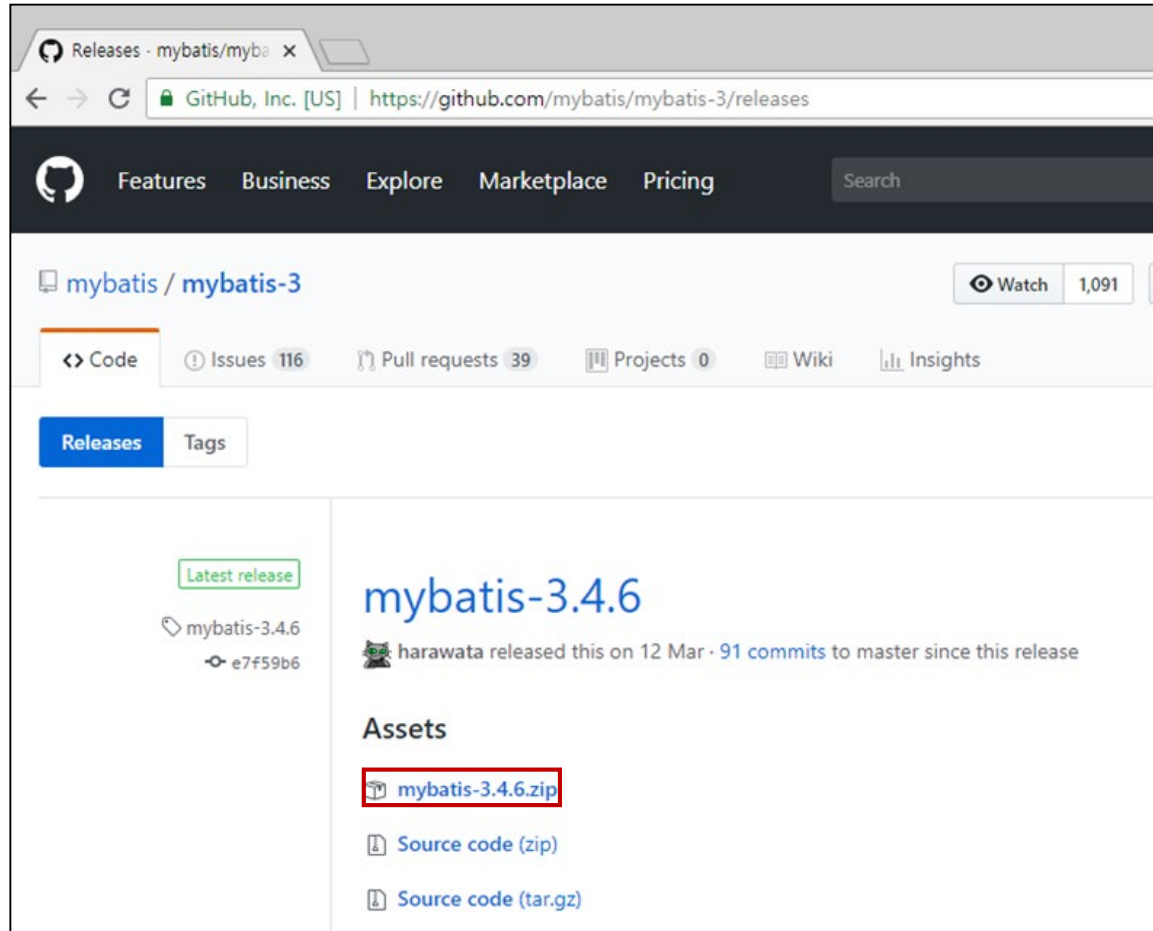


The screenshot shows the MyBatis website's 'Products' page. The browser's address bar displays 'blog.mybatis.org/p/products.html'. The page features the MyBatis logo and a navigation menu with links to Home, Products, Contribute, Sponsors, Press, Articles and Blogs, and About. Below the navigation menu, the 'Products' section contains a table listing various MyBatis projects. The 'download' link for 'MyBatis 3' is highlighted with a red box.

Project	Description	Links
MyBatis 3	SQL Mapping Framework for Java	download docs
Generator	Code generator for MyBatis and iBATIS	download docs eclipse plugin
Migrations	Migrations database change management tool	download docs
Migrations Maven Plugin	Maven plugin for Migrations	download docs
MyBatipse	Eclipse plugin for MyBatis	download docs
MyBatis for Scala	MyBatis for Scala	download docs
MyBatis Dynamic SQL	SQL Generator for MyBatis and Spring JDBC Templates	download docs

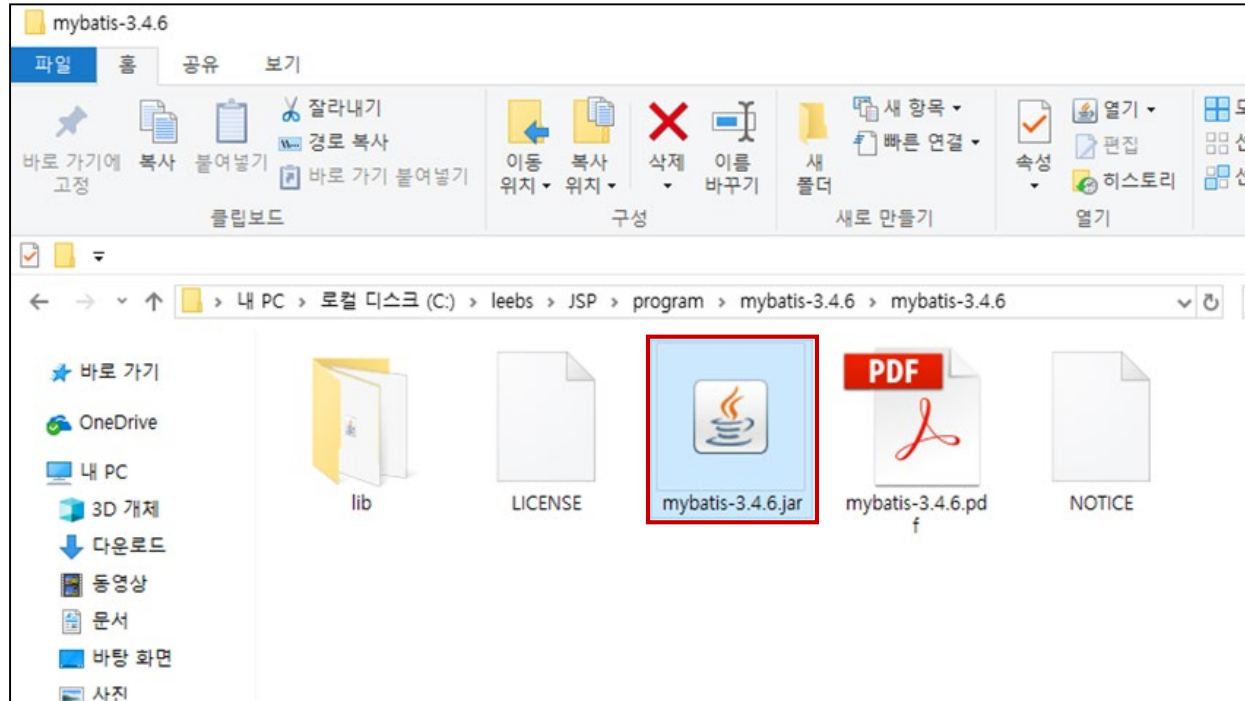
23.2 마이바티스 설치하기

3. mybatis-3.4.6.zip 파일을 클릭해 다운로드합니다.



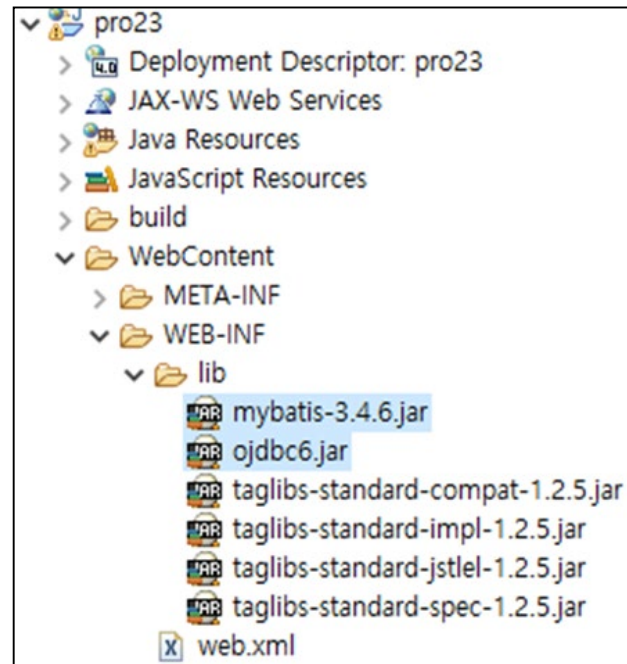
23.2 마이바티스 설치하기

4. 다운로드한 압축 파일의 압축을 해제합니다



23.2 마이바티스 설치하기

5. mybatis-3.4.6.jar를 복사해 새로 만든 프로젝트의 lib 폴더에 붙여 넣습니다. 그리고 이 버전에 대응하는 ojdbc6.jar 파일도 복사해 붙여 넣습니다. JSTL 관련 라이브러리도 14장을 참고하여 복사해 붙여 넣습니다.



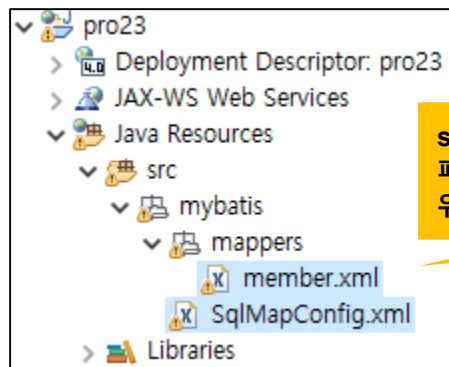
23.3 마이바티스 이용해 회원 기능 실습

• 23.3.1 마이바티스 설정 파일 작성

마이바티스 관련 설정 파일

설정 파일	기능
SqlMapConfig.xml	데이터베이스 연동 시 반환되는 값을 저장할 빈이나 트랜잭션, 데이터소스 등 마이바티스 관련 정보를 설정합니다.
member.xml	회원 정보 관련 SQL문을 설정합니다.

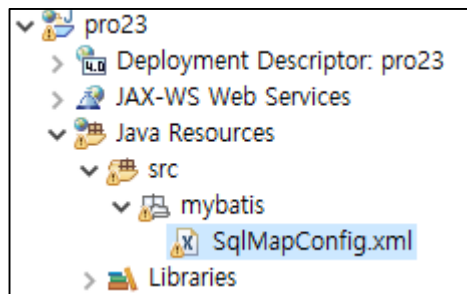
1. 다음과 같이 설정 파일들을 추가합니다. 이때 각 설정 파일은 src 패키지 아래에 위치해야 한다는 것을 잊지 마세요



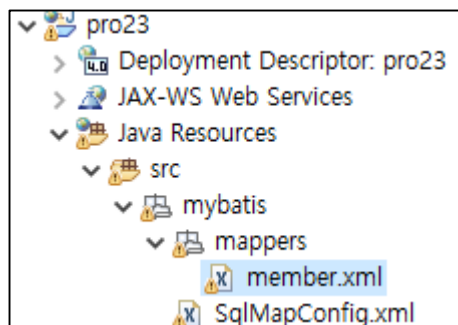
src 패키지 아래에 SqlMapConfig.xml은 mybatis 패키지에, member.xml은 /mybatis/mappers 패키지에 위치합니다.

23.3 마이바티스 이용해 회원 기능 실습

2. 또한 src 아래 mybatis 패키지에서 마우스 오른쪽 버튼을 클릭한 후 New > other > XML File을 선택합니다.
파일 이름을 SqlMapConfig.xml로 입력하여 파일을 생성합니다.



3. 다시 mybatis 패키지를 선택하고 mybatis.mappers 패키지를 만든 다음 mappers 패키지에 member.xml을 생성합니다.



23.3 마이바티스 이용해 회원 기능 실습

4. SqlMapConfig.xml을 다음과 같이 작성합니다.

```
5
6<configuration>
7  <typeAliases>
8    <typeAlias type="com.spring.ex01.MemberVO" alias="memberVO" />
9  </typeAliases>
10 <environments default="development">
11   <environment id="development">
12     <transactionManager type="JDBC" />
13     <dataSource type="POOLED">
14       <property name="driver" value="com.mysql.cj.jdbc.Driver" />
15       <property name="url" value="jdbc:mysql://localhost:3306/servletex" />
16       <property name="username" value="root" />
17       <property name="password" value="1234" />
18     </dataSource>
19   </environment>
20 </environments>
21
22 <mappers>
23   <mapper resource="mybatis/mappers/member.xml"/>
24 </mappers>
25 </configuration>
```

23.3 마이바티스 이용해 회원 기능 실습

5. member.xml을 다음과 같이 작성하여 회원 기능과 관련된 SQL문을 설정합니다.

코드 23-4 pro23/src/mybatis/mappers/member.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">
```

오타 방지를 위해 이 책에서 제공하는
파일에서 복사해 붙여 넣으세요.

```
<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>
```

member.xml의 네임스페이스를 지정합니다.

SQL문을 실행한 후 반환되는 레코드들을
<typeAlias> 태그에서 지정한 memberVO 빈에
저장합니다.

레코드의 컬럼 이름에 대해
memberVO의 같은 속성에
값을 설정합니다.

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

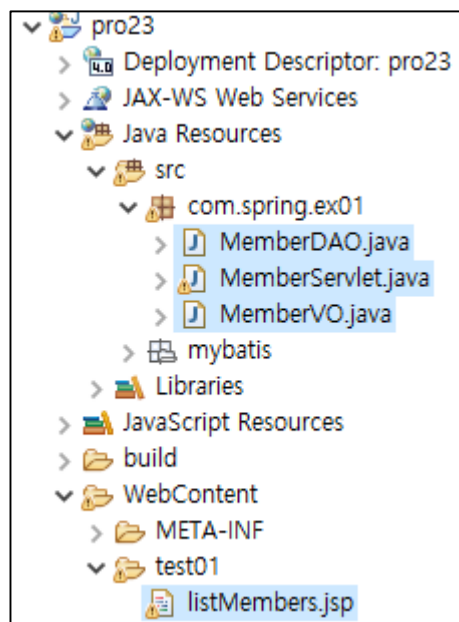
DAO에서 id를 이용해 해당 SQL문을 호출합니다.

반환되는 레코드를 memResult에 저장합니다.

23.3 마이바티스 이용해 회원 기능 실습

- 23.3.2 마이바티스를 이용한 회원 정보 조회 실습

1. 실제 마이바티스와 연동하기 위한 자바 클래스와 JSP를 다음과 같이 준비합니다.



23.3 마이바티스 이용해 회원 기능 실습

2. MemberServlet을 다음과 같이 작성합니다. 브라우저에서 요청 시 MemberDAO 객체를 생성한 후 selectAllMemberList()를 호출하는 서블릿입니다.

코드 23-5 pro23/src/com/spring/ex01/MemberServlet.java

```
package com.spring.ex01;

...

@WebServlet("/mem.do")
public class MemberServlet extends HttpServlet {

protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doHandle(request, response);
}
```

23.3 마이바티스 이용해 회원 기능 실습

```
protected public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException , IOException {
    doHandle(request, response);
}

private void doHandle(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html; charset=utf-8");
    MemberDAO dao = new MemberDAO();
    List membersList = dao.selectAllMemberList();
    request.setAttribute("membersList", membersList);
    RequestDispatcher dispatch =
        request.getRequestDispatcher("test01/listMembers.jsp");
    dispatch.forward(request, response);
}
}
```

MemberDAO 객체를 생성하고
selectAllMemberList()를 호출합니다.

23.3 마이바티스 이용해 회원 기능 실습

3. MemberDAO 클래스를 다음과 같이 작성합니다.

코드 23-6 pro23/src/com/spring/ex01/MemberDAO.java

```
package com.spring.ex01;
...
public class MemberDAO{
    private static SqlSessionFactory sqlMapper=null;
    public static SqlSessionFactory getInstance(){
        if(sqlMapper==null) {
            try{
                String resource = "mybatis/SqlMapConfig.xml";
                Reader reader = Resources.getResourceAsReader(resource);
                sqlMapper=new SqlSessionFactoryBuilder().build(reader);
                reader.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        return sqlMapper;
    }
}
```

MemberDAO의 각 메서드 호출 시 src/mybatis/SqlMapConfig.xml에서 설정 정보를 읽은 후 데이터베이스와의 연동 준비를 합니다.

마이바티스를 이용하는 sqlMapper 객체를 가져옵니다.

23.3 마이바티스 이용해 회원 기능 실습

```
public List<MemberVO> selectAllMemberList(){
    sqlMapper=getInstance();
    SqlSession session=sqlMapper.openSession();
    List<MemberVO> memlist=null;
    memlist=session.selectList("mapper.member.selectAllMemberList");
    return memlist;
}
}
```

실제 member.xml의 SQL문을 호출하는 데 사용되는 SqlSession 객체를 가져옵니다.

여러 개의 레코드를 조회하므로 selectList() 메서드에 실행하고자 하는 SQL문의 id를 인자로 전달합니다.

23.3 마이바티스 이용해 회원 기능 실습

4. MemberVO 클래스를 다음과 같이 작성합니다. SQL문으로 전달할 값이나 SQL문을 실행한 후 반환되는 레코드들의 값을 각 속성에 저장합니다.

코드 23-7 pro23/src/com/spring/ex01/MemberVO.java

```
package com.spring.ex01;

...

public class MemberVO {
    private String id;
    private String pwd;
    private String name;
    private String email;
    private Date joinDate;

    public MemberVO() {
    }

    public MemberVO(String id, String pwd, String name, String email) {
        this.id = id;
        this.pwd = pwd;
        this.name = name;
        this.email = email;
    }

    // 각 속성에 대한 getter와 setter
    ...
}
```

23.3 마이바티스 이용해 회원 기능 실습

5. 회원 정보를 표시하는 listMembers.jsp는 22장의 listMembers.jsp를 복사해서 사용합니다.

코드 23-8 pro23/WebContent/test01/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>  
  </tr>  
</c:forEach>  
...
```

23.3 마이바티스 이용해 회원 기능 실습

6. <http://localhost:8080/pro23/mem.do>로 요청하여 실행 결과를 확인합니다.



The screenshot shows a web browser window with the address bar displaying `localhost:8090/pro23/mem.do`. The page content is a table with 5 columns: 아이디 (ID), 비밀번호 (Password), 이름 (Name), 이메일 (Email), and 가입일 (Join Date). The table contains 6 rows of member data.

아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13
park2	1234	박지성	park2@test.com	2018-09-10
park	1234	박찬호	park@test.com	2018-09-04
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원가입](#)

23.4 마이바티스 이용해 회원 정보 CRUD 실습

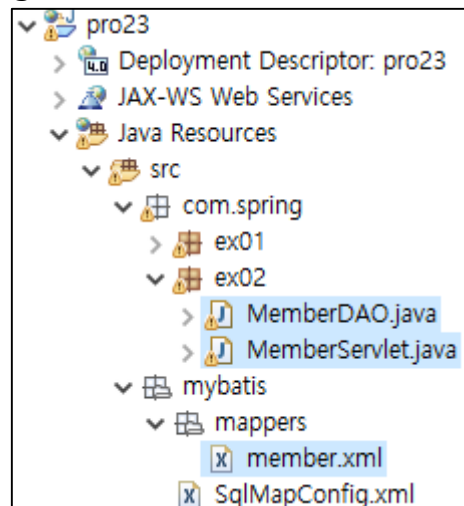
SqlSession 클래스에서 제공하는 여러 가지 메서드

메서드	기능
List selectList(query_id)	id에 대한 select문을 실행한 후 여러 레코드를 List로 반환합니다.
List selectList(query_id, 조건)	id에 대한 select문을 실행하면서 사용되는 조건도 전달합니다.
T selectOne(query_id)	id에 대한 select문을 실행한 후 지정한 타입으로 한 개의 레코드를 반환합니다.
T selectOne(query_id, 조건)	id에 대한 select문을 실행하면서 사용되는 조건도 전달합니다.
Map<K,V> selectMap(query_id, 조건)	id에 대한 select문을 실행하면서 사용되는 조건도 전달합니다. Map 타입으로 레코드를 반환합니다.
int insert(query_id, Object obj)	id에 대한 insert문을 실행하면서 obj 객체의 값을 테이블에 추가합니다.
int update(query_id, Object obj)	obj 객체의 값을 조건문의 수정 값으로 사용해 id에 대한 update문을 실행합니다.
int delete(query_id, Object obj)	obj 객체의 값을 조건문의 조건 값으로 사용해 id에 대한 delete문을 실행합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

- 23.4.1 회원의 ID와 비밀번호 조회

1. 우선 다음과 같이 com.spring.ex02 패키지를 만들고 필요한 실습 파일을 준비합니다.



23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. member.xml에 다음과 같이 SQL문을 작성합니다.

코드 23-9 pro23/src/mybatis/mappers/member.xml

```

...
<mapper namespace="mapper.member">
    ...
    <select id="selectName" resultType="String">
        <![CDATA[
            select name from t_member
            where id = 'hong'
        ]]>
    </select>
    <select id="selectPwd" resultType="int">
        <![CDATA[
            select pwd from t_member
            where id= 'hong'
        ]]>
    </select>
</mapper>

```

MemberDAO에서 접근 시 사용할 SQL문의 id를 지정합니다.

resultType 속성을 문자열로 지정해 SQL문으로 조회한 이름(문자열)을 호출한 메서드로 반환합니다.

MemberDAO에서 접근 시 사용할 SQL문의 id를 지정합니다.

resultType 속성을 int로 지정해 SQL문으로 조회한 정수를 호출한 메서드로 반환합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. MemberServlet을 다음과 같이 작성합니다.

코드 23-10 pro23/src/com/spring/ex02/MemberServlet.java

...

```
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html; charset=utf-8");
    MemberDAO dao = new MemberDAO();
    String name = dao.selectName();
    //int pwd = dao.selectPwd();
    PrintWriter pw = response.getWriter();
    pw.write("<script>");
    pw.write("alert(' 이름: " + name + "')");
    //pw.write("alert(' 비밀번호 : " + pwd + "')");
    pw.write("</script>");
}
```

MemberDAO의 selectName() 메서드를 호출합니다.

MemberDAO의 selectPwd() 메서드를 호출합니다.

조회한 이름을 브라우저로 출력합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. MemberDAO 클래스를 다음과 같이 작성합니다.

코드 23-11 pro23/src/com/spring/ex02/MemberDAO.java

```
...
public class MemberDAO {
    public static SqlSessionFactory sqlMapper = null;
    private static SqlSessionFactory getInstance() {
        if (sqlMapper == null) {
            try {
                String resource = "mybatis/SqlMapConfig.xml";
                Reader reader = Resources.getResourceAsReader(resource);
                sqlMapper = new SqlSessionFactoryBuilder().build(reader);
                reader.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return sqlMapper;
    }
}
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

```
public String selectName() {  
    sqlMapper = getInstance();  
    SqlSession session = sqlMapper.openSession();  
    String name = session.selectOne("mapper.member.selectName");  
    return name;  
}
```

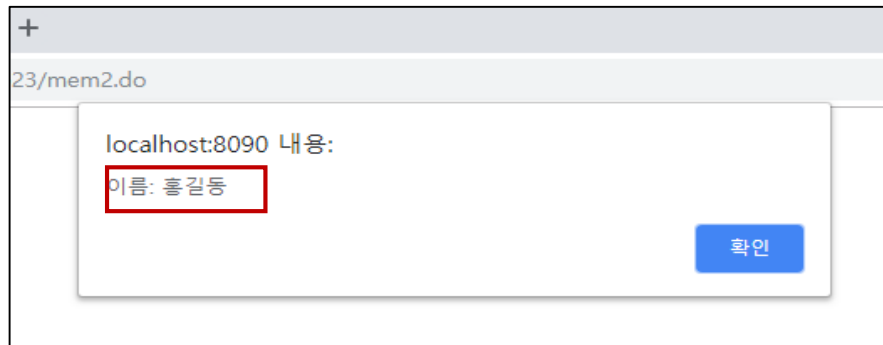
selectOne() 메서드로 인자로 지정한 SQL문을 실행한
후 한 개의 데이터(문자열)를 반환합니다.

```
public int selectPwd() {  
    sqlMapper = getInstance();  
    SqlSession session = sqlMapper.openSession();  
    int pwd = session.selectOne("mapper.member.selectPwd");  
    return pwd;  
}
```

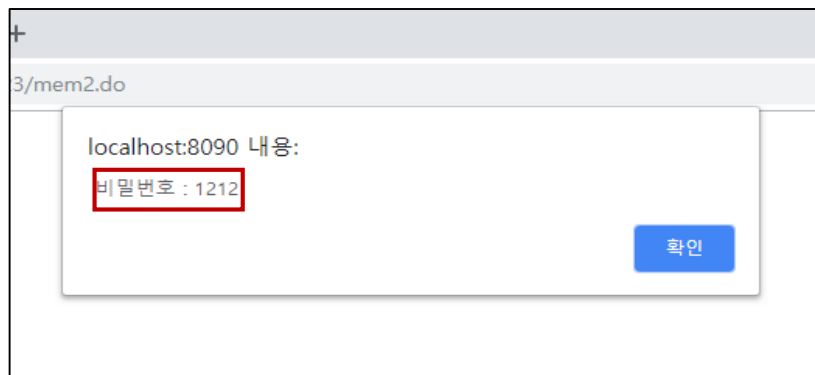
selectOne() 메서드로 지정한 SQL문을 실행한 후
한 개의 데이터(정수)를 반환합니다

23.4 마이바티스 이용해 회원 정보 CRUD 실습

5. `http://localhost:8080/pro23/mem2.do`로 요청합니다. 서블릿에서 `selectName()` 메서드로 조회한 경우 아이디에 해당하는 회원 이름을 알림창으로 출력합니다.



6. 서블릿에서 `selectPwd()` 메서드로 조회한 경우 아이디에 해당하는 비밀번호를 알림창으로 출력합니다.



23.4 마이바티스 이용해 회원 정보 CRUD 실습

• 23.4.2 HashMap을 이용한 모든 회원 정보 조회

1. member.xml을 다음과 같이 수정합니다.

코드 23-12 pro23/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>
  -->
  <resultMap id="memResult" type="java.util.HashMap">
    <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>
</mapper>
```

type 속성을 memberVO로 설정하는 <resultMap>은 주석 처리합니다.

조회한 레코드를 지정한 컬럼 이름을 key, 값을 value로 해서 저장합니다.

조회한 회원 정보를 HashMap에 저장합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. 브라우저에서 서블릿에 요청하면 selectAllMemberList() 메서드를 호출하여 조회한 회원 정보를 바인딩한 후 listMembers.jsp로 포워딩합니다.

코드 23-13 pro23/src/com/spring/ex01/MemberServlet.java

```
...
@WebServlet("/mem.do")
public class MemberServlet extends HttpServlet {
    ...
    private void doHandle(HttpServletRequest request, HttpServletResponse response)
        throws ServletException ,IOException {
        MemberDAO dao = new MemberDAO();
        List membersList = dao.selectAllMemberList();
        request.setAttribute("membersList", membersList);
        RequestDispatcher dispatch = request.getRequestDispatcher("test01/listMembers.jsp");
        dispatch.forward(request, response);
    }
}
```

조회한 회원 정보를 List에 저장합니다.

조회한 회원 정보를 바인딩하고 JSP로 포워딩합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. MemberDAO 클래스에서는 selectList() 메서드를 호출하면서 id가 selectAllMemberList인 SQL문을 실행하도록 다음과 같이 코드를 수정합니다.

코드 23-14 pro23/src/com/spring/ex01/MemberDAO.java

```
...
public class MemberDAO {
    ...
    public List<MemberVO> selectAllMemberList() {
        sqlMapper = getInstance();
        SqlSession session = sqlMapper.openSession();
        List<MemberVO> membersList = null;
        membersList = session.selectList("mapper.member.selectAllMemberList");
        return membersList;
    }
}
```

모든 회원 정보를 조회합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. 다음은 실행 결과입니다. 이번에는 조회한 회원 정보를 HashMap에 저장해서 표시합니다.



The screenshot shows a web browser window with the address bar displaying 'localhost:8090/pro23/mem.do'. The page title is '회원 정보 출력창'. Below the browser window, a table displays member information with the following columns: 아이디 (ID), 비밀번호 (Password), 이름 (Name), 이메일 (Email), and 가입일 (Registration Date). The table contains six rows of data.

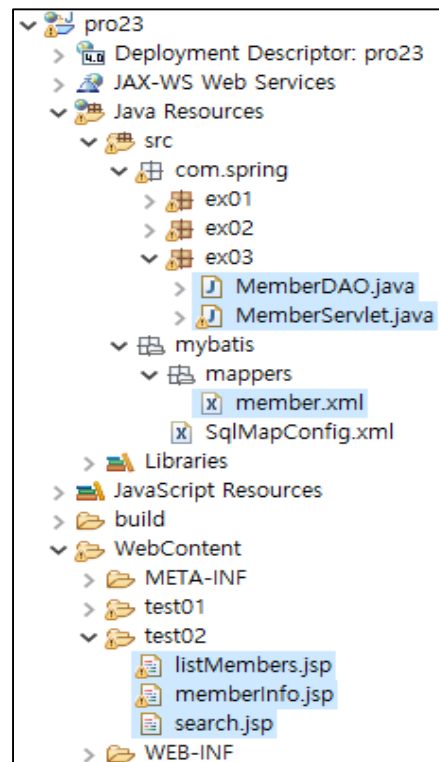
아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13 14:45:48.0
park2	1234	박지성	park2@test.com	2018-09-10 16:30:42.0
park	1234	박찬호	park@test.com	2018-09-04 22:27:18.0
kim	1212	김유신	kim@jweb.com	2018-09-04 21:35:51.0
lee	1212	이순신	lee@test.com	2018-09-04 21:35:48.0
hong	1212	홍길동	hong@gmail.com	2018-09-04 21:35:46.0

[회원가입](#)

23.4 마이바티스 이용해 회원 정보 CRUD 실습

- 23.4.3 조건 값으로 회원 정보 조회

1. 다음과 같이 실습 파일을 준비합니다.



23.4 마이바티스 이용해 회원 정보 CRUD 실습

마이바티스로 조건값 전달 방법

- MemberDAO에서 메서드 호출 시 전달된 조건 값은 매개변수 이름으로 SQL문의 조건식에 전달

SQL문에서 조건값 사용 방법

- #{**전달된 매개변수이름**}

2. member.xml을 다음과 같이 편집합니다.

코드 23-15 pro23/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>
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

```
<select id="selectAllMemberList" resultMap="memResult" >
```

```
<![CDATA[
```

```
select * from t_member order by joinDate desc
```

```
]]>
```

```
</select>
```

```
<select id="selectMemberById" resultType="memberVO" parameterType="String" >
```

```
<![CDATA[
```

```
select * from t_member
```

```
where
```

```
id= #{id}
```

```
]]>
```

```
</select>
```

```
<select id="selectMemberByPwd" resultMap="memResult" parameterType="int" >
```

```
<![CDATA[
```

```
select * from t_member
```

```
where
```

```
pwd = #{pwd}
```

```
]]>
```

```
</select>
```

```
...
```

```
</mapper>
```

MemberDAO에서 호출하는 id를 지정합니다.

조회되는 한 개의 레코드를 memberVO에 저장합니다.

MemberDAO에서 SQL문 호출 시 전달되는 매개변수의 데이터 타입을 지정합니다.

MemberDAO에서 메시지를 호출하면서 parameterType으로 전달된 매개변수 이름을 select문의 id의 조건 값으로 사용합니다

SQL문 실행 시 매개변수 이름을 pwd의 조건 값으로 사용합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. 서블릿에서는 브라우저의 요청에 대해 MemberDAO 클래스의 메서드를 호출한 후 그 결과를 브라우저로 출력합니다.

코드 23-16 pro23/src/com/spring/ex03/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    MemberDAO dao=new MemberDAO();
    MemberVO memberVO=new MemberVO();
    String action=request.getParameter("action");
    String nextPage="";
    if(action== null || action.equals("listMembers")){
        List membersList=dao.selectAllMemberList();
        request.setAttribute("membersList", membersList);
        nextPage="test02/listMembers.jsp";
    }
}
```

코드 23-18 pro23/WebContent/test02/search.jsp

```
...
<title>회원 검색창</title>
</head>
<body>
    <form action="${pageContext.request.contextPath}/mem3.do">
        입력 : <input type="text" name="value"/>
        <select name="action">
            <option value="listMembers" >전체</option>
            <option value="selectMemberById" >아이디</option>
            <option value="selectMemberByPwd">비밀번호</option>
        </select> <br>
        <input type="submit" value="검색" />
    </form>
</body>
</html>
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

```
}else if(action.equals("selectMemberById")){
    String id=request.getParameter("value");
    memberVO=dao.selectMemberById(id);
    request.setAttribute("member",memberVO);
    nextPage="test02/memberInfo.jsp";
```

검색 조건이 selectMemberById이면 전송된 값을 getParameter()로 가져온 후 SQL문의 조건식에서 id의 조건 값으로 전달합니다.

```
}else if(action.equals("selectMemberByPwd")){
    int pwd =Integer.parseInt(request.getParameter("value"));
    List<MemberVO> membersList=dao.selectMemberByPwd(pwd);
    request.setAttribute("membersList",membersList);
    nextPage="test02/listMembers.jsp";
}
```

검색 조건이 selectMemberBy

Pwd0 코드 23-18 pro23/WebContent/test02/search.jsp

```
getPa ...
SQL문 <title>회원 검색창</title>
값으로 </head>
<body>
<form action="${pageContext.request.contextPath}/mem3.do">
    입력 : <input type="text" name="value"/>
    <select name="action">
        <option value="listMembers" >전체</option>
        <option value="selectMemberById" >아이디</option>
        <option value="selectMemberByPwd">비밀번호</option>
    </select> <br>
    <input type="submit" value="검색" />
</form>
</body>
</html>
```

```
RequestDispatcher dispatch = request.getRequestDispatcher(nextPag
dispatch.forward(request, response);
```

```
}
```

```
}
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. selectOne() 메서드는 하나의 레코드를 조회할 때 사용합니다. selectOne() 메서드의 두 번째 인자는 첫 번째 인자의 SQL문에서 매개변수 이름 id로 조건 값을 전달합니다.

코드 23-17 pro23/src/com/spring/ex03/MemberDAO.java

```
...
public class MemberDAO{
    ...
    public MemberVO selectMemberById(String id){
        sqlMapper=getInstance();
        SqlSession session=sqlMapper.openSession();
        MemberVO memberVO=session.selectOne("mapper.member.selectMemberById", id);
        return memberVO;
    }

    public List<MemberVO> selectMemberByPwd(int pwd) {
        sqlMapper = getInstance();
        SqlSession session = sqlMapper.openSession();
        List<MemberVO> membersList = null;
        membersList= session.selectList("mapper.member.selectMemberByPwd", pwd);
        return membersList;
    }
    ...
}
```

서블릿에서 넘어온 id의 값을 selectOne() 메서드 호출 시
해당 SQL문의 조건 값으로 전달합니다.

레코드 한 개만 조회할 때 사용합니다.

정수 데이터인 pwd를 SQL문의
조건 값으로 전달합니다.

비밀번호가 같은 회원은 여러 명이 있을 수 있으므로
selectList() 메서드로 조회합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

5. search.jsp를 다음과 같이 작성합니다. 검색창에 입력한 값과 셀렉트 박스의 검색 조건을 선택해 서블릿으로 전송합니다.

코드 23-18 pro23/WebContent/test02/search.jsp

```
...
<title>회원 검색창</title>
</head>
<body>
<form action="${pageContext.request.contextPath}/mem3.do">
  입력 : <input type="text" name="value"/>
  <select name="action">
    <option value="listMembers" >전체</option>
    <option value="selectMemberById" >아이디</option>
    <option value="selectMemberByPwd">비밀번호</option>
  </select> <br>
  <input type="submit" value="검색" />
</form>
</body>
</html>
```

검색할 값을 입력합니다.

셀렉트 박스의 검색 조건을 선택합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

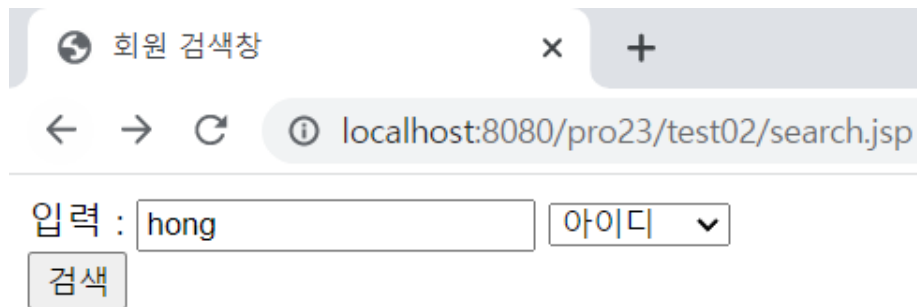
6. memberInfo.jsp에서는 검색 조건으로 조회한 회원 정보를 출력합니다.

코드 23-19 pro23/WebContent/test02/memberInfo.jsp

```
...
<table border="1" align="center" width="100%" >
  <tr align="center" bgcolor="lightgreen">
    <td ><b>아이디</b></td>
    <td><b>비밀번호</b></td>
    <td><b>이름</b></td>
    <td><b>이메일</b></td>
    <td><b>가입일</b></td>
  </tr>
  <tr align="center">
    <td>${member.id}</td>
    <td>${member.pwd}</td>
    <td>${member.name}</td>
    <td>${member.email}</td>
    <td>${member.joinDate}</td>
  </tr>
</table>
...
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

7. <http://localhost:8090/pro23/test02/search.jsp>로 요청하여 셀렉트 박스에서 아이디를 선택하여 검색 조건을 설정합니다. 입력 칸에 park이라고 입력한 후 검색을 클릭합니다.

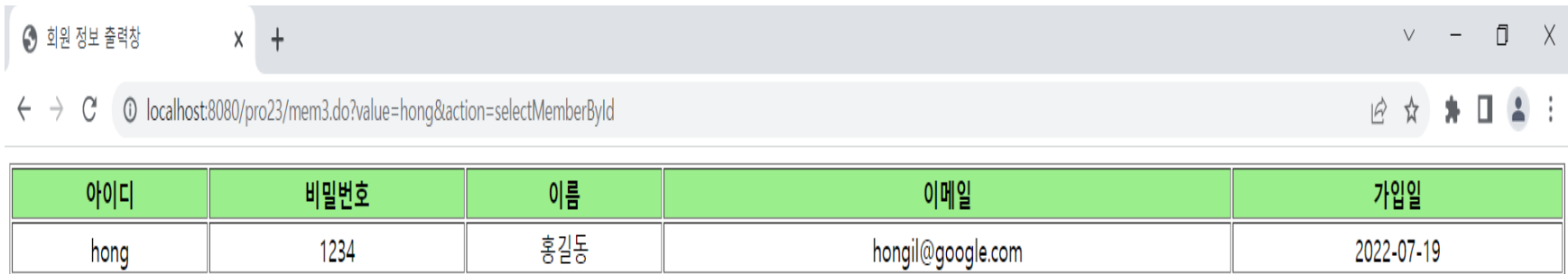


회원 검색창

← → ↻ ⓘ localhost:8080/pro23/test02/search.jsp

입력 : 아이디 ▼

8. 조회된 결과가 memberInfo.jsp에 표시됩니다.



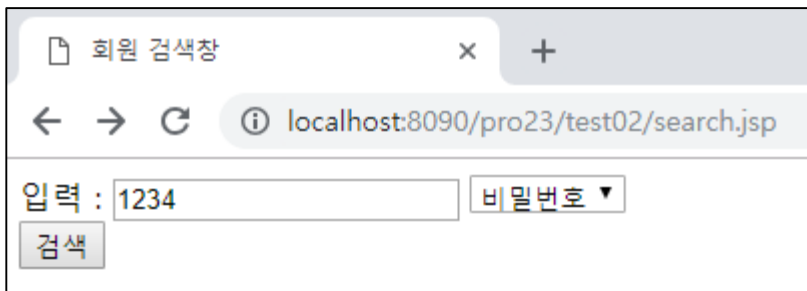
회원 정보 출력창

← → ↻ ⓘ localhost:8080/pro23/mem3.do?value=hong&action=selectMemberById

아이디	비밀번호	이름	이메일	가입일
hong	1234	홍길동	hongil@google.com	2022-07-19

23.4 마이바티스 이용해 회원 정보 CRUD 실습

9. 이번에는 셀렉트 박스에서 검색 조건을 비밀번호로 설정한 후 검색을 클릭합니다. 그러면 입력한 값과 같은 비밀번호를 가지는 회원 정보가 모두 표시됩니다.

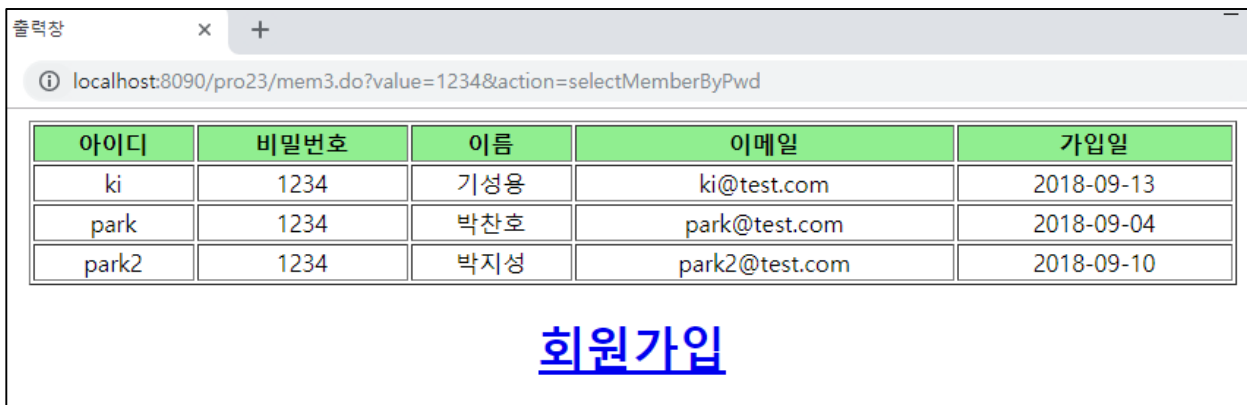


회원 검색창

← → ↻ ⓘ localhost:8090/pro23/test02/search.jsp

입력 : 1234 비밀번호 ▾

검색



출력창

ⓘ localhost:8090/pro23/mem3.do?value=1234&action=selectMemberByPwd

아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13
park	1234	박찬호	park@test.com	2018-09-04
park2	1234	박지성	park2@test.com	2018-09-10

[회원가입](#)

23.4 마이바티스 이용해 회원 정보 CRUD 실습

10. 검색 조건을 전체로 설정해서 요청하면 전체 회원 정보가 표시됩니다.

출력창 x +

localhost:8090/pro23/mem3.do?value=&action=listMembers

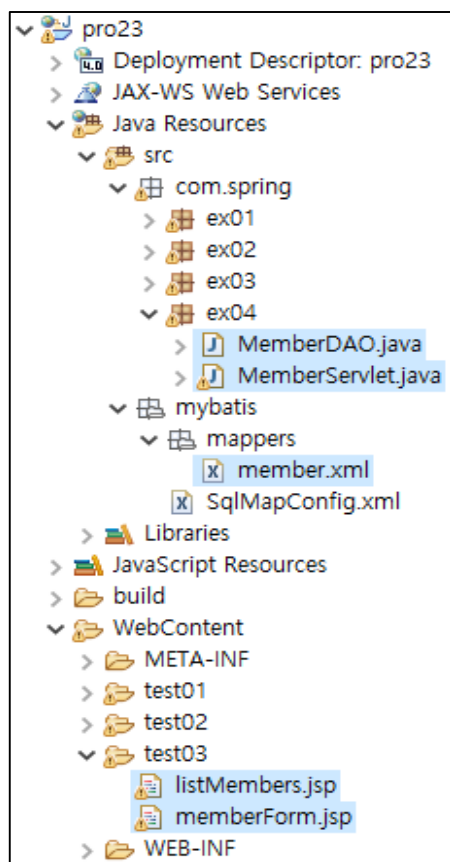
아이디	비밀번호	이름	이메일	가입일
ki	1234	기성용	ki@test.com	2018-09-13
park2	1234	박지성	park2@test.com	2018-09-10
park	1234	박찬호	park@test.com	2018-09-04
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원가입](#)

23.4 마이바티스 이용해 회원 정보 CRUD 실습

- 23.4.4 회원 정보 추가

1. 다음과 같이 com.spring.ex04 패키지를 만들고 실습 파일을 준비합니다.



23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. member.xml을 다음과 같이 작성합니다.

코드 23-20 pro23/src/mybatis/mappers/member.xml

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

MemberDAO에서 회원 정보를 memberVO의 속성에 저장해서 넘깁니다.

memberVO의 속성 이름에 저장된 값을 value로 설정합니다.

23.4 마이바티스 이용해 회원 정보 CRUD

실습

3. 브라우저에서 전송된 action 값이 insertMember면 함께 전송된 회원 정보를 가져와 MemberVO 객체에 설정합니다.

코드 23-21 pro23/src/com/spring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    ...
    } else if(action.equals("insertMember")) {
        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);
        dao.insertMember(memberVO);
        nextPage="/mem4.do?action=listMembers";
    }
    RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
    dispatch.forward(request, response);
}
...
```

회원 가입창에서 전송된 회원 정보를 MemberVO에 설정한 후 insertMember() 메서드로 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. MemberDAO 클래스에서 insert문을 사용하려면 SqlSession 클래스의 insert() 메서드를 이용해야 합니다.

코드 23-22 pro23/src/com/spring/ex04/MemberDAO.java

```
...
public class MemberDAO {
    ...
    public int insertMember(MemberVO memberVO) {
        sqlMapper = getInstance();
        SqlSession session = sqlMapper.openSession();
        int result = 0;
        result = session.insert("mapper.member.insertMember", memberVO);
        session.commit();
        return result;
    }
}
```

지정한 id의 SQL문에 memberVO의 값을 전달하여 회원 정보를 테이블에 추가합니다.

수동 커밋이므로 반드시 commit() 메서드를 호출하여 영구 반영합니다.

```
memberVO.setId(id);
memberVO.setPwd(pwd);
memberVO.setName(name);
memberVO.setEmail(email);
dao.insertMember(memberVO);
```

23.4 마이바티스 이용해 회원 정보 CRUD 실습

5. 회원 가입창에서 회원 정보를 전송하면 action 값으로 insertMember를 전달합니다.

코드 23-23 pro23/WebContent/test03/memberForm.jsp

...

```
<form method="post" action="${ contextPath}/mem4.do?action=insertMember">
```

```
<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>
```

...

action 값으로 insertMember를
MemberServlet으로 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

6. <http://localhost:8090/pro23/test03/memberForm.jsp>로 요청하여 회원 정보를 입력하고 가입하기를 클릭하면 새 회원 정보가 추가되고 회원 목록이 표시됩니다.

회원 가입창

아이디

비밀번호

이름

이메일

아이디	비밀번호	이름	이메일	가입일
jspark	1234	박지성	jspark@test.com	2018-09-29
ki	1234	기성용	ki@test.com	2018-09-13
park	1234	박찬호	park@test.com	2018-09-04
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

회원가입

23.4 마이바티스 이용해 회원 정보 CRUD 실습

• 23.4.5 HashMap을 이용한 회원 정보 추가

1. member.xml을 다음과 같이 수정합니다. insert문의 parameterType을 HashMap으로 지정합니다. 회원 정보들은 HashMap의 key를 이용해 가져옵니다.

코드 23-24 pro23/src/mybatis/mappers/member.xml

...

```
<insert id="insertMember2" parameterType="java.util.HashMap">
```

```
<![CDATA[
```

```
insert into t_member(id,pwd, name, email)
```

```
values(#{id}, #{pwd}, #{name}, #{email})
```

```
]]>
```

```
</insert>
```

...

MemberDAO에서 회원 정보를
HashMap에 담아서 전달합니다.

HashMap에 각각의 key로 저장된 value를
가져와 테이블에 추가합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. MemberServlet 클래스를 다음과 같이 작성합니다. 브라우저에서 전달된 회원 정보를 HashMap에 key/value로 저장한 후 MemberDAO의 insertMember2() 메서드로 전달합니다.

코드 23-25 pro23/src/com/spring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    ...
    } else if(action.equals("insertMember2")) {
        String id=request.getParameter("id");
        String pwd=request.getParameter("pwd");
        String name=request.getParameter("name");
        String email = request.getParameter("email");

        Map memberMap=new HashMap();
        memberMap.put("id", id);
        memberMap.put("pwd", pwd);
        memberMap.put("name", name);
        memberMap.put("email", email);
        dao.insertMember2(memberMap);
        nextPage="/mem4.do?action=listMembers";
    }
    RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
    dispatch.forward(request, response);
}
...
```

회원 가입창에서 전송된 회원 정보를 HashMap에 key/value로 저장한 후 MemberDAO의 insertMember2() 인자로 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. 이번에는 SqlSession 클래스의 insert() 메서드 호출 시 두 번째 인자로 HashMap을 전달합니다.

코드 23-26 pro23/src/com/spring/ex04/MemberDAO.java

```
...
public int insertMember2(Map<String,String> memberMap){
    sqlMapper=getInstance();
    SqlSession session=sqlMapper.openSession();
    int result= session.insert("mapper.member.insertMember2", memberMap);
    session.commit();
    return result;
}
...
```

메서드로 전달된 HashMap을
다시 SQL문으로 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. memberForm.jsp의 <form> 태그의 action 속성을 \${contextPath}/mem4.do?action=insertMember2로 변경합니다.
5. http://localhost:8080/pro23/test03/memberForm.jsp로 요청하여 회원 정보를 입력하고 가입하기를 클릭한 후 회원 정보를 등록하고 결과를 확인합니다.

회원 가입창

아이디

cha

비밀번호

....

이름

차범근

이메일

cha@test.com

가입하기

다시입력

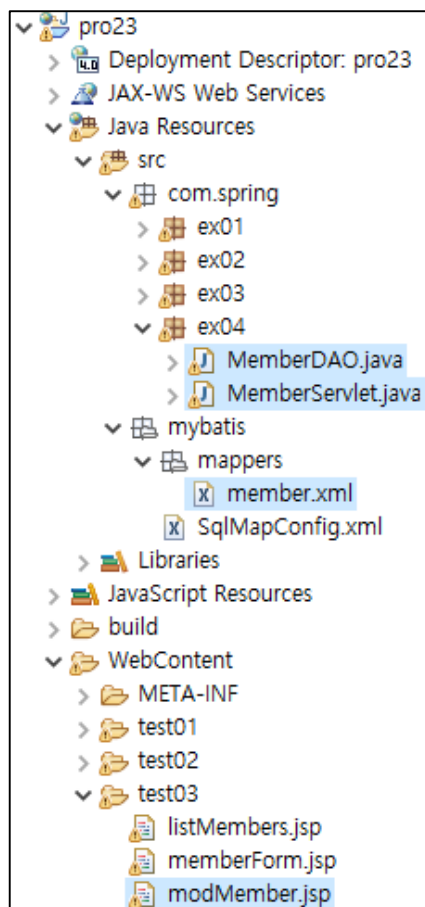
아이디	비밀번호	이름	이메일	가입일
cha	1212	차범근	cha@test.com	2018-11-22
ki	1234	기성용	ki@test.com	2018-09-13
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

회원가입

23.4 마이바티스 이용해 회원 정보 CRUD 실습

- 23.4.6 회원 정보 수정

1. 회원 정보 수정에 필요한 modMember.jsp를 추가합니다.



23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. member.xml을 다음과 같이 수정합니다.

코드 23-27 pro23/src/mybatis/mappers/member.xml

```
...  
<update id="updateMember" parameterType="memberVO">  
  <![CDATA[  
    update t_member  
    set pwd=#{pwd}, name=#{name}, email=#{email}  
    where  
    id=#{id}  
  ]]>  
</update>  
...
```

SQL문에 사용될 데이터를 memberVO 빈에
설정해 전달합니다.

memberVO 빈의 속성 값을 각 컬럼의
수정 값으로 설정합니다.

memberVO 빈의 id 속성 값을
조건 값으로 사용합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. MemberServlet을 다음과 같이 작성합니다.

코드 23-28 pro23/src/com/spring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException , IOException {
    ...
    } else if(action.equals("updateMember")){
        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);
        dao.updateMember(memberVO);
        nextPage="/mem4.do?action=listMembers";
    }
    RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
    dispatch.forward(request, response);
}
...
```

회원 수정창에서 전송된 회원 정보를 MemberVO의 속성에
설정 후 updateMember()를 호출하면서 MemberVO 객체
를 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. MemberDAO에서 SqlSession 클래스의 update() 메서드를 이용해서 update문을 실행하도록 다음과 같이 설정합니다.

코드 23-29 pro23/src/com/spring/ex04/MemberDAO.java

```
...
public int updateMember(MemberVO memberVO) {
    sqlMapper = getInstance();
    SqlSession session = sqlMapper.openSession();
    int result = session.update("mapper.member.updateMember", memberVO);
    session.commit();
    return result;
}
...
```

update문 호출 시 SqlSession의
update() 메서드를 이용합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

5. modMember.jsp를 다음과 같이 작성합니다.

코드 23-30 pro23/WebContent/test03/modMember.jsp

```
...
<form method="post" action="${contextPath}/mem4.do?action=updateMember">
  <h1 class="text_center">회원 정보 수정창</h1>
  <table align="center">
    <tr>
      <td width="200"><p align="right">사용자 아이디</p>
      <td width="400"><input type="text" name="id"></td>
    </tr>
    ....
    ....
    <input type="submit" value="수정하기">
  ...
```

updateMember 서블릿에 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

6. <http://localhost:8090/pro23/test03/modMember.jsp>로 요청하여 ID가 cha인 회원의 수정 정보를 입력하고 수정하기를 클릭하면 수정된 회원 정보가 표시됩니다.

회원 정보 수정창

아이디

비밀번호

이름

이메일

아이디	비밀번호	이름	이메일	가입일
cha	4321	차범근	cha2@test.com	2018-11-22
ki	1234	기정홍	ki@test.com	2018-09-13
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

[회원가입](#)

23.4 마이바티스 이용해 회원 정보 CRUD 실습

• 23.4.7 회원 정보 삭제

1. member.xml에 다음 내용을 추가합니다.

코드 23-31 pro23/src/mybatis/mappers/member.xml

```
...
<delete id="deleteMember" parameterType="String">
  <![CDATA[
    delete from t_member
    where
    id=#{id}
  ]]>
</delete>
...
```

회원 ID는 문자열이므로 parameterType을 String으로 설정합니다.

전달된 ID를 조건 값으로 해당 회원 정보를 삭제합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

2. MemberServlet 클래스는 다음과 같이 작성합니다. 브라우저에서 서블릿으로 전달된 action 값이 deleteMember면 같이 전달된 ID 값을 받아 MemberDAO로 전달합니다.

코드 23-32 pro23/src/com/sring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    ...
    } else if(action.equals("deleteMember")){
        String id=request.getParameter("id");
        dao.deleteMember(id);
        nextPage="/mem4.do?action=listMembers";
    }
    RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
    dispatch.forward(request, response);
}
...
```

회원 ID를 가져옵니다.

회원 목록창에서 전달된 ID를 deleteMember() 메서드를 호출하면서 SQL문으로 전달합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

3. MemberDAO 클래스는 다음과 같이 작성합니다.

코드 23-33 pro23/src/com/sring/ex04/MemberDAO.java

```
...
public int deleteMember(String id) {
    sqlMapper = getInstance();
    SqlSession session = sqlMapper.openSession();
    int result = 0;
    result = session.delete("mapper.member.deleteMember", id);
    session.commit();
    return result;
}
...
```

delete문을 실행하려면 SqlSession의 delete() 메서드를 이용해야 합니다.

SQL문을 실행한 후 반드시 커밋합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

4. listMembers.jsp를 다음과 같이 작성합니다. 회원 목록창에 삭제하기 링크를 추가합니다. 삭제하기를 클릭하면 action 값과 회원의 ID를 서블릿으로 전송하도록 합니다.

코드 23-34 pro23/WebContent/test03/listMembers.jsp

```
...
<table border="1" align="center" width="80%">
  <tr align="center" bgcolor="lightgreen">
    <td><b>아이디</b></td>
    <td><b>비밀번호</b></td>
    <td><b>이름</b></td>
    <td><b>이메일</b></td>
    <td><b>가입일</b></td>
    <td><b>삭제</b></td>
  </tr>
  <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="${contextPath}/mem4.do?action=deleteMember&id=${member.id}">
        삭제하기</a></td>
    </tr>
  </c:forEach>
</table>
...
```

삭제하기 클릭 시 action 값과 회원 ID를
서블릿으로 전송합니다.

23.4 마이바티스 이용해 회원 정보 CRUD 실습

5. <http://localhost:8090/pro23/mem4.do>로 회원 목록을 요청한 후 삭제하기를 클릭합니다.
그러면 해당 회원 정보를 삭제한 후 회원 목록을 표시합니다.

아이디	비밀번호	이름	이메일	가입일	삭제
cha	4321	차범근	cha2@test.com	2018-11-22	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	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

아이디	비밀번호	이름	이메일	가입일	삭제
ki	1234	기성용	ki@test.com	2018-09-13	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

23.5 마이바티스의 동적 SQL 문 사용하기

마이바티스의 동적 SQL 기능 등장 배경

```
select * from t_member ①
```

```
select * from t_member ②
```

```
where
```

```
id = 'hong'
```

```
select * from t_member ③
```

```
where
```

```
id='hong'
```

```
and pwd='1234'
```



공통 SQL문에 대해 조건값의 유무에 따라 동적으로 공통 SQL문에 조건절을 추가

23.5 마이바티스의 동적 SQL문 사용하기

마이바티스의 동적 SQL문의 특징

- 주로 SQL문의 조건절에서 사용
- 조건절(where)에 조건을 동적으로 추가
- JSTL과 XML 기반으로 동적 SQL문을 작성

마이바티스의 동적 SQL문 구성 요소

- if
- choose(when, otherwise)
- trim(where, set)
- foreach

23.5 마이바티스의 동적 SQL문 사용하기

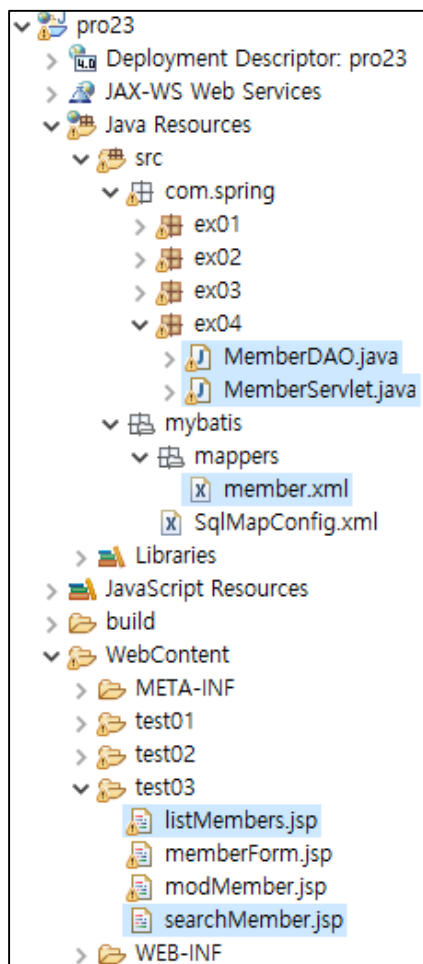
- 23.5.1 <if> 태그로 동적 SQL문 만들기

<if> 태그 사용법

```
<where>  
  <if test='조건식'>  
    추가할 구문  
  </if>  
</where>
```

23.5 마이바티스의 동적 SQL 문 사용하기

1. 다음과 같이 실습 파일을 준비합니다



23.5 마이바티스의 동적 SQL 문 사용하기

2. member.xml에 모든 회원 정보를 조회하는 select문에 대해 where절을 이용한 조건절을 다음과 같이 추가합니다.

코드 23-35 pro23/src/mybatis/mappers/member.xml

```
...
<select id="searchMember" parameterType="memberVO" resultMap="memResult">
  <![CDATA[
    select * from t_member
  ]]>
  <where>
    <if test="name != '' and name != null">
      name = #{name}
    </if>
    <if test="email != '' and email != null">
      and email=#{email}
    </if>
  </where>
  order by joinDate desc
</select>
</mapper>
```

공통 SQL문입니다.

<where> 태그를 이용해 SQL문의 where절을 구성합니다.

name 속성 값을 체크해 공백이 아니거나 null이 아니면 'name=name 속성 값' 조건절을 공통 SQL문 뒤에 추가합니다.

email 속성 값을 체크해 공백과 null이 아니면 'email=email 속성 값' 구문을 공통 SQL문 뒤에 추가합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

3. MemberServlet 클래스를 다음과 같이 작성합니다.

코드 23-36 pro23/src/com/spring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    ...
    } else if(action.equals("searchMember")){
        String name=request.getParameter("name");
        String email=request.getParameter("email");
        memberVO.setName(name);
        memberVO.setEmail(email);
        List membersList =dao.searchMember(memberVO);
        request.setAttribute("membersList",membersList);
        nextPage="test03/listMembers.jsp";
    }
    ...
}
```

검색창에 입력한 검색 조건을 가져옵니다.

23.5 마이바티스의 동적 SQL 문 사용하기

4. MemberDAO 클래스를 다음과 같이 작성합니다.

코드 23-37 pro23/src/com/spring/ex04/MemberDAO.java

```
...  
public List searchMember(MemberVO memberVO){  
    sqlMapper=getInstance();  
    SqlSession session=sqlMapper.openSession();  
    List list=session.selectList("mapper.member.searchMember",memberVO);  
    return list;  
}  
...
```

회원 검색창에서 전달된 이름과 나이 값을 memberVO에
설정하여 SQL문으로 전달합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

5. searchMember.jsp에서는 검색 조건을 입력하고 <hidden> 태그의 action 값을 searchMember로 설정해 서블릿으로 전송합니다.

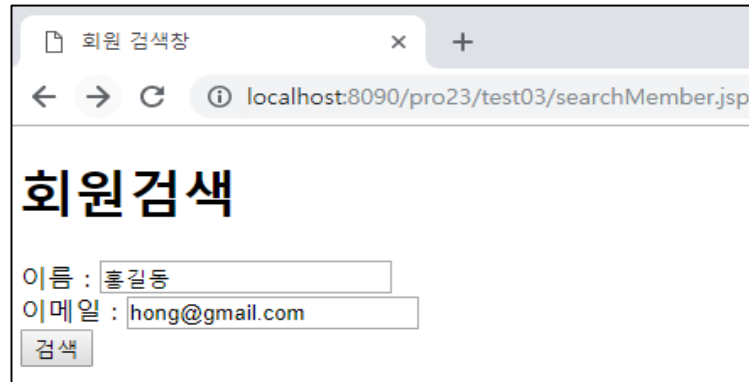
코드 23-38 pro23/WebContent/test03/searchMember.jsp

```
...  
<body>  
  <h1>회원검색</h1>  
  <form action="${contextPath}/mem4.do">  
    <input type="hidden" name="action" value="searchMember" />  
    이름 : <input type="text" name="name" /><br>  
    이메일 : <input type="text" name="email" /><br>  
    <input type="submit" value="검색" />  
  </form>  
</body>  
...
```

_____ <hidden> 태그를 이용해 서블릿으로 action 값을 전달합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

6. <http://localhost:8090/pro23/test03/searchMember.jsp>로 요청하여 이름과 이메일로 조회합니다. 그러면 이름과 이메일을 동시에 만족하는 회원 정보를 출력합니다



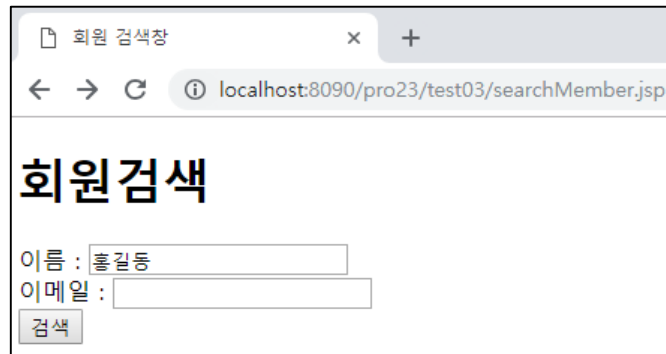
아이디	비밀번호	이름	이메일	가입일
hong	1212	홍길동	hong@gmail.com	2018-09-04

```
select * from t_member
where
name = '홍길동'
and email = 'hong@gmail.com'
```

[회원가입](#)

23.5 마이바티스의 동적 SQL 문 사용하기

7. 이번에는 이름으로만 조회해 보겠습니다. 그러면 이름에 해당하는 회원 정보를 출력합니다.



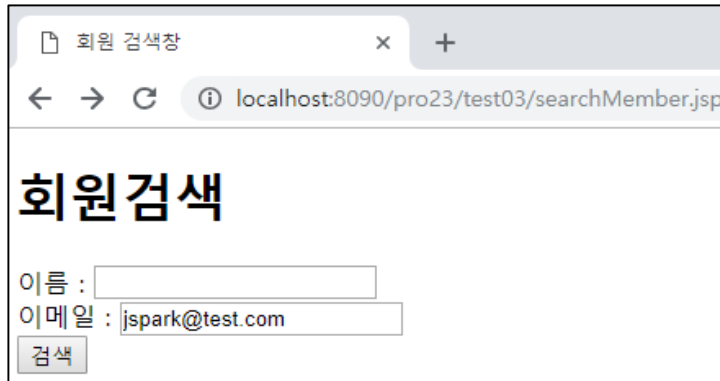
아이디	비밀번호	이름	이메일	가입일자
hong	1212	홍길동	hong@gmail.com	2018-09-04

```
select * from t_member  
where  
name = '홍길동';
```

회원가입

23.5 마이바티스의 동적 SQL 문 사용하기

8. 마찬가지로 이메일로만 조회하면 이메일에 해당하는 모든 회원 정보를 표시합니다.



회원 검색창

localhost:8090/pro23/test03/searchMember.jsp

회원검색

이름 :

이메일 :

아이디	비밀번호	이름	이메일	
jspark	1234	박지성	jspark@test.com	2018-09-29

```
select * from t_member  
where  
email = 'jspark@test.com'
```

회원가입

23.5 마이바티스의 동적 SQL 문 사용하기

9. 마지막으로 회원 검색창에 아무것도 입력하지 않고 검색을 클릭하면 모든 회원 정보를 조회합니다.

아이디	비밀번호	이름	이메일	가입일
jspark	1234	박지성	jspark@test.com	2018-09-29
ki	1234	기성용	ki@test.com	2018-09-13
park	1234	박찬호	park@test.com	2018-09-04
kim	1212	김유신	kim@jweb.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

`select * from t_member;`

회원가입

23.5 마이바티스의 동적 SQL문 사용하기

- 23.5.2 <choose> 태그로 동적 SQL문 만들기

<choose> 태그 사용법

```
<choose>
  <when test="조건식1">
    구문1
  </when>
  <when test="조건식2">
    구문2
  </when>
  ...
  <otherwise>
    구문 n+1;
  </otherwise>
</choose>
```

23.5 마이바티스의 동적 SQL 문 사용하기

코드 23-39 pro23/src/mybatis/mappers/member.xml

```

...
<select id="searchMember" parameterType="memberVO" resultMap="memResult">
  <![CDATA[
    select * from t_member
  ]]>
  <where>
    <choose>
      <when test="name != '' and name != null and email != '' and email != null">
        name=#{name} and email=#{email}
      </when>
      <when test="name != '' and name != null">
        name = #{name}
      </when>
      <when test="email != '' and email != null">
        email = #{email}
      </when>
    </choose>
  </where>
  order by joinDate desc
</select>
</mapper>

```

SQL문 id를 searchMember로 지정합니다.

name과 email 속성 값이 모두 있는 경우 'name=name 속성 값 and email=email 속성 값' 조건식을 where절에 추가합니다.

name 속성 값만 있을 경우 'name=name 속성 값' 조건식을 where절에 추가합니다.

email 속성 값만 있을 경우 'email=email 속성 값' 조건식을 where절에 추가합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

- 23.5.3 <forEach> 태그로 회원 정보 조회하기

<forEach> 태그 사용법

```
<foreach item="item" collection="list" index="index" open=" (" close=")" separator=",">
    #{item}
</foreach>
```

<forEach> 태그에 관련된 속성들

속성	설명
collection	전달받은 인자 값을 의미하며, 배열과 List 계열 인스턴스를 전달할 수 있습니다. List 인스턴스 전달 시에는 list로 표시하고 배열 전달 시에는 array로 표시합니다
index	foreach문이 반복될 때마다 1씩 증가시키면서 접근하는 값의 위치를 나타냅니다. 최초 값의 위치는 0입니다.
item	반복문이 실행될 때마다 collection 속성에 지정된 값에 접근하여 차례대로 사용합니다.
open	해당 구문이 시작될 때의 지정한 기호를 추가합니다.
close	해당 구문이 끝날 때의 지정한 기호를 추가합니다.
separator	한 번 이상 반복될 때 반복되는 사이에 지정한 기호를 추가합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

1. 다음과 같이 member.xml을 작성하여 SQL문으로 Map 데이터가 전달되면 <foreach> 태그로 Map 데이터의 값을 반복해서 접근한 후 in 조건절에 조건 값으로 추가합니다

코드 23-40 pro23/src/mybatis/mappers/member.xml

```
...
<select id="foreachSelect" resultMap="memResult" parameterType="java.util.Map">
  <![CDATA[
    select * from t_member
  ]]>
  where name in
  <foreach item="item" collection="list" open="(" separator="," close=")" >
    #{item}
  </foreach>
  order by joinDate desc
</select>
</mapper>
```

SQL문에 List 인스턴스나 배열을 전달하면 자동으로 Map에 전달되어 이름을 키(key)로 사용합니다. List 인스턴스는 list를 키로 사용하고 배열은 array를 키로 사용합니다.

foreach문을 이용해 반복해서 list의 값을 표시합니다.

select * from t_member where name in('홍길동', '이순신', '차범근')

23.5 마이바티스의 동적 SQL문 사용하기

2. MemberServlet 클래스에서는 브라우저에서 action 값 foreachSelect로 요청하면 ArrayList에 회원 이름을 저장하여 SQL문으로 전달합니다.

코드 23-41 pro23/src/com/spring/ex04/MemberServlet.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    ...
    } else if(action.equals("foreachSelect")) {
        List<String> nameList = new ArrayList();
        nameList.add("홍길동");
        nameList.add("차범근");
        nameList.add("이순신");
        List membersList=dao.foreachSelect(nameList);
        request.setAttribute("membersList",membersList);
        nextPage="test03/listMembers.jsp";
    }
    RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
    dispatch.forward(request, response);
}
...
```

ArrayList에 검색할 이름을 저장한 후
SQL문으로 ArrayList를 전달합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

3. MemberDAO 클래스에서는 이름이 저장된 ArrayList를 다시 SqlSession의 selectList() 메서드를 호출하면서 SQL문으로 전달합니다.

코드 23-42 pro23/src/com/spring/ex04/MemberDAO.java

...

```
public List foreachSelect(List nameList){
```

```
    sqlMapper=getInstance();
```

```
    SqlSession session=sqlMapper.openSession();
```

```
    List list=session.selectList('mapper.member.foreachSelect",nameList);
```

```
    return list;
```

```
}
```

...

검색 이름이 저장된 nameList를 SQL문으로 전달합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

4. <http://localhost:8090/pro23/mem4.do?action=foreachSelect>로 요청하여 결과를 확인합니다

아이디	비밀번호	이름	이메일	가입일
lee	1212	이순신	lee@test.com	2018-09-04
hong	1212	홍길동	hong@gmail.com	2018-09-04

```
select * from t_member  
where name in('홍길동', '이순신', '차범근')
```

[회원가입](#)

23.5 마이바티스의 동적 SQL 문 사용하기

• 23.5.4 <forEach> 태그로 회원 정보 추가하기

1. member.xml을 다음과 같이 작성합니다.

코드 23-43 pro23/src/mybatis/mappers/member.xml

```

...
<!--
<insert id="foreachInsert" parameterType="java.util.Map">
  INSERT INTO t_member(id, pwd, name, email)
  VALUES
  <foreach item="item" collection="list" >
    ({item.id},
    #{item.pwd},
    #{item.name},
    #{item.email})
  </foreach>
</insert>
-->
<insert id="foreachInsert" parameterType="java.util.Map">
  <foreach item="item" collection="list" open="INSERT ALL" separator=" "
    close="SELECT * FROM DUAL" >
    INTO t_member(id, pwd, name, email)
    VALUES ({item.id},
    #{item.pwd},
    #{item.name},
    #{item.email})
  </foreach>
</insert>
</mapper>

```

MySQL과는 달리 오라클에서는 insert문을 반복해서 사용하면 오류가 발생합니다.

<foreach>로 반복 작업을 할 때는 처음에 INSERT ALL을 추가합니다.

<foreach>로 반복 작업이 끝난 후 SELECT * FROM DUAL을 마지막에 추가합니다.

<foreach> 태그 안에 위치해야 합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

2. 서블릿에서는 브라우저에서 전송된 action 값 foreachInsert에 대해 세 명의 회원 정보를 memList에 저장한 후 SQL문으로 전달하도록 구현합니다.

코드 23-44 pro23/src/com/spring/ex04/MemberServlet.java

...

```
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
```

...

```
} else if(action.equals("foreachInsert")) {
```

테이블에 추가할 회원 정보를 memList에 저장합니다.

```
List<MemberVO> memList = new ArrayList();
memList.add(new MemberVO("m1", "1234", "박길동", "m1@test.com"));
memList.add(new MemberVO("m2", "1234", "이길동", "m2@test.com"));
memList.add(new MemberVO("m3", "1234", "김길동", "m3@test.com"));
```

```
int result=dao.foreachInsert(memList);
nextPage="/mem4.do?action=listMembers";
```

SQL문으로 memList를 전달합니다.

```
}
```

```
RequestDispatcher dispatch = request.getRequestDispatcher(nextPage);
```

```
dispatch.forward(request, response);
```

```
}
```

...

23.5 마이바티스의 동적 SQL 문 사용하기

3. MemberDAO 클래스에서는 서블릿에서 회원 정보로 설정된 MemberVO 객체를 저장한 memList를 전달받습니다. 그리고 이를 다시 SqlSession의 insert() 메서드로 전달합니다.

코드 23-45 pro23/src/com/spring/ex04/MemberDAO.java

...

```
public int foreachInsert(List memList){
```

```
    sqlMapper=getInstance();
```

```
    SqlSession session=sqlMapper.openSession();
```

```
    int result = session.insert("mapper.member.foreachInsert",memList);
```

```
    session.commit();
```

```
    return result ;
```

```
}
```

...

회원 정보가 저장된 memList를 SQL문으로 전달합니다.

```
} else if(action.equals("foreachInsert")) {
```

↓ 저장합니다.

```
List<MemberVO> memList = new ArrayList();
```

```
memList.add(new MemberVO("m1", "1234", "박길동", "m1@test.com"));
```

```
memList.add(new MemberVO("m2", "1234", "이길동", "m2@test.com"));
```

```
memList.add(new MemberVO("m3", "1234", "김길동", "m3@test.com"));
```

```
int result=dao.foreachInsert(memList);
```

SQL문으로 memList를 전달합니다.

23.5 마이바티스의 동적 SQL 문 사용하기

4. <http://localhost:8090/pro23/mem4.do?action=foreachInsert>로 요청하면 다음과 같이 박길동, 이길동, 세 명의 회원 정보가 한꺼번에 추가된 것을 볼 수 있습니다.

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

회원가입

23.5 마이바티스의 동적 SQL 문 사용하기

- 23.5.5 <sql> 태그와 <include> 태그로 SQL 문 중복 제거하기

코드 23-46 pro23/src/mybatis/mappers/member.xml

...

```
<sql id="a">
  <![CDATA[
    select * from t_member
  ]]>
</sql>
```

• <sql> 태그를 이용해 공통 SQL문의 refid를
a로 지정합니다.

```
<select id="searchMember" parameterType="memberVO" resultMap="memResult">
```

```
  <include refid="a" />
```

• <include> 태그를 이용해 공통 SQL문을
재사용합니다.

```
  <where>
```

```
  <choose>
```

```
    <when test="name != '' and name != null and email != '' and email != null">
```

```
      name=#{name} and email=#{email}
```

```
    </when>
```

```
    <when test="name != '' and name != null">
```

```
      name = #{name}
```

```
    </when>
```

```
    <when test="email != '' and email != null">
```

```
      email = #{email}
```

```
    </when>
```

```
  </choose>
```

23.5 마이바티스의 동적 SQL 문 사용하기

```
</where>
```

```
order by joinDate desc
```

```
</select>
```

```
<select id="foreachSelect" resultMap="memResult" parameterType="java.util.Map">
```

```
<include refid="a" />
```

————— *<include> 태그를 이용해 공통 SQL문을
재사용합니다.*

```
where name in
```

```
<foreach item="item" collection="list" open="(" separator="," close=")" >
```

```
  #{item}
```

```
</foreach>
```

```
</select>
```

```
</mapper>
```

23.5 마이바티스의 동적 SQL 문 사용하기

마이바티스에서 오라클 연동해 like 검색하는 방법

```

162<!--
163  <select id="selectLike" resultMap="memResult" parameterType="String" >
164    <![CDATA[
165      select * from t_member
166      where
167        name like '%#{name}%'
168    ]]>
169  </select>
170  -->
171  <!-- like 검색 -->
172  <select id="selectLike" resultMap="memResult" parameterType="String" >
173    <![CDATA[
174      select * from t_member
175      where
176        name like '%' || #{name} || '%'
177    ]]>
178  </select>

```

이렇게 작성하면 실행 시 에러 발생!

#{name} 앞에는 '%' ||를 붙인다.
에는 || '%'를 붙인다.

❖ 마이바티스 동적 SQL 기능 관련 사이트

- <http://www.mybatis.org/mybatis-3/ko/dynamic-sql.html>