

7장 서블릿 비즈니스 로직 처리

- 7.1 서블릿의 비즈니스 로직 처리 방법
- 7.2 서블릿의 데이터베이스 연동하기
- 7.3 DataSource 이용해 데이터베이스 연동하기
- 7.4 DataSource 이용해 회원 정보 등록하기
- 7.5 회원 정보 삭제하기

7.1 서블릿의 비즈니스 로직 처리 방법

도서 쇼핑몰에서 검색 하기

YES24.COM 통합검색 **자바 웹** **미아용철** ▼

빠른분야찾기 베스트 신상품 이벤트 바이백 중고매장 채널에스 블로그 에스베리굿즈 만원아하도서

통합검색 546

국내도서 135

외국도서 169

eBook 21

DVD/BD 1

중고샵 119

리뷰 94

기사, 인터뷰 7

결과내 재검색

입력후 엔터

책소개 검색

입력후 엔터

목차 검색

인기도 | 정확도 | 신상품 | 최저가 | 최고가 | 평점순 | 리뷰순

20개

1

[도서] 자바 웹을 다루는 기술 : JSP, 서블릿, 스프링까지 실무에서 알아야 할 기술은 따로 있
대
이병승 저 | 길벗 | 2019년 01월
45,000원 → **40,500원**(10% 할인) | YES포인트 2,250원(5%지급)
도착 예상일 : 지금 주문하면 **내일** 도착예정
판매지수 2,760 | 회원리뷰 (1개) | 내용 ★★★★★ 편집/구성 ★★★★★
사은품 길벗 IT 전문서 브랜드전

2

[도서] 스프링 부트로 배우는 자바 웹 개발 : 서블릿부터 Spring Data JPA, Rest API, 액추
에이터를 활용한 모니터링, 클라우드 서비스를 이용한 배포까지
윤석진 저 | 제이펍 | 2018년 06월
27,000원 → **24,300원**(10% 할인) | YES포인트 1,350원(5%지급)
도착 예상일 : 지금 주문하면 **내일** 도착예정

1

7.1 서블릿의 비즈니스 로직 처리 방법

- 서블릿의 비즈니스 처리 작업

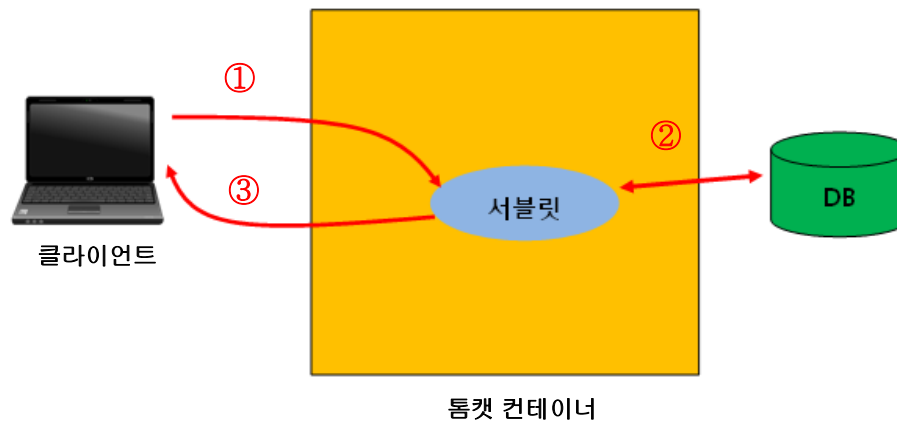
- 서블릿이 클라이언트로부터 요청을 받으면 그 요청에 대해 작업을 수행하는 것.
- 웹 프로그램에서 대부분의 비즈니스 처리 작업은 데이터베이스 연동 관련 작업이지만 그 외에 다른 서버와 연동해서 데이터를 얻는 작업도 수행
- 서블릿의 가장 핵심 기능

- 서블릿의 비즈니스 처리 작업 예

- 웹 사이트 회원 등록 요청 처리 작업
- 웹 사이트 로그인 요청 처리 작업
- 쇼핑몰 상품 주문 처리 작업

7.1 서블릿의 비즈니스 로직 처리 방법

서블릿의 비즈니스 처리 과정

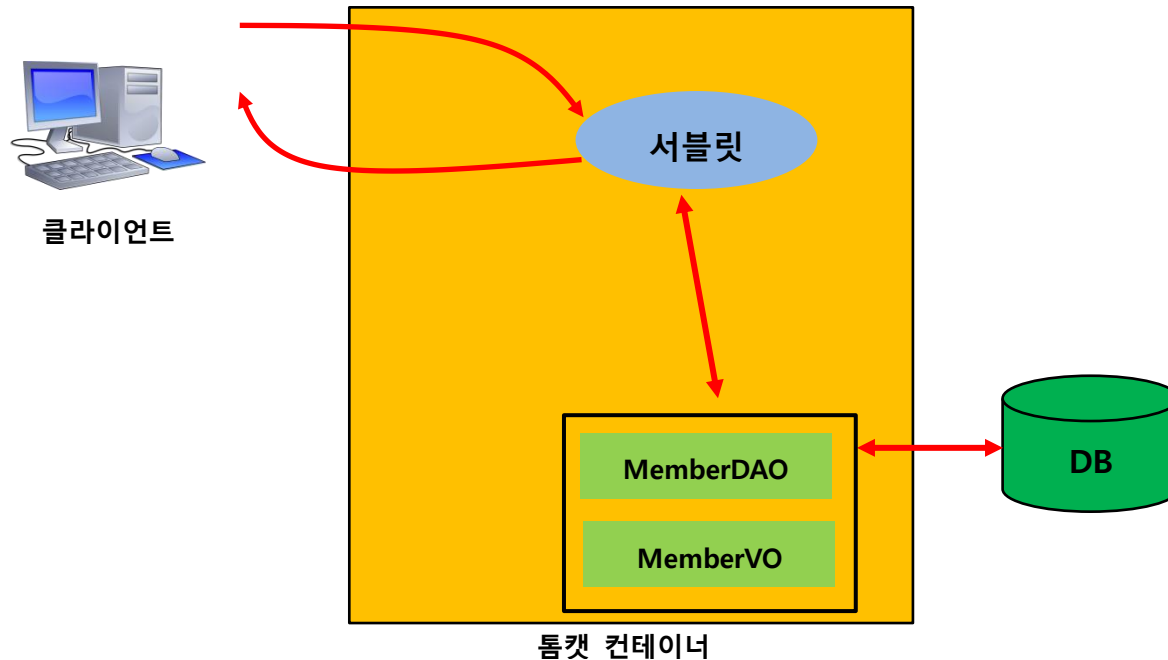


- ① 클라이언트로부터 요청을 받습니다.
- ② 데이터베이스 연동과 같은 비즈니스 로직을 처리합니다.
- ③ 처리 결과를 클라이언트에게 돌려줍니다.

7.2 서블릿의 데이터베이스 연동하기

- 서블릿의 데이터베이스 연동 과정

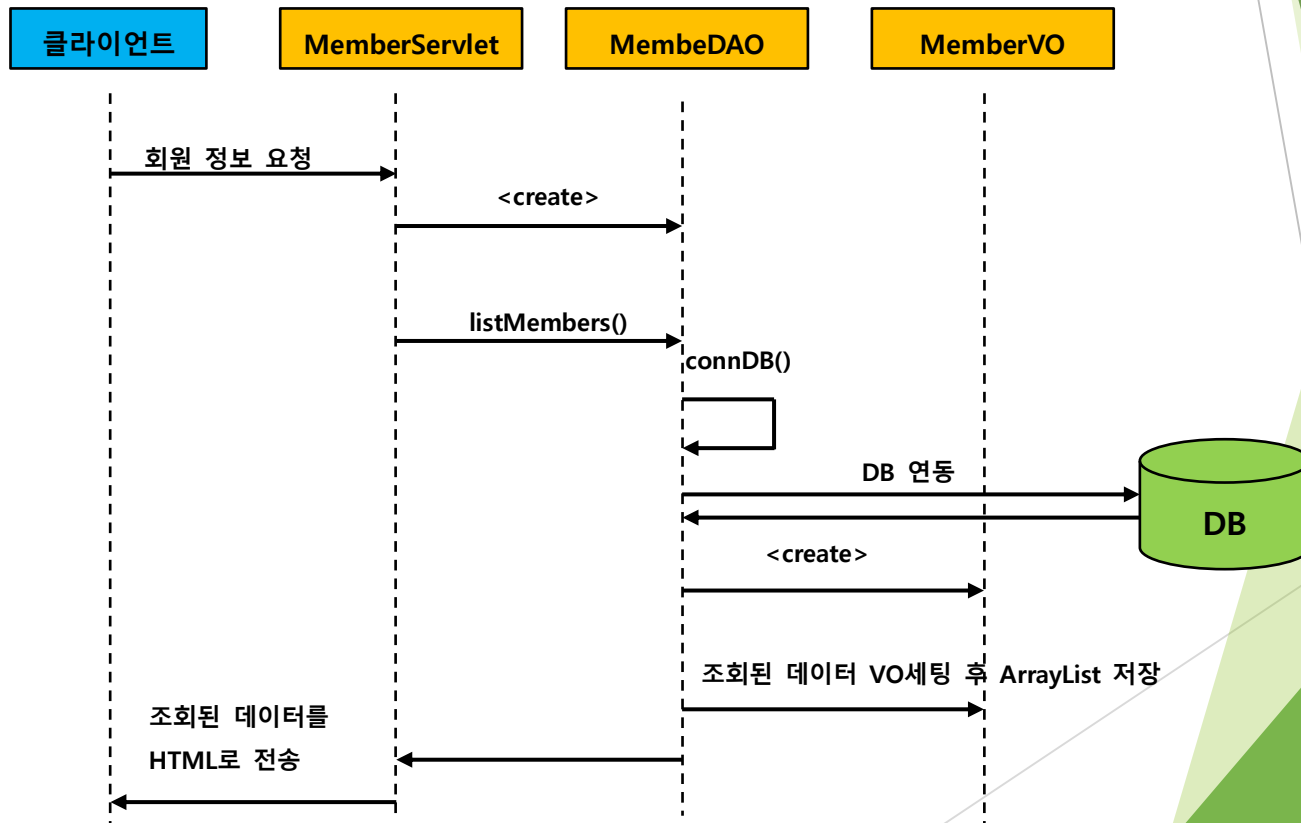
서블릿의 데이터베이스 연동 과정



7.2 서블릿의 데이터베이스 연동하기

- 7.2.1 서블릿으로 회원 정보 테이블의 회원 정보 조회

회원 정보 조회 과정



7.2 서블릿의 데이터베이스 연동하기

- ① 웹 브라우저가 서블릿에게 회원 정보를 요청합니다.
- ② MemberServlet은 요청을 받은 후 MemberDAO 객체를 생성하여 listMembers() 메서드를 호출합니다.
- ③ listMembers()에서 다시 connDB() 메서드를 호출하여 데이터베이스와 연결한 후 SQL문을 실행해 회원 정보를 조회합니다.
- ④ 조회된 회원 정보를 MemberVO 속성에 설정한 후 다시 ArrayList에 저장합니다.
- ⑤ ArrayList를 다시 메서드를 호출한 MemberServlet으로 반환한 후 ArrayList의 MemberVO를 차례대로 가져와 회원 정보를 HTML 태그의 문자열로 만듭니다.
- ⑥ 만들어진 HTML 태그를 웹 브라우저로 전송해서 회원 정보를 출력합니다.

7.2 서블릿의 데이터베이스 연동하기

테이블 t_member의 구성

NO	속성이름	컬럼이름	자료형	크기	유일키 여부	NULL 여부	키	기본값
1	ID	id	varchar2	10	Y	N	기본키	
2	비밀번호	pwd	varchar2	10		N		
3	이름	name	varchar2	50				
4	이메일	email	varchar2	50		N		
5	가입일자	joinDate	date			N		sysdate

7.2 서블릿의 데이터베이스 연동하기

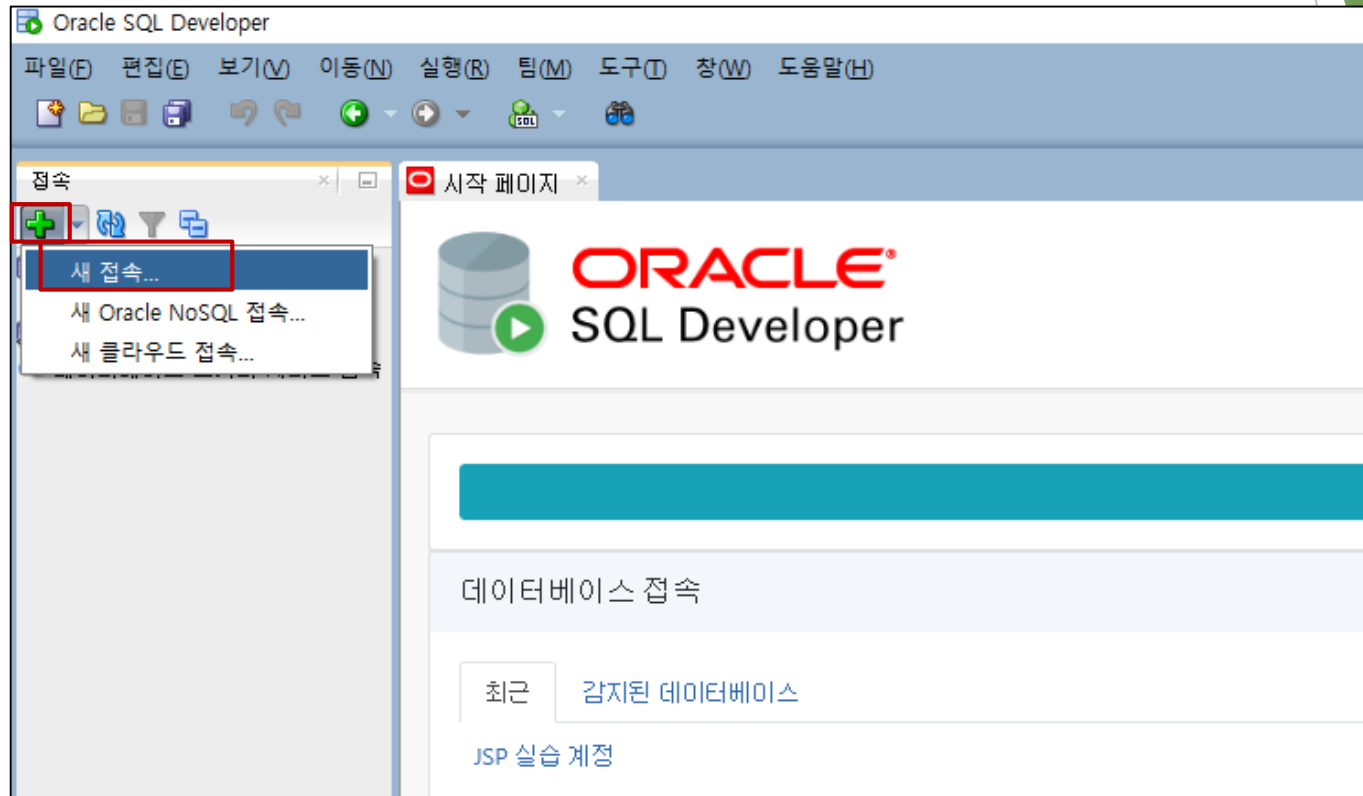
오라클에 테이블 생성 후 회원 정보 입력하기

1. 먼저 SQL Developer에서 회원 테이블과 회원 정보를 입력하기 위해 SQL Developer를 실행합니다.



7.2 서블릿의 데이터베이스 연동하기

2. 왼쪽 메뉴의 +를 클릭한 후 새 접속...을 선택합니다.



7.2 서블릿의 데이터베이스 연동하기

3. 왼쪽 메뉴에서 미리 만들어 놓은 접속 이름을 클릭하거나 직접 연결 정보를 입력한 후 접속을 클릭합니다.

The screenshot shows the 'Oracle' connection configuration dialog box. On the left, a list of connections includes 'JSP 실습 계정', which is selected. The right pane displays the configuration for this connection. The '접속 이름(N)' is 'JSP 실습 계정', the '사용자 이름(U)' is 'scott', and the '비밀번호(P)' is masked with '****'. There is a checkbox for '비밀번호 저장(Y)' and a '접속 색상' button. The 'Oracle' tab is active, showing '접속 유형(Y)' set to '기본' (Basic) and '롤(L)' set to '기본값' (Default). The '호스트 이름(A)' is 'localhost', the '포트(R)' is '1521', and the 'SID(I)' is 'xe'. There is also an option for '서비스 이름(E)'. At the bottom, there are checkboxes for 'OS 인증' and 'Kerberos 인증', and a '고급...' button. The '상태:' label is present. At the very bottom, there are buttons for '도움말(H)', '저장(S)', '지우기(C)', '테스트(T)', '접속(O)' (highlighted with a red box), and '취소'.

7.2 서블릿의 데이터베이스 연동하기

4. 접속한 후 생성되는 워크시트에 다음과 같은 테이블 생성 SQL문을 입력합니다.

코드 7-1 회원 정보를 저장하는 테이블을 생성하고 추가하는 SQL문

```
--회원 테이블 생성
create table t_member(
    id varchar2(10) primary key,
    pwd varchar2(10),
    name varchar2(50),
    email varchar2(50),
    joinDate date default sysdate
);

--회원 정보 추가
insert into t_member
values('hong','1212','홍길동','hong@gmail.com',sysdate);

insert into t_member
values('lee','1212','이순신','lee@test.com',sysdate);

insert into t_member
values('kim','1212','김유신','kim@jweb.com',sysdate);
commit;

select * from t_member;
```

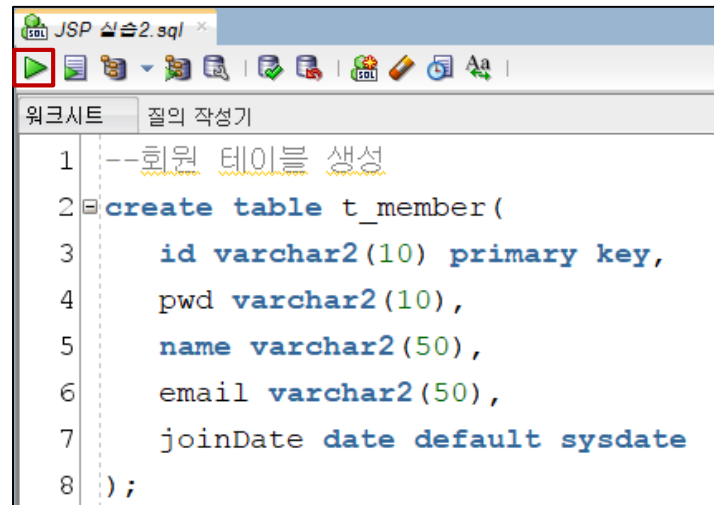
명시적으로 추가하지 않으면
현재 시각을 입력합니다.

SQL Developer에서 테이블에 회원 정보를 추가한 후 반드시
커밋(commit)을 해줘야 영구적으로 반영이 됩니다.

테이블에서 회원 정보를 조회합니다.

7.2 서블릿의 데이터베이스 연동하기

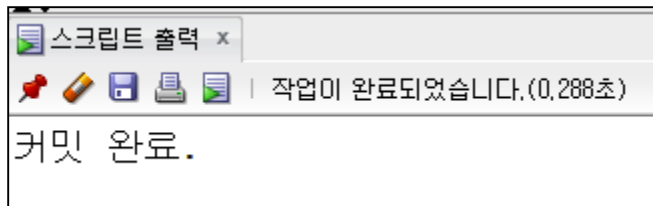
5. 마우스 포인터를 각각의 SQL문에 위치시킨 후 왼쪽 상단 녹색 버튼을 클릭해 SQL문을 실행하여 테이블을 생성합니다. insert문에 대해서도 동일하게 실행합니다.



```
1  --회원 테이블 생성
2  create table t_member(
3      id varchar2(10) primary key,
4      pwd varchar2(10),
5      name varchar2(50),
6      email varchar2(50),
7      joinDate date default sysdate
8  );
```

7.2 서블릿의 데이터베이스 연동하기

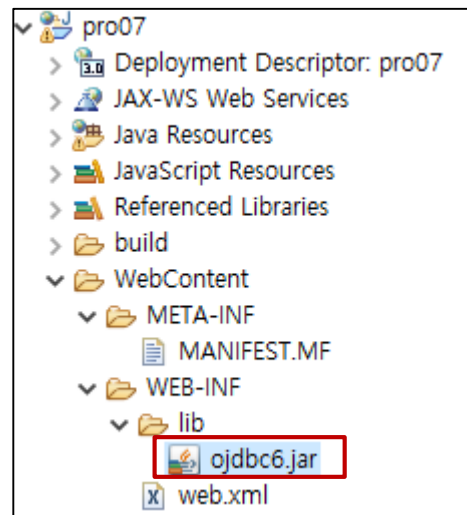
6. 커밋이 완료됐다는 메시지가 나타나고 select문으로 조회 시 회원 정보가 표시됩니다.



ID	PWD	NAME	EMAIL	JOINDATE
1 hong	1212	홍길동	hong@gmail.com	18/09/04
2 lee	1212	이순신	lee@test.com	18/09/04
3 kim	1212	김유신	kim@jweb.com	18/09/04

7.2 서블릿의 데이터베이스 연동하기

7. 이클립스에서 만든 프로젝트에서 회원 정보를 조회해 보겠습니다. 새 프로젝트 pro07을 생성한 다음 오라클 데이터베이스와 연동하는 데 필요한 드라이버인 ojdbc6.jar를 프로젝트의 /WebContent/WEB-INF/lib 폴더에 복사하여 붙여 넣습니다.



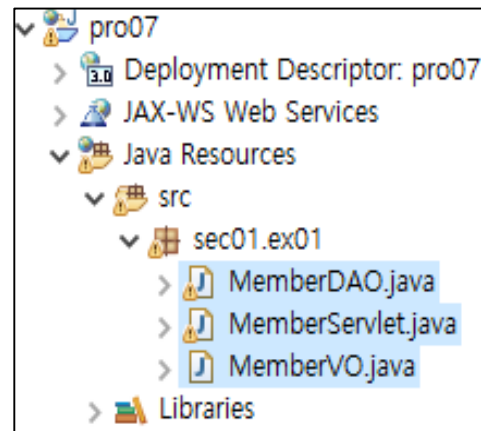
❖ Tip

오라클 드라이버는 아래 링크를 클릭해 다운로드할 수 있습니다.

• <https://www.oracle.com/technetwork/apps-tech/jdbc-112010-090769.html>

7.2 서블릿의 데이터베이스 연동하기

8. sec01.ex01 패키지를 만들고 다음과 같이 회원 조회와 관련된 자바 클래스 파일인 MemberDAO, MemberServlet, MemberVO 클래스를 각각 생성합니다.



7.2 서블릿의 데이터베이스 연동하기

9. 브라우저의 요청을 받는 MemberServlet 클래스를 다음과 같이 작성합니다.

코드 7-2 pro07/src/sec01/ex01/MemberServlet.java

```
package sec01.ex01;
```

```
...
```

```
@WebServlet("/member") // 주식 해제
```

```
public class MemberServlet extends HttpServlet
```

```
{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws IOException, ServletException
```

```
{
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    MemberDAO dao = new MemberDAO();
```

SQL문으로 조회할 MemberDAO 객체를 생성합니다.

```
    List list = dao.listMembers();
```

listMembers() 메서드로 회원 정보를 조회합니다.

```
    out.print("<html><body>");
```

```
    out.print("<table border=1><tr align='center' bgcolor='lightgreen'>");
```

```
    out.print("<td>아이디</td><td>비밀번호</td><td>이름</td><td>이메일</td>
```

```
    <td>가입일</td></tr>");
```

7.2 서블릿의 데이터베이스 연동하기

```
for (int i = 0; i < list.size(); i++)  
{
```

조회한 회원 정보를 for문과 <tr> 태그를 이
용해 리스트로 출력합니다.

```
    MemberVO memberVO = (MemberVO) list.get(i);  
    String id = memberVO.getId();  
    String pwd = memberVO.getPwd();  
    String name = memberVO.getName();  
    String email = memberVO.getEmail();  
    Date joinDate = memberVO.getJoinDate();  
    out.print("<tr><td>" + id + "</td><td>" + pwd + "</td><td>"  
            + name + "</td><td>" + email + "</td><td>"  
            + joinDate + "</td></tr>");  
}
```

```
    out.print("</table></body></html>");  
}  
}
```

7.2 서블릿의 데이터베이스 연동하기

10. MemberDAO 클래스를 다음과 같이 작성합니다. 회원 정보 조회 SQL문을 실행하여 조회한 레코드들의 컬럼 값을 다시 MemberVO 객체의 속성에 설정한 다음 ArrayList에 저장하고 호출한 곳으로 반환합니다.

코드 7-3 pro07/src/sec01/ex01/MemberDAO.java

```
package sec01.ex01;
...
public class MemberDAO
{
    private Connection con;
    private Statement stmt;

    public List listMembers()
    {
        List list = new ArrayList();
        try
        {
            connDB();
            String query = "select * from t_member ";
            System.out.println(query);
            ResultSet rs = stmt.executeQuery(query);
            while (rs.next())
            {
                String id = rs.getString("id");
                String pwd = rs.getString("pwd");
                String name = rs.getString("name");
                String email = rs.getString("email");
                Date joinDate = rs.getDate("joinDate");
                MemberVO vo = new MemberVO();
                vo.setId(id);
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return list;
    }
}
```

네 가지 정보로 데이터베이스를 연결합니다.

SQL문으로 회원 정보를 조회합니다.

조회한 레코드의 각 컬럼 값을 받아 옵니다.

각 컬럼 값을 다시 MemberVO 객체의 속성에 설정합니다.

7.2 서블릿의 데이터베이스 연동하기

```
vo.setPwd(pwd);
vo.setName(name);
vo.setEmail(email);
vo.setJoinDate(joinDate);
list.add(vo);
}
rs.close();
stmt.close();
con.close();
} catch (Exception e)
{
    e.printStackTrace();
}
return list;
}
```

설정된 MemberVO 객체를 다시 ArrayList에 저장합니다.

조회한 레코드의 개수만큼 MemberVO 객체를 저장한 ArrayList를 반환합니다.

7.2 서블릿의 데이터베이스 연동하기

```
private void connDB()
{
    try
    {
        Class.forName(driver);
        System.out.println("Oracle 드라이버 로딩 성공");
        con = DriverManager.getConnection(url, user, pwd);
        System.out.println("Connection 생성 성공");
        stmt = con.createStatement();
        System.out.println("Statement 생성 성공");
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

7.2 서블릿의 데이터베이스 연동하기

11. MemberVO 클래스를 다음과 같이 작성합니다.

코드 7-4 pro07/src/sec01/ex01/MemberVO.java

```
package sec01.ex01;

import java.sql.Date;

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

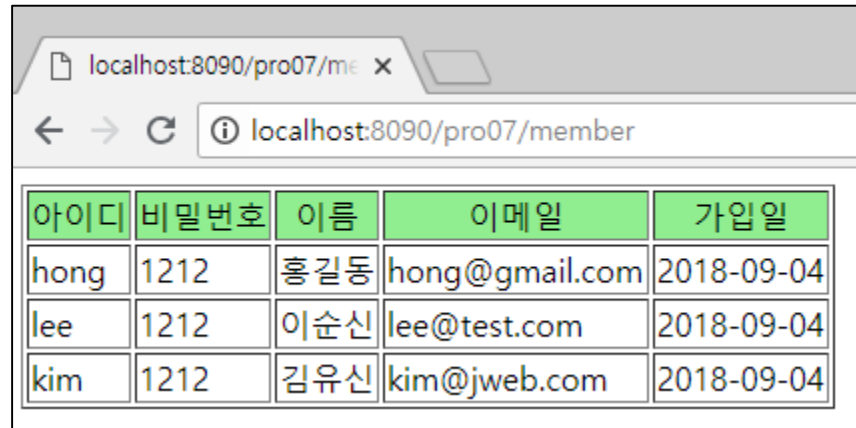
    public MemberVO()
    {
        System.out.println("MemberVO 생성자 호출");
    }
}
```

t_member 테이블의 컬럼 이름과 동일한 자료형과
이름으로 속성들을 선언합니다.

...

7.2 서블릿의 데이터베이스 연동하기

12. <http://localhost:8090/pro07/member>로 요청하여 실행 결과를 확인합니다. 회원 정보가 웹 브라우저로 출력되는 것을 확인할 수 있습니다.



The screenshot shows a web browser window with the address bar displaying `localhost:8090/pro07/member`. The browser content displays a table with 5 columns: 아이디 (ID), 비밀번호 (Password), 이름 (Name), 이메일 (Email), and 가입일 (Join Date). The table contains 3 rows of member data.

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

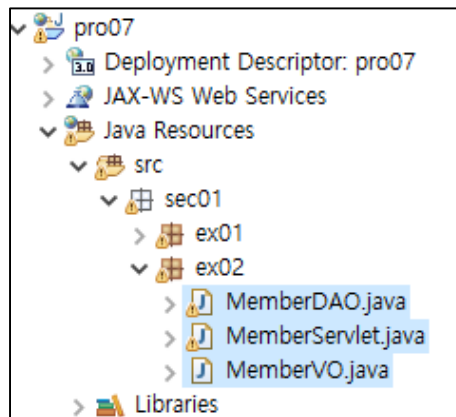
7.2 서블릿의 데이터베이스 연동하기

• 7.2.2 PreparedStatement 를 이용한 회원 정보 조회 실습

PreparedStatement 인터페이스의 특징

- PreparedStatement 인터페이스는 Statement 인터페이스를 상속하므로 지금까지 사용한 메서드를 그대로 사용함
- Statement 인터페이스에 대해서 PreparedStatement 인터페이스는 컴파일된 SQL문을 DBMS에 전달하여 성능을 향상시킴
- PreparedStatement 인터페이스에서는 실행하려는 SQL문에 '?'를 넣을 수 있으므로 '?'의 값만 바꾸어 손쉽게 설정할 수 있어 Statement보다 SQL문 작성하기가 더 간단함

1. sec01.ex02 패키지를 만든 후 MemberServlet.java와 MemberVO.java는 기존의 것을 복사하여 붙여 넣습니다.



7.2 서블릿의 데이터베이스 연동하기

2. PreparedStatement를 이용해 데이터베이스와 연동하는 MemberDAO 클래스를 작성합니다.

코드 7-5 pro07/src/sec01/ex02/MemberDAO.java

```
package sec01.ex02;

...

public class MemberDAO
{
    private Connection con;
    private PreparedStatement pstmt;

    public List listMembers()
    {
        List list = new ArrayList();
        try
        {
            connDB();
            con = dataFactory.getConnection();
            String query = "select * from t_member ";
            System.out.println("prepareStatement: " + query);
```

7.2 서블릿의 데이터베이스 연동하기

```
pstmt = con.prepareStatement(query);
ResultSet rs = pstmt.executeQuery();

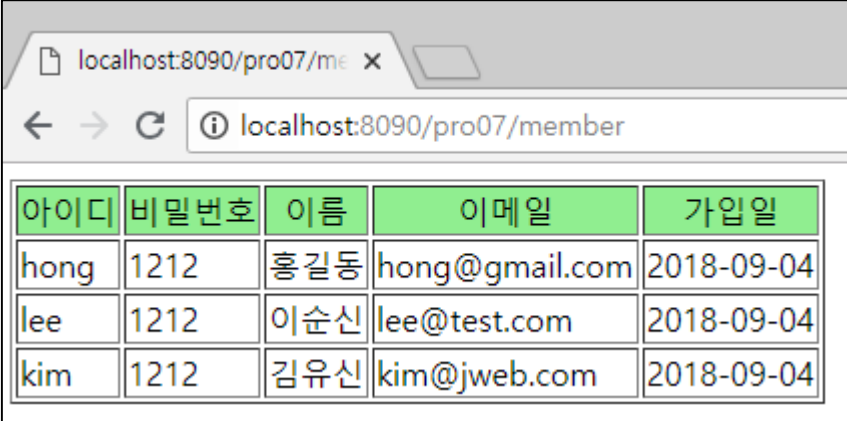
while (rs.next())
{
    String id = rs.getString("id");
    String pwd = rs.getString("pwd");
    String name = rs.getString("name");
    String email = rs.getString("email");
    Date joinDate = rs.getDate("joinDate");
    MemberVO vo = new MemberVO();
    vo.setId(id);
    vo.setPwd(pwd);
    vo.setName(name);
    vo.setEmail(email);
    vo.setJoinDate(joinDate);
    list.add(vo);
}
rs.close();
pstmt.close();
con.close();
} catch (Exception e)
{
    e.printStackTrace();
}
return list;
}
```

prepareStatement() 메서드에 SQL문을 전달해 PreparedStatement 객체를 생성합니다.

executeQuery() 메서드를 호출해 미리 설정한 SQL문을 실행합니다.

7.2 서블릿의 데이터베이스 연동하기

3. <http://localhost:8090/pro07/member>로 요청해서 실행 결과를 확인합니다. 눈으로 보면 Statement를 사용했을 때와 결과는 같습니다. 하지만 데이터베이스와 연동할 경우 수행 속도가 좀 더 빠르다는 차이가 있습니다.



A screenshot of a web browser window. The address bar shows 'localhost:8090/pro07/member'. The page content is a table with 5 columns: 아이디, 비밀번호, 이름, 이메일, and 가입일. The table contains 3 rows of data.

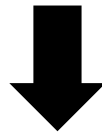
아이디	비밀번호	이름	이메일	가입일
hong	1212	홍길동	hong@gmail.com	2018-09-04
lee	1212	이순신	lee@test.com	2018-09-04
kim	1212	김유신	kim@jweb.com	2018-09-04

7.3 DataSource 이용해 데이터베이스 연동

- **ConnectionPool 등장 배경**

기존 데이터베이스 연동 방법의 문제점

- 애플리케이션에서 데이터베이스 연결 과정에서 시간이 너무 많이 걸린다.

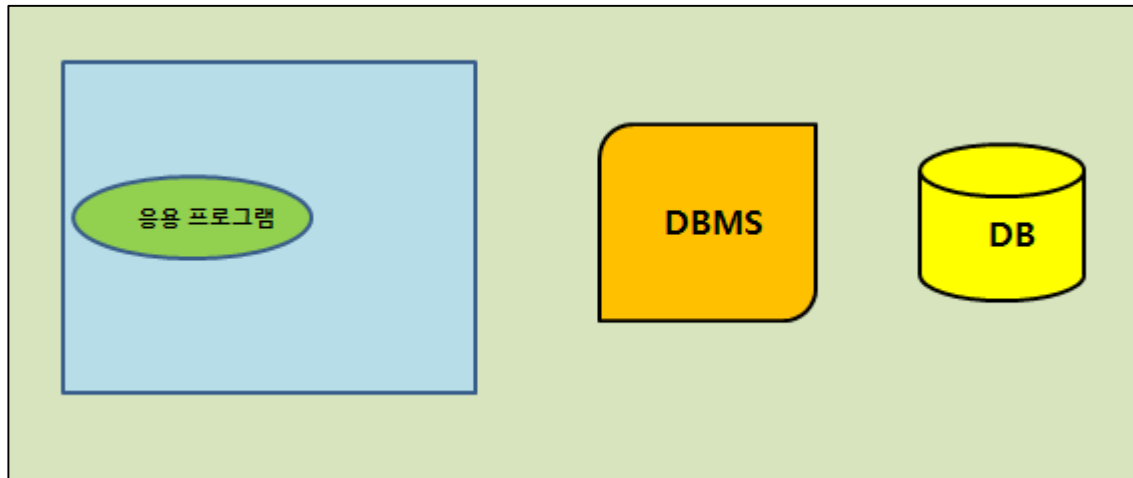


- 애플리케이션 실행 시 미리 Connection 객체를 생성한 후, 미리 데이터베이스 연결을 맺는다.
- 애플리케이션은 데이터베이스 연동 작업 발생 시 이 Connection 객체를 이용해서 작업을 한다.

7.3 DataSource 이용해 데이터베이스 연동

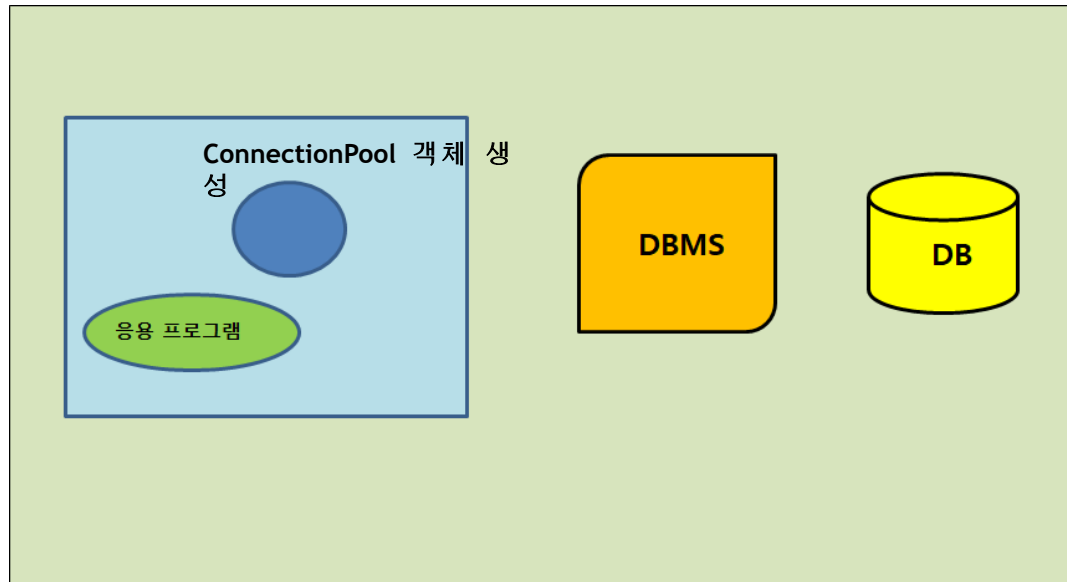
- 7.3.1 커넥션풀 동작 과정

1. 톰캣 컨테이너를 실행한 후 응용 프로그램을 실행합니다.



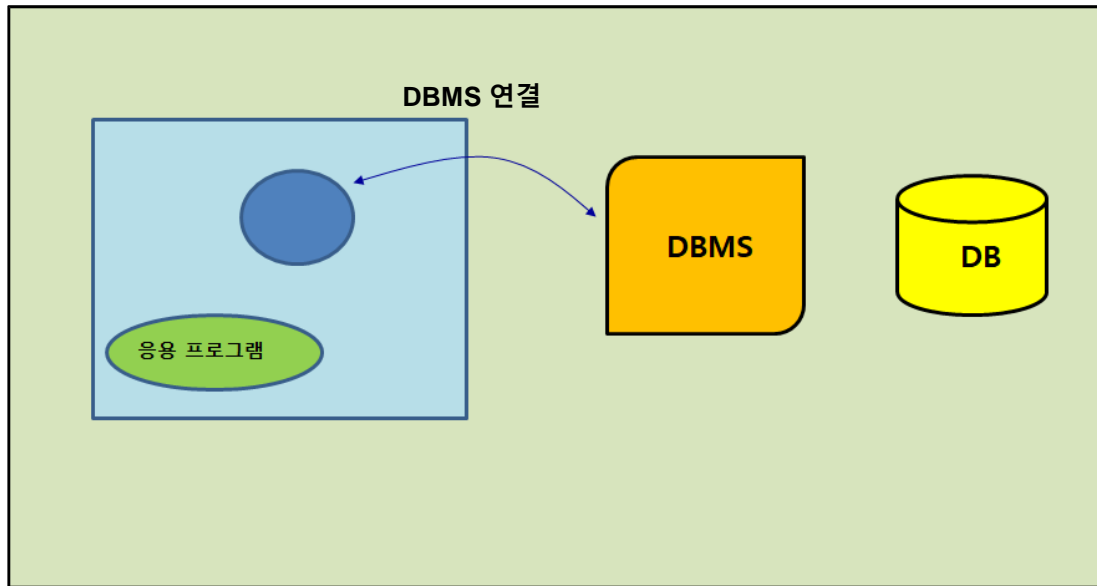
7.3 DataSource 이용해 데이터베이스 연동

2. 톰캣 컨테이너 실행 시 ConnectionPool 객체를 생성합니다.



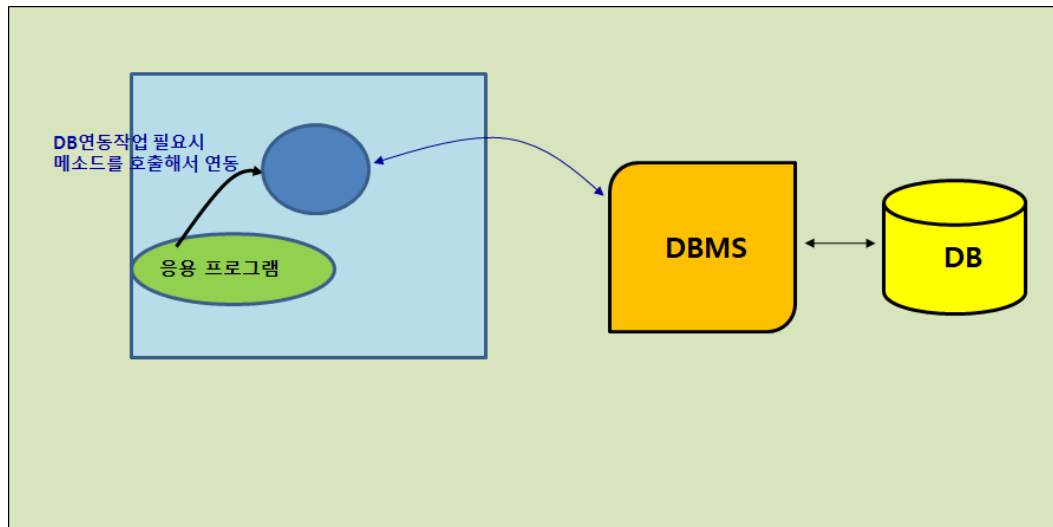
7.3 DataSource 이용해 데이터베이스 연동

3. 생성된 커넥션 객체는 DBMS와 미리 연결합니다.



7.3 DataSource 이용해 데이터베이스 연동

4. 데이터베이스와의 연동 작업이 필요할 경우 응용 프로그램은 ConnectinPool에서 제공하는 메서드를 호출하여 연동합니다



7.3 DataSource 이용해 데이터베이스 연동

• 7.3.2 JNDI

JNDI(Java Naming and Directory Interface)란 필요한 자원을 키/값(key/value) 쌍으로 저장한 후 필요할 때 키를 이용해 값을 얻는 방법

JNDI의 예

- 웹 브라우저에서 name/value 쌍으로 전송한 후 서블릿에서 `getParameter(name)`로 값을 가져올 때
- 해시맵(HashMap)이나 해시테이블(HashTable)에 키/값으로 저장한 후 키를 이용해 값을 가져올 때
- 웹 브라우저에서 도메인 네임으로 DNS 서버에 요청할 경우 도메인 네임에 대한 IP 주소를 가져올 때

커넥션풀에 적용하기

- 톰캣 컨테이너가 ConnectionPool 객체를 생성하면 이 객체에 대한 JNDI 이름(key)을 미리 설정해 놓음
- 그러면 웹 애플리케이션에서 데이터베이스와 연동 작업을 할 때 이 JNDI 이름(key)으로 접근하여 작업을 수행함

7.3 DataSource 이용해 데이터베이스 연동

- 7.3.3 톰캣의 DataSource 설정 및 사용 방법

톰캣의 ConnectionPool 설정 과정

- JDBC 드라이버를 /WEB-INF/lib 폴더에 설치합니다.



- ConnectionPool 기능 관련 jar 파일을 /WEB-INF/lib 폴더에 설치합니다.



- CATALINA_HOME\context.xml에 Connection 객체 생성 시 연결할 데이터베이스 정보를 JNDI로 설정합니다.



- DAO 클래스에서 데이터베이스 연동 시 JNDI 이름으로 데이터베이스를 연결해서 작업합니다.

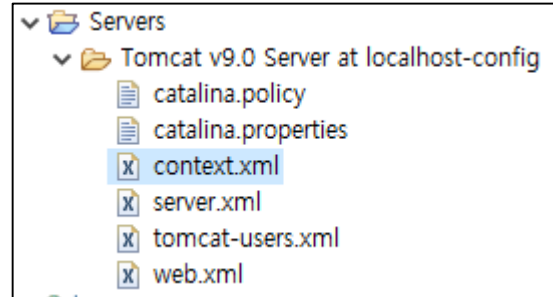
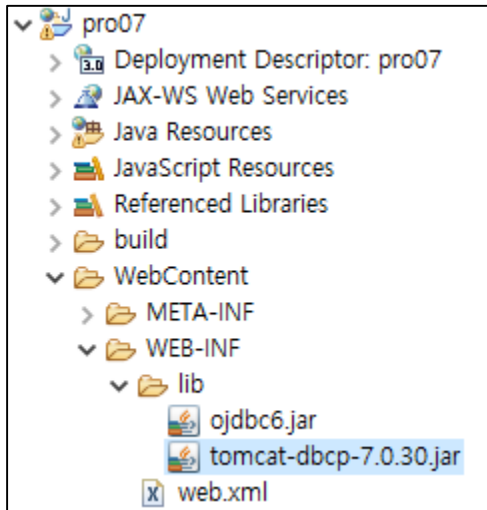
❖ ConnectionPool 관련 라이브러리 다운로드 받기

- <http://www.java2s.com/Code/Jar/t/Downloadtomcatdbcp7030jar.htm>

7.3 DataSource 이용해 데이터베이스 연동

- 7.3.4 이클립스에서 톰캣 DataSource 설정

프로젝트의 WEB-INF/lib 폴더에 드라이버와 ConnectionPool관련 라이브러리 설치



7.3 DataSource 이용해 데이터베이스 연동

context.xml 파일에 <Resource> 태그를 이용해 톰캣 실행 시 연결할 데이터베이스를 설정

```
context.xml
23 <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
24
25 <!-- Uncomment this to disable session persistence across Tomcat restarts -->
26 <!--
27 <Manager pathname="" />
28 -->
29 <Resource
30   name="jdbc/oracle"
31   auth="Container"
32   type="javax.sql.DataSource"
33   driverClassName="oracle.jdbc.OracleDriver"
34   url="jdbc:oracle:thin:@localhost:1521:XE"
35   username="scott"
36   password="tiger"
37   maxActive="50"
38   maxWait="-1" />
39 </Context>
```

7.3 DataSource 이용해 데이터베이스 연동

자바 클래스에서는 `name` 속성의 `jdbc/oracle`로 DataSource에 접근

코드 7-6 톰캣 context.xml에 Resource를 설정하는 방법

<Resource

name="jdbc/oracle"

name의 jdbc/oracle로 DataSource에 접근합니다.

auth="Container"

type="javax.sql.DataSource"

driverClassName="oracle.jdbc.OracleDriver"

url="jdbc:oracle:thin:@localhost:1521:XE"

username="scott"

데이터베이스를 연결하는 데 필요한
네 가지 값을 설정합니다.

password="tiger"

maxActive="50"

maxWait="-1" />

7.3 DataSource 이용해 데이터베이스 연동

ConnectionPool로 연결할 데이터베이스 속성

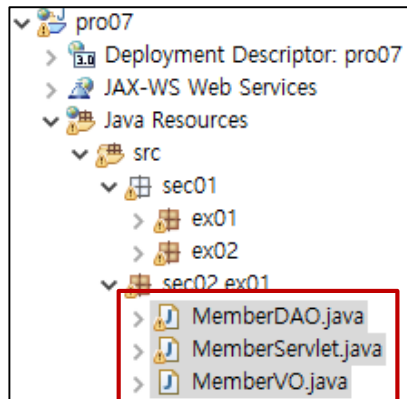
속성	설명
name	DataSource에 대한 JNDI 이름
auth	인증 주체
driverClassName	연결할 데이터베이스 종류에 따른 드라이버 클래스 이름
factory	연결할 데이터베이스 종류에 따른 ConnectionPool 생성 클래스 이름
maxActive	동시에 최대로 데이터베이스에 연결할 수 있는 Connection 수
maxIdle	동시에 idle 상태로 대기할 수 있는 최대 수
maxWait	새로운 연결이 생길 때까지 기다릴 수 있는 최대 시간
user	데이터베이스 접속 ID
password	데이터베이스 접속 비밀번호
type	데이터베이스 종류별 DataSource
url	접속할 데이터베이스 주소와 포트 번호 및 SID

다른 속성들은 고정적으로 사용되며, 주로 **driverClassName**, **user**, **password**, **url**만 변경해서 사용합니다.

7.3 DataSource 이용해 데이터베이스 연동

- 7.3.5 톰캣의 DataSource로 연동해 회원 정보 조회 실습

1. sec02.ex01 패키지를 만들고 앞에서 사용한 MemberDAO, MemberServlet, MemberVO 클래스를 복사하여 붙여 넣습니다.



7.3 DataSource 이용해 데이터베이스 연동

2. 복사한 MemberServlet 클래스의 서블릿 매핑 이름을 /member2로 변경합니다.

```
14
15 @WebServlet("/member2")
16 public class MemberServlet extends HttpServlet {
17     public void doGet(HttpServletRequest request, Http
18
19         response.setContentType("text/html; charset=utf-8");
20         PrintWriter out=response.getWriter();
21         MemberDAO dao=new MemberDAO();
22         List list=dao.listMembers();
23 }
```


7.3 DataSource 이용해 데이터베이스 연동

3. DataSource를 이용해 데이터베이스와 연동하는 MemberDAO 클래스를 다음과 같이 수정합니다.

코드 7-7 pro07/src/sec02/ex01/MemberDAO.java

```
package sec02.ex01;

...

public class MemberDAO
{
    /*
    private static final String driver = "oracle.jdbc.driver.OracleDriver";
    private static final String url = "jdbc:oracle:thin:@localhost:1521:XE";
    private static final String user = "scott"; private static final String pwd = "tiger";
    */

    private Connection con;
    private PreparedStatement pstmt;
    private DataSource dataFactory;

    public MemberDAO()
    {
        try
        {
            Context ctx = new InitialContext();
            Context envContext = (Context) ctx.lookup("java:/comp/env");
            dataFactory = (DataSource) envContext.lookup("jdbc/oracle");
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

더 이상 사용되지 않으므로 주석 처리합니다.

JNDI에 접근하기 위해 기본 경로
(java:/comp/env)를 지정합니다.

톰캣 context.xml에 설정한
name 값인 jdbc/oracle을
이용해 톰캣이 미리 연결한
DataSource를 받아 옵니다.

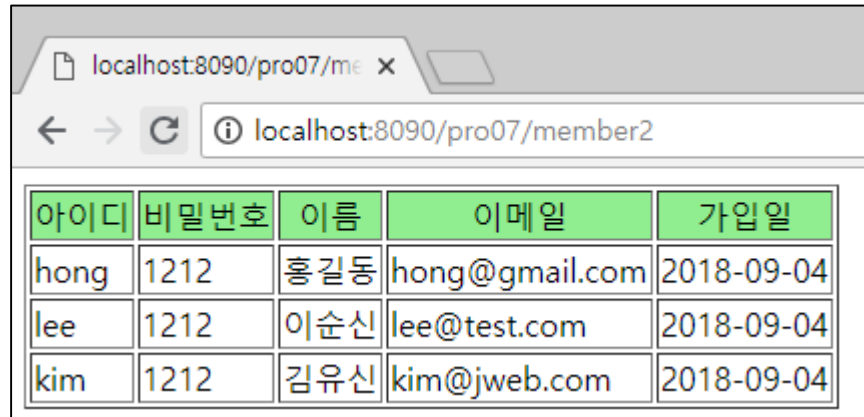
7.3 DataSource 이용해 데이터베이스 연동

```
public List listMembers()
{
    List list = new ArrayList();
    try
    {
        // connDB();
        con = dataFactory.getConnection();
        String query = "select * from t_member ";
        System.out.println("prepareStatement: " + query);
        pstmt = con.prepareStatement(query);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next())
        {
            String id = rs.getString("id");
            String pwd = rs.getString("pwd");
            String name = rs.getString("name");
            String email = rs.getString("email");
            Date joinDate = rs.getDate("joinDate");
            MemberVO vo = new MemberVO();
            vo.setId(id);
            vo.setPwd(pwd);
            vo.setName(name);
        }
    }
}
```

DataSource를 이용해 데이터베이스에 연결합니다.

7.3 DataSource 이용해 데이터베이스 연동

4. <http://localhost:8090/pro07/member2>로 요청합니다. 결과는 앞에서 실습했을 때와 같지만 이번에는 커넥션풀을 이용해서 데이터베이스와 연동했다는 점에서 차이가 있습니다.

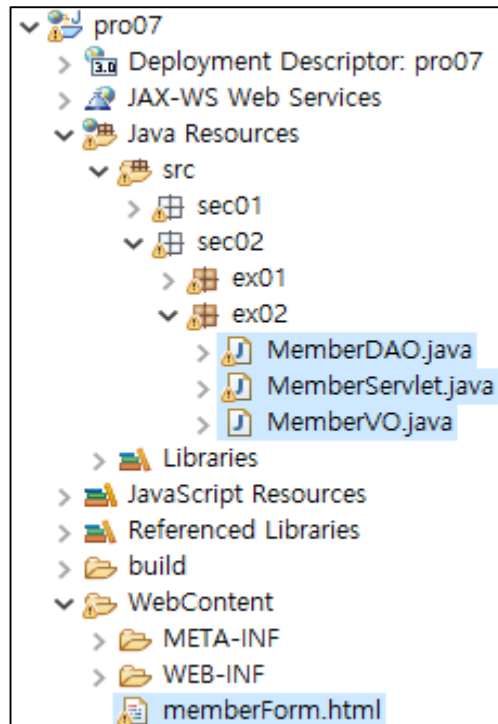


A screenshot of a web browser window. The address bar shows 'localhost:8090/pro07/member2'. The page content is a table with 5 columns: 아이디, 비밀번호, 이름, 이메일, and 가입일. The table contains 3 rows of data.

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

7.4 DataSource 이용해 회원 정보 등록하기

1. sec02.ex02 패키지를 만들고 MemberVO.java를 복사하여 붙여 넣습니다.



7.4 DataSource 이용해 회원 정보 등록하기

2. 회원 가입창을 작성하기 위해 다음과 같이 memberForm.html을 작성합니다. <hidden> 태그를 이용해 회원 가입창에서 새 회원 등록 요청을 서블릿에 전달합니다.

코드 7-8 pro07/WebContent/MemberForm.html

```
<!DOCTYPE html>
<html>
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>회원 가입창</title>
```

```
<script type="text/javascript">
```

```
function fn_sendMember() {
```

```
var frmMember = document.frmMember;
```

```
var id = frmMember.id.value;
```

```
var pwd = frmMember.pwd.value;
```

```
var name = frmMember.name.value;
```

```
var email = frmMember.email.value;
```

자바스크립트에서 <form> 태그의 name으로 접근해 입력한 값들을 얻습니다.

7.4 DataSource 이용해 회원 정보 등록하기

```
if (id.length == 0 || id == "") {  
    alert("아이디는 필수입니다.");  
} else if (pwd.length == 0 || pwd == "") {  
    alert("비밀번호는 필수입니다.");  
}  
else if (name.length == 0 || name == "") {  
    alert("이름은 필수입니다.");  
} else if (email.length == 0 || email == "") {  
    alert("이메일은 필수입니다.");  
} else {  
    frmMember.method = "post";  
    frmMember.action = "member3";  
    frmMember.submit();  
}  
}  
</script>  
</head>  
<body>  
    <form name="frmMember">  
        <table>  
            <th>회원 가입창</th>  
            <tr>  
                <td>아이디</td>  
                <td><input type="text" name="id"></td>  
            </tr>
```

전송 방법을 post로 지정합니다.

서블릿 매핑 이름을 member3으로 지정합니다.

서블릿으로 전송합니다.

입력한 ID를 서블릿으로 전송합니다.

7.4 DataSource 이용해 회원 정보 등록하기

3. MemberServlet 클래스를 다음과 같이 작성합니다.

코드 7-9 pro07/sec02/ex02/MemberServlet.java

```
package sec02.ex02;

...

private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    MemberDAO dao = new MemberDAO();
    PrintWriter out = response.getWriter();
    String command = request.getParameter("command");

    if (command != null && command.equals("addMember"))
    {
        String _id = request.getParameter("id");
        String _pwd = request.getParameter("pwd");
        String _name = request.getParameter("name");
        String _email = request.getParameter("email");
        MemberVO vo = new MemberVO();
        vo.setId(_id);
        vo.setPwd(_pwd);
        vo.setName(_name);
        vo.setEmail(_email);
        dao.addMember(vo);
    }
}
```

command 값을 받아옵니다.

회원 가입창에서 전송된
command가 addMember
이면 전송된 값들을 받아
옵니다.

회원 가입창에서 전송된 값
들을 얻어 와 MemberVO
객체에 저장한 후 SQL문을
이용해 전달합니다.

7.4 DataSource 이용해 회원 정보 등록하기

```
...
for (int i = 0; i < list.size(); i++)
{
    MemberVO memberVO = (MemberVO) list.get(i);
    String id = memberVO.getId();
    String pwd = memberVO.getPwd();
    String name = memberVO.getName();
    String email = memberVO.getEmail();
    Date joinDate = memberVO.getJoinDate();
    out.print("<tr><td>" + id + "</td><td>" + pwd + "</td><td>"
        + name + "</td><td>" + email + "</td><td>" + joinDate
        + "</td><td>" + "<a href='/pro07/member3?command=delMember&id='"
        + id + "'> 삭제 </a></td></tr>");
}
out.print("</table></body></html>");
out.print("<a href='/pro07/memberForm.html'>새 회원 가입하기</a>");
}
}
```

클릭하면 다시 회원 가입창으로 이동합니다.

새 회원 등록하기

7.4 DataSource 이용해 회원 정보 등록하기

PreparedStatement에서 insert문 사용하는 방법

- ① PreparedStatement의 insert문은 회원 정보를 저장하기 위해 ?(물음표)를 사용합니다.
- ② ?는 id, pwd, name, age에 순서대로 대응합니다.
- ③ 각 ?에 대응하는 값을 지정하기 위해 PreparedStatement의 setter를 이용합니다.
- ④ setter() 메서드의 첫 번째 인자는 '?'의 순서를 지정합니다.
- ⑤ ?은 1부터 시작합니다.
- ⑥ insert, delete, update문은 executeUpdate() 메서드를 호출합니다.

7.4 DataSource 이용해 회원 정보 등록하기

4. MemberDAO 클래스를 다음과 같이 수정합니다

코드 7-10 pro07/sec02/ex02/MemberDAO.java

```
...
public void addMember(MemberVO memberVO)
{
    try
    {
        Connection con = dataFactory.getConnection();
        String id = memberVO.getId();
        String pwd = memberVO.getPwd();
        String name = memberVO.getName();
        String email = memberVO.getEmail();

        String query = "insert into t_member";
        query += " (id,pwd,name,email)";
        query += " values(?,?,?,?)";
        System.out.println("prepareStatement: " + query);
        pstmt = con.prepareStatement(query);
        pstmt.setString(1, id);
        pstmt.setString(2, pwd);
        pstmt.setString(3, name);
        pstmt.setString(4, email);
        pstmt.executeUpdate();
        pstmt.close();
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

DataSource를 이용해 데이터베이스와 연결합니다.

테이블에 저장할 회원 정보를 받아 옵니다.

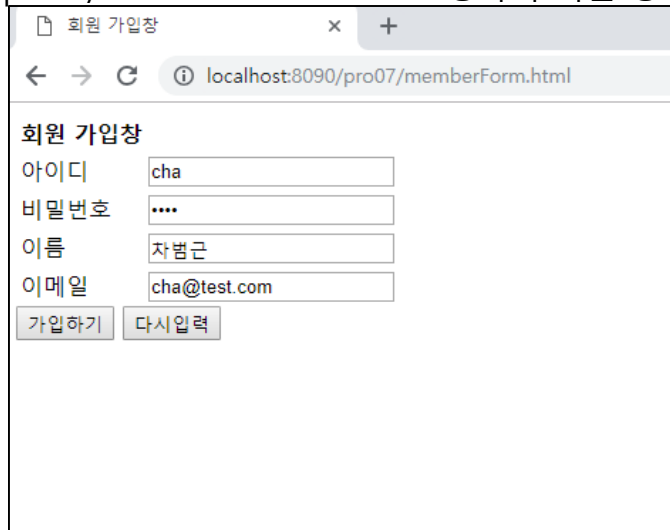
insert문을 문자열로 만듭니다.

insert문의 각 '?'에 순서대로 회원 정보를 세팅합니다.

회원 정보를 테이블에 추가합니다.

7.4 DataSource 이용해 회원 정보 등록하기

5. `http://localhost:8090/pro07/memberForm.html`로 요청하여 회원 정보를 입력한 후 가입하기를 클릭합니다.



회원 가입창

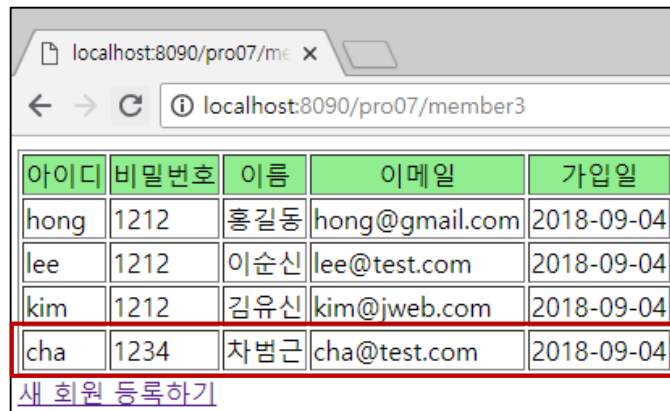
아이디

비밀번호

이름

이메일

6. 다음과 같이 회원 정보가 출력됩니다.



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

[새 회원 등록하기](#)

7.5 회원 정보 삭제하기

1. MemberServlet 클래스를 다음과 같이 수정합니다.

코드 7-11 pro07/src/sec02/ex02/MemberServlet3.java

```
...
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    MemberDAO dao = new MemberDAO();
    PrintWriter out = response.getWriter();
    String command = request.getParameter("command");
    if (command != null && command.equals("addMember"))
    {
        ...
    }
    else if (command != null && command.equals("delMember"))
    {
        String id = request.getParameter("id");
        dao.delMember(id);
    }
}
```

command 값이 delMember인
경우 ID를 가져와 SQL문으로 전
달해서 삭제합니다.

7.5 회원 정보 삭제하기

```
...  
for (int i = 0; i < list.size(); i++)  
{  
    MemberVO memberVO = (MemberVO) list.get(i);  
    String id = memberVO.getId();  
    String pwd = memberVO.getPwd();  
    String name = memberVO.getName();  
    int age = memberVO.getAge();  
    Date joinDate = memberVO.getJoinDate();  
    out.print("<tr><td>" + id + "</td><td>"  
        + pwd + "</td><td>" + name + "</td><td>"  
        + age + "</td><td>" + joinDate + "</td><td>"  
        + "<a href='/pro07/member3?command=delMember&id=" + id + "'>삭제 </a></td></tr>")  
    }  
    out.print("</table></body></html>");  
    out.print("<a href='/pro07/memberForm.html'>새 회원 등록하기</a>");  
}  
}
```

삭제를 클릭하면 command 값과
회원 ID를 서블릿으로 전송합니다.

7.5 회원 정보 삭제하기

2. MemberDAO 클래스를 다음과 같이 수정합니다.

코드 7-12 pro07/src/sec02/ex02/MemberDAO.java

```
public void delMember(String id)
{
    try
    {
Connection con = dataFactory.getConnection();
Statement stmt = con.createStatement();

        String query = "delete from t_member" + " where id=?";
        System.out.println("prepareStatement:" + query);
        pstmt = con.prepareStatement(query);
        pstmt.setString(1, id);
        pstmt.executeUpdate();
        pstmt.close();
    } catch (Exception e)
    {
```

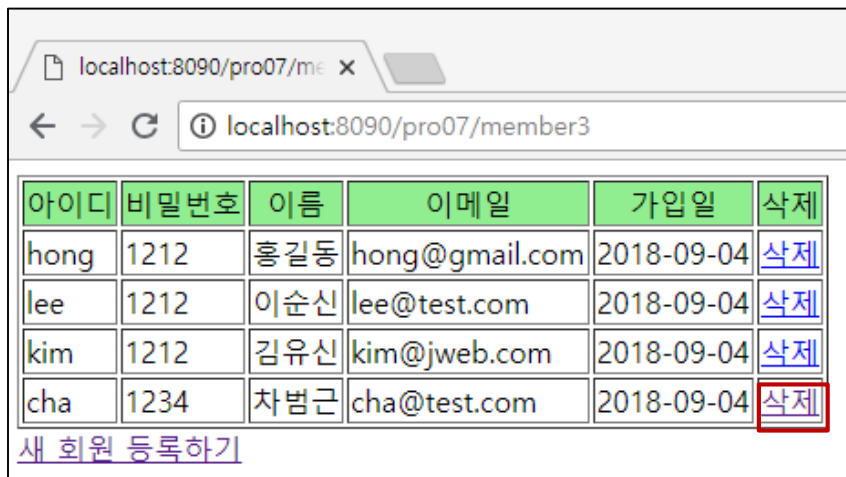
delete문을 문자열로 만듭니다.

첫 번째 '?'에 전달된 ID를 인자로 넣습니다.

delete문을 실행해 테이블에서 해당 ID의 회원 정보를 삭제합니다.

7.5 회원 정보 삭제하기

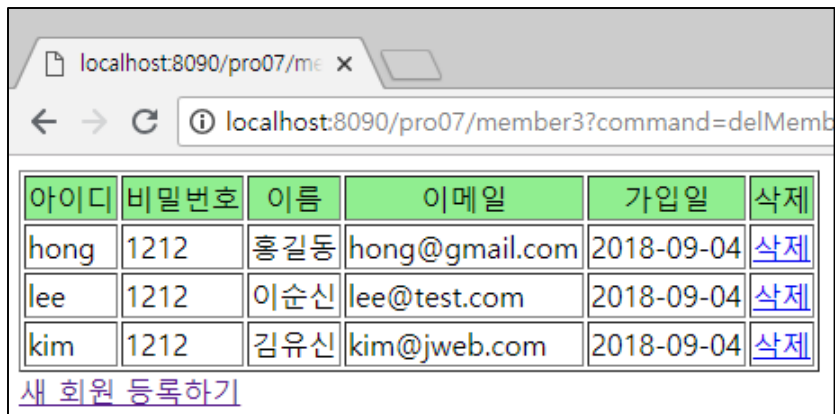
3. `http://localhost:8090/member3`로 요청한 후 삭제를 클릭합니다.



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

[새 회원 등록하기](#)

4. 회원 정보를 삭제한 후 남은 회원 정보가 다시 출력됩니다.



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

[새 회원 등록하기](#)

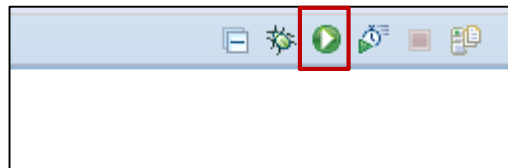
이클립스 디버깅 기능

이클립스 디버깅 기능 사용하기

1. sec02.ex02.MemberServlet 클래스의 doHandle() 메서드 28번 줄 번호 옆을 마우스로 더블클릭해 중단점 (breakpoint)을 만듭니다.

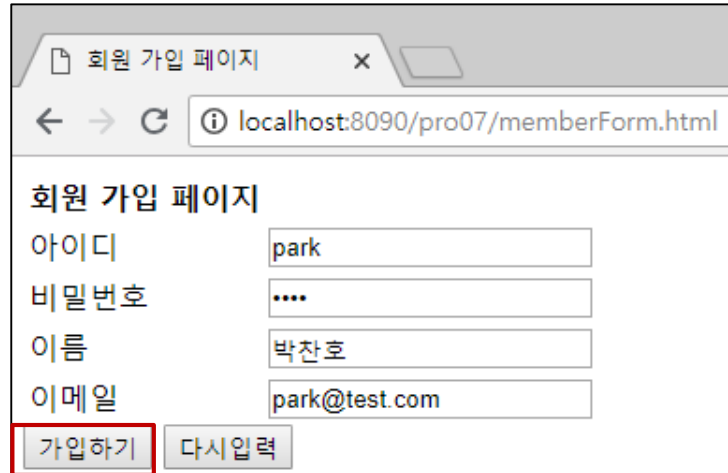
```
27 private void doHandle(HttpServletRequest request, HttpServletResponse response) {
28     request.setCharacterEncoding("utf-8");
29     response.setContentType("text/html;charset=utf-8");
30     MemberDAO dao=new MemberDAO();
31     PrintWriter out=response.getWriter();
32     String command=request.getParameter("command");
33     if(command!= null && command.equals("addMember")){
34         String _id=request.getParameter("id");
35         String _pwd=request.getParameter("pwd");
36         String _name=request.getParameter("name");
37         String _email=request.getParameter("email");
38
39         MemberVO vo=new MemberVO();
40         vo.setId(_id);
41         vo.setPwd(_pwd);
42         vo.setName(_name);
43         vo.setEmail(_email);
44         dao.addMember(vo);
45     }else if(command!= null && command.equals("delMember")) {
46         String id = request.getParameter("id");
```

2. 톰캣 실행 시 버그 아이콘을 클릭해 디버그 모드로 실행합니다.



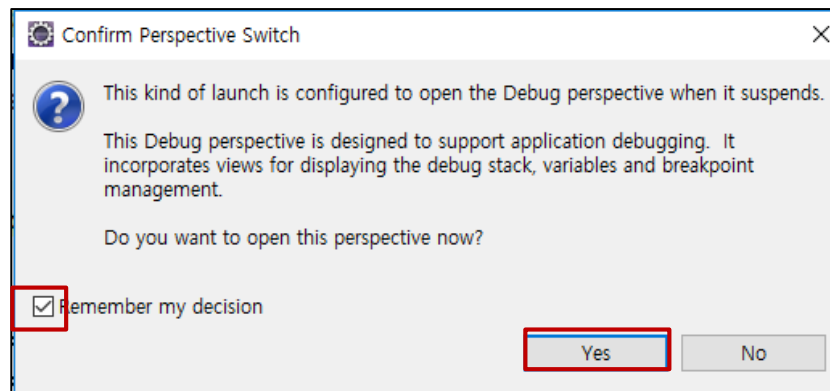
이클립스 디버깅 기능

3. 회원 가입 페이지를 열어 새 회원 정보를 입력한 후 가입하기를 클릭합니다.



A screenshot of a web browser window. The address bar shows 'localhost:8090/pro07/memberForm.html'. The page title is '회원 가입 페이지'. The form contains the following fields: '아이디' (ID) with value 'park', '비밀번호' (Password) with masked characters '....', '이름' (Name) with value '박찬호', and '이메일' (Email) with value 'park@test.com'. At the bottom, there are two buttons: '가입하기' (Join) and '다시입력' (Re-enter). The '가입하기' button is highlighted with a red rectangle.

4. 웹 브라우저의 요청을 받은 이클립스가 디버그 모드로 전환하기 위한 동의 요청창이 나타나면 Remember my decision 옵션 체크박스에 체크한 후 Yes를 클릭합니다.



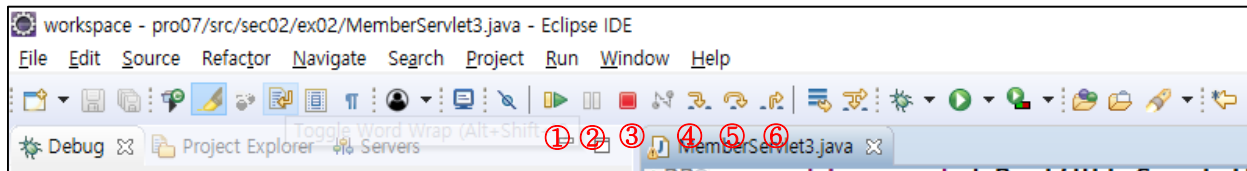
이클립스 디버깅 기능

5. 이클립스가 디버그 모드로 전환되고 실행은 중단점에서 정지합니다.

```
27= private void doHandle(HttpServletRequest request, HttpServletResponse response)
28 request.setCharacterEncoding("utf-8");
29 response.setContentType("text/html;charset=utf-8");
30 MemberDAO dao=new MemberDAO();
31 PrintWriter out=response.getWriter();
32 String command=request.getParameter("command");
33 if(command!= null && command.equals("addMember")){
34     String _id=request.getParameter("id");
35     String _pwd=request.getParameter("pwd");
36     String _name=request.getParameter("name");
37     String _email=request.getParameter("email");
38
39     MemberVO vo=new MemberVO();
40     vo.setId(_id);
41     vo.setPwd(_pwd);
42     vo.setName(_name);
43     vo.setEmail(_email);
44     dao.addMember(vo);
```

이클립스 디버깅 기능

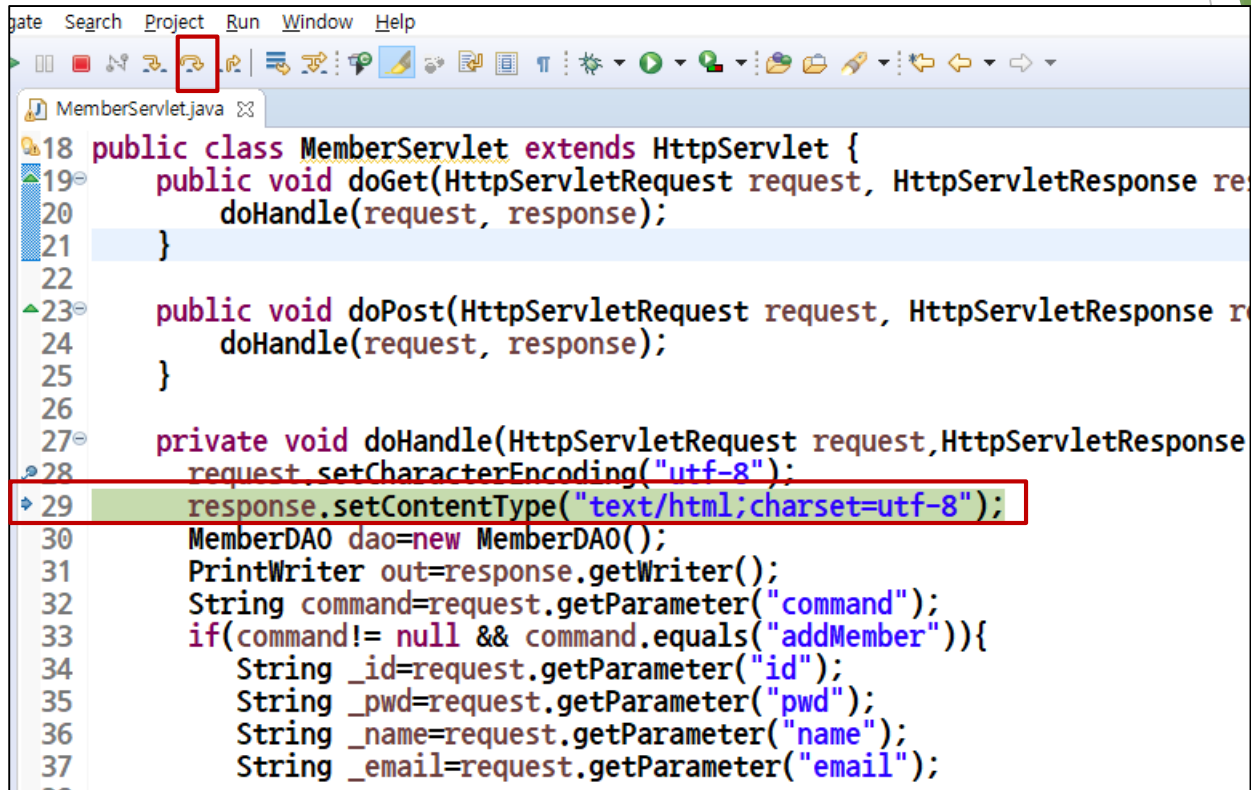
6. 이클립스 상단의 여러 가지 버튼을 이용해 디버깅을 수행합니다.



- ① **Resume:** 다음 중단점을 만날 때까지 진행합니다(F8).
- ② **Suspend:** 현재 동작하고 있는 스레드를 멈춥니다.
- ③ **Terminate:** 프로그램을 종료합니다(CTRL+ F2).
- ④ **Step Into:** 메서드가 존재할 경우 그 메서드로 이동합니다(F5).
- ⑤ **Step Over:** 한 라인씩 실행합니다(F6).
- ⑥ **Step Return:** 'Step Into'로 이동한 메서드에서 원래 위치로 복귀합니다(F7).

이클립스 디버깅 기능

7. 가장 자주 사용하는 Step Over 아이콘을 클릭해 중단점에서 다음 라인으로 이동합니다.



이클립스 디버깅 기능

8. 계속해서 Step Over 아이콘을 클릭해 32행의 실행문을 실행한 후 변수 `command` 위에 마우스 포인터를 놓으면 `command`의 값을 팝업창으로 표시해 줍니다.



The screenshot shows the Eclipse IDE with a Java file named `MemberServlet.java`. The code is as follows:

```
27 private void doHandle(HttpServletRequest request, HttpServletResponse response) {
28     request.setCharacterEncoding("utf-8");
29     response.setContentType("text/html; charset=utf-8");
30     MemberDAO dao = new MemberDAO();
31     PrintWriter out = response.getWriter();
32     String command = request.getParameter("command");
33     if (command != null) {
34         String name = request.getParameter("name");
35         String email = request.getParameter("email");
36         String password = request.getParameter("password");
37         String confirmPassword = request.getParameter("confirmPassword");
38         if ("addMember".equals(command)) {
39             MemberVO vo = new MemberVO();
40             vo.setName(name);
41             vo.setEmail(email);
42             vo.setPassword(password);
43             dao.addMember(vo);
44         } else if (command != null && command.equals("delMember")) {
45             // ...
46         }
47     }
48 }
```

A debugger popup is visible over line 33, showing the value of the variable `command`. The popup displays:

- `command= "addMember" (id=598)`
- `coder= 0`
- `hash= -184929637`
- `value= (id=599)`

The value `addMember` is highlighted with a red box.

이클립스 디버깅 기능

9. Step Over 아이콘을 계속 클릭하면 if문이 참이므로 회원 정보를 가져옵니다.



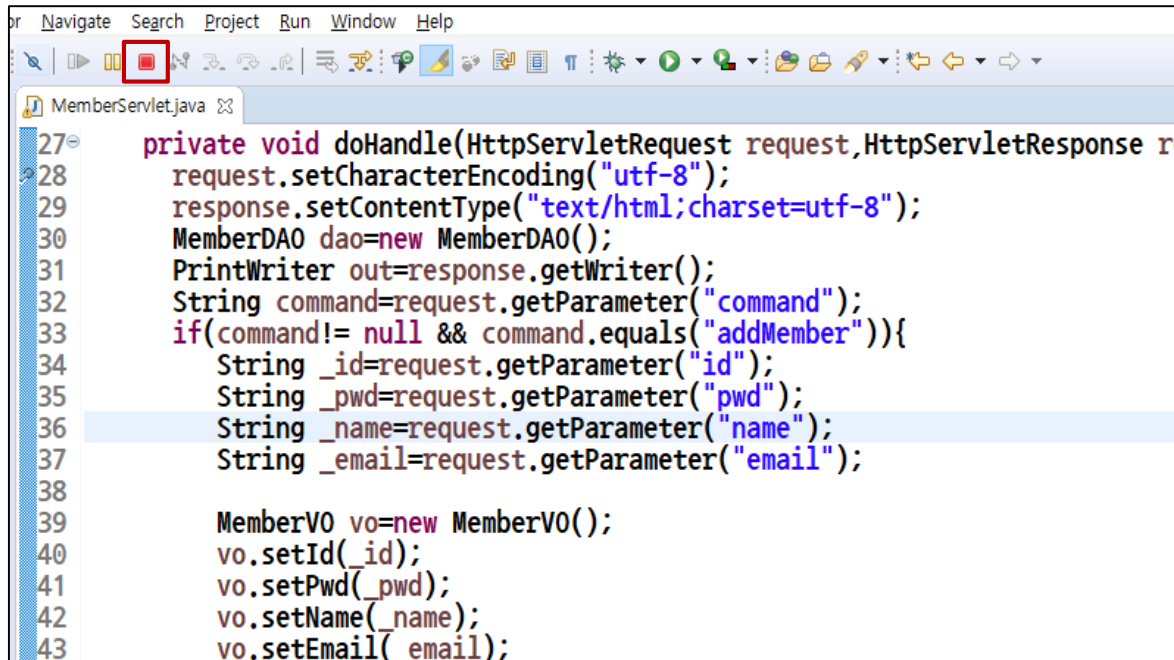
```
MemberServlet.java
27 private void doHandle(HttpServletRequest request, HttpServletResponse response) {
28     request.setCharacterEncoding("utf-8");
29     response.setContentType("text/html;charset=utf-8");
30     MemberDAO dao=new MemberDAO();
31     PrintWriter out=response.getWriter();
32     String command=request.getParameter("command");
33     if(command!= null && command.equals("addMember")){
34         String _id=request.getParameter("id");
35         String
36         String
37         String
38
39         MemberVO vo=new MemberVO();
40         vo.setName("park");
41         vo.setPassword("1234");
42         vo.setNickname("park");
43         vo.setEmail("park@naver.com");
44         dao.addMember(vo);
45     }else if(command!= null && command.equals("delMember")) {
46         String id = request.getParameter("id");
```

Debugger tooltip for line 34:

- id= "park" (id=652)
 - coder= 0
 - hash= 3433450
 - value= (id=654)

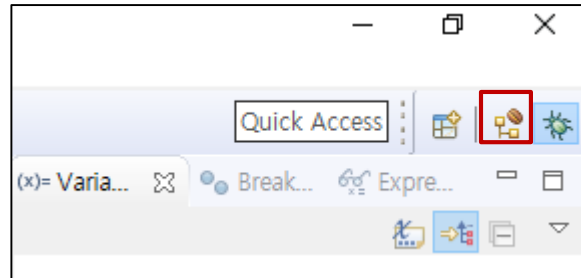
이클립스 디버깅 기능

10. 디버깅이 끝났으면 Resume 아이콘을 클릭해 다음 중단점으로 이동합니다(중단점은 여러 개를 지정할 수 있습니다). 중단점이 더 없으면 종료합니다.



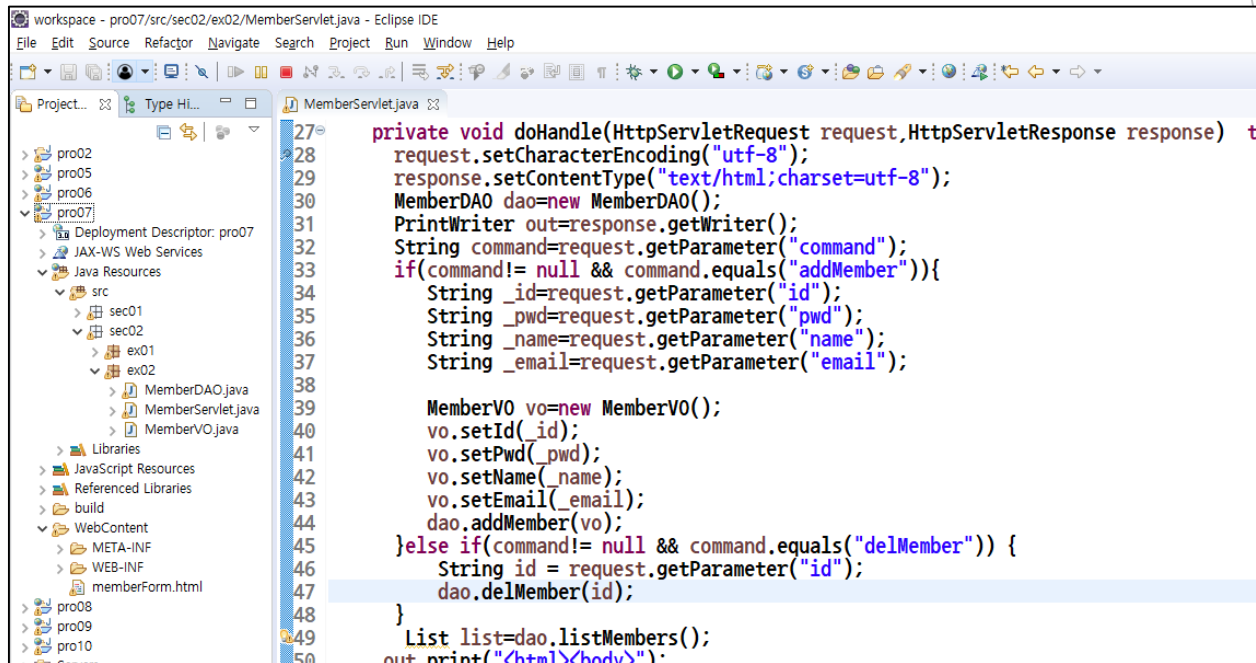
이클립스 디버깅 기능

11. 이클립스를 다시 편집 모드로 되돌리기 위해 오른쪽 상단의 Java EE Perspective 아이콘을 클릭합니다.



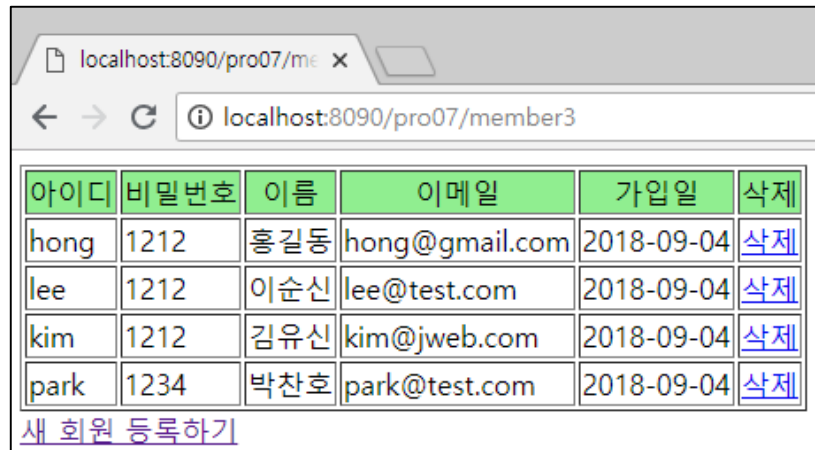
이클립스 디버깅 기능

12. 이클립스를 편집 모드로 전환합니다.



이클립스 디버깅 기능

13. 정상적으로 회원이 등록된 것을 확인할 수 있습니다.



The screenshot shows a web browser window with the address bar displaying 'localhost:8090/pro07/member3'. The main content area contains a table with 6 columns: 아이디 (ID), 비밀번호 (Password), 이름 (Name), 이메일 (Email), 가입일 (Join Date), and 삭제 (Delete). The table lists four members: hong, lee, kim, and park. Each member row has a blue '삭제' (Delete) link in the last column. Below the table is a link labeled '새 회원 등록하기' (Register New Member).

아이디	비밀번호	이름	이메일	가입일	삭제
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제
lee	1212	이순신	lee@test.com	2018-09-04	삭제
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제
park	1234	박찬호	park@test.com	2018-09-04	삭제

[새 회원 등록하기](#)