

# 6장 스프링 JDBC 기능

강사 김영석

A top-down view of a wooden desk. On the desk, there is a silver laptop with a black keyboard, a pair of black-rimmed glasses, a white coffee cup with a yellow handle, and a small green succulent in a dark pot. The word 'CONTENT' is written in large, white, sans-serif capital letters on the left side of the image.

# CONTENT

1

스프링 JDBC로 데이터베이스오  
연동 설정하기

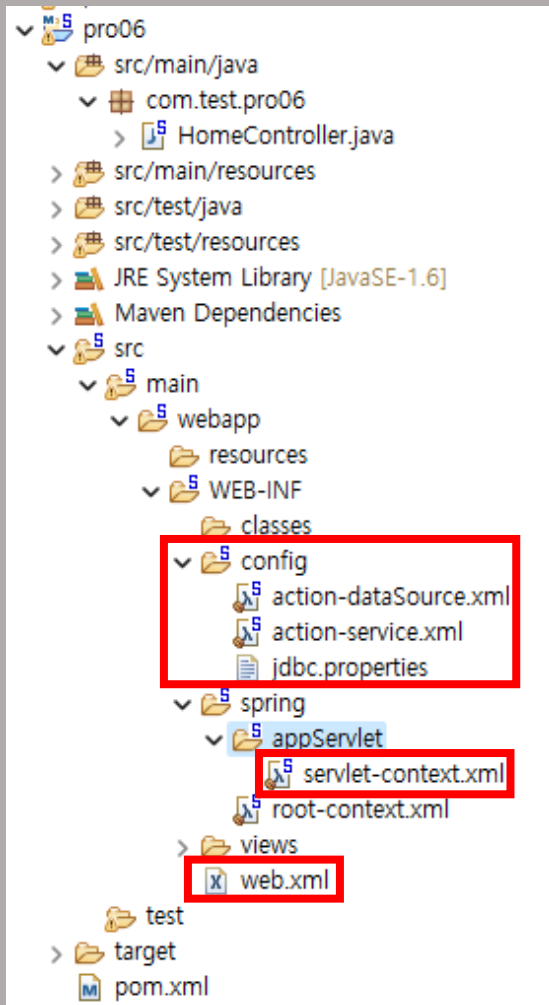
2

JdbcTemplate 클래스 이용해  
회원 정보 조회하기

# 1. 스프링 JDBC로 데이터베이스와 연동 설정하기


## ✓ 스프링에서 제공하는 JDBC 특징

- 기존 JDBC의 장점과 단점을 유지하면서 단점을 보완 했다.
- 간결한 API 뿐만 아니라 확장된 JDBC의 기능도 제공한다.
- 실제 개발에서는 JDBC 기능보다는 마이바티스나 하이버네이트 같은 데이터베이스 연동 관련 프레임워크를 사용하지만 스프링 JDBC의 기본적인 기능을 알아두면 도움이 된다.




파일	설명
web.xml	ContextLoaderListener를 이용해 빈 설정 XML 파일들을 읽어 들인다.
servlet-context.xml	스프링에서 필요한 여러 가지 빈을 설정한다.
action-datasource.xml	스프링 JDBC 설정에 필요한 정보를 설정한다.
jdbc.properties	데이터베이스 연결 정보를 저장한다.
action-service.xml	서비스 빈 생성을 설정한다.





```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2
5
6   <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
7   <context-param>
8     <param-name>contextConfigLocation</param-name>
9     <param-value>
10       /WEB-INF/spring/root-context.xml
11       /WEB-INF/spring/config/action-service.xml
12       /WEB-INF/spring/config/action-dataSource.xml
13     </param-value>
14   </context-param>
15
16   <!-- Creates the Spring Container shared by all Servlets and Filters -->
17   <listener>
18     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
19   </listener>
20
21   <!-- Processes application requests -->
22   <servlet>
23     <servlet-name>appServlet</servlet-name>
24     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
25     <init-param>
26       <param-name>contextConfigLocation</param-name>
27       <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
28     </init-param>
29     <load-on-startup>1</load-on-startup>
30   </servlet>
```

```
servlet-context.xml
19 <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
20   <beans:property name="prefix" value="/WEB-INF/views/" />
21   <beans:property name="suffix" value=".jsp" />
22 </beans:bean>
23 <beans:bean id="memberController" class="com.test.pro06.member.controller.MemberControllerImpl">
24   <beans:property name="methodNameResolver">
25     <ref bean="methodResolver"></ref>
26   </beans:property>
27   <beans:property name="memberService" ref="memberService"></beans:property>
28 </beans:bean>
29 <beans:bean id="methodResolver"
30   class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
31   <beans:property name="mappings">
32     <beans:props>
33       <beans:prop key="/member/listMembers.do">listMembers</beans:prop>
34       <beans:prop key="/member/addMember.do">addMember</beans:prop>
35       <beans:prop key="/member/memberForm.do">memberForm</beans:prop>
36       <beans:prop key="/member/memberDetail.do">memberDetail</beans:prop>
37     </beans:props>
38   </beans:property>
39 </beans:bean>
40 <beans:bean id="urlMapping"
41   class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
42   <beans:property name="mappings">
43     <beans:props>
44       <beans:prop key="/member/*.do">memberController</beans:prop>
45     </beans:props>
46   </beans:property>
47 </beans:bean>
48 <context:component-scan base-package="com.test.pro06" />
49 </beans:beans>
```




```
*action-service.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans
5
6       <bean id="memberService"
7             class="com.test.pro06.member.service.MemberServiceImpl">
8         <property name="memberDAO" ref="memberDAO"></property>
9     </bean>
10 </beans>
```


pro06/pom.xml

```
106<dependency>
107    <groupId>javax.servlet</groupId>
108    <artifactId>jstl</artifactId>
109    <version>1.2</version>
110</dependency>
111
112<!-- 스프링 JDBC -->
113<dependency>
114    <groupId>org.springframework</groupId>
115    <artifactId>spring-jdbc</artifactId>
116    <version>${org.springframework-version}</version>
117</dependency>
118
119<!-- 오라클 드라이버 -->
120<dependency>
121    <groupId>com.oracle.ojdbc</groupId>
122    <artifactId>ojdbc8</artifactId>
123    <version>19.3.0.0</version>
124</dependency>
125
126<!-- Test -->
127<dependency>
```





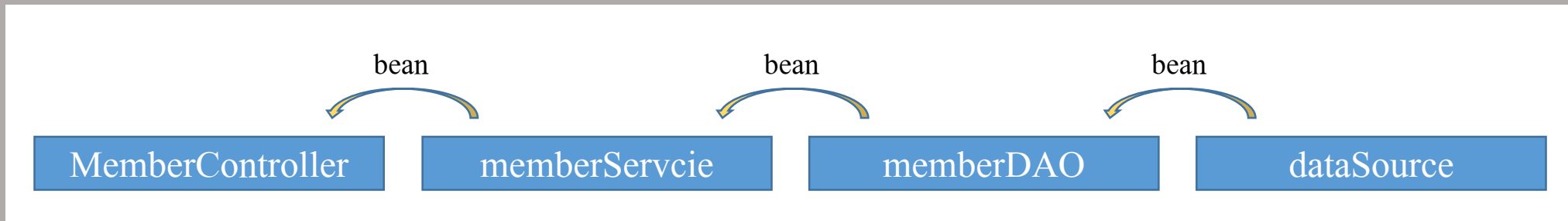
```
*action-datasource.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans"
5
6     <bean id="propertyConfigurer"
7         class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
8         <property name="locations">
9             <list>
10                 <value>/WEB-INF/config/jdbc.properties</value>
11             </list>
12         </property>
13     </bean>
14     <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
15         <property name="driverClass" value="${jdbc.driverClassName}"></property>
16         <property name="url" value="${jdbc.url}"></property>
17         <property name="username" value="${jdbc.username}"></property>
18         <property name="password" value="${jdbc.password}"></property>
19     </bean>
20     <bean id="memberDAO" class="com.test.pro06.member.dao.MemberDAOImpl">
21         <property name="dataSource" ref="dataSource"></property>
22     </bean>
23 </beans>
```



```
jdbc.properties
1 jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
2 jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
3 jdbc.username=system
4 jdbc.password=oracle|
```

XML 파일에 오류 표시가 나는 이유는 아직 클래스를 만들지 않아서다.  
클래스를 만들고 나면 오류 표시는 사라진다.

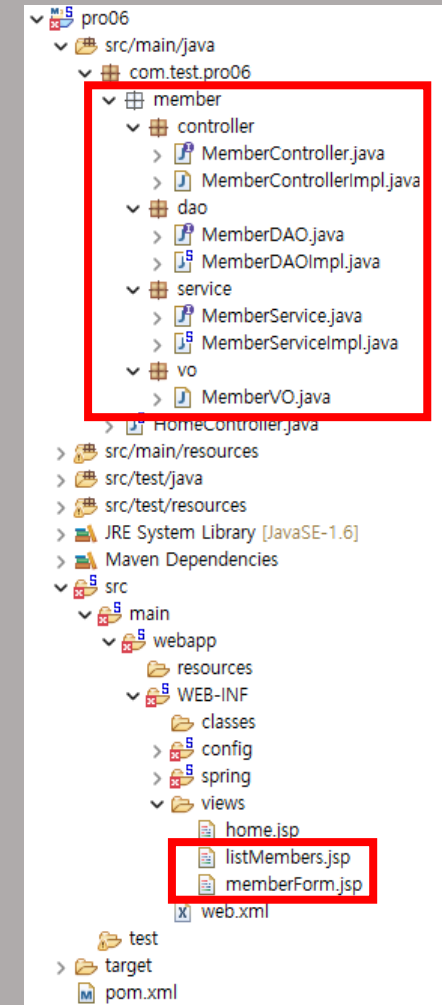
✓ XML 파일 설정에 의한 빈 주입 과정은 다음과 같다



# Quiz

✓ 그림과 같이 파일을 준비 해 보세요.

- 각 패키지는 vo 패키지를 제외하고 인터페이스와 클래스로 되어 있다.
- MemberControllerImpl 클래스의 메소드를 url Mapping 하여 각 url 이 잘 작동 하는지 구현 해 보세요.




## 2. JdbcTemplate 클래스 이용해 회원 정보 조회하기

✓ JdbcTemplate 클래스에서 제공하는 SQL 관련 메서드

기능	메서드
insert, update, delete 관련 메서드	int update(String query)
	int update(String query, Object[] args)
	int update(String query, Object[] args, int[] argTypes)
select 기능 메서드	int queryForInt(String query)
	long queryForInt(String query, Object[] args)
	long queryForLong(String query, Object[] args)
	Object queryForObject(String query, Class requiredType)
	List queryForList(String query)
	List queryForList(String query, Object[] args)






```
MemberVO.java
1 package com.test.pro06.member.vo;
2
3 import java.sql.Date;
4
5 public class MemberVO {
6     private String id;
7     private String pwd;
8     private String name;
9     private String email;
10    private Date joinDate;
11    public String getId() {
12        return id;
13    }
14    public void setId(String id) {
15        this.id = id;
16    }
17    public String getPwd() {
18        return pwd;
19    }
20    public void setPwd(String pwd) {
21        this.pwd = pwd;
22    }
23    public String getName() {
24        return name;
25    }
}
```

MemberController.java

```
1 package com.test.pro06.member.controller;
2
3 import javax.servlet.http.HttpServletRequest;
4 import javax.servlet.http.HttpServletResponse;
5
6 import org.springframework.web.servlet.ModelAndView;
7
8 public interface MemberController {
9     public ModelAndView listMembers(HttpServletRequest request, HttpServletResponse response)
10         throws Exception;
11     public ModelAndView memberForm(HttpServletRequest request, HttpServletResponse response)
12         throws Exception;
13 }
```



```
MemberService.java ✕ *MemberDAO.java
1 package com.test.pro06.member.service;
2
3 import java.util.List;
4
5 public interface MemberService {
6     public List listMembers();
7 }
```


```
*MemberDAO.java ✕
1 package com.test.pro06.member.dao;
2
3 import java.util.List;
4
5 public interface MemberDAO {
6     public List listMembers();
7 }
```

controller, service, dao 모두 동일한 메서드명으로 지정했다.

```
MemberControllerImpl.java
1 package com.test.pro06.member.controller;
2
3 import java.util.List;
4
12
13 public class MemberControllerImpl extends MultiActionController implements MemberController{
14     private MemberService memberService;
15     public void setMemberService(MemberService memberService) {
16         this.memberService = memberService;
17     }
18     public ModelAndView listMembers(HttpServletRequest request, HttpServletResponse response)
19         throws Exception{
20         String viewName = getViewName(request);
21         List membersList = memberService.listMembers();
22         ModelAndView mav = new ModelAndView(viewName);
23         mav.addObject("membersList", membersList);
24         return mav;
25     }
26     public ModelAndView memberForm(HttpServletRequest request, HttpServletResponse response)
27         throws Exception{
28         String viewName = getViewName(request);
29         ModelAndView mav = new ModelAndView(viewName);
30         return mav;
31     }
32     private String getViewName(HttpServletRequest request) {
33         String contextPath = request.getContextPath();
34         String uri = (String)request.getAttribute("javax.servlet.include.request_uri");
35         if(uri == null || uri.trim().equals("")) {
36             uri = request.getRequestURI();
37         }
```


getViewName 메서드는 이전 프로젝트에서 가져온다.





```
*MemberServiceImpl.java
1 package com.test.pro06.member.service;
2
3 import java.util.List;
4
5 import com.test.pro06.member.dao.MemberDAO;
6
7 public class MemberServiceImpl implements MemberService{
8     private MemberDAO memberDAO;
9     public void setMemberDAO(MemberDAO memberDAO) {
10         this.memberDAO = memberDAO;
11     }
12     public List listMembers() {
13         // TODO Auto-generated method stub
14         List membersList = null;
15         membersList = memberDAO.listMembers();
16         return membersList;
17     }
18 }
```

```
*MemberDAOImpl.java
1 package com.test.pro06.member.dao;
2
3 import java.sql.ResultSet;
4
14
15 public class MemberDAOImpl implements MemberDAO{
16     private JdbcTemplate jdbcTemplate;
17     public void setDataSource(DataSource dataSource) {
18         this.jdbcTemplate = new JdbcTemplate(dataSource);
19     }
20     @Override
21     public List listMembers() {
22         String query = "select id,pwd,name,email,joinDate from t_member order by joinDate desc";
23         List membersList = new ArrayList();
24         membersList = this.jdbcTemplate.query(query, new RowMapper() {
25             public Object mapRow(ResultSet rs, int rowNum) throws SQLException {
26                 MemberVO memberVO = new MemberVO();
27                 memberVO.setId(rs.getString("id"));
28                 memberVO.setPwd(rs.getString("pwd"));
29                 memberVO.setName(rs.getString("name"));
30                 memberVO.setEmail(rs.getString("email"));
31                 memberVO.setJoinDate(rs.getDate("joinDate"));
32                 return memberVO;
33             }
34         });
35         return membersList;
36     }
37 }
```



시작 페이지 x 오라클 x

워크시트 | 질의 작성기

```
create table t_member(  
    id varchar2(50) primary key not null,  
    pwd varchar2(50) not null,  
    name varchar2(50) not null,  
    email varchar2(100) not null,  
    joinDate date default sysdate);  
  
insert into t_member values('hong', '1234', '홍길동', 'hong@test.com', sysdate);  
insert into t_member values('lee', '1234', '이순신', 'lee@test.com', sysdate);  
insert into t_member values('kim', '1234', '김유신', 'kim@test.com', sysdate);  
  
commit;
```

```
listMembers.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"
3   isELIgnored="false" %>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <%
6   request.setCharacterEncoding("UTF-8");
7 %>
8 <html>
9 <head>
10 <meta charset=UTF-8">
11 <title>회원 정보 출력창</title>
12 </head>
13 <body>
14 <table border="1" align="center" width="80%">
15   <tr align="center" bgcolor="lightgreen">
16     <td><b>아이디</b></td>
17     <td><b>비밀번호</b></td>
18     <td><b>이름</b></td>
19     <td><b>이메일</b></td>
20     <td><b>가입일</b></td>
21   </tr>
22
23   <c:forEach var="member" items="${membersList}" >
24     <tr align="center">
25       <td>${member.id}</td>
26       <td>${member.pwd}</td>
27       <td>${member.name}</td>
28       <td>${member.email}</td>
29       <td>${member.joinDate}</td>
30     </tr>
31   </c:forEach>
32 </table>
33 <a href="/pro06/member/memberForm.do">
34 <h1 style="text-align:center">회원가입</h1></a>
35 </body>
36 </html>
```



회원 정보 출력창

localhost:8080/pro06/member/listMembers.do

☆ 시크릿 모드

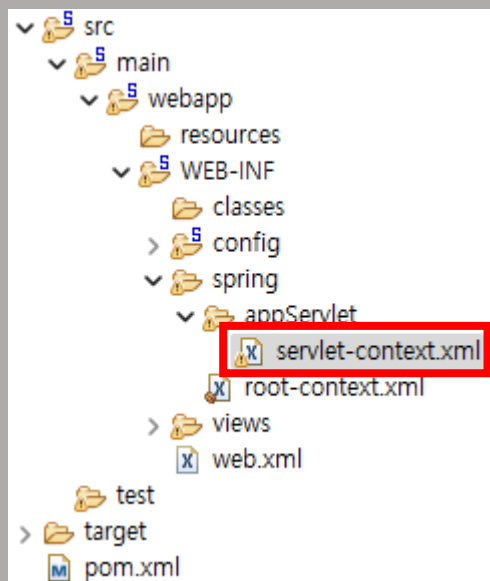
아이디	비밀번호	이름	이메일	가입일
kim	1234	김유신	kim@test.com	2023-09-10
lee	1234	이순신	lee@test.com	2023-09-10
hong	1234	홍길동	hong@test.com	2023-09-10

[회원가입](#)

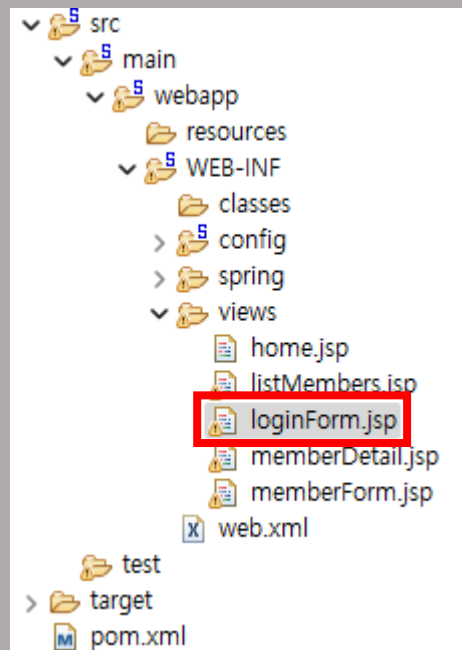
# Quiz

- ✓ JdbcTemplate 을 이용하여 특정 회원 정보 조회를 해보세요.
  - memberDetail.do 를 이용해서 구현한다.
  - sql문은 “select \* from t\_member where id=?” 로 해야 한다.
  - JdbcTemplate.queryForObject (sql문, new RowMapper(), 전달할 값)을 사용한다.
  - jdbcTemplate.query() 는 여러 개의 값을 조회하는 List 만 가능하고,  
JdbcTemplate.queryForObject() 는 한 개의 값을 조회 할 수 있다.

## ✓ JdbcTemplate 로그인 여부

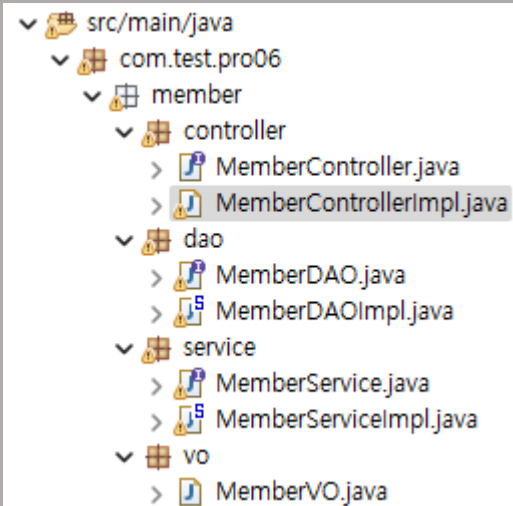


```
servlet-context.xml
30     class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
31     <beans:property name="mappings">
32     <beans:props>
33         <beans:prop key="/member/listMembers.do">listMembers</beans:prop>
34         <beans:prop key="/member/addMember.do">addMember</beans:prop>
35         <beans:prop key="/member/memberForm.do">memberForm</beans:prop>
36         <beans:prop key="/member/memberDetail.do">memberDetail</beans:prop>
37         <beans:prop key="/member/memberAdd.do">memberAdd</beans:prop>
38         <beans:prop key="/member/loginForm.do">loginForm</beans:prop>
39         <beans:prop key="/member/login.do">login</beans:prop>
40     </beans:props>
41     </beans:property>
42 </beans:bean>
43 <beans:bean id="urlMapping"
44     class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
45     <beans:property name="mappings">
46     <beans:props>
47         <beans:prop key="/member/*.do">memberController</beans:prop>
48     </beans:props>
49     </beans:property>
50 </beans:bean>
51 <context:component-scan base-package="com.test.pro06" />
52 </beans:beans>
```



```
loginForm.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>로그인</title>
8 </head>
9 <body>
10     <form method="post" action="/pro06/member/login.do">
11         <table border="1" width="80%" align="center">
12             <tr align="center">
13                 <td>아이디</td>
14                 <td>비밀번호</td>
15             </tr>
16             <tr align="center">
17                 <td><input type="text" name="id" size="20"></td>
18                 <td><input type="password" name="pwd" size="20"></td>
19             </tr>
20             <tr>
21                 <td colspan="2" align="center">
22                     <input type="submit" value="로그인">
23                     <input type="reset" value="다시 입력">
24                 </td>
25             </tr>
26         </table>
27     </form>
28 </body>
29 </html>
```





```
MemberControllerImpl.java
83 @Override
84 public ModelAndView loginForm(HttpServletRequest request, HttpServletResponse response) throws Exception {
85     // TODO Auto-generated method stub
86     String viewName = getViewName(request);
87     ModelAndView mav = new ModelAndView(viewName);
88     return mav;
89 }
90 @Override
91 public ModelAndView login(HttpServletRequest request, HttpServletResponse response) throws Exception {
92     // TODO Auto-generated method stub
93     response.setContentType("text/html;charset=utf-8");
94     String id = request.getParameter("id");
95     String pwd = request.getParameter("pwd");
96
97     int result = memberService.login(id, pwd);
98
99     PrintWriter out = response.getWriter();
100
101     if (result == 1) {
102         out.write("<script>");
103         out.write("alert('로그인 되었습니다. ' + ' ');");
104         out.write("</script>");
105     } else {
106         out.write("<script>");
107         out.write("alert('로그인 실패 했습니다. ')");
108         out.write("</script>");
109     }
110
111     return null;
112 }
113 }
```

```

61     }
62     }, id);
63     return member;
64 }
65 @Override
66 public int login(String id, String pwd) throws SQLException {
67     // TODO Auto-generated method stub
68     String sql = "select count(*) from t_member where id=? and pwd=?";
69
70     int result = this.jdbcTemplate.queryForObject(sql, Integer.class, id, pwd);
71     return result;
72 }
73
74 }

```

로그인

localhost:8080/pro06/memb...

아이디

비밀번호

hong

로그인

다시 입력

localhost:8080/pro06/member/

localhost:8080 내용:  
로그인 되었습니다.

확인

## ✓ JdbcTemplate 입력, 수정, 삭제

- update(sql, 데이터....) 를 이용하여 입력, 수정, 삭제를 할 수 있다.

```

servlet-context.xml
28     </beans:bean>
29     <beans:bean id="methodResolver"
30         class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
31         <beans:property name="mappings">
32             <beans:props>
33                 <beans:prop key="/member/listMembers.do">listMembers</beans:prop>
34                 <beans:prop key="/member/addMember.do">addMember</beans:prop>
35                 <beans:prop key="/member/memberDetail.do">memberDetail</beans:prop>
36                 <beans:prop key="/member/loginForm.do">loginForm</beans:prop>
37                 <beans:prop key="/member/login.do">login</beans:prop>
38                 <beans:prop key="/member/memberAdd.do">memberAdd</beans:prop>
39                 <beans:prop key="/member/memberForm.do">memberForm</beans:prop>
40             </beans:props>
41         </beans:property>

```

MemberControllerImpl.java

```
53     }
54     return fileName;
55 }
56
57 public ModelAndView memberForm(HttpServletRequest request, HttpServletResponse response)
58     throws Exception{
59     String viewName = getViewName(request);
60     ModelAndView mav = new ModelAndView(viewName);
61     return mav;
62 }
63
64 @Override
65 public ModelAndView memberAdd(HttpServletRequest request, HttpServletResponse response) throws Exception {
66     // TODO Auto-generated method stub
67     MemberVO vo = new MemberVO();
68     vo.setId(request.getParameter("id"));
69     vo.setPwd(request.getParameter("pwd"));
70     vo.setName(request.getParameter("name"));
71     vo.setEmail(request.getParameter("email"));
72
73     memberService.memberAdd(vo);
74     return new ModelAndView("redirect:listMembers.do");
75 }
76
77 @Override
78 public ModelAndView memberDetail(HttpServletRequest request, HttpServletResponse response) throws Exception {
79     // TODO Auto-generated method stub
80     String id = "hong";
```



```
*memberForm.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7   <title>회원 가입창</title>
8 </head>
9 <body>
10   <form method="post" action="/pro06/member/memberAdd.do">
11     <table border="1" width="80%" align="center">
12       <tr align="center">
13         <td>아이디</td>
14         <td>비밀번호</td>
15         <td>이름</td>
16         <td>이메일</td>
17       </tr>
18       <tr align="center">
19         <td><input type="text" name="id" size="20"></td>
20         <td><input type="password" name="pwd" size="20"></td>
21         <td><input type="text" name="name" size="20"></td>
22         <td><input type="email" name="email" size="20"></td>
23       </tr>
24       <tr>
25         <td colspan="4" align="center">
26           <input type="submit" value="회원가입">
27           <input type="reset" value="다시 입력">
28         </td>
29       </tr>
30     </table>
31   </form>
32 </body>
33 </html>
```

```
*MemberDAOImpl.java 83
30         memberVO.setName(rs.getString("name"));
31         memberVO.setEmail(rs.getString("email"));
32         memberVO.setJoinDate(rs.getDate("joinDate"));
33         return memberVO;
34     }
35     });
36     return membersList;
37 }
38 @Override
39 public void memberAdd(MemberVO vo) {
40     // TODO Auto-generated method stub
41     String query = "insert into t_member values(?,?,?,sysdate)";
42     int result = jdbcTemplate.update(query, vo.getId(), vo.getPwd(), vo.getName(), vo.getEmail());
43     System.out.println(result + "행이 추가 되었습니다.");
44 }
45 @Override
46 public MemberVO memberDetail(String id) {
47     // TODO Auto-generated method stub
48     String sql = "select * from t_member where id=?";
49     MemberVO member = (MemberVO) this.jdbcTemplate.queryForObject(sql, new RowMapper() {
```

회원 가입창

localhost:8080/pro06/member/memberForm.do

아이디	비밀번호	이름	이메일
zzz	...	zzz	zzz@test.com
<div>회원가입</div> <div>다시 입력</div>			

회원 정보 출력창

localhost:8080/pro06/member/listMembers.do

아이디	비밀번호	이름	이메일	가입일
zzz	zzz	zzz	zzz@test.com	2023-09-18
kkk	1234	kkk	kkk@test.com	2023-09-13
ddd	111	ddd	ddd@test.com	2023-09-10
ccc	111	ccc	ccc@test.com	2023-09-10
bbb	111	111	bbb@test.com	2023-09-10
aaa	1111	aaa	aaa@test.com	2023-09-10
kim	1234	김유신	kim@test.com	2023-09-10
lee	1234	이순신	lee@test.com	2023-09-10
hong	1234	홍길동	hong@test.com	2023-09-10

# Quiz

- ✓ 수정과 삭제를 구현 해 보세요.
  - memberDel.do 와 memberMod.do 로 구현한다.
  - 삭제와 수정 후 listMembers.do 로 이동한다.



A photograph of a server room with rows of server racks on both sides of a central aisle. The racks have glass doors and internal components are visible, with many small blue lights glowing. The ceiling has several long, rectangular light fixtures. The overall atmosphere is dim and technical.

수고하셨습니다.