

# F

## 21장 회원제 게시판 구현1: 회원 관련 기능

## 21.1 회원 관련 주요 기능

- 회원 가입
- 회원 정보 수정하기
- 로그인하기
- 로그아웃하기
- 로그인한 사람만 특정 기능 실행하기

### 21.2 예제를 위한 데이터베이스 생성

```
[board/sql/ddl.sql]
      -- jsp ddl
      create database board default character set utf8;
      create user 'jspexam'@'localhost' identified by 'jsppw';
      create user 'jspexam'@'%' identified by 'jsppw';
 05
      GRANT ALL PRIVILEGES ON board.* TO 'jspexam'@'localhost';
 80
      GRANT ALL PRIVILEGES ON board.* TO 'jspexam'@'%';
 09
 10
      create table board.member (
 11
          memberid varchar(50) primary key,
          name varchar(50) not null,
 12
 13
          password varchar(10) not null,
          regdate datetime not null
      ) engine=InnoDB default character set = utf8;
 15
 16
 17
      create table board.article (
 18
          article_no int auto_increment primary key,
 19
          writer_id varchar(50) not null,
 20
          writer_name varchar(50) not null,
          title varchar(255) not null,
 21
 22
          regdate datetime not null,
 23
          moddate datetime not null,
           read_cnt int
 25
      ) engine=InnoDB default character set = utf8;
 27
      create table board.article content (
           article_no int primary key,
 29
           content text
 30
      ) engine=InnoDB default character set = utf8;
 31
 32
      select * from member;
```

# 21.3 예제 이클립스 프로젝트 생성

■ [Eclipse > File > New > Dynamic Web Project] 메뉴를 실행한다.

- Project name: board
- Dynamic web module version: 3.1
- board 프로젝트의 WebContent/WEB-INF/lib 폴더에 다음 파일을 복사한다.
  - commons-dbcp2-2.1.jar
  - commons-logging-1.2.jar
  - commons-pool2-2.4.1.jar
  - mysql-connector-java-5.1.35-bin.jar
  - jstl-1.2.jar

### 21.4 커넥션 관련 코드

#### (1) DBCPInitListener.java

■ DB 연동을 하므로 커넥션 관련 코드를 작성해야 한다. 먼저 커넥션 풀을 초기화하기 위한 DBCPInitListener 코드는 아래와 같다.

```
[board/src/jdbc/DBCPInitListener.java]
 01
       package jdbc;
 03
      import java.io.IOException;
       import java.io.StringReader;
       import java.sql.DriverManager;
 06
      import java.util.Properties;
 07
 80
      import javax.servlet.ServletContextEvent;
 09
       import javax.servlet.ServletContextListener;
 10
      import org.apache.commons.dbcp2.ConnectionFactory;
 11
      import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
 13
      import org.apache.commons.dbcp2.PoolableConnection;
      import org.apache.commons.dbcp2.PoolableConnectionFactory;
 14
 15
       import org.apache.commons.dbcp2.PoolingDriver;
 16
       import org.apache.commons.pool2.impl.GenericObjectPool;
 17
       import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
 18
 19
       public class DBCPInitListener implements ServletContextListener {
 20
 21
                 @Override
 22
                 public void contextInitialized(ServletContextEvent sce) {
 23
                           String poolConfig =
                                               sce.getServletContext().getInitParameter("poolConfig");
 24
 25
                           Properties prop = new Properties();
 26
                           try {
 27
                                     prop.load(new StringReader(poolConfig));
 28
                           } catch (IOException e) {
 29
                                     throw new RuntimeException("config load fail", e);
 30
 31
                           loadJDBCDriver(prop);
 32
                           initConnectionPool(prop);
 33
 34
 35
                 private void loadJDBCDriver(Properties prop) {
 36
                           String driverClass = prop.getProperty("jdbcdriver");
 37
                                     Class.forName(driverClass);
 38
 39
                           } catch (ClassNotFoundException ex) {
                                     throw new RuntimeException("fail to load JDBC Driver", ex);
 40
```

```
41
42
43
44
                private void initConnectionPool(Properties prop) {
45
                          try {
46
                                    String jdbcUrl = prop.getProperty("jdbcUrl");
47
                                    String username = prop.getProperty("dbUser");
48
                                    String pw = prop.getProperty("dbPass");
49
50
                                    ConnectionFactory connFactory =
51
                                                        new DriverManagerConnectionFactory(jdbcUrl, username,
52
     pw);
53
54
                                    PoolableConnectionFactory poolableConnFactory =
55
                                                        new PoolableConnectionFactory(connFactory, null);
                                    String validationQuery = prop.getProperty("validationQuery");
56
57
                                    if (validationQuery != null && !validationQuery.isEmpty()) {
58
                                              poolableConnFactory.setValidationQuery(validationQuery);
59
60
                                    GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
61
                                    poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60L * 5L);
                                    poolConfig.setTestWhileIdle(true);
62
                                    int minIdle = getIntProperty(prop, "minIdle", 5);
63
64
                                    poolConfig.setMinIdle(minIdle);
                                    int maxTotal = getIntProperty(prop, "maxTotal", 50);
65
66
                                    poolConfig.setMaxTotal(maxTotal);
67
68
                                    GenericObjectPool<PoolableConnection> connectionPool =
69
                                                                       GenericObjectPool♦(poolableConnFactory,
70
      poolConfig);
71
                                    poolableConnFactory.setPool(connectionPool);
72
73
                                    Class.forName("org.apache.commons.dbcp2.PoolingDriver");
74
                                    PoolingDriver driver = (PoolingDriver)
                                              DriverManager.getDriver("jdbc:apache:commons:dbcp:");
75
76
                                    String poolName = prop.getProperty("poolName");
77
                                    driver.registerPool(poolName, connectionPool);
78
                          } catch (Exception e) {
79
                                    throw new RuntimeException(e);
80
81
82
83
                private int getIntProperty(Properties prop, String propName, int defaultValue) {
84
                          String value = prop.getProperty(propName);
85
                          if (value == null) return defaultValue;
86
                          return Integer.parseInt(value);
87
88
                @Override
89
                public void contextDestroyed(ServletContextEvent sce) {
90
91
92
93
     }
```

#### (2) web.xml

■ DBCPInitListener는 서블릿 컨텍스트 리스너이므로 web.xml에 등록한다.

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03
04
                xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
05
                          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
06
                version="3,1">
07
08
                (listener)
                          \listener-class\rightarrowjdbc.DBCPInitListener\langle/listener-class\rightarrow
09
10
                </listener>
11
                ⟨context-param⟩
12
                          <param-name>poolConfig</param-name>
13
14
                          ⟨param-value⟩
                                     jdbcdriver=com.mysql.jdbc.Driver
15
                                     jdbcUrl=jdbc:mysql://localhost:3306/board?characterEncoding=utf8
16
17
                                     dbUser=jspexam
18
                                     dbPass=jsppw
                                     validationQuery=select 1
19
20
                                     minIdle=3
21
                                     maxTotal=30
                                     poolName=board
22
23
                          ⟨/param-value⟩
24
                </context-param>
25
26
                <filter>
                          <filter-name>encodingFilter</filter-name>
27
28
                          <filter-class>util.CharacterEncodingFilter</filter-class>
29
                          ⟨init-param⟩
30
                                     <param-name>encoding</param-name>
31
                                     ⟨param-value⟩utf-8⟨/param-value⟩
                          </init-param>
32
                </filter>
33
34
35
                <filter-mapping>
36
                          <filter-name>encodingFilter</filter-name>
37
                          <url-pattern>/*</url-pattern>
38
                </filter-mapping>
39
40
                <servlet>
41
                          <servlet-name>ControllerUsingURI</servlet-name>
                          ⟨servlet-class⟩mvc.controller.ControllerUsingURI⟨/servlet-class⟩
42
43
                          ⟨init-param⟩
44
                                     <param-name>configFile</param-name>
45
                                     ⟨param-value⟩
                      /WEB-INF/commandHandlerURI.properties
46
47
                  ⟨/param-value⟩
48
                          ⟨/init-param⟩
49
                          ⟨load-on-startup⟩1⟨/load-on-startup⟩
                </servlet>
50
51
52
                ⟨servlet-mapping⟩
53
                          <servlet-name>ControllerUsingURI
                          ⟨url-pattern⟩*.do⟨/url-pattern⟩
54
55
                ⟨/servlet-mapping⟩
56
                ⟨filter⟩
57
58
                          <filter-name>LoginCheckFilter</filter-name>
59
                          <filter-class>filter.LoginCheckFilter</filter-class>
                </filter>
60
61
                ⟨filter-mapping⟩
62
                          <filter-name>LoginCheckFilter</filter-name>
                          <url-pattern>/changePwd.do</url-pattern>
63
                          <url-pattern>/article/write.do</url-pattern>
64
65
                          \url-pattern\/article/modify.do\/url-pattern\/
66
                </filter-mapping>
67
      </web-app>
```

#### (3) ConnectionProvider.java

■ ConnectionProvider 클래스는 커넥션을 구할 때 사용한다. web.xml에서 지정한 poolName 값 인 board를 풀 이름으로 사용한 것을 알 수 있다.

```
[board/src/jdbc/connection/ConnectionProvider.java]
 01
      package jdbc.connection;
 02
      import java.sql.Connection;
 03
 04
      import java.sql.DriverManager;
 05
      import java.sql.SQLException;
 06
 07
      public class ConnectionProvider {
 08
 09
           public static Connection getConnection() throws SQLException {
 10
               return DriverManager.getConnection(
 11
                       "jdbc:apache:commons:dbcp:board");
 12
 13
      }
```

#### (4) dbconnTest.jsp

■ DB 연결이 올바르게 되는지 확인할 용도로 JSP 코드를 작성한다.

```
[board/WebContent/dbconnTest.java]
 01

    page contentType="text/html; charset=utf-8"

      $\text{\mathbb{@} page import="jdbc.connection.ConnectionProvider" \text{\sigma}$}
 02
 03
      <%@ page import="java.sql.*" %>
      ⟨head⟩⟨title⟩@ 테스트⟨/title⟩⟨/head⟩
 05
      <body>
 06
 07
      <%
 80
                 try (Connection conn = ConnectionProvider.getConnection()) {
 09
                          out.println("커넥션 연결 성공함"); <mark>//try-with-resource, conn.close()를 자동실행함.</mark>
 10
                 } catch(SQLException ex) {
                           out.println("커넥션 연결 실패함 : " + ex.getMessage());
 11
                           application.log("커넥션 연결 실패", ex);
 12
 13
 14
      %>
 15
      </body>
      </html>
 16
```

## Ę

## 21.5 캐릭터 인코딩 필터 설정

```
07
      ... (생략)
98
09
                <filter>
10
                          <filter-name>encodingFilter</filter-name>
11
                          <filter-class>util.CharacterEncodingFilter</filter-class>
12
                          ⟨init-param⟩
13
                                    <param-name>encoding</param-name>
14
                                    <param-value>utf-8</param-value>
15
                          </init-param>
16
                </filter>
17
18
19
                <filter-mapping>
20
                          <filter-name>encodingFilter</filter-name>
                          <url-pattern>/*</url-pattern>
21
                </filter-mapping>
22
23
24
      ...(생략)
25
26
      </web-app>
```

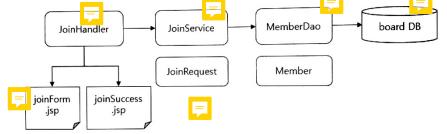
## 21.6 MVC 컨트롤러 코드

■ web.xml에 ControllerUsingURI를 위한 설정을 추가한다.

```
[board/WebContent/WEB-INF/web.xml]
 01
      <?xml version="1.0" encoding="UTF-8"?>
 02
       \web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 03
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 04
                 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee"
 05
                           http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
 06
                 version="3.1">
 07
 08
       ... (생략)
 09
                <servlet>
 10
                           <servlet-name>ControllerUsingURI</servlet-name>
 11
                          <servlet-class>mvc.controller.ControllerUsingURI/servlet-class>
 12
                          ⟨init-param⟩
 13
                                    <param-name>configFile</param-name>
 14
 15
                                    <param-value>
                       /WEB-INF/commandHandlerURI.properties
 16
                   </param-value>
 17
 18
                           </init-param>
                           <load-on-startup>1</load-on-startup>
 19
 20
                 </servlet>
 21
 22
                 <servlet-mapping>
                          <servlet-name>ControllerUsingURI</servlet-name>
 23
                          <url-pattern>*.do</url-pattern>
 24
 25
                 ⟨/servlet-mapping⟩
 26
 27
       ...(생략)
 28
 29
       </web-app>
```

### 21.7 회원 가입 기능 구현

- 회원 가입 기능의 7 세는 다음과 같다.
  - 회원 가입 요청을 하면 입력을 위한 폼을 보여준다.
  - 입력 폼에 아이디, 이름, 암호, 암호 확인을 입력하고 전송하면 가입에 성공한다.
  - 동일한 아이디를 가진 회원이 존재하면 에러 메시지와 함께 다시 폼을 보여준다.
  - 입력한 암호와 암호 확인이 일치하지 않으면 에러 메시지와 함께 다시 폼을 보여준다.
- 이 기능을 위해 구현할 코드는 아래와 같다.



- 각 코드의 역할은 다음과 같다.
  - JoinHandler : 사용자의 요청을 받는다.
    - joinForm.jsp: 회원 가입 폼을 보여준다.
    - joinSuccess.jsp : 회원 가입 처리에 성공한 경우 결과를 보여준다.
  - JoinService : 회원 가입 기능을 구현한다.
    - JoinRequest : 회원 가입할 때 필요한 데이터를 담는다. 폼에 입력한 값을 이 객체 에 담아 JoinService에 전달한다.
  - MemberDao: member 테이블과 관련된 쿼리를 실행한다.
  - Member: member 테이블과 관련된 클래스로서 회원 데이터를 담는다.

#### 21.7.1 회원 정보 보관을 위한 DB 테이블과 관련 Member 클래스

```
[board/sql/ddl.sql]
       create table board.member (
           memberid varchar(50) primary key,
 03
           name varchar(50) not null,
           password varchar(10) not null,
 05
           regdate datetime not null
       ) engine=InnoDB default character set = utf8;
```

```
[board/src/member/model/Member.java]
01
      package member.model;
02
03
      import java.util.Date;
      public class Member {
05
06
07
                 private String id;
98
                 private String name;
09
                 private String password;
10
                 private Date regDate;
 11
                 public Member(String id, String name, String password, Date regDate) {
12
13
                           this.id = id;
```

```
this.name = name;
14
15
                          this.password = password;
                          this.regDate = regDate;
16
17
18
                public String getId() {
19
20
                         return id;
21
22
23
                public String getName() {
                         return name;
24
25
26
27
                public String getPassword() {
28
                         return password;
29
30
               public Date getRegDate() {
31
32
                          return regDate;
33
34
35
                public boolean matchPassword(String pwd) {
36
                          return password.equals(pwd);
37
38
                public void changePassword(String newPwd) {
39
                          this.password = newPwd;
40
41
42
     }
43
```

#### 21.7.2 MemberDao 구현

```
[board/src/member/dao/MemberDao.java]
      package member.dao;
 02
 03
      import java.sql.Connection;
      import java.sql.PreparedStatement;
 05
      import java.sql.ResultSet;
 06
      import java.sql.SQLException;
 07
      import java.sql.Timestamp;
 80
      import java.util.Date;
 09
      import jdbc.JdbcUtil;
 10
 11
      import member.model.Member;
 12
 13
      public class MemberDao {
 14
                 public Member selectById(Connection conn, String id) throws SQLException {
 15
 16
                           PreparedStatement pstmt = null;
                           ResultSet rs = null;
 17
 18
                           try {
 19
                                     pstmt = conn.prepareStatement(
                                                         "select * from member where memberid = ?");
 20
 21
                                     pstmt.setString(1, id);
                                     rs = pstmt.executeQuery();
 22
                                     Member member = null;
 23
 24
                                     if (rs.next()) {
 25
                                               member = new Member(
                                                                   rs.getString("memberid"),
 26
 27
                                                                   rs.getString("name"),
```

```
rs.getString("password"),
28
29
                                                                   toDate(rs.getTimestamp("regdate")));
30
31
                                    return member;
32
                          } finally {
                                    JdbcUtil.close(rs);
33
                                    JdbcUtil.close(pstmt);
34
35
36
37
               private Date toDate(Timestamp date) {
38
39
                          return date == null ? null : new Date(date.getTime());
40
41
                public void insert(Connection conn, Member mem) throws SQLException {
42
43
                          try (PreparedStatement pstmt =
                                              conn.prepareStatement("insert into member values(?,?,?,?)")) {
44
45
                                    pstmt.setString(1, mem.getId());
46
                                    pstmt.setString(2, mem.getName());
                                    pstmt.setString(3, mem.getPassword());
47
48
                                    pstmt.setTimestamp(4, new Timestamp(mem.getRegDate().getTime()));
49
                                    pstmt.executeUpdate();
                          }
50
51
53
                public void update(Connection conn, Member member) throws SQLException {
54
                          try (PreparedStatement pstmt = conn.prepareStatement(
55
                                              "update member set name = ?, password = ? where memberid = ?"))
56
57
                                    pstmt.setString(1, member.getName());
58
                                    pstmt.setString(2, member.getPassword());
                                    pstmt.setString(3, member.getId());
60
                                    pstmt.executeUpdate();
61
62
     }
63
```

### 21.7.3 JoinService와 JoinRequest 구현

■ JoinRequest 클래스는 JoinService가 회원 가입 기능을 구현할 때 필요한 요청 데이터를 담는 클래스이다.

```
[board/src/member/service/JoinRequest.java]
       package member.service;
 02
 03
      import java.util.Map;
 05
       public class JoinRequest {
 06
 07
                 private String id;
 08
                 private String name;
                 private String password;
 09
                 private String confirmPassword;
 10
 11
 12
                 public String getId() {
 13
                           return id;
 14
 15
                 public void setId(String id) {
 16
                           this.id = id;
 17
```

```
18
19
20
                public String getName() {
21
                          return name;
22
23
                public void setName(String name) {
24
25
                          this.name = name;
26
27
                public String getPassword() {
28
29
                          return password;
30
31
32
                public void setPassword(String password) {
33
                          this.password = password;
34
35
                public String getConfirmPassword() {
37
                          return confirmPassword;
38
39
                public void setConfirmPassword(String confirmPassword) {
40
41
                          this.confirmPassword = confirmPassword;
42
43
                public boolean isPasswordEqualToConfirm() {
44
45
                          return password != null && password.equals(confirmPassword);
46
47
48
               public void validate(Map<String, Boolean> errors) {
                          checkEmpty(errors, id, "id");
50
                          checkEmpty(errors, name, "name");
                          checkEmpty(errors, password, "password");
51
                          checkEmpty(errors, confirmPassword, "confirmPassword");
52
53
                          if (!errors.containsKey("confirmPassword")) {
54
                                    if (!isPasswordEqualToConfirm()) {
55
                                              errors.put("notMatch", Boolean. TRUE);
57
58
59
               private void checkEmpty(Map<String, Boolean> errors,
60
61
                                    String value, String fieldName) {
62
                          if (value = null || value.isEmpty())
63
                                   errors.put(fieldName, Boolean. TRUE);
64
                }
65
     }
```

■ 회원 가입 기능을 제공하는 JoinService 클래스의 소스 코드는 아래와 같다.

```
[board/src/member/service/JoinService.java]
      package member.service;
 02
 03
      import java.sql.Connection;
 04
      import java.sql.SQLException;
 05
      import java.util.Date;
 06
 07
      import jdbc.JdbcUtil;
      import jdbc.connection.ConnectionProvider;
      import member.dao.MemberDao;
 09
 10
      import member.model.Member;
```

```
12
      public class JoinService {
13
14
                private MemberDao memberDao = new MemberDao();
15
                public void join(JoinRequest joinReq) {
16
17
                          Connection conn = null;
18
                          try {
                                    conn = ConnectionProvider.getConnection(); // DB 커넥션을 구한다.
19
20
                                    conn.setAutoCommit(false);
21
                                    Member member = memberDao.selectById(conn, joinReq.getId());
22
23
                                    if (member != null) {
                                              JdbcUtil.rollback(conn);
24
                                              throw new DuplicateIdException();
25
26
27
28
                                                                  Member(joinReq.getId(), joinReq.getName(),
                                    memberDao.insert(conn, new
29
      joinReq.getPassword(), new Date()));
30
                                    conn.commit();
31
                          } catch (SQLException e) {
32
                                    JdbcUtil.rollback(conn);
33
                                    throw new RuntimeException(e);
34
                          } finally {
35
                                    JdbcUtil.close(conn);
36
37
      }
38
```

### 21.7.4 JoinHandler와 JSP 구현

- JoinHandler는 다음과 같이 구현한다.
  - GET 방식으로 요청이 오면 폼을 보여주는 뷰인 joinForm.jsp를 리턴한다.
  - POST 방식으로 요청이 오면 회원 가입을 처리하고 결과를 보여주는 뷰를 리턴한다.

```
[board/src/member/command/JoinHandler.java]
      package member.command;
 02
 03
      import java.util.HashMap;
 04
      import java.util.Map;
 05
 06
      import javax.servlet.http.HttpServletRequest;
 07
       import javax.servlet.http.HttpServletResponse;
 80
      import member.service.DuplicateIdException;
 09
 10
      import member.service.JoinRequest;
 11
       import member.service.JoinService;
      import mvc.command.CommandHandler;
 12
 13
 14
      public class JoinHandler implements CommandHandler {
 15
                 private static final String FORM_VIEW = "/WEB-INF/view/joinForm.jsp";
 16
                 private JoinService joinService = new JoinService();
 17
 18
 19
                 @Override
 20
                 public String process(HttpServletRequest req, HttpServletResponse res) {
 21
                           if (req.getMethod().equalsIgnoreCase("GET")) {
 22
                                     return processForm(req, res);
 23
                           } else if (req.getMethod().equalsIgnoreCase("POST")) {
                                     return processSubmit(req, res);
 25
                           } else {
```

```
res.setStatus(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
26
27
                                     return null;
                           }
28
29
31
                private String processForm(HttpServletRequest req, HttpServletResponse res) {
32
                           return FORM_VIEW;
33
34
35
                private String processSubmit(HttpServletRequest req, HttpServletResponse res) {
36
                           JoinRequest joinReq = new JoinRequest();
37
                           joinReq.setId(req.getParameter("id"));
38
                           joinReq.setName(req.getParameter("name"));
39
                           joinReq.setPassword(req.getParameter("password"));
40
                           joinReq.setConfirmPassword(req.getParameter("confirmPassword"));
41
                           Map\langleString, Boolean\rangle errors = new HashMap\langle \rangle();
42
43
                           req.setAttribute("errors", errors);
44
45
                           joinReq.validate(errors);
46
47
                           if (!errors.isEmpty()) {
                                     return FORM_VIEW;
48
49
50
51
                           try {
                                     joinService.join(joinReq);
52
53
                                     return "/WEB-INF/view/joinSuccess.jsp";
54
                           } catch (DuplicateIdException e) {
55
                                     errors.put("duplicateId", Boolean.TRUE);
56
                                     return FORM_VIEW;
57
58
59
     }
60
```

#### 21.7.5 JoinHandler를 위한 매핑 설정

```
[board/WebContent/WEB-INF/commandHandlerURI.properties]

01 /join.do=member.command.JoinHandler
02 /login.do=auth.command.LoginHandler
03 /logout.do=auth.command.LogoutHandler
04 /changePwd.do=member.command.ChangePasswordHandler
05 /article/write.do=article.command.WriteArticleHandler
06 /article/list.do=article.command.ListArticleHandler
07 /article/read.do=article.command.ReadArticleHandler
08 /article/modify.do=article.command.ModifyArticleHandler
```

#### 21.7.6 회원 가입 기능 테스트

■ 톰갯을 구동한 뒤 http://localhost:8080/board/join.do를 웹 브라우저에 입력한다.

### 21.8 로그인 기능 구현

#### 21.8.1 LoginService와 User 구현

```
[board/src/auth/service/User.java]
 01
      package auth.service;
 02
 03
      public class User {
 04
 05
                 private String id;
 06
                 private String name;
 07
 08
                 public User(String id, String name) {
                           this.id = id;
 09
                           this.name = name;
 10
 11
 12
                 public String getId() {
 13
                          return id;
 14
 15
 16
 17
                 public String getName() {
 18
                          return name;
 19
 20
 21
      }
```

```
[board/src/auth/service/LoginService.java]
 01
      package auth.service;
 02
 03
      import java.sql.Connection;
 04
      import java.sql.SQLException;
 05
 06
      import jdbc.connection.ConnectionProvider;
 07
      import member.dao.MemberDao;
 80
      import member.model.Member;
 09
      public class LoginService {
 10
 11
                private MemberDao memberDao = new MemberDao();
 12
 13
 14
                 public User login(String id, String password) {
 15
                           try (Connection conn = ConnectionProvider.getConnection()) {
 16
                                     Member member = memberDao.selectById(conn, id);
 17
                                     if (member == null) {
                                               throw new LoginFailException();
 18
 19
 20
                                     if (!member.matchPassword(password)) {
 21
                                               throw new LoginFailException();
 22
 23
                                     return new User(member.getId(), member.getName());
                           } catch (SQLException e) {
 25
                                     throw new RuntimeException(e);
 26
 27
 28
```

### 21.8.2 LoginHandler 구현

```
[board/src/member/command/LoginHandler.java]

01 package auth.command;
```

```
02
03
      import java.util.HashMap;
04
     import java.util.Map;
05
06
     import javax.servlet.http.HttpServletRequest;
07
     import javax.servlet.http.HttpServletResponse;
08
09
      import auth.service.LoginFailException;
10
      import auth.service.LoginService;
11
      import auth.service.User;
      import mvc.command.CommandHandler;
12
13
14
     public class LoginHandler implements CommandHandler {
15
16
                private static final String FORM_VIEW = "/WEB-INF/view/loginForm.jsp";
17
                private LoginService loginService = new LoginService();
18
19
20
                public String process(HttpServletRequest req, HttpServletResponse res)
21
                throws Exception {
22
                          if (req.getMethod().equalsIgnoreCase("GET")) {
23
                                    return processForm(req, res);
                          } else if (req.getMethod().equalsIgnoreCase("POST")) {
24
25
                                    return processSubmit(req, res);
26
                          } else {
27
                                    res.setStatus(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
28
                                    return null:
29
                          }
30
31
32
                private String processForm(HttpServletRequest req, HttpServletResponse res) {
33
                          return FORM_VIEW;
34
35
36
                private String processSubmit(HttpServletRequest req, HttpServletResponse res)
37
                throws Exception {
38
                          String id = trim(req.getParameter("id"));
39
                          String password = trim(req.getParameter("password"));
40
41
                          Map(String, Boolean) errors = new HashMap()();
42
                          req.setAttribute("errors", errors);
43
                          if (id == null || id.isEmpty())
44
                                    errors.put("id", Boolean.TRUE);
45
46
                          if (password = null || password.isEmpty())
47
                                    errors.put("password", Boolean.TRUE);
48
49
                          if (!errors.isEmpty()) {
50
                                    return FORM_VIEW;
51
52
53
                          try {
54
                                    User user = loginService.login(id, password);
55
                                    req.getSession().setAttribute("authUser", user);
56
                                    res.sendRedirect(req.getContextPath() + "/index.jsp");
57
                                    return null;
                          } catch (LoginFailException e) {
58
                                    errors.put("idOrPwNotMatch", Boolean.TRUE);
59
60
                                    return FORM_VIEW;
61
                          }
62
63
64
                private String trim(String str) {
65
                          return str = null ? null : str.trim();
66
```

### 21.8.3 loginForm.jsp과 index.jsp 구현

```
[board/WebContent/WEB-INF/view/loginForm.jsp]
 01
       $\tag{\mathbb{m}} page contentType="text/html; charset=utf-8"\tag{\mathbb{m}}$
       {%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
       <!DOCTYPE html>
 03
      <html>
 04
 05
      <head>
      〈title〉로그인〈/title〉
 06
      </head>
 07
 08
       <body>
       <form action="login.do" method="post">
 10
       <c:if test="${errors.id0rPwNotMatch}">
       아이디와 암호가 일치하지 않습니다.
 11
 12
       \langle /c:if \rangle
 13
      아이디:<br/>\input type="text" name="id" value="${param.id}">
 14
 15
                 <c:if test="${errors.id}">ID를 입력하세요.</c:if>
 16
       \langle /p \rangle
 17
       18
                 암호:<br/>
\input type="password" name="password">
                 <c:if test="${errors.password}">암호를 입력하세요.</c:if>
 19
 20
      (input type="submit" value="로コ인">
 21
 22
       </form>
 23
       </body>
       </html>
 24
```

```
[board/WebContent/index.jsp]
      $\text{\mathbb{m}} page contentType="text/html; charset=utf-8"\text{\mathbb{m}}$
 02
      {%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 03
      <%@ taglib prefix="u" tagdir="/WEB-INF/tags" %>
 04
      <!DOCTYPE html>
 05
      <html>
      <head>
 06
 07
      <title>회원제 게시판 예제</title>
      </head>
      <body>
 09
 10
      <c:if test="${! empty authUser}">
 11
 12
                ${authUser.name}님, 안녕하세요.
                〈a href="logout.do">[로그아웃하기]</a〉
 13
                〈a href="changePwd.do"〉[암호변경하기]⟨/a⟩
 14
 15
      \langle c: if test="$\{empty authUser}" \rangle
 16
 17
                ⟨a href="join.do"⟩[회원가입하기]⟨/a⟩
 18
                〈a href="login.do">[로그인하기]⟨/a⟩
      </c:if>
 19
 20
      --%>
 21
      <u:isLogin>
                CT: ${authUser.name}님, 안녕하세요.
                〈a href="logout.do">[로그아웃하기]〈/a〉
 23
 24
                〈a href="changePwd.do">[암호변경하기]</a>
 25
      </usisLogin>
 26
       <u:notLogin>
                CT: <a href="join.do">[회원가입하기]</a>
 27
 28
                〈a href="login.do">[로그인하기]⟨/a⟩
```

## 21.8.4 로그인 기능 테스트

■ 톰갯을 실행하고 <a href="http://localhost:8080/board/index.jsp">http://localhost:8080/board/index.jsp</a>를 웹 브라우저에서 실행한다.

## 21.9 로그아웃 기능 구현

```
[board/src/auth/command/LogoutHandler.java]
      package auth.command;
 02
 03
      import javax.servlet.http.HttpServletRequest;
       import javax.servlet.http.HttpServletResponse;
 05
      import javax.servlet.http.HttpSession;
 07
      import mvc.command.CommandHandler;
 80
 09
      public class LogoutHandler implements CommandHandler {
 10
 11
                @Override
                public String process(HttpServletRequest req, HttpServletResponse res)
 12
 13
                throws Exception {
 14
                           HttpSession session = req.getSession(false);
 15
                           if (session != null) {
 16
                                     session.invalidate();
 17
                           res.sendRedirect(req.getContextPath() + "/index.jsp");
 18
 19
                           return null;
 21
      }
 22
```

## 21.10 로그인 여부 검사 기능

### 21.10.1 LoginCheckFilter 구현

```
[board/src/filter/LoginCheckFilter.java]
      package filter;
 03
      import java.io.IOException;
 05
      import javax.servlet.Filter;
 06
      import javax.servlet.FilterChain;
      import javax.servlet.FilterConfig;
 08
      import javax.servlet.ServletException;
 09
      import javax.servlet.ServletRequest;
 10
     import javax.servlet.ServletResponse;
 11
      import javax.servlet.http.HttpServletRequest;
12
      import javax.servlet.http.HttpServletResponse;
13 import javax.servlet.http.HttpSession;
```

```
14
15
      public class LoginCheckFilter implements Filter {
16
17
                @Override
18
                public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
19
                                    throws IOException, ServletException {
                          HttpServletRequest request = (HttpServletRequest) req;
20
21
                          HttpSession session = request.getSession(false);
                          if (session == null || session.getAttribute("authUser") == null) {
22
                                    HttpServletResponse response = (HttpServletResponse)res;
23
24
                                    response.sendRedirect(request.getContextPath() + "/login.do");
25
                          } else {
                                    chain.doFilter(req, res);
26
                          }
27
28
29
                @Override
30
31
                public void init(FilterConfig config) throws ServletException {
33
34
                @Override
35
                public void destroy() {
36
37
     }
38
```

#### 21.10.2 로그인 여부 검사 커스텀 태그 구현

```
<c:if test="${! empty authUser}">
         ${authUser.name}님, 안녕하세요.
         〈a href="logout.do">[로그아웃하기]</a>
         〈a href="changePwd.do"〉[암호변경하기]⟨/a⟩
</c:if>
<c:if test="${empty authUser}">
         <a href="join.do">[회원가입하기]</a>
         <a href="login.do">[로그인하기]</a>
</c:if>
--%>
<u:isLogin>
         CT: ${authUser.name}님, 안녕하세요.
<a href="logout.do">[로그아웃하기]</a>
         〈a href="changePwd.do"〉[암호변경하기]⟨/a⟩
</u:isLogin>
<u:notLogin>
         CT: 〈a href="join.do"〉[회원가입하기]〈/a〉
         〈a href="login.do">[로그인하기]</a>
</u:notLogin>
```

## 21.11 암호 변경 기능 구현

#### 21.11.1 Member와 MemberDao에 암호 변경 관련 기능 구현

```
[board/src/member/dao/MemberDao.java]

01 package member.dao;
02
```

```
03
     import java.sql.Connection;
04
      import java.sql.PreparedStatement;
05
      import java.sql.ResultSet;
06
     import java.sql.SQLException;
07
     import java.sql.Timestamp;
08
     import java.util.Date;
09
10
     import jdbc.JdbcUtil;
11
      import member.model.Member;
12
     public class MemberDao {
13
14
15
                public Member selectById(Connection conn, String id) throws SQLException {
                          PreparedStatement pstmt = null;
16
17
                          ResultSet rs = null;
18
                          try {
19
                                    pstmt = conn.prepareStatement(
20
                                                         "select * from member where memberid = ?");
21
                                    pstmt.setString(1, id);
22
                                    rs = pstmt.executeQuery();
23
                                    Member member = null;
24
                                    if (rs.next()) {
25
                                              member = new Member(
26
                                                                   rs.getString("memberid"),
                                                                   rs.getString("name"),
27
                                                                   rs.getString("password"),
28
                                                                   toDate(rs.getTimestamp("regdate")));
29
30
                                    }
31
                                    return member;
32
                          } finally {
33
                                    JdbcUtil.close(rs);
34
                                    JdbcUtil.close(pstmt);
35
                          }
36
37
38
                private Date toDate(Timestamp date) {
39
                          return date == null ? null : new Date(date.getTime());
40
41
42
                public void insert(Connection conn, Member mem) throws SQLException {
43
                          try (PreparedStatement pstmt =
44
                                              conn.prepareStatement("insert into member values(?,?,?,?)")) {
45
                                    pstmt.setString(1, mem.getId());
46
                                    pstmt.setString(2, mem.getName());
47
                                    pstmt.setString(3, mem.getPassword());
48
                                    pstmt.setTimestamp(4, new Timestamp(mem.getRegDate().getTime()));
49
                                    pstmt.executeUpdate();
50
                          }
51
52
53
                public void update(Connection conn, Member member) throws SQLException {
54
                          try (PreparedStatement pstmt = conn.prepareStatement(
55
                                              "update member set name = ?, password = ? where memberid = ?"))
56
     {
57
                                    pstmt.setString(1, member.getName());
58
                                    pstmt.setString(2, member.getPassword());
59
                                    pstmt.setString(3, member.getId());
60
                                    pstmt.executeUpdate();
61
                          }
62
                }
63
     }
```

### 21.11.2 ChangePasswordService 구현

```
[board/src/member/service/ChangePasswordService.java]
 01
       package member.service;
 02
 03
       import java.sql.Connection;
       import java.sql.SQLException;
 05
 06
       import jdbc.JdbcUtil;
 07
       import jdbc.connection.ConnectionProvider;
 80
       import member.dao.MemberDao;
 09
       import member.model.Member;
 10
 11
       public class ChangePasswordService {
 12
 13
                 private MemberDao memberDao = new MemberDao();
 14
 15
                 public void changePassword(String userId, String curPwd, String newPwd) {
 16
                           Connection conn = null;
 17
                           try {
 18
                                     conn = ConnectionProvider.getConnection();
                                     conn.setAutoCommit(false);
 19
 20
 21
                                     Member member = memberDao.selectById(conn, userId);
                                     if (member = null) {
 22
 23
                                               throw new MemberNotFoundException();
 25
                                     if (!member.matchPassword(curPwd)) {
 26
                                               throw new InvalidPasswordException();
 27
 28
                                     member.changePassword(newPwd);
 29
                                     memberDao.update(conn, member);
 30
                                     conn.commit();
                           } catch (SQLException e) {
 31
                                     JdbcUtil.rollback(conn);
 32
 33
                                     throw new RuntimeException(e);
 34
                           } finally {
 35
                                     JdbcUtil.close(conn);
 36
 37
 38
```

### 21.11.3 ChangePasswordHandler 구현

```
[board/src/member/command/ChangePasswordHandler.java]
      package member.command;
 02
 03
      import java.util.HashMap;
 04
      import java.util.Map;
 06
      import javax.servlet.http.HttpServletRequest;
 07
      import javax.servlet.http.HttpServletResponse;
 98
 09
      import auth.service.User;
 10
      import member.service.ChangePasswordService;
 11
       import member.service.InvalidPasswordException;
 12
       import member.service.MemberNotFoundException;
 13
       import mvc.command.CommandHandler;
 14
```

```
15
      public class ChangePasswordHandler implements CommandHandler {
16
                private static final String FORM_VIEW = "/WEB-INF/view/changePwdForm.jsp";
17
                private ChangePasswordService changePwdSvc = new ChangePasswordService();
18
19
20
                public String process(HttpServletRequest req, HttpServletResponse res)
21
                throws Exception {
22
                           if (req.getMethod().equalsIgnoreCase("GET")) {
23
                                     return processForm(req, res);
                           } else if (req.getMethod().equalsIgnoreCase("POST")) {
24
25
                                     return processSubmit(req, res);
26
                                     res.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
27
28
                                     return null;
29
30
31
32
                private String processForm(HttpServletRequest req, HttpServletResponse res) {
33
                           return FORM_VIEW;
34
35
36
37
                private String processSubmit(HttpServletRequest req, HttpServletResponse res)
38
                throws Exception {
                           User user = (User)req.getSession().getAttribute("authUser");
39
40
41
                           Map\langleString, Boolean\rangle errors = new HashMap\langle\rangle();
42
                           req.setAttribute("errors", errors);
43
                           String curPwd = req.getParameter("curPwd");
44
45
                           String newPwd = req.getParameter("newPwd");
46
47
                           if (curPwd == null || curPwd.isEmpty()) {
                                     errors.put("curPwd", Boolean.TRUE);
48
49
50
                           if (curPwd == null || curPwd.isEmpty()) {
51
                                     errors.put("newPwd", Boolean.TRUE);
52
53
                           if (!errors.isEmpty()) {
54
                                     return FORM_VIEW;
55
56
57
                           try {
                                     changePwdSvc.changePassword(user.getId(), curPwd, newPwd);
58
                                     return "/WEB-INF/view/changePwdSuccess.jsp";
59
                           } catch (InvalidPasswordException e) {
60
61
                                     errors.put("badCurPwd", Boolean.TRUE);
                                     return FORM_VIEW;
62
63
                           } catch (MemberNotFoundException e) {
64
                                     res.sendError(HttpServletResponse.SC_BAD_REQUEST);
65
                                     return null;
66
67
68
69
      }
```

## 21.11.4 changePwdForm.jsp와 changePwdSuccess.jsp 구현

```
03
   <!DOCTYPE html>
04
    <html>
05
    \head>
   <title>암호 변경</title>
06
   </head>
   <body>
80
09
    <form action="changePwd.do" method="post">
10
             현재 암호:<br/>
<input type="password" name="curPwd">
11
             <c:if test="${errors.curPwd}">현재 암호를 입력하세요.</c:if>
12
             <c:if test="${errors.badCurPwd}">현재 암호가 일치하지 않습니다.</c:if>
13
    14
15
    16
             새 암호:〈br/〉〈input type="password" name="newPwd"〉
             <c:if test="${errors.newPwd}">새 암호를 입력하세요.</c:if>
17
    18
    <input type="submit" value="암호 변경">
19
20 </form>
21 </body>
22 </html>
```

### 21.11.5 암호 변경 기능 테스트

■ 톰갯을 재시작하고 로그인을 한 뒤에 <a href="http://localhost:8080/board/changePwd.do">http://localhost:8080/board/changePwd.do</a> 주소를 웹 브라우저에 입력한다.

# 21.12 정리

■ '핸들러(커맨드)-서비스-DAO' 구조는 처음 웹 개발을 시작할 때 익히기 쉬운 구조이다. 실제 현업에서 개발하는 많은 웹 어플리케이션이 이 구조를 사용하고 있다.