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';

GRANT ALL PRIVILEGES ON board.* TO 'jspexam'@'localhost';
GRANT ALL PRIVILEGES ON board.* TO 'jspexam'@'%';

create table board.member (
    memberid varchar(50) primary key,
    name varchar(50) not null,
    password varchar(10) not null,
    regdate datetime not null
) engine=InnoDB default character set = utf8;

create table board.article (
    article_no int auto_increment primary key,
    writer_id varchar(50) not null,
    writer_name varchar(50) not null,
    title varchar(255) not null,
    regdate datetime not null,
    moddate datetime not null,
    read_cnt int
) engine=InnoDB default character set = utf8;

create table board.article_content (
    article_no int primary key,
    content text
) engine=InnoDB default character set = utf8;

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]

```
package jdbc;

import java.io.IOException;
import java.io.StringReader;
import java.sql.DriverManager;
import java.util.Properties;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.commons.dbcp2.ConnectionFactory;
import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
import org.apache.commons.dbcp2.PoolableConnection;
import org.apache.commons.dbcp2.PoolableConnectionFactory;
import org.apache.commons.dbcp2.PoolingDriver;
import org.apache.commons.pool2.impl.GenericObjectPool;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;

public class DBCPInitListener implements ServletContextListener {

        @Override
        public void contextInitialized(ServletContextEvent sce) {
                String poolConfig =
                                sce.getServletContext().getInitParameter("poolConfig");
                Properties prop = new Properties();
                try {
                        prop.load(new StringReader(poolConfig));
                } catch (IOException e) {
                        throw new RuntimeException("config load fail", e);
                }
                loadJDBCDriver(prop);
                initConnectionPool(prop);
        }

        private void loadJDBCDriver(Properties prop) {
                String driverClass = prop.getProperty("jdbcdriver");
                try {
                        Class.forName(driverClass);
                } catch (ClassNotFoundException ex) {
                        throw new RuntimeException("fail to load JDBC Driver", ex);
```

```java
        }
    }

    private void initConnectionPool(Properties prop) {
        try {
            String jdbcUrl = prop.getProperty("jdbcUrl");
            String username = prop.getProperty("dbUser");
            String pw = prop.getProperty("dbPass");

            ConnectionFactory connFactory =
                    new  DriverManagerConnectionFactory(jdbcUrl,  username,
pw);

            PoolableConnectionFactory poolableConnFactory =
                    new PoolableConnectionFactory(connFactory, null);
            String validationQuery = prop.getProperty("validationQuery");
            if (validationQuery != null && !validationQuery.isEmpty()) {
                poolableConnFactory.setValidationQuery(validationQuery);
            }
            GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
            poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60L * 5L);
            poolConfig.setTestWhileIdle(true);
            int minIdle = getIntProperty(prop, "minIdle", 5);
            poolConfig.setMinIdle(minIdle);
            int maxTotal = getIntProperty(prop, "maxTotal", 50);
            poolConfig.setMaxTotal(maxTotal);

            GenericObjectPool<PoolableConnection> connectionPool =
                    new           GenericObjectPool<>(poolableConnFactory,
poolConfig);
            poolableConnFactory.setPool(connectionPool);

            Class.forName("org.apache.commons.dbcp2.PoolingDriver");
            PoolingDriver driver = (PoolingDriver)
                DriverManager.getDriver("jdbc:apache:commons:dbcp:");
            String poolName = prop.getProperty("poolName");
            driver.registerPool(poolName, connectionPool);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private int getIntProperty(Properties prop, String propName, int defaultValue) {
        String value = prop.getProperty(propName);
        if (value == null) return defaultValue;
        return Integer.parseInt(value);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
    }

}
```

(2) web.xml

■ DBCPInitListener는 서블릿 컨텍스트 리스너이므로 web.xml에 등록한다.

[board/WebContent/WEB-INF/web.xml]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```xml
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">

    <listener>
        <listener-class>jdbc.DBCPInitListener</listener-class>
    </listener>

    <context-param>
        <param-name>poolConfig</param-name>
        <param-value>
            jdbcdriver=com.mysql.jdbc.Driver
            jdbcUrl=jdbc:mysql://localhost:3306/board?characterEncoding=utf8
            dbUser=jspexam
            dbPass=jsppw
            validationQuery=select 1
            minIdle=3
            maxTotal=30
            poolName=board
        </param-value>
    </context-param>

    <filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>util.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>utf-8</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <servlet>
        <servlet-name>ControllerUsingURI</servlet-name>
        <servlet-class>mvc.controller.ControllerUsingURI</servlet-class>
        <init-param>
            <param-name>configFile</param-name>
            <param-value>
      /WEB-INF/commandHandlerURI.properties
     </param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>ControllerUsingURI</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>LoginCheckFilter</filter-name>
        <filter-class>filter.LoginCheckFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>LoginCheckFilter</filter-name>
        <url-pattern>/changePwd.do</url-pattern>
        <url-pattern>/article/write.do</url-pattern>
        <url-pattern>/article/modify.do</url-pattern>
    </filter-mapping>
</web-app>
```

(3) ConnectionProvider.java

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

[board/src/jdbc/connection/ConnectionProvider.java]

```java
package jdbc.connection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionProvider {

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(
                "jdbc:apache:commons:dbcp:board");
    }
}
```

(4) dbconnTest.jsp

■ DB 연결이 올바르게 되는지 확인할 용도로 JSP 코드를 작성한다.

[board/WebContent/dbconnTest.java]

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<%@ page import="jdbc.connection.ConnectionProvider" %>
<%@ page import="java.sql.*" %>
<html>
<head><title>연결 테스트</title></head>
<body>
<%
          try (Connection conn = ConnectionProvider.getConnection()) {
                    out.println("커넥션 연결 성공함"); //try-with-resource, conn.close()를 자동실행함.
          } catch(SQLException ex) {
                    out.println("커넥션 연결 실패함 : " + ex.getMessage());
                    application.log("커넥션 연결 실패", ex);
          }
%>
</body>
</html>
```

21.5 캐릭터 인코딩 필터 설정

[board/WebContent/WEB-INF/web.xml]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">
```

```

... (생략)

          <filter>
                    <filter-name>encodingFilter</filter-name>
                    <filter-class>util.CharacterEncodingFilter</filter-class>
                    <init-param>
                              <param-name>encoding</param-name>
                              <param-value>utf-8</param-value>
                    </init-param>
          </filter>

          <filter-mapping>
                    <filter-name>encodingFilter</filter-name>
                    <url-pattern>/*</url-pattern>
          </filter-mapping>

...(생략)

</web-app>
```

21.6 MVC 컨트롤러 코드

■ web.xml에 ControllerUsingURI를 위한 설정을 추가한다.

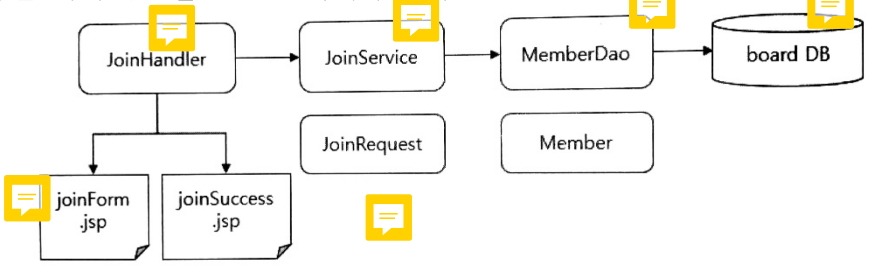
[board/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">

... (생략)

          <servlet>
                    <servlet-name>ControllerUsingURI</servlet-name>
                    <servlet-class>mvc.controller.ControllerUsingURI</servlet-class>
                    <init-param>
                              <param-name>configFile</param-name>
                              <param-value>
                /WEB-INF/commandHandlerURI.properties
            </param-value>
                    </init-param>
                    <load-on-startup>1</load-on-startup>
          </servlet>

          <servlet-mapping>
                    <servlet-name>ControllerUsingURI</servlet-name>
                    <url-pattern>*.do</url-pattern>
          </servlet-mapping>

...(생략)

</web-app>
```

21.7 회원 가입 기능 구현

- 회원 가입 기능의 상세는 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 객체에 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]

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

[board/src/member/model/Member.java]

```java
package member.model;

import java.util.Date;

public class Member {

        private String id;
        private String name;
        private String password;
        private Date regDate;

        public Member(String id, String name, String password, Date regDate) {
                this.id = id;
```

```java
			this.name = name;
			this.password = password;
			this.regDate = regDate;
		}

		public String getId() {
			return id;
		}

		public String getName() {
			return name;
		}

		public String getPassword() {
			return password;
		}

		public Date getRegDate() {
			return regDate;
		}

		public boolean matchPassword(String pwd) {
			return password.equals(pwd);
		}

		public void changePassword(String newPwd) {
			this.password = newPwd;
		}

	}
```

21.7.2 MemberDao 구현

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

```java
	package member.dao;

	import java.sql.Connection;
	import java.sql.PreparedStatement;
	import java.sql.ResultSet;
	import java.sql.SQLException;
	import java.sql.Timestamp;
	import java.util.Date;

	import jdbc.JdbcUtil;
	import member.model.Member;

	public class MemberDao {

		public Member selectById(Connection conn, String id) throws SQLException {
			PreparedStatement pstmt = null;
			ResultSet rs = null;
			try {
				pstmt = conn.prepareStatement(
						"select * from member where memberid = ?");
				pstmt.setString(1, id);
				rs = pstmt.executeQuery();
				Member member = null;
				if (rs.next()) {
					member = new Member(
							rs.getString("memberid"),
							rs.getString("name"),
                        rs.getString("password"),
                        toDate(rs.getTimestamp("regdate")));
            }
            return member;
        } finally {
            JdbcUtil.close(rs);
            JdbcUtil.close(pstmt);
        }
    }

    private Date toDate(Timestamp date) {
        return date == null ? null : new Date(date.getTime());
    }

    public void insert(Connection conn, Member mem) throws SQLException {
        try (PreparedStatement pstmt =
                conn.prepareStatement("insert into member values(?,?,?,?)")) {
            pstmt.setString(1, mem.getId());
            pstmt.setString(2, mem.getName());
            pstmt.setString(3, mem.getPassword());
            pstmt.setTimestamp(4, new Timestamp(mem.getRegDate().getTime()));
            pstmt.executeUpdate();
        }
    }

    public void update(Connection conn, Member member) throws SQLException {
        try (PreparedStatement pstmt = conn.prepareStatement(
                "update member set name = ?, password = ? where memberid = ?"))
{
            pstmt.setString(1, member.getName());
            pstmt.setString(2, member.getPassword());
            pstmt.setString(3, member.getId());
            pstmt.executeUpdate();
        }
    }
}
```

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;

import java.util.Map;

public class JoinRequest {

    private String id;
    private String name;
    private String password;
    private String confirmPassword;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
```

```java
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getConfirmPassword() {
		return confirmPassword;
	}

	public void setConfirmPassword(String confirmPassword) {
		this.confirmPassword = confirmPassword;
	}

	public boolean isPasswordEqualToConfirm() {
		return password != null && password.equals(confirmPassword);
	}

	public void validate(Map<String, Boolean> errors) {
		checkEmpty(errors, id, "id");
		checkEmpty(errors, name, "name");
		checkEmpty(errors, password, "password");
		checkEmpty(errors, confirmPassword, "confirmPassword");
		if (!errors.containsKey("confirmPassword")) {
			if (!isPasswordEqualToConfirm()) {
				errors.put("notMatch", Boolean.TRUE);
			}
		}
	}

	private void checkEmpty(Map<String, Boolean> errors,
			String value, String fieldName) {
		if (value == null || value.isEmpty())
			errors.put(fieldName, Boolean.TRUE);
	}
}
```

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

[board/src/member/service/JoinService.java]

```java
package member.service;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.Date;

import jdbc.JdbcUtil;
import jdbc.connection.ConnectionProvider;
import member.dao.MemberDao;
import member.model.Member;

```

```
public class JoinService {

        private MemberDao memberDao = new MemberDao();

        public void join(JoinRequest joinReq) {
                Connection conn = null;
                try {
                        conn = ConnectionProvider.getConnection();  // DB 커넥션을 구한다.
                        conn.setAutoCommit(false);

                        Member member = memberDao.selectById(conn, joinReq.getId());
                        if (member != null) {
                                JdbcUtil.rollback(conn);
                                throw new DuplicateIdException();
                        }

                        memberDao.insert(conn,   new   Member(joinReq.getId(),   joinReq.getName(),
joinReq.getPassword(), new Date()));
                        conn.commit();
                } catch (SQLException e) {
                        JdbcUtil.rollback(conn);
                        throw new RuntimeException(e);
                } finally {
                        JdbcUtil.close(conn);
                }
        }
}
```

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;

import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import member.service.DuplicateIdException;
import member.service.JoinRequest;
import member.service.JoinService;
import mvc.command.CommandHandler;

public class JoinHandler implements CommandHandler {

        private static final String FORM_VIEW = "/WEB-INF/view/joinForm.jsp";
        private JoinService joinService = new JoinService();

        @Override
        public String process(HttpServletRequest req, HttpServletResponse res) {
                if (req.getMethod().equalsIgnoreCase("GET")) {
                        return processForm(req, res);
                } else if (req.getMethod().equalsIgnoreCase("POST")) {
                        return processSubmit(req, res);
                } else {
```

```java
                res.setStatus(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
                return null;
            }
        }

        private String processForm(HttpServletRequest req, HttpServletResponse res) {
            return FORM_VIEW;
        }

        private String processSubmit(HttpServletRequest req, HttpServletResponse res) {
            JoinRequest joinReq = new JoinRequest();
            joinReq.setId(req.getParameter("id"));
            joinReq.setName(req.getParameter("name"));
            joinReq.setPassword(req.getParameter("password"));
            joinReq.setConfirmPassword(req.getParameter("confirmPassword"));

            Map<String, Boolean> errors = new HashMap<>();
            req.setAttribute("errors", errors);

            joinReq.validate(errors);

            if (!errors.isEmpty()) {
                return FORM_VIEW;
            }

            try {
                joinService.join(joinReq);
                return "/WEB-INF/view/joinSuccess.jsp";
            } catch (DuplicateIdException e) {
                errors.put("duplicateId", Boolean.TRUE);
                return FORM_VIEW;
            }
        }

}
```

21.7.5 JoinHandler를 위한 매핑 설정

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

```
/join.do=member.command.JoinHandler
/login.do=auth.command.LoginHandler
/logout.do=auth.command.LogoutHandler
/changePwd.do=member.command.ChangePasswordHandler
/article/write.do=article.command.WriteArticleHandler
/article/list.do=article.command.ListArticleHandler
/article/read.do=article.command.ReadArticleHandler
/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]

```java
package auth.service;

public class User {

	private String id;
	private String name;

	public User(String id, String name) {
		this.id = id;
		this.name = name;
	}

	public String getId() {
		return id;
	}

	public String getName() {
		return name;
	}

}
```

[board/src/auth/service/LoginService.java]

```java
package auth.service;

import java.sql.Connection;
import java.sql.SQLException;

import jdbc.connection.ConnectionProvider;
import member.dao.MemberDao;
import member.model.Member;

public class LoginService {

	private MemberDao memberDao = new MemberDao();

	public User login(String id, String password) {
		try (Connection conn = ConnectionProvider.getConnection()) {
			Member member = memberDao.selectById(conn, id);
			if (member == null) {
				throw new LoginFailException();
			}
			if (!member.matchPassword(password)) {
				throw new LoginFailException();
			}
			return new User(member.getId(), member.getName());
		} catch (SQLException e) {
			throw new RuntimeException(e);
		}
	}
}
```

21.8.2 LoginHandler 구현

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

```java
package auth.command;
```

```java

import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import auth.service.LoginFailException;
import auth.service.LoginService;
import auth.service.User;
import mvc.command.CommandHandler;

public class LoginHandler implements CommandHandler {

	private static final String FORM_VIEW = "/WEB-INF/view/loginForm.jsp";
	private LoginService loginService = new LoginService();

	@Override
	public String process(HttpServletRequest req, HttpServletResponse res)
	throws Exception {
		if (req.getMethod().equalsIgnoreCase("GET")) {
			return processForm(req, res);
		} else if (req.getMethod().equalsIgnoreCase("POST")) {
			return processSubmit(req, res);
		} else {
			res.setStatus(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
			return null;
		}
	}

	private String processForm(HttpServletRequest req, HttpServletResponse res) {
		return FORM_VIEW;
	}

	private String processSubmit(HttpServletRequest req, HttpServletResponse res)
	throws Exception {
		String id = trim(req.getParameter("id"));
		String password = trim(req.getParameter("password"));

		Map<String, Boolean> errors = new HashMap<>();
		req.setAttribute("errors", errors);

		if (id == null || id.isEmpty())
			errors.put("id", Boolean.TRUE);
		if (password == null || password.isEmpty())
			errors.put("password", Boolean.TRUE);

		if (!errors.isEmpty()) {
			return FORM_VIEW;
		}

		try {
			User user = loginService.login(id, password);
			req.getSession().setAttribute("authUser", user);
			res.sendRedirect(req.getContextPath() + "/index.jsp");
			return null;
		} catch (LoginFailException e) {
			errors.put("idOrPwNotMatch", Boolean.TRUE);
			return FORM_VIEW;
		}
	}

	private String trim(String str) {
		return str == null ? null : str.trim();
	}
}
```

21.8.3 loginForm.jsp과 index.jsp 구현

[board/WebContent/WEB-INF/view/loginForm.jsp]

```
<%@ page contentType="text/html; charset=utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<title>로그인</title>
</head>
<body>
<form action="login.do" method="post">
<c:if test="${errors.idOrPwNotMatch}">
아이디와 암호가 일치하지 않습니다.
</c:if>
<p>
	아이디:<br/><input type="text" name="id" value="${param.id}">
	<c:if test="${errors.id}">ID를 입력하세요.</c:if>
</p>
<p>
	암호:<br/><input type="password" name="password">
	<c:if test="${errors.password}">암호를 입력하세요.</c:if>
</p>
<input type="submit" value="로그인">
</form>
</body>
</html>
```

[board/WebContent/index.jsp]

```
<%@ page contentType="text/html; charset=utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="u" tagdir="/WEB-INF/tags" %>
<!DOCTYPE html>
<html>
<head>
<title>회원제 게시판 예제</title>
</head>
<body>
<%--
<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>
<br/>
</body>
</html>
```

21.8.4 로그인 기능 테스트

- 톰갯을 실행하고 http://localhost:8080/board/index.jsp를 웹 브라우저에서 실행한다.

21.9 로그아웃 기능 구현

[board/src/auth/command/LogoutHandler.java]

```java
package auth.command;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import mvc.command.CommandHandler;

public class LogoutHandler implements CommandHandler {

        @Override
        public String process(HttpServletRequest req, HttpServletResponse res)
        throws Exception {
                HttpSession session = req.getSession(false);
                if (session != null) {
                        session.invalidate();
                }
                res.sendRedirect(req.getContextPath() + "/index.jsp");
                return null;
        }

}
```

21.10 로그인 여부 검사 기능

21.10.1 LoginCheckFilter 구현

[board/src/filter/LoginCheckFilter.java]

```java
package filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```java

public class LoginCheckFilter implements Filter {

		@Override
		public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
						throws IOException, ServletException {
				HttpServletRequest request = (HttpServletRequest) req;
				HttpSession session = request.getSession(false);
				if (session == null || session.getAttribute("authUser") == null) {
						HttpServletResponse response = (HttpServletResponse)res;
						response.sendRedirect(request.getContextPath() + "/login.do");
				} else {
						chain.doFilter(req, res);
				}
		}

		@Override
		public void init(FilterConfig config) throws ServletException {
		}

		@Override
		public void destroy() {
		}

}
```

21.10.2 로그인 여부 검사 커스텀 태그 구현

```jsp
<%--
<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]

```java
package member.dao;

```

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.Date;

import jdbc.JdbcUtil;
import member.model.Member;

public class MemberDao {

	public Member selectById(Connection conn, String id) throws SQLException {
		PreparedStatement pstmt = null;
		ResultSet rs = null;
		try {
			pstmt = conn.prepareStatement(
					"select * from member where memberid = ?");
			pstmt.setString(1, id);
			rs = pstmt.executeQuery();
			Member member = null;
			if (rs.next()) {
				member = new Member(
						rs.getString("memberid"),
						rs.getString("name"),
						rs.getString("password"),
						toDate(rs.getTimestamp("regdate")));
			}
			return member;
		} finally {
			JdbcUtil.close(rs);
			JdbcUtil.close(pstmt);
		}
	}

	private Date toDate(Timestamp date) {
		return date == null ? null : new Date(date.getTime());
	}

	public void insert(Connection conn, Member mem) throws SQLException {
		try (PreparedStatement pstmt =
				conn.prepareStatement("insert into member values(?,?,?,?)")) {
			pstmt.setString(1, mem.getId());
			pstmt.setString(2, mem.getName());
			pstmt.setString(3, mem.getPassword());
			pstmt.setTimestamp(4, new Timestamp(mem.getRegDate().getTime()));
			pstmt.executeUpdate();
		}
	}

	public void update(Connection conn, Member member) throws SQLException {
		try (PreparedStatement pstmt = conn.prepareStatement(
				"update member set name = ?, password = ? where memberid = ?"))
{
			pstmt.setString(1, member.getName());
			pstmt.setString(2, member.getPassword());
			pstmt.setString(3, member.getId());
			pstmt.executeUpdate();
		}
	}
}
```

21.11.2 ChangePasswordService 구현

[board/src/member/service/ChangePasswordService.java]

```java
package member.service;

import java.sql.Connection;
import java.sql.SQLException;

import jdbc.JdbcUtil;
import jdbc.connection.ConnectionProvider;
import member.dao.MemberDao;
import member.model.Member;

public class ChangePasswordService {

	private MemberDao memberDao = new MemberDao();

	public void changePassword(String userId, String curPwd, String newPwd) {
		Connection conn = null;
		try {
			conn = ConnectionProvider.getConnection();
			conn.setAutoCommit(false);

			Member member = memberDao.selectById(conn, userId);
			if (member == null) {
				throw new MemberNotFoundException();
			}
			if (!member.matchPassword(curPwd)) {
				throw new InvalidPasswordException();
			}
			member.changePassword(newPwd);
			memberDao.update(conn, member);
			conn.commit();
		} catch (SQLException e) {
			JdbcUtil.rollback(conn);
			throw new RuntimeException(e);
		} finally {
			JdbcUtil.close(conn);
		}
	}
}
```

21.11.3 ChangePasswordHandler 구현

[board/src/member/command/ChangePasswordHandler.java]

```java
package member.command;

import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import auth.service.User;
import member.service.ChangePasswordService;
import member.service.InvalidPasswordException;
import member.service.MemberNotFoundException;
import mvc.command.CommandHandler;

```

```java
public class ChangePasswordHandler implements CommandHandler {
		private static final String FORM_VIEW = "/WEB-INF/view/changePwdForm.jsp";
		private ChangePasswordService changePwdSvc = new ChangePasswordService();

		@Override
		public String process(HttpServletRequest req, HttpServletResponse res)
		throws Exception {
				if (req.getMethod().equalsIgnoreCase("GET")) {
						return processForm(req, res);
				} else if (req.getMethod().equalsIgnoreCase("POST")) {
						return processSubmit(req, res);
				} else {
						res.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
						return null;
				}
		}

		private String processForm(HttpServletRequest req, HttpServletResponse res) {
				return FORM_VIEW;
		}


		private String processSubmit(HttpServletRequest req, HttpServletResponse res)
		throws Exception {
				User user = (User)req.getSession().getAttribute("authUser");

				Map<String, Boolean> errors = new HashMap<>();
				req.setAttribute("errors", errors);

				String curPwd = req.getParameter("curPwd");
				String newPwd = req.getParameter("newPwd");

				if (curPwd == null || curPwd.isEmpty()) {
						errors.put("curPwd", Boolean.TRUE);
				}
				if (curPwd == null || curPwd.isEmpty()) {
						errors.put("newPwd", Boolean.TRUE);
				}
				if (!errors.isEmpty()) {
						return FORM_VIEW;
				}

				try {
						changePwdSvc.changePassword(user.getId(), curPwd, newPwd);
						return "/WEB-INF/view/changePwdSuccess.jsp";
				} catch (InvalidPasswordException e) {
						errors.put("badCurPwd", Boolean.TRUE);
						return FORM_VIEW;
				} catch (MemberNotFoundException e) {
						res.sendError(HttpServletResponse.SC_BAD_REQUEST);
						return null;
				}
		}

}
```

21.11.4 changePwdForm.jsp와 changePwdSuccess.jsp 구현

[board/WebContent/WEB-INF/view/changePwdForm.jsp]

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<!DOCTYPE html>
<html>
<head>
<title>암호 변경</title>
</head>
<body>
<form action="changePwd.do" method="post">
<p>
          현재 암호:<br/><input type="password" name="curPwd">
          <c:if test="${errors.curPwd}">현재 암호를 입력하세요.</c:if>
          <c:if test="${errors.badCurPwd}">현재 암호가 일치하지 않습니다.</c:if>
</p>
<p>
          새 암호:<br/><input type="password" name="newPwd">
          <c:if test="${errors.newPwd}">새 암호를 입력하세요.</c:if>
</p>
<input type="submit" value="암호 변경">
</form>
</body>
</html>
```

21.11.5 암호 변경 기능 테스트

- 톰갯을 재시작하고 로그인을 한 뒤에 http://localhost:8080/board/changePwd.do 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.