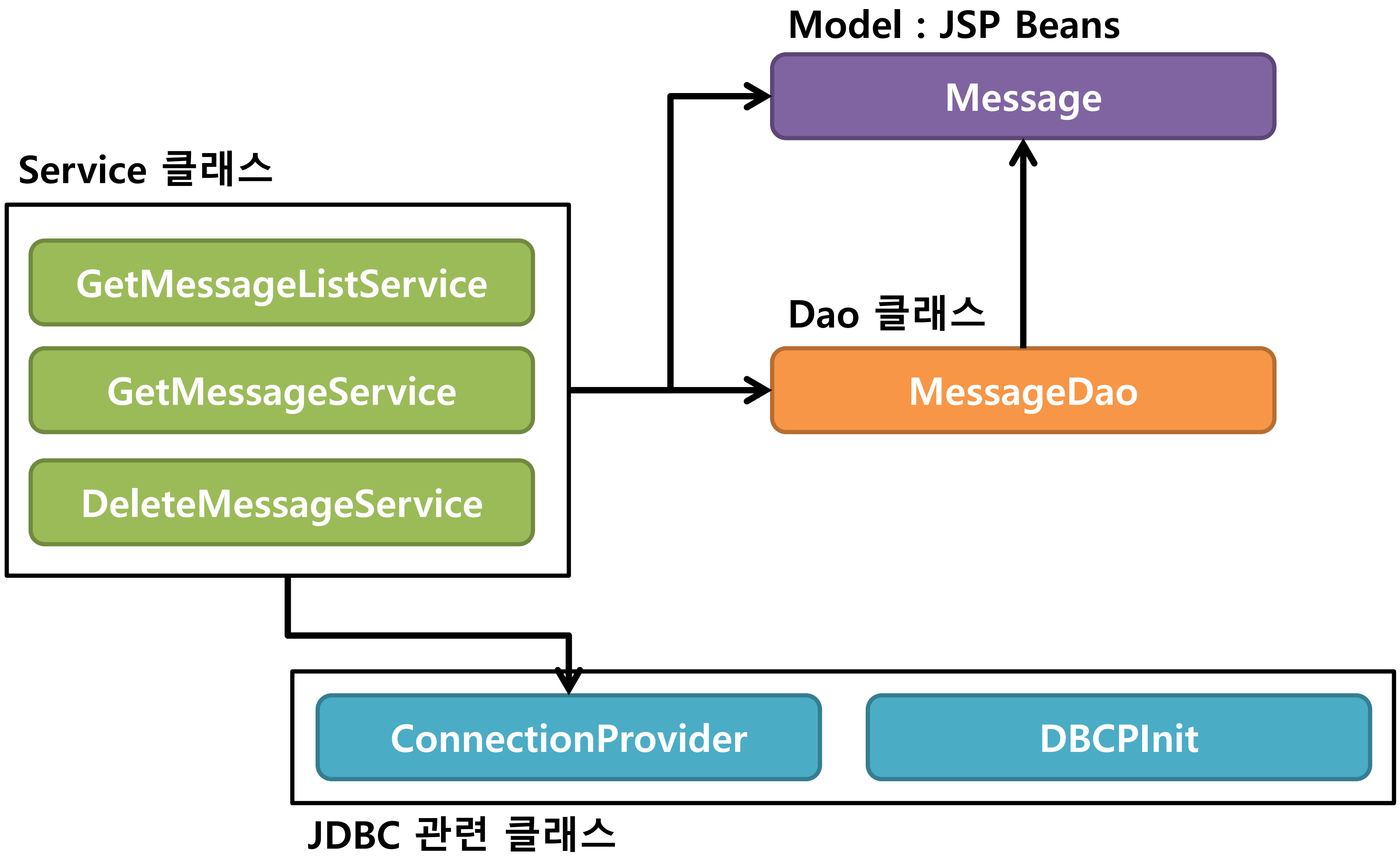
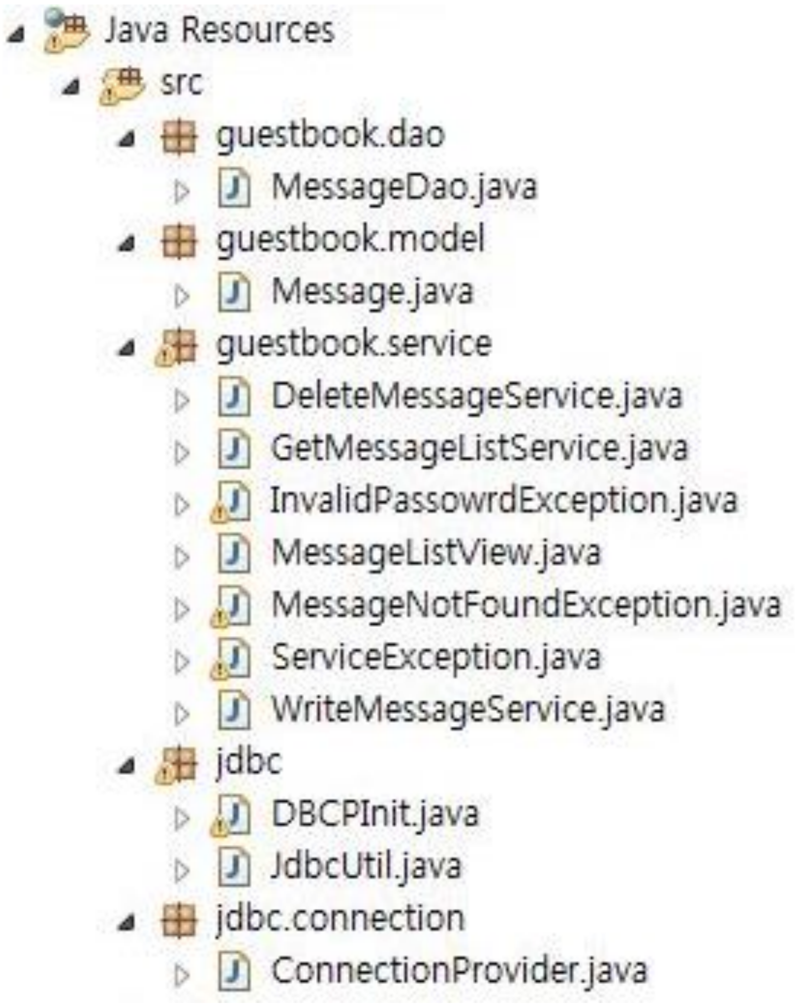
JSP (Java Sever Page)

- 웹 어플리케이션 – 방명록 만들기

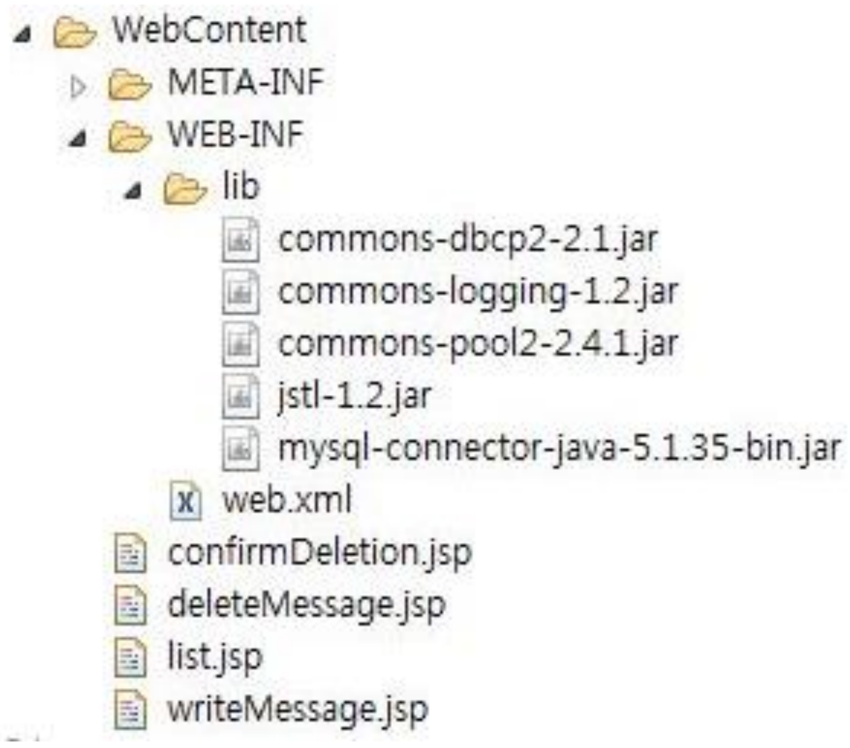
■ 방명록의 구성 예


Model : JSP Beans
Message
Service 클래스
GetMessageListService
GetMessageService
DeleteMessageService
Dao 클래스
MessageDao
ConnectionProvider
DBCPInit
JDBC 관련 클래스

■ 클래스 구성



```
Java Resources
  src
    guestbook.dao
      MessageDao.java
    guestbook.model
      Message.java
    guestbook.service
      DeleteMessageService.java
      GetMessageListService.java
      InvalidPassowrdException.java
      MessageListView.java
      MessageNotFoundException.java
      ServiceException.java
      WriteMessageService.java
    jdbc
      DBCPInit.java
      JdbcUtil.java
    jdbc.connection
      ConnectionProvider.java
```



```
WebContent
  META-INF
  WEB-INF
    lib
      commons-dbcp2-2.1.jar
      commons-logging-1.2.jar
      commons-pool2-2.4.1.jar
      jstl-1.2.jar
      mysql-connector-java-5.1.35-bin.jar
    web.xml
  confirmDeletion.jsp
  deleteMessage.jsp
  list.jsp
  writeMessage.jsp
```

■ 클래스 구성

패키지	클래스	기능
dao	MessageDao	테이블의 CRUD 기능 구현
model	Message	테이블의 컬럼과 메칭되는 데이터 관리 기능
service	DeleteMessageService	글 삭제 기능
	GetMessageListService	요청한 페이지 번호에 포함된 메시지 목록을 보여준다
	WriteMessageService	글 작성 기능
	InvalidMessagePassowrdException	비밀번호 유효성 예외처리
	MessageListView	게시물의 리스트
	MessageNotFoundException	메시지 검색결과 예외처리
	ServiceException	서비스 불가 예외처리
Util	JdbcUtil	close(), 와 rollback 기능
	ConnectionProvider	DBMS 제공 기능 (구성에 따라 구현)
	DBCPInit	Connection Pool 기능. DBCP 서블릿 초기화

1. TABLE : guestbook_message

```
create sequence message_id_seq increment by 1 start with 1;


create table guestbook_message (
    message_id int not null primary key,
    guest_name varchar(50) not null,
    password varchar(10) not null,
    message long varchar not null
)
```

■ 2. 프로젝트 생성 및 모듈 준비

- Database 드라이버
- 컨넥션 풀
- 로깅

3. DBCPInit.java : DBCP 초기화 서블릿 등록

앞에서 테스트한 설정 파일 사용

데이터 베이스이름 변경 및 풀 이름 변경 처리 후 사용

```
private void initConnectionPool() {
        try {
                //String jdbcUrl =  "jdbc:mysql://localhost:3306/test?" +
                        "useUnicode=true&characterEncoding=utf8";
                String jdbcDriver = "jdbc:oracle:thin:localhost:1521:orcl";
                String username = “scott";
                String pw = “tiger";
```

4. web.xml : DBMS 초기화 서블릿 등록

```
<servlet>
		<servlet-name>DBCPInit</servlet-name>
		<servlet-class>jdbc. DBCPInit</servlet-class>
		<load-on-startup>1</load-on-startup>
</servlet>
```

5. TABLE 과 맵핑되는 클래스 : Message.java

```
package model;

public class Message {
		private int id;
		private String guestName;
		private String password;
		private String message;

		public int getId() { return id; }
		public void setId(int id) { this.id = id; }
		public String getGuestName() { return guestName; }
		public void setGuestName(String guestName) { this.guestName = guestName; }
		public String getPassword() { return password; }
		public void setPassword(String password) { this.password = password; }
		public String getMessage() { return message; }
		public void setMessage(String message) { this.message = message; }
		public boolean hasPassword() { return password != null && !password.isEmpty(); }
		public boolean matchPassword(String pwd) {
				return password != null && password.equals(pwd); }
}
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
public class MessageDao {
		private static MessageDao messageDao = new MessageDao();
		public static MessageDao getInstance() {
				return messageDao;
		}
		private MessageDao() {}

		public int insert(Connection conn, Message message) throws SQLException {
				PreparedStatement pstmt = null;
				try {
				pstmt = conn.prepareStatement("insert into guestbook_message "
				+ "(message_id, guest_name, password, message) "+ "values
				(message_id_seq.NEXTVAL, ?, ?, ?)");
						pstmt.setString(1, message.getGuestName());
						pstmt.setString(2, message.getPassword());
						pstmt.setString(3, message.getMessage());
						return pstmt.executeUpdate();
				} finally {
						JdbcUtil.close(pstmt);
				}
		}
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
public Message select(Connection conn, int messageId) throws SQLException {
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        try {
                pstmt = conn.prepareStatement(
                "select * from guestbook_message where message_id = ?");
                pstmt.setInt(1, messageId);
                rs = pstmt.executeQuery();
                if (rs.next()) {
                        return makeMessageFromResultSet(rs);
                } else {
                        return null;
                }
        } finally {
                JdbcUtil.close(rs);
                JdbcUtil.close(pstmt);
        }
}
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
private Message makeMessageFromResultSet(ResultSet rs) throws SQLException {
		Message message = new Message();
		message.setId(rs.getInt("message_id"));
		message.setGuestName(rs.getString("guest_name"));
		message.setPassword(rs.getString("password"));
		message.setMessage(rs.getString("message"));
		return message;
}
public int selectCount(Connection conn) throws SQLException {
		Statement stmt = null;
		ResultSet rs = null;
		try {
				stmt = conn.createStatement();
		rs = stmt.executeQuery("select count(*) from guestbook_message");
				rs.next();
				return rs.getInt(1);
		} finally {
				JdbcUtil.close(rs);
				JdbcUtil.close(stmt);
		}
}
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

Oracle

```
select message_id, guest_name, password, message
from (
        select rownum rnum, message_id, guest_name, password, message
        from (
                select *
                from guestbook_message m
                order by m.message_id desc
        )
        where rownum <= ?
        )
where rnum >= ?
```

MYSQL

```
select * from guestbook_message order by message_id desc limit ?, ?
                                                    [시작번호행],[읽어올 개수]
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
public List<Message> selectList(Connection conn, int firstRow, int endRow)
                        throws SQLException {
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            try {
                        pstmt = conn.prepareStatement(
                        "select message_id, guest_name, password, message from ( "
+ "     select rownum rnum, message_id, guest_name, password, message from ( "
+ "         select * from guestbook_message m order by m.message_id desc "
+ "     ) where rownum <= ? “ + ") where rnum >= ?");
                        pstmt.setInt(1, endRow);
                        pstmt.setInt(2, firstRow);
                        rs = pstmt.executeQuery();
            if (rs.next()) {
                        List<Message> messageList = new ArrayList<Message>();
                                    do {
                        messageList.add(super.makeMessageFromResultSet(rs));
                                    } while (rs.next());
                                    return messageList;
                        } else {
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
                                        return Collections.emptyList();
                              }
                    } finally {
                              JdbcUtil.close(rs);
                              JdbcUtil.close(pstmt);
                    }
          }
          public int delete(Connection conn, int messageId) throws SQLException {
                    PreparedStatement pstmt = null;
                    ResultSet rs = null;
                    try {
                              pstmt = conn.prepareStatement(
                              "delete from guestbook_message where message_id = ?");
                              pstmt.setInt(1, messageId);
                              return pstmt.executeUpdate();
                    } finally {
                              JdbcUtil.close(rs);
                              JdbcUtil.close(pstmt);
                    }
          }
}
```

6. MessageDao.java : 테이블에 대한 CRUD쿼리 실행 기능

```
                    if (rs.next()) {
                    List<Message> messageList = new ArrayList<Message>();
                              do {
messageList.add(super.makeMessageFromResultSet(rs));
                              } while (rs.next());
                              return messageList;
                    } else {
                              return Collections.emptyList();
                    }
          } finally {
                    JdbcUtil.close(rs);
                    JdbcUtil.close(pstmt);
          }
     }

}
```

7. ServiceException.java

```
Package service;

public class ServiceException extends Exception {

    public ServiceException(String message, Exception cause) {
        super(message, cause);
    }

    public ServiceException(String message) {
        super(message);
    }

}
```

8. MessageListView.java : 게시물의 리스트

```
package model;

import java.util.List;

public class MessageListView {

        private int messageTotalCount;
        private int currentPageNumber;
        private List<Message> messageList;
        private int pageTotalCount;
        private int messageCountPerPage;
        private int firstRow;
        private int endRow;

        public MessageListView(List<Message> messageList, int messageTotalCount,
                        int currentPageNumber, int messageCountPerPage, int startRow,
                        int endRow) {
```

8. MessageListView.java : 게시물의 리스트

```
		this.messageList = messageList;
		this.messageTotalCount = messageTotalCount;
		this.currentPageNumber = currentPageNumber;
		this.messageCountPerPage = messageCountPerPage;
		this.firstRow = startRow;
		this.endRow = endRow;
		calculatePageTotalCount();
	}

	private void calculatePageTotalCount() {
		if (messageTotalCount == 0) {
			pageTotalCount = 0;
		} else {
			pageTotalCount = messageTotalCount / messageCountPerPage;
			if (messageTotalCount % messageCountPerPage > 0) {
				pageTotalCount++;
			}
		}
	}
```

8. MessageListView.java : 게시물의 리스트

```
        public int getMessageTotalCount() { return messageTotalCount; }

        public int getCurrentPageNumber() { return currentPageNumber; }

        public List<Message> getMessageList() { return messageList; }

        public int getPageTotalCount() { return pageTotalCount; }

        public int getMessageCountPerPage() { return messageCountPerPage; }

        public int getFirstRow() { return firstRow; }

        public int getEndRow() { return endRow; }

        public boolean isEmpty() { return messageTotalCount == 0; }

}
```

9. GetMessageListService.java : 요청한 페이지,목록,개수 등

```
package service;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.Collections;
import java.util.List;
import dao.MessageDao;
import dao.MessageDaoProvider;
import model.Message;
import model.MessageListView;
import jdbc.JdbcUtil;
import jdbc.connection.ConnectionProvider;

public class GetMessageListService {
        private static GetMessageListService instance =
                                new GetMessageListService();

        public static GetMessageListService getInstance() {
                return instance;
        }
```

9. GetMessageListService.java : 요청한 페이지,목록,개수 등

```
private GetMessageListService() {
}
//한 페이지에 보여줄 메시지의 수
private static final int MESSAGE_COUNT_PER_PAGE = 3;

public MessageListView getMessageList(int pageNumber) throws ServiceException {

        Connection conn = null;

        int currentPageNumber = pageNumber;

        try {
                conn = ConnectionProvider.getConnection();
                MessageDao messageDao =
MessageDaoProvider.getInstnace().getMessageDao();
                //전체 메시지 구하기
                int messageTotalCount = messageDao.selectCount(conn);
                List<Message> messageList = null;
                int firstRow = 0;
                int endRow = 0;
```

9. GetMessageListService.java : 요청한 페이지,목록,개수 등

```
                    if (messageTotalCount > 0) {
                    firstRow =(pageNumber - 1) * MESSAGE_COUNT_PER_PAGE + 1;
                    endRow = firstRow + MESSAGE_COUNT_PER_PAGE - 1;
                            messageList =messageDao.selectList(conn, firstRow, endRow);
                    } else {
                            currentPageNumber = 0;
                            messageList = Collections.emptyList();
                    }
                    return new MessageListView(messageList, messageTotalCount,
                    currentPageNumber, MESSAGE_COUNT_PER_PAGE, firstRow, endRow);

                    } catch (SQLException e) {
                            throw new ServiceException("메시지 목록 구하기 실패: "
                                            + e.getMessage(), e);
                    } finally {
                            JdbcUtil.close(conn);
                    }
            }
}
```

10. WriteMessageService.java : 글쓰기

```
package service;

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

import dao.MessageDao;
import dao.MessageDaoProvider;
import model.Message;
import jdbc.JdbcUtil;
import jdbc.connection.ConnectionProvider;

public class WriteMessageService {
		private static WriteMessageService instance =
						new WriteMessageService();

		public static WriteMessageService getInstance() {
				return instance;
		}
```

10. WriteMessageService.java : 글쓰기

```
		private WriteMessageService() {
		}

		public void write(Message message) throws ServiceException {
				Connection conn = null;
				try {
						conn = ConnectionProvider.getConnection();
						MessageDao messageDao =

		MessageDaoProvider.getInstnace().getMessageDao();
						messageDao.insert(conn, message);
				} catch (SQLException e) {
						throw new ServiceException(
										"메시지 등록 실패: " + e.getMessage(), e);
				} finally {
						JdbcUtil.close(conn);
				}
		}

}
```

11. DeleteMessageService.java : 게시글 삭제

```
package service;

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

import dao.MessageDao;
import dao.MessageDaoProvider;
import model.Message;
import jdbc.JdbcUtil;
import jdbc.connection.ConnectionProvider;

public class DeleteMessageService {

        private static DeleteMessageService instance =
                        new DeleteMessageService();

        public static DeleteMessageService getInstance() {
                return instance;
        }
```

11. DeleteMessageService.java : 게시글 삭제

```
        private DeleteMessageService() {}

        public void deleteMessage(int messageId, String password)
                        throws ServiceException, InvalidMessagePassowrdException,
                        MessageNotFoundException {
                Connection conn = null;
                try {
                        conn = ConnectionProvider.getConnection();
                        conn.setAutoCommit(false);
                        MessageDao messageDao =
        MessageDaoProvider.getInstnace().getMessageDao();
                        Message message = messageDao.select(conn, messageId);
                        if (message == null) {
           throw new MessageNotFoundException("메시지가 없습니다:"+ messageId);
                        }
                        if (!message.hasPassword()) {
                                throw new InvalidMessagePassowrdException();
                        }
                        if (!message.getPassword().equals(password)) {
                                throw new InvalidMessagePassowrdException();
                        }
```

11. DeleteMessageService.java : 게시글 삭제

```
                    messageDao.delete(conn, messageId);
                    conn.commit();
                } catch (SQLException ex) {
                    JdbcUtil.rollback(conn);
    throw new ServiceException("삭제 처리 중 에러가 발생했습니다:" + ex.getMessage(), ex);
                } catch (InvalidMessagePassowrdException ex) {
                    JdbcUtil.rollback(conn);
                    throw ex;
                } catch (MessageNotFoundException ex) {
                    JdbcUtil.rollback(conn);
                    throw ex;
                } finally {
                    if (conn != null) {
                        try {conn.setAutoCommit(false);
                        } catch (SQLException e) { }
                        JdbcUtil.close(conn);
                    }
                }
        }
}
```

12. MessageNotFoundException.java

```
package service;

public class MessageNotFoundException extends Exception {

    public MessageNotFoundException(String message) {
        super(message);
    }

}
```

13. InvalidMessagePassowrdException.java

```
package service;

public class InvalidMessagePassowrdException extends Exception {

    public InvalidMessagePassowrdException(String message) {
        super(message);
    }

}
```

14. ConnectionProvider.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:guestbook");
		}
}
```

15. JdbcUtil.java

```
package jdbc;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JdbcUtil {

        public static void close(ResultSet rs) {
                if (rs != null) {
                        try {
                                rs.close();
                        } catch (SQLException ex) {
                        }
                }
        }
```

15. JdbcUtil.java

```
public static void close(Statement stmt) {
        if (stmt != null) {
                try {
                        stmt.close();
                } catch (SQLException ex) {
                }
        }
}

public static void close(Connection conn) {
        if (conn != null) {
                try {
                        conn.close();
                } catch (SQLException ex) {
                }
        }
}
```

15. JdbcUtil.java

```
	public static void rollback(Connection conn) {
		if (conn != null) {
			try {
				conn.rollback();
			} catch (SQLException ex) {
			}
		}
	}
}
```

15. JdbcUtil.java

```
        public static void rollback(Connection conn) {
                if (conn != null) {
                        try {
                                conn.rollback();
                        } catch (SQLException ex) {
                        }
                }
        }
}
```

16. list.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page import="model.Message"%>
<%@ page import="model.MessageListView"%>
<%@ page import="service.GetMessageListService"%>
<%
        String pageNumberStr = request.getParameter("page");
        int pageNumber = 1;
        if (pageNumberStr != null) {
                pageNumber = Integer.parseInt(pageNumberStr);
        }

        GetMessageListService messageListService =
                        GetMessageListService.getInstance();

        MessageListView viewData =
        messageListService.getMessageList(pageNumber);
%>
```

16. list.jsp

```
<html>
<head>   <title>방명록 메시지 목록</title></head>
<body>

<form action="writeMessage.jsp" method="post">
이름: <input type="text" name="guestName" />  <br />
암호: <input type="password" name="password" />  <br />
메시지: <textarea name="message" cols="30" row="3"></textarea>  <br />
<input type="submit" value="메시지 남기기" />
</form>
<hr>

<%  if (viewData.isEmpty()) { %>
등록된 메시지가 없습니다.
```

16. list.jsp

```
<%  } else { /* 메시지 있는 경우 처리 시작 */ %>
<table border="1">
<%  for (Message message : viewData.getMessageList()) { %>
        <tr>
                <td>
                메시지 번호: <%= message.getId() %>  <br/>
                손님 이름: <%= message.getGuestName() %>  <br/>
                메시지: <%= message.getMessage() %>  <br/>
                <a href="confirmDeletion.jsp?messageId=<%= message.getId() %>">
                [삭제하기]</a>
                </td>
        </tr>
<%      } %>
</table>
<%      for (int i = 1 ; i <= viewData.getPageTotalCount() ; i++) { %>
<a href="list.jsp?page=<%= i %>">[<%= i %>]</a>
<%      } %>
<%  } /* 메시지 있는 경우 처리 끝 */ %>
</body>
</html>
```

17. writeMessage.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page errorPage="errorView.jsp" %>
<%@ page import="model.Message" %>
<%@ page import="service.WriteMessageService" %>
<%          request.setCharacterEncoding("euc-kr");               %>
<jsp:useBean id="message" class="model.Message">
            <jsp:setProperty name="message" property="*" />
</jsp:useBean>
<%
            WriteMessageService writeService = WriteMessageService.getInstance();
            writeService.write(message);
%>
<html>
<head>  <title>방명록 메시지 남김</title>         </head>
<body>
방명록에 메시지를 남겼습니다.
<br/>
<a href="list.jsp">[목록 보기]</a>
</body>
</html>
```

18. confirmDeletion.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<html>
<head>
		<title>방명록 메시지 삭제 확인</title>
</head>
<body>

<form action="deleteMessage.jsp" method="post">
<input type="hidden" name="messageId" value="<%=
request.getParameter("messageId") %>" />
메시지를 삭제하시려면 암호를 입력하세요:<br/>
암호: <input type="password" name="password" /> <br />
<input type="submit" value="메시지 삭제하기" />
</form>
</body>
</html>
```

19. deleteMessage.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page errorPage="errorView.jsp" %>
<%@ page import="service.DeleteMessageService" %>
<%@ page import="service.InvalidMessagePassowrdException" %>
<%
		int messageId = Integer.parseInt(request.getParameter("messageId"));
		String password = request.getParameter("password");
		boolean invalidPassowrd = false;
		try {
				DeleteMessageService deleteService = 
DeleteMessageService.getInstance();
				deleteService.deleteMessage(messageId, password);
		} catch(InvalidMessagePassowrdException ex) {
				invalidPassowrd = true;
		}
%>
```

19. deleteMessage.jsp

```
<html>
<head>
		<title>방명록 메시지 삭제함</title>
</head>
<body>
<%  if (!invalidPassowrd) { %>
메시지를 삭제하였습니다.
<%  } else { %>
입력한 암호가 올바르지 않습니다. 암호를 확인해주세요.
<%  }%>
<br/>
<a href="list.jsp">[목록 보기]</a>
</body>
</html>
```

■ 기능추가 요청

- 이전에 구현한 회원가입 부분을 현재 구성 패턴(DAO 패턴) 형태로 구성 합시다.
- 방명록 작성을 회원 로그인한 후 가능하도록 구성을 바꾸어 봅시다.
- 글작성 폼과 리스트를 별도 페이지로 각각 구현.

- 로그인 전

  글작성 → 로그인 폼

  → 로그인 → 글 작성 페이지

  → 회원가입폼 → 회원가입 → 로그인 → 글 작성 페이지