

JSP & Servlet

부록에서는 이 교재에서 사용을 하는 WAS(Web Application Server)서버인 Tomcat Server의 환경 설정의 세부적인 설명과 JSP 및 서블릿에 연동되는 Database Server가 꼭 MySQL 뿐만 아니라 Oracle 및 MS-SQL이 될 수 있기 때문에 이 두 개의 Database Server 연결 설정을 부록에 추가 하였습니다.

APPENDIX

Appendix 01 Oracle과 MS_SQL 연결설정
Appendix 02 Tomcat Server 환경설정

APPENDIX

01

Oracle과 MS_SQL 연결 설정

앞에서 진행한 실습에서는 MySQL을 DBMS로 사용하였습니다. 하지만 개발자가 되어 개발을 진행하다 보면 Oracle과 MS-SQL 등 다른 DBMS를 사용하게 될 경우가 생길 수 있습니다. 따라서 이번 부록에서는 MySQL과 더불어 가장 많이 사용되는 DBMS인 Oracle과 MS-SQL의 JDBC 연결법에 대해서 알아보고 실습해보겠습니다.

01 _ Oracle 연결하기

Oracle은 전 세계 DBMS시장에서 점유율 1위를 차지하고 있는 데이터베이스 서버 프로그램입니다. 국내에서도 가장 많이 사용되는 DBMS 프로그램이죠. 그럼 Oracle의 JDBC 연결법에 대해서 차근 차근 실습해보겠습니다.

여기서 잠깐!

본 교재는 MySQL을 기반으로 만들어졌습니다. 따라서 Oracle과 MS-SQL의 설치 파일은 따로 제공되지 않습니다. Oracle과 MS-SQL이 설치되어 있다는 가정 하에 JDBC 연결과정을 진행하도록 하겠습니다. Oracle과 MS-SQL이 설치는 다른 교재나 인터넷의 강좌를 참고합니다.

01-1 자바와 Oracle를 연결하기 위해서는 Oracle 전용 JDBC 드라이버를 설치해야 됩니다.

Oracle 전용 JDBC 드라이버인 ojdbc14.jar는 출판사 사이트에서 다운로드 받을 수 있습니다. 다운로드받은 ojdbc14.jar 파일을 다음과 같이 C:\WJsp\Wmyapp\WWebContent\WEB-INF\lib 경로에 붙여넣기 합니다.



▲ [그림 etc01-1] Oracle 전용 JDBC 드라이버(ojdbc14.jar) 복사

Oracle 전용 JDBC 드라이버(ojdbc14.jar)를 이클립스 myapp프로젝트의 WebContent/WEB-INF/lib 폴더에 복사를 합니다. 이 위치에 MySQL용 드라이버가 같이 있어도 무방합니다.

01-2 Oracle 접속을 위한 클래스파일과 JSP 파일 작성

먼저 Oracle과 MS-SQL JDBC 연결 테스트를 위해 etc01 패키지를 작성합니다.

DBConnectionMgr.java, RegisterMgr.java, RegisterBean.java 세 개의 java 파일은 etc01 패키지에 작성하시면 됩니다.

(1) table.sql (Oracle용 tblRegister 테이블)

여기서 잠깐!

이번 예제에서 사용할 테이블은 10장에서 만들었던 테이블을 Oracle용 MS-SQL용으로 생성하여 사용 하도록 하겠습니다.

```
CREATE TABLE tblRegister(  
    id VARCHAR2(20 BYTE) NOT NULL PRIMARY KEY,  
    pwd VARCHAR2(20 BYTE) NOT NULL,  
    name CHAR(15 BYTE) NULL,  
    num1 CHAR(6 BYTE) NULL,  
    num2 CHAR(7 BYTE) NULL,  
    email VARCHAR2(30 BYTE) NULL,  
    phone VARCHAR2(30 BYTE) NULL,  
    zipcode CHAR(5 BYTE) NULL,  
    address VARCHAR2(60 BYTE) NULL,  
    job VARCHAR2(30 BYTE) NULL  
);
```

데이터 입력 쿼리문

```
INSERT INTO tblRegister (id, pwd, name, num1, num2, email, phone, zipcode, address, job) VALUES  
( 'rorod', '1234', '이경미', '123456', '1234567', 'rorod@jspstudy.co.kr', '010-1111-2222', '12345',  
'부산 연제구', '프로그래머');
```

(2) DBConnectionMgr.java

DBConnectionMgr.java는 14장에서 사용했던 파일을 복사해서 그대로 사용하시되 드라이버명, 접속URL, 유저 명, 유저 패스워드를 지정하는 멤버변수인 String _driver, _url, _user, _password의 내용만 수정하시면 됩니다.

TIP

SID 확인하는 쿼리문

```
select name from v$database;
```

실습 파일 : source/etc01/DBConnectionMgr.java

```
.....
.....
.....
public class DBConnectionMgr {
    private Vector connections = new Vector(10);
    //////////////// Oracle 연결 //////////////////////////
    private String _driver = "oracle.jdbc.OracleDriver",
    url = "jdbc:oracle:thin:@127.0.0.1:1521:XE",
    user = "root",
    password = "1234";
    //////////////// MS-SQL 연결 //////////////////////////
    /*private String _driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver",
    url = "jdbc:sqlserver://127.0.0.1:1433;databaseName=mydb",
    user = "root",
    password = "1234";*/
    .....
    .....
    .....
    .....
```

드라이버 명을 지정합니다. 여기서는 Oracle JDBC 드라이버를 지정하였습니다.

DBMS 접속을 위한 url입니다. 127.0.0.1은 자신을 가리키는 루프백 아이피이고 1521은 Oracle의 기본 포트번호, XE는 SID입니다. 왼쪽 팁에 나와있는 SID 확인하는 쿼리를 이용해 SID를 확인한 뒤 입력하도록 합니다.

유저 아이디입니다.

유저 패스워드입니다.

MS-SQL 연결을 위한 멤버변수입니다. 차후 MS-SQL 연결 테스트를 위해 주석처리 해놓도록 합시다.

(3) RegisterMgr.java

실습 파일 : source/etc01/RegisterMgr.java

```
package etc01;
import java.sql.*;
import java.util.*;
public class RegisterMgr {

    private DBConnectionMgr pool;

    public RegisterMgr() {
        try {
            pool = DBConnectionMgr.getInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public Vector<RegisterBean> getRegisterList() {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        Vector<RegisterBean> vlist = new Vector<RegisterBean>();
        try {
            con = pool.getConnection();
```

```

        String query = "select * from tblRegister";
        pstmt = con.prepareStatement(query);
        rs = pstmt.executeQuery();
        while (rs.next()) {
            RegisterBean bean = new RegisterBean();
            bean.setId (rs.getString("id"));
            bean.setPwd (rs.getString("pwd"));
            bean.setName (rs.getString("name"));
            bean.setNum1 (rs.getString("num1"));
            bean.setNum2 (rs.getString("num2"));
            bean.setEmail (rs.getString("email"));
            bean.setPhone (rs.getString("phone"));
            bean.setZipcode (rs.getString("zipcode"));
            bean.setAddress (rs.getString("address"));
            bean.setJob (rs.getString("job"));
            vlist.addElement(bean);
        }
    } catch (Exception ex) {
        System.out.println("Exception" + ex);
    } finally {
        pool.freeConnection(con, pstmt, rs);
    }
    return vlist;
}
}

```

(4) RegisterBean.java

실습 파일 : source/etc01/RegisterBean.java

```

package etc01;

public class RegisterBean {

    private String id;
    private String pwd;
    private String name;
    private String num1;
    private String num2;
    private String email;
    private String phone;
    private String zipcode;
    private String address;
    private String job;

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

```

```

public void setPwd(String pwd) {
    this.pwd = pwd;
}
public void setName(String name) {
    this.name = name;
}
public void setNum1(String num1) {
    this.num1 = num1;
}
public void setNum2(String num2) {
    this.num2 = num2;
}
public void setEmail(String email) {
    this.email = email;
}
public void setPhone(String phone) {
    this.phone = phone;
}
public void setZipcode(String zipcode) {
    this.zipcode = zipcode;
}
public void setAddress(String address) {
    this.address = address;
}
public void setJob(String job) {
    this.job = job;
}
public String getId() {
    return id;
}
public String getPwd() {
    return pwd;
}
public String getName() {
    return name;
}
public String getNum1() {
    return num1;
}
public String getNum2() {
    return num2;
}
public String getEmail() {
    return email;
}
}

```



```

    public String getPhone() {
        return phone;
    }
    public String getZipcode() {
        return zipcode;
    }
    public String getAddress() {
        return address;
    }
    public String getJob() {
        return job;
    }
}

```

(5) usingJDBCBean.jsp

usingJDBCBean.jsp는 JSP 파일이므로 WebContent에 etc01 폴더를 만들고 작성합니다.

실습 파일 : source/etc01/usingJDBCBean.jsp

```

<%@ page contentType="text/html; charset=EUC-KR" %>
<%@ page import="java.util.*, etc01.*"%>
<jsp:useBean id="rMgr" class="etc01.RegisterMgr"/>
<html>
<head>
<title>JSP에서 데이터베이스 연동</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body bgcolor="#FFFFCC">
<h2>Bean를 사용한 데이터베이스 연동 예제입니다....</h2>
<br/><br/>
<h3>회원정보</h3>
<table bordercolor="#0000ff" border="1">
<tr>
<td><strong>ID</strong></td>
<td><strong>PWD</strong></td>
<td><strong>NAME</strong></td>
<td><strong>NUM1</strong></td>
<td><strong>NUM2</strong></td>
<td><strong>EMAIL</strong></td>
<td><strong>PHONE</strong></td>
<td><strong>ZIPCODE/ADDRESS</strong></td>
<td><strong>JOB</strong></td>
</tr>
<%
    Vector<RegisterBean> vlist = rMgr.getRegisterList();

```

```

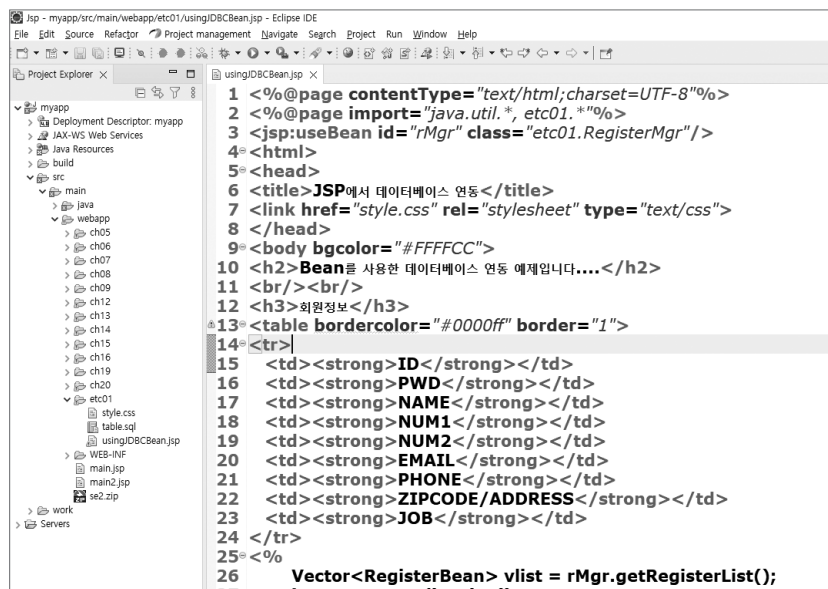
        int counter = vlist.size();
        for(int i=0; i<vlist.size(); i++){
            RegisterBean bean = (RegisterBean)vlist.elementAt(i);
        }
    }
    <tr>
        <td><%=bean.getId()%></td>
        <td><%=bean.getPwd()%></td>
        <td><%=bean.getName()%></td>
        <td><%=bean.getNum1()%></td>
        <td><%=bean.getNum2()%></td>
        <td><%=bean.getEmail()%></td>
        <td><%=bean.getPhone()%></td>
        <td><%=bean.getZipcode()%></td><%=bean.getAddress()%></td>
        <td><%=bean.getJob()%></td>
    </tr>
</table>
<br/><br/>
total records : <%= counter %>
</body>
</html>

```

01-3 연결 테스트

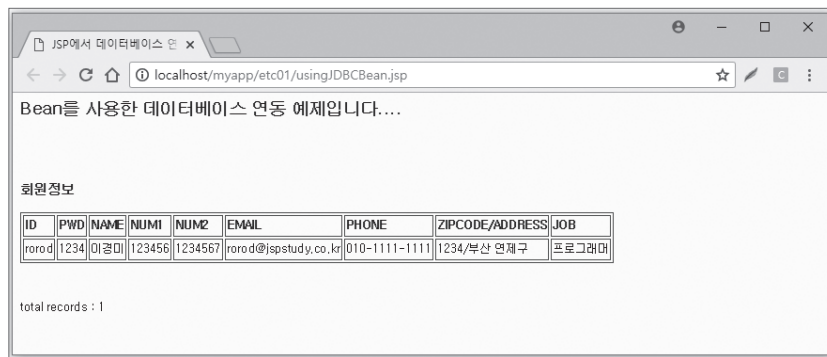
java 파일과 JSP 파일을 모두 작성했으면 연결을 테스트 하도록 하겠습니다.

01 이클립스에서 usingJDBCBean.jsp 파일을 실행합니다.



▲ [그림 etc01-2] usingJDBCBean.jsp 파일 실행

02 다음과 같이 tblRegister 테이블에 입력한 내용이 출력되면 Oracle JDBC가 정상적으로 연결된 것입니다.



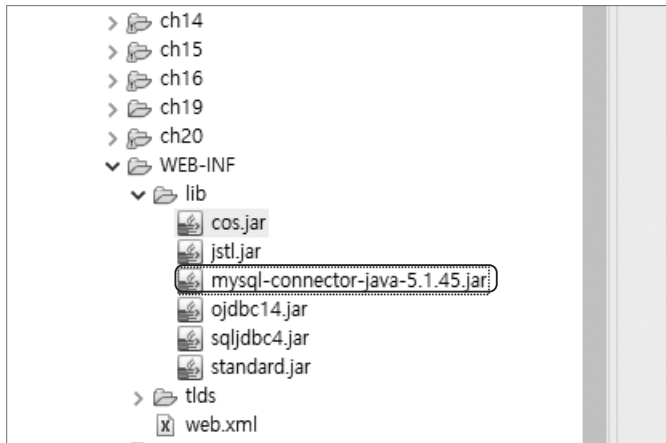
▲ [그림 etc01-3] 결과 확인

02 _ MS-SQL 연결하기

MS-SQL은 마이크로소프트(Microsoft)사에서 개발한 DBMS입니다. Oracle, MySQL과 더불어 가장 많이 사용되는 DBMS 중 하나입니다.

02-1 MS-SQL 전용 JDBC 드라이버 설치

자바와 MS-SQL을 연결하기 위해서 MS-SQL 전용 JDBC 드라이버를 설치하도록 하겠습니다. MS-SQL 전용 JDBC 드라이버인 sqljdbc4.jar는 출판사 사이트에서 다운로드 받을 수 있습니다. 다운로드받은 sqljdbc4.jar 파일을 다음과 같이 C:\WJsp\Wmyapp\WWebContent\WWEB-INF\Wlib 경로에 붙여넣기 합니다.



▲ [그림 etc01-4] 이클립스에서 드라이버파일(sqljdbc4.jar) 파일 확인

MS-SQL 전용 JDBC 드라이버(sqljdbc4.jar)를 이클립스 myapp프로젝트의 WebContent/WEB-INF/lib 폴더에 복사를 합니다. 이 위치에 MySQL용 드라이버와 같이 있어도 무방합니다.

02-2 MS-SQL 접속을 위한 DBConnectionMgr.java 수정

앞에서 Oracle 연결 테스트를 위해 DBConnectionMgr.java, Register.java, RegisterBean.java, usingJDBCBean.jsp 파일을 작성하였기 때문에 MS-SQL 연결을 위해서 DBConnectionMgr.java 파일의 일부만 수정하시면 됩니다. MS-SQL부터 테스트하실 분은 Oracle 연결 테스트 파트에 있는 해당 파일들을 작성해주세요.

etc01 패키지에 작성해놓은 DBConnectionMgr.java 파일을 열어 다음과 같이 Oracle 연결 부분을 주석처리하고 MS-SQL 연결 부분의 주석처리를 해제합니다.

실습 파일 : source/etc01/DBConnectionMgr.java

```
.....
.....
.....
public class DBConnectionMgr {
    private Vector connections = new Vector(10);
    //////////////// Oracle 연결 //////////////////////
    /*private String _driver = "oracle.jdbc.OracleDriver",
    _url = "jdbc:oracle:thin:@127.0.0.1:1521:XE",
    _user = "root",
    _password = "1234";*/
    //////////////// MS-SQL 연결 //////////////////////
    private String _driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver",
    _url = "jdbc:sqlserver://127.0.0.1:1433;databaseName=mydb",
    _user = "root",
    _password = "1234";
    .....
    .....
    .....
```

앞에서 실습한 Oracle 연결 파트입니다. MS-SQL 연결 테스트를 위해 /* */로 주석처리 하도록 합니다.

드라이버 명을 지정합니다. 여기서는 MS-SQL JDBC 드라이버를 지정하였습니다.

사용자 계정입니다.

사용자 비밀번호입니다.

DBMS 접속을 위한 url입니다. 127.0.0.1은 자신을 가리키는 루프백 아이피이고 1433은 MS-SQL의 기본 포트번호입니다. 마지막에는 접속할 DB 이름이 들어갑니다.

02-3 테이블 및 데이터 입력하기

테이블 작성 쿼리문 (MS-SQL용 tblRegister 테이블)

```
CREATE TABLE tblRegister(
    id VARCHAR(20) NOT NULL PRIMARY KEY,
    pwd VARCHAR(20) NOT NULL,
    name CHAR(15) NULL,
    num1 CHAR(6) NULL,
    num2 CHAR(7) NULL,
    email VARCHAR(30) NULL,
    phone VARCHAR(30) NULL,
    zipcode CHAR(5) NULL,
    address VARCHAR(60) NULL,
    job VARCHAR(30) NULL
);
```

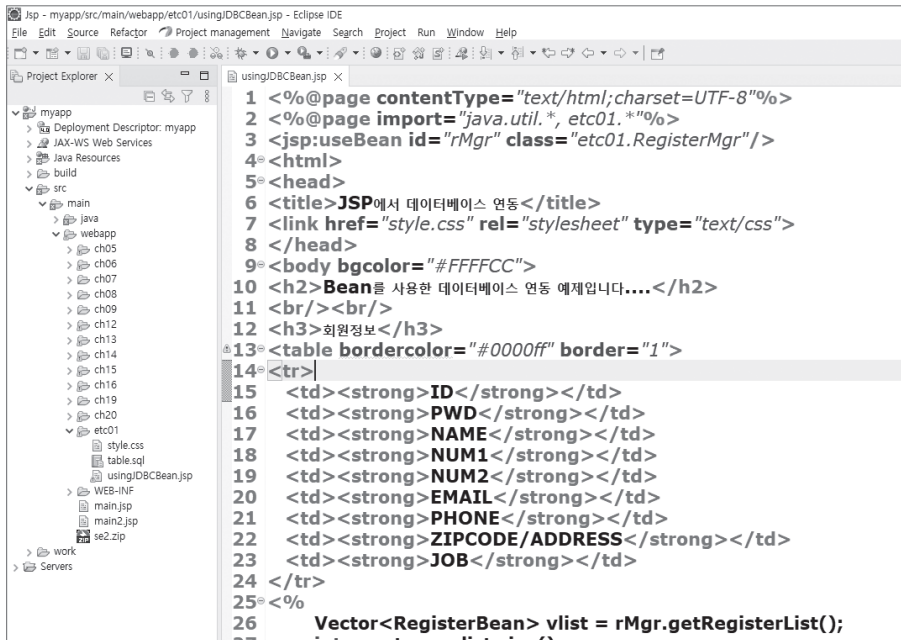
데이터 입력 쿼리문

```
INSERT tblRegister (id, pwd, name, num1, num2, email, phone, zipcode, address, job) VALUES ('rorod', '1234', '이경미', '123456', '1234567', 'rorod@jspstudy.co.kr', '010-1111-2222', '12345', '부산 연제구', '프로그래머');
```

02-4 연결 테스트

java 파일과 JSP 파일을 모두 작성했으면 연결을 테스트 하도록 하겠습니다.

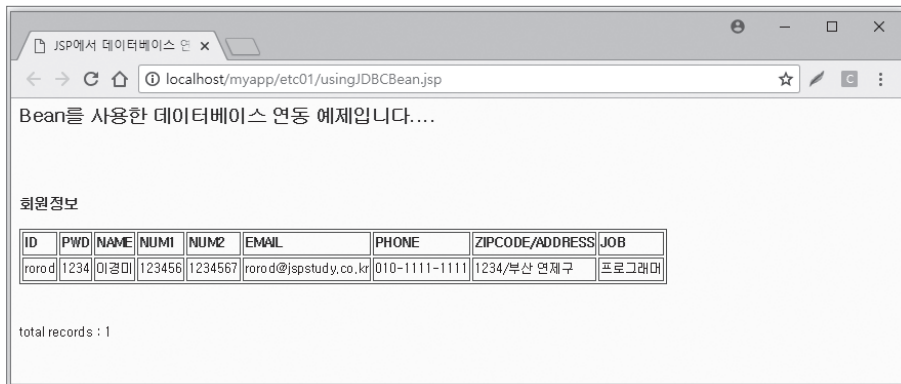
01 이클립스에서 usingJDBCBean.jsp 파일을 실행합니다.



```
1 <%@page contentType="text/html; charset=UTF-8"%>
2 <%@page import="java.util.*, etc01.*"%>
3 <jsp:useBean id="rMgr" class="etc01.RegisterMgr"/>
4 <html>
5 <head>
6 <title>JSP에서 데이터베이스 연동 예제입니다....</title>
7 <link href="style.css" rel="stylesheet" type="text/css">
8 </head>
9 <body bgcolor="#FFFFCC">
10 <h2>Bean를 사용한 데이터베이스 연동 예제입니다....</h2>
11 <br/><br/>
12 <h3>회원정보</h3>
13 <table bordercolor="#0000ff" border="1">
14 <tr>
15 <td><strong>ID</strong></td>
16 <td><strong>PWD</strong></td>
17 <td><strong>NAME</strong></td>
18 <td><strong>NUM1</strong></td>
19 <td><strong>NUM2</strong></td>
20 <td><strong>EMAIL</strong></td>
21 <td><strong>PHONE</strong></td>
22 <td><strong>ZIPCODE/ADDRESS</strong></td>
23 <td><strong>JOB</strong></td>
24 </tr>
25 <%
26 Vector<RegisterBean> ylist = rMgr.getRegisterList();
```

▲ [그림 etc01-5] usingJDBCBean.jsp 파일 실행

02 다음과 같이 tblRegister 테이블에 입력한 내용이 출력되면 MS-SQL JDBC가 정상적으로 연결된 것입니다.



▲ [그림 etc01-6] 결과 확인

APPENDIX

02

Tomcat Server 환경설정

톰캣의 환경설정파일 server.xml의 connector와 context 태그에 대해서 간단한 예제와 함께 알아보고
어플리케이션 설정파일인 web.xml의 설정 방법과 설정에 따른 차이점을 간단한 예제를 통해 알아보겠습
니다.

01 _ 톰캣 server.xml 설정하기

server.xml은 'Tomcat rootWconfW'에 위치하며 톰캣에서 사용할 포트번호, 프로토콜을 설정하고 실행할 어플리케이션의 경로 등을 설정하는 파일입니다. 이번 부록에서는 server.xml 파일에서 사용되는 Connector와 Context 태그에 대해서 알아보겠습니다.

01-1 Connector 태그

Connector 태그는 클라이언트의 요청을 설정된 프로토콜에 따라 수신하고 지정한 엔진에 연결하는 태그입니다.

(1) Connector 태그의 속성

Connector 태그는 service 태그의 내부에 위치하며 일반적으로 아래와 같은 형식으로 설정합니다.

```
<Connector connectionTimeout="20000" port="80" protocol="HTTP/1.1" redirectPort="8443"/>
```

위 소스에서 'connectionTimeout="20000"'은 클라이언트가 서버에 연결 요청을 하였을 때 서버가 요청을 기다리며 대기하는 최대 시간을 뜻합니다. 단위는 ms(밀리세컨드:1000분의 1초)이므로 위 소스의 20000은 20초가 됩니다.

'port="80"'은 Connector가 응답할 포트번호로써 여기서 지정한 포트에 요청이 들어오면 Connector가 생성됩니다. 'HTTP/1.1'의 기본 포트번호는 8080인데 일반적으로 윈도우 서버의 서비스 중 하나인 IIS의 기본 포트번호가 '80'이기 때문에 이와 겹치는 것을 방지하기 위해서 '8080'으로 설정이 되어 있습니다. 하지만 IIS를 사용하지 않을 경우 HTTP/1.1 커넥터의 포트번호는 '80'으로 설정하는 것이 좋습니다. '8080'으로 설정할 경우 서버에 접속할 때 주소 뒤에 'http://localhost:8080'과 같이 포트번호를 붙여주어야 하지만 위와 같이 'port="80"'으로 지정할 경우 주소에서 포트번호를 생략할 수 있습니다.

'protocol="HTTP/1.1"'은 커넥터의 프로토콜을 설정하는 속성입니다. Connector에서 지정할 수 있는 프로토콜의 종류는 'HTTP/1.1'과 'AJP/1.3' 두 가지가 있는데 'HTTP/1.1'은 웹 요청을 받아들이는 프로토콜이고 'AJP/1.3'은 톰캣과 아파치를 연동할 때 사용하는 프로토콜입니다.

'redirectPort="8443"'은 Connector이 non-SSL 상태일 때 SSL 요청을 받을 경우 지정한 포트로 재접속합니다.

위에서 설명한 속성 외에도 connector에는 아래와 같은 속성이 있습니다.

속성	설명
URIEncoding	URI를 디코딩할 때 사용되는 문자 인코딩 타입을 지정합니다. 설정하지 않을 경우 기본값은 ISO-8859-1입니다.
allowTrace	TRACE HTTP 메소드를 사용할지 여부를 설정하는 속성입니다. true/false로 설정할 수 있으며 기본값은 false입니다.
address	하나 이상의 IP 주소를 가진 서버일 경우 이 속성은 지정된 포트에서 수신을 위해 사용되는 주소를 지정합니다.
maxConnections	서버에 연결할 수 있는 최대 연결수입니다. 연결된 수가 지정된 숫자에 도달하면 서버는 더 이상 연결을 허용하지 않습니다.
max-httpHeaderSize	요청 및 응답 HTTP 헤더의 최대 크기입니다. 지정되지 않을 경우 이 속성은 8192B(8KB)로 설정됩니다. 단위는 바이트입니다.
maxThreads	요청 처리 스레드의 최대 개수를 설정합니다. 기본값은 200입니다.
socketBuffer	소켓 버퍼의 크기를 지정합니다. -1 일 경우 버퍼를 사용하지 않습니다. 기본값은 9000B(바이트)입니다.
asyncTimeout	비동기 요청에 대한 시간제한입니다. 기본값은 10000ms(10초)입니다.
maxHeaderCount	컨테이너에 의해 허용되는 요청 헤더의 최대 수입니다. 지정된 수보다 더 많은 헤더가 포함 된 요청은 거부됩니다. 0일 경우 제한이 없음을 뜻합니다. 기본값은 100입니다.
acceptCount	모든 스레드가 사용 중일 때 들어오는 연결 요청을 저장하는 큐의 최대 길이를 설정합니다. 큐가 가득 찰 경우 요청은 거부됩니다. 기본값은 100입니다.

▲ [표 etc02-1] connector 태그의 속성

이 외에도 Connector 태그에는 많은 속성들이 있습니다. 자세한 내용은 <http://tomcat.apache.org/tomcat-9.0-doc>을 참고하시기 바랍니다.

(2) URIEncoding 설정에 따른 차이점

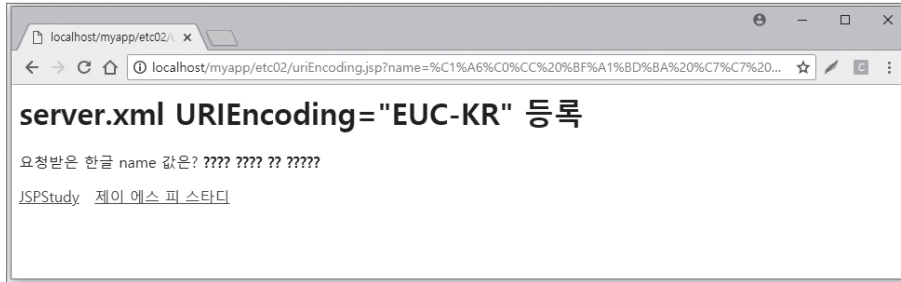
Connector태그에서 URIEncoding의 기본값은 'ISO-8859-1'인데 이 상태로 페이지를 실행할 경우 한글이 깨져서 나오게 됩니다. 따라서 한글을 정상적으로 출력하기 위해서는 URIEncoding의 값을 'EUC-KR'로 설정해줄 필요가 있습니다.

01 먼저 URIEncoding 설정에 따른 결과값의 차이를 확인하기 위한 테스트 페이지를 다음과 같이 작성합니다.

실습 파일 : source/etc02/uriEncoding.jsp

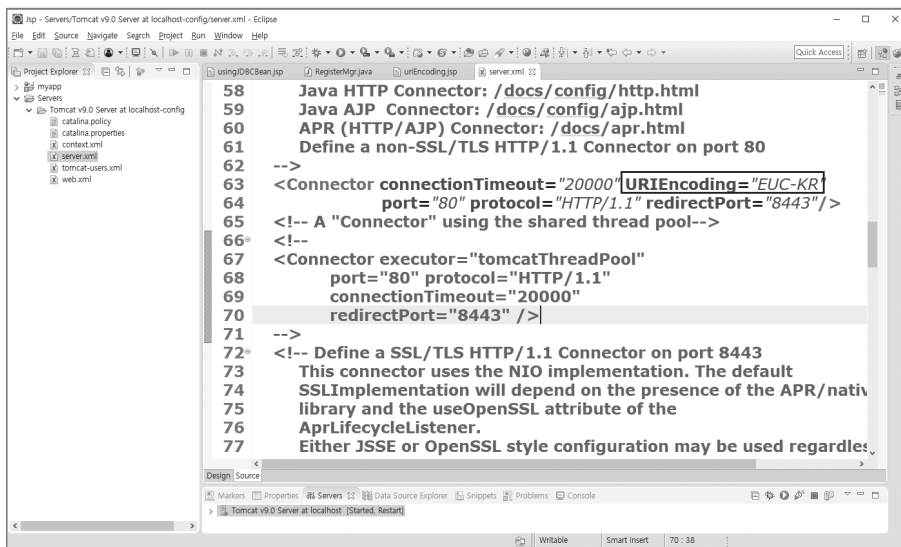
```
<%@ page contentType="text/html; charset=EUC-KR" %>
<h1>server.xml URIEncoding="EUC-KR" 등록</h1>
<%
    String name = request.getParameter("name");
    if(name==null) name = "요청 받은 값이 없음";
%>
요청받은 한글 name 값은? <b>X<%=name %></b><p>
<a href="uriEncoding.jsp?name=JSPStudy">JSPStudy</a>&nbsp;&nbsp;&nbsp;
<a href="uriEncoding.jsp?name=제이 에스 피 스타디">제이 에스 피 스타디</a>
```

02 작성한 페이지를 실행하면 아직 URLEncoding 값이 default(ISO-8859-1) 상태이기 때문에 한글이 깨지는 것을 확인할 수 있습니다.



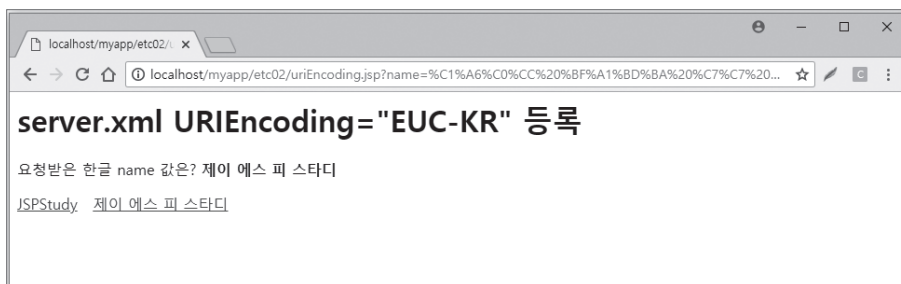
▲ [그림 etc02-1] URLEncoding 값이 default(ISO-8859-1)때의 한글 출력

03 톰캣 9.0 디렉토리에 위치한 server.xml 파일을 열어서 다음과 같이 Connector에 'URLEncoding="EUC-KR"'를 추가를 합니다.



▲ [그림 etc02-2] server.xml에서 URLEncoding 값을 'EUC-KR'로 설정

04 설정 후 다시 서버를 재시작하고 페이지를 실행시키면 한글이 정상적으로 출력되는 것을 확인할 수 있습니다.



▲ [그림 etc02-3] URLEncoding 값이 'EUC-KR'일 때의 한글 출력

01-2 host 태그

host 태그는 일반적으로 아래와 같은 형식으로 설정합니다.

```
<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
```

host 태그는 가상 호스트를 생성하는 태그입니다. 각각의 host 태그는 하나의 가상호스트가 되며 연결할 도메인, 호스트의 root폴더와 같은 속성을 지정합니다. 또한 하나의 host 태그 내부에는 다수의 Context 태그가 들어가는데 이 Context 태그는 하나의 웹 어플리케이션을 지정합니다.

host 태그는 대표적으로 아래와 같은 속성을 가집니다.

속성	설명
appBase	가상호스트의 root 폴더를 지정합니다.
name	가상호스트에 연결할 도메인(주소)을 지정합니다.
autoDeploy	톰캣이 실행 상태일 때 appBase에 지정한 root 디렉토리에 어플리케이션이 추가될 경우 Context 태그를 추가할지 여부를 지정합니다.
unpackWARs	appBase에 지정한 root 디렉토리에 WAR 파일을 추가할 경우 압축을 해제할 것인지의 여부를 지정합니다. true일 경우 압축을 해제한 뒤 해당 디렉토리에서 실행하고 false일 경우 WAR 압축 상태로 실행합니다.

▲ [표 etc02-2] connector 태그의 속성

(1) Context 태그의 개념과 속성

Context 태그는 일반적으로 아래와 같은 형식으로 설정합니다.

```
<Context docBase="myapp" path="/myapp" workDir="C:\Study\myapp\work" reloadable="true" />
```

Context 태그는 가상 호스트 안에서 실행되는 어플리케이션의 경로와 옵션을 설정하는 태그입니다. 하나의 Context 태그는 하나의 웹 어플리케이션을 의미합니다.

Context 태그가 일반적으로 가지는 속성은 다음과 같습니다.

속성	설명
docBase	웹 어플리케이션의 root 디렉토리 또는 WAR 파일의 경로를 지정합니다. 절대경로 또는 host의 appBase에 대한 상대경로로 지정할 수 있습니다.
path	웹 어플리케이션의 컨텍스트 경로를 지정합니다. path 경로는 유일성을 가지며 "(공백)"으로 지정할 경우 호스트의 기본 어플리케이션이 되어 다른 컨텍스트에 해당되지 않는 모든 요청을 받아들이게 됩니다.
workDir	컨텍스트를 위한 작업(임시) 디렉토리 경로를 지정합니다.
reloadable	/WEB-INF/classes/와 /WEB-INF/lib 경로의 파일이 변경됐을 때 자동 리로딩 여부를 결정합니다.

▲ [표 etc02-3] Context 태그의 속성

02 _ web.xml 설정하기

web.xml은 톰캣의 배포서술자(DD, Deployment Descriptor)로써 웹 어플리케이션의 환경설정을 위한 파일입니다.

web.xml 파일은 기본적으로 Tomcat ROOTWconf에 위치하는데 웹 어플리케이션에 web.xml 파일이 없을 경우 Tomcat ROOTWconf에 위치한 web.xml 파일이 적용됩니다.

따라서 웹 어플리케이션의 WebContent/WEB-INF 경로에 web.xml 파일을 위치시키게 되면 Tomcat ROOTWconf에 위치한 web.xml 파일과 별개로 웹 어플리케이션의 실행 환경을 설정할 수 있습니다.

이 경우 일반적으로 Tomcat ROOTWconf에 위치한 web.xml 파일을 WebContent/WEB-INF에 복사해서 필요한 부분만 설정해서 사용하게 됩니다.

그럼 web.xml의 환경설정에 대해서 알아보겠습니다.

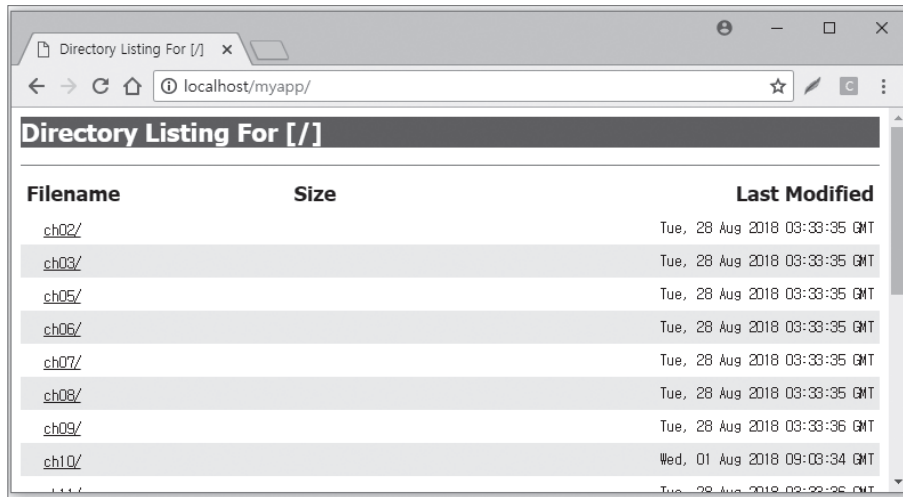
02-1 listings

```
<init-param>
  <param-name>listings</param-name>
  <param-value>true</param-value>
</init-param>
```

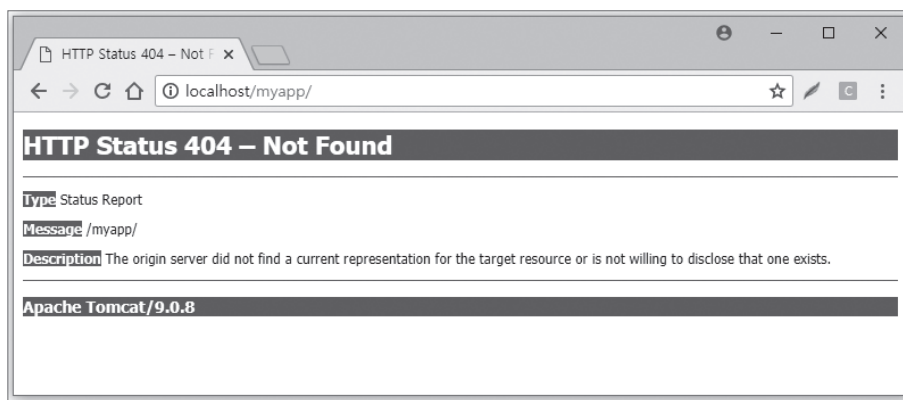
listings는 위와 같은 형태로 작성하며 사용자가 주소창에 'http://localhost/myapp'와 같이 파일명을 제외한 디렉토리 경로만을 입력했을 때 해당 경로에 위치하는 파일과 디렉토리 목록을 브라우저에 보여줄 것인지의 여부를 설정합니다.

listings가 true로 되어 있을 경우 브라우저를 통해 디렉토리와 파일목록을 확인할 수 있기 때문에 개발을 하는데 있어서 편리합니다.

하지만 웹 어플리케이션의 개발이 완료되고 배포된 상태에서 디렉토리와 파일의 목록이 접속자에게 노출되면 보안상 문제가 발생할 수 있기 때문에 이때는 listings를 false로 설정해 주어야 합니다.



▲ [그림 etc02-4] listings이 true인 상태에서 디렉토리 명을 입력했을 때의 결과



▲ [그림 etc02-5] listings이 false인 상태에서 디렉토리 명을 입력했을 때의 결과

그림 4번, 5번과 같이 listings가 true일 경우 디렉토리와 파일목록이 노출되지만 false일 경우 해당 페이지가 없음을 뜻하는 HTTP Status 404 에러를 나타냅니다.

02-2 welcome-file

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

디렉토리명 까지 입력했을 때 default로 보여줄 페이지의 경로 및 파일명

welcome-file는 위와 같은 형태로 작성하며 사용자가 URL을 입력할 때 'http://localhost/myapp/'와 같이 웹 어플리케이션의 디렉토리명까지만 입력할 경우 보여줄 페이지를 지정합니다. welcome-file이 지정되어 있으면 listings가 true로 지정되어 있더라도 디렉토리 경로를 입력하였을 때 welcome-file에서 지정한 페이지로 이동하게 됩니다.

02-3 Exception 발생 시 전환되는 페이지 설정

jsp 페이지를 실행할 때 Exception(예외상황)이 발생하게 되면 에러가 발생한 부분의 소스와 Exception 코드가 웹 브라우저에 노출됩니다.

이 경우 listings와 마찬가지로 보안상 문제가 발생할 수 있기 때문에 Exception 발생 시 보여줄 페이지를 따로 지정해둘 필요가 있는데 이때 사용되는 것이 아래의 코드입니다.

```
<error-page>
  <exception-type>java.lang.Exception</exception-type>
  <location>/etc02/error.jsp</location>
</error-page>
```

Exception(예외) 상황이 발생했을 경우 보여줄 페이지의 경로 및 파일명

위와 같이 설정하면 Exception 발생 시 etc02폴더에 위치한 error.jsp 페이지를 보여주게 됩니다. 그럼 간단한 예제를 통해 Exception 발생 시 보여줄 페이지를 설정해 보겠습니다.

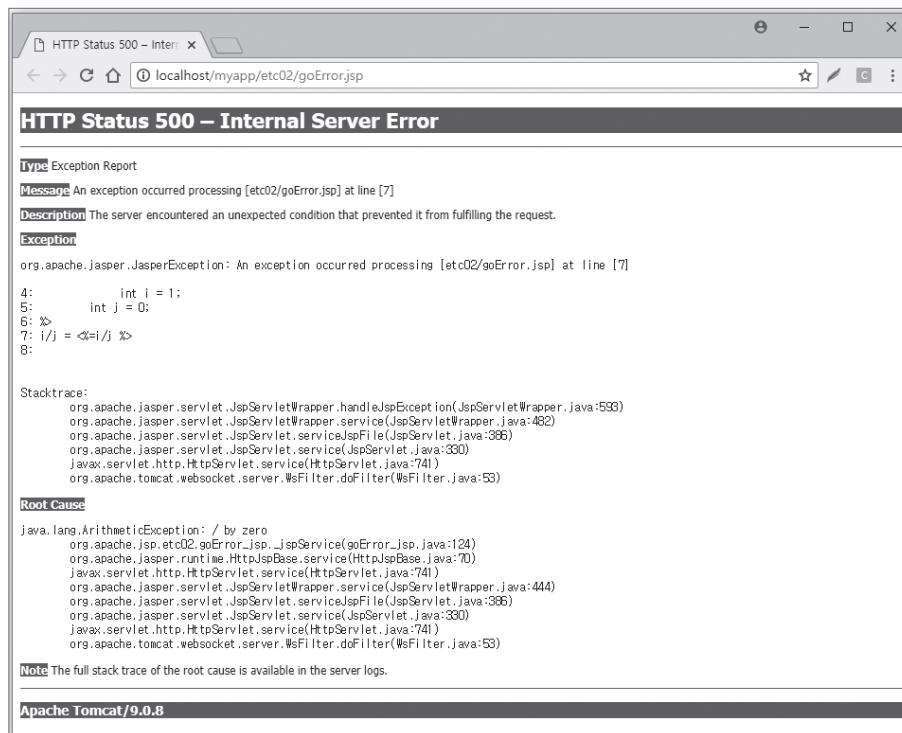
01 먼저 Exception을 발생시키기 위해 다음과 같이 에러가 발생하는 페이지를 작성합니다.

실습 파일 : source/etc02/goError.jsp

```
<%@ page contentType="text/html; charset=EUC-KR" %>
<h1>web.xml 에러페이지 등록</h1>
<%
    int i = 1;
    int j = 0;
%>
i/j = <%=i/j %>
```

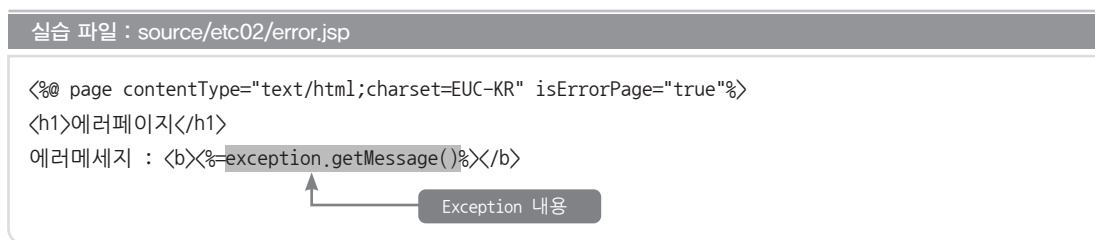
1을 0으로 나누어서 에러를 발생시킵니다.

02 페이지를 작성하고 실행하면 다음과 같이 웹 브라우저에 Exception 코드가 출력됩니다.

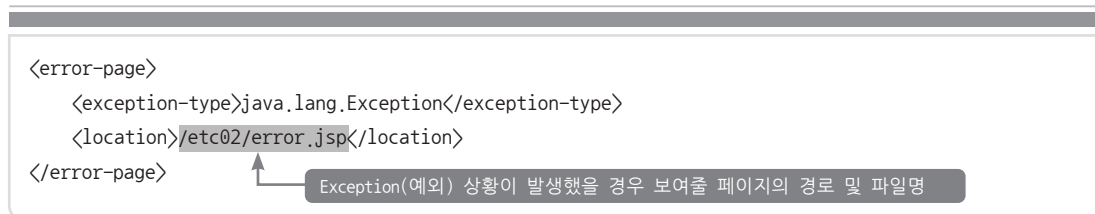


▲ [그림 etc02-6] Exception 페이지를 지정하지 않았을 때의 출력 결과

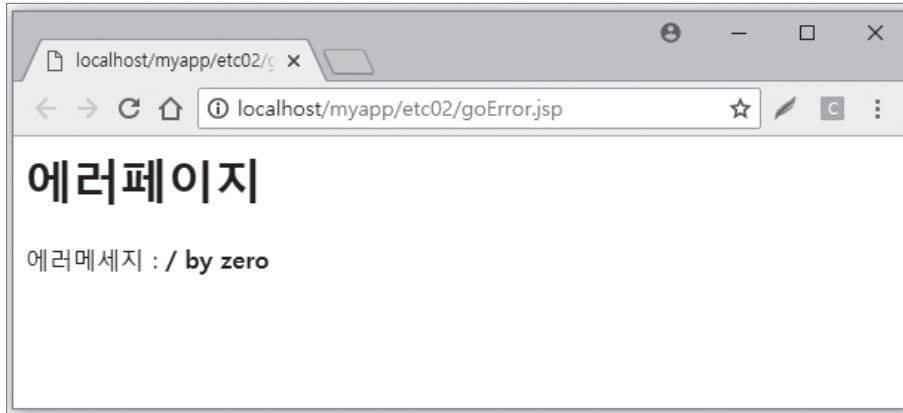
03 Exception이 발생했을 때 보여줄 페이지를 작성합니다.



04 이제 Exception이 발생했을 때 보여줄 페이지를 지정하기 위해 프로젝트의 WEB-INF에 위치한 web.xml에 아래의 코드를 추가합니다.



05 다시 서버를 재시작하고 goError.jsp 페이지를 실행하면 Exception이 발생하고 다음과 같이 지정한 페이지가 출력됩니다.



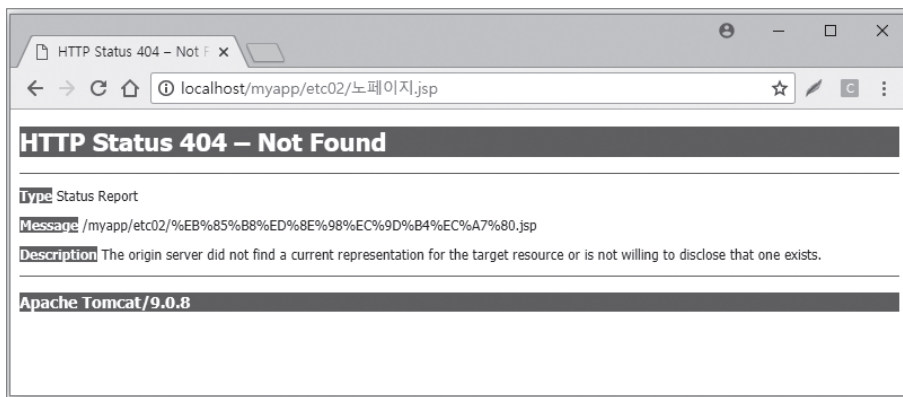
▲ [그림 etc02-7] Exception 페이지를 지정하였을 때의 출력 결과

02-4 404에러 발생 시 전환되는 페이지 설정

404에러는 서버에 존재하지 않는 페이지 명을 주소창에 입력하였을 때 출력되는 에러입니다. 웹 브라우저를 사용하면서 가장 흔하게 볼 수 있는 에러 중 하나죠. 이 경우에도 보안상 문제가 발생할 수 있기 때문에 특정 페이지를 지정해줄 필요가 있습니다.

그럼 간단한 예제를 통해 404에러 발생 시 보여줄 페이지를 지정해 보겠습니다.

01 다음과 같이 'http://localhost/myapp/etc02/' 뒤에 존재하지 않는 페이지 명을 입력할 경우 웹 브라우저는 404에러를 출력합니다. 컨테이너의 종류와 버전도 고스란히 노출이 됩니다.



▲ [그림 etc02-8] 존재하지 않는 페이지 명을 입력하였을 경우 출력 결과

02 먼저 404에러가 발생했을 경우 보여줄 페이지를 작성합니다.

실습 파일 : source/etc02/noPage.jsp

```
<%@ page contentType="text/html; charset=EUC-KR" isErrorPage="true" %>
<h1>요청하신 페이지는 존재하지 않습니다.</h1>
<font color="red">
(익스플로러는 기본적으로 처리하는 404에러 관련 내용이 있기 때문에 내부적인 오류 페이지가 보여 질수 있습니다.)
</font><p/>
<a href="http://JSPStudy.co.kr">JSPStudy</a>
```

03 404에러 발생 시 보여줄 페이지를 다음과 같이 web.xml에 지정해줍니다.

```
<error-page>
    <error-code>404</error-code>
    <location>/etc02/noPage.jsp</location>
</error-page>
```

04 서버를 재시작하고 'http://localhost/myapp/etc02/noname.jsp' 와 같이 존재하지 않는 페이지를 입력하면 다음과 같이 지정한 페이지가 출력되는 것을 확인할 수 있습니다.



▲ [그림 etc02-9] 404에러 페이지를 지정하였을 때의 출력 결과

3장에서 서블릿 관련해서 추가 된 내용을 포함한 web.xml입니다.

실습 파일 : source/etc02/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0">
  <servlet>
    <servlet-name>default</servlet-name>
    <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
    <init-param>
      <param-name>debug</param-name>
      <param-value>0</param-value>
    </init-param>
    <init-param>
      <param-name>listings</param-name>
      <param-value>>false</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>MyServlet2</servlet-name>
    <servlet-class>ch03.MyServlet2</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet2</servlet-name>
    <url-pattern>/ch03/myServlet2</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/etc02/error.jsp</location>
  </error-page>
  <error-page>
    <error-code>404</error-code>
    <location>/etc02/noPage.jsp</location>
  </error-page>
</web-app>
```

APPENDIX

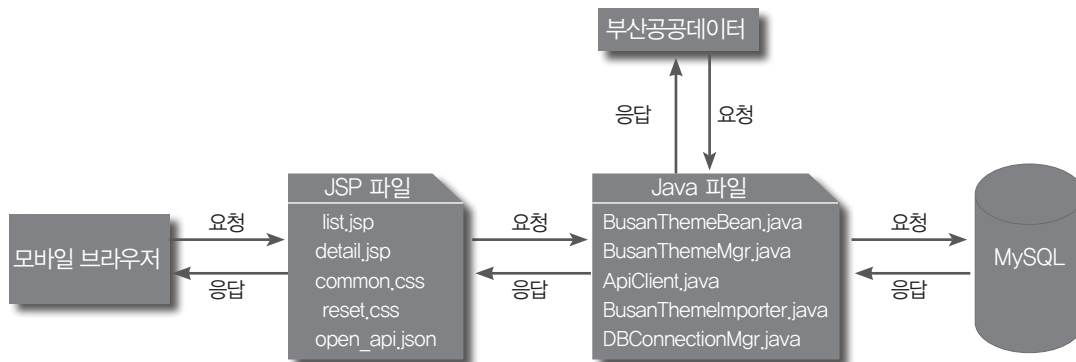
03

모바일웹 테마여행정보

이번 장에서는 국가에서 제공하는 공공 데이터를 활용하여 부산의 테마여행정보를 모바일 웹 스타일의 리스트와 상세보기 페이지로 구성해 보겠습니다. 전 세계적으로 이처럼 공공 데이터를 체계적으로 관리하고 개방하는 나라는 많지 않습니다. 이렇게 가치 있는 데이터를 활용하여 모바일 웹 페이지를 만들어 보도록 하겠습니다.

이를 통해 공공 데이터의 활용 가능성을 탐색하고, 실용적인 웹 서비스를 제작하는 경험을 쌓을 수 있습니다. 또한, 다양한 데이터 타입을 처리하는 방법도 같이 설명하도록 하겠습니다.

● 테마여행정보 전체적인 구조



▲ [그림 etc03-1] 테마여행정보 전체 구조도

소스목록

- | | |
|----------|---|
| JSP&HTML | <ul style="list-style-type: none"> - list.jsp (리스트 페이지) - detail.jsp (상세보기 페이지) - common.css, reset.css (CSS 페이지) - open_api.json (테마여행정보 JSON 파일) |
| <hr/> | |
| 자바&빈즈 | <ul style="list-style-type: none"> - BusanThemeBean.java (테마여행정보 테이블 빈즈) - BusanThemeMgr.java (테마여행정보 SQL 실행 자바) - ApiClient.java (공공data 테마여행정보 JSON 생성 자바) - BusanThemeImporter.java (JSON 파일 테이블 저장 자바) - DBConnectionMgr.java (데이터베이스 Connectionpool 자바) |

테마 여행 정보의 기능

- 공공 데이터 테마여행정보를 JSON 파일로 생성하는 기능이 있습니다."
- JSON을 DB에 저장하는 기능이 있습니다.
- 테마여행정보 리스트를 보여줍니다.
- 테마여행정보 상세보기 기능이 있습니다.

01 _ 데이터베이스 설계

먼저 프로그래밍에 앞서 필요한 SQL 테이블을 만들어 보겠습니다. 테마여행정보에 필요한 테이블은 1개입니다. 부산 공공 데이터에서 제공되는 데이터의 SHEET를 참고한 칼럼으로 테이블을 설계 하였습니다.

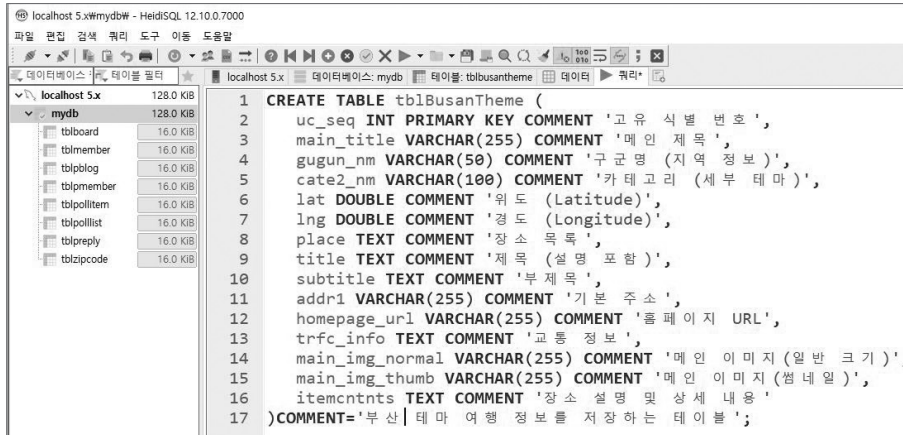
콘텐츠ID	콘텐츠명	구군	구분	위도	경도	장소	제목	부제목
58	초량이바구길 (한,영, 중구)	도보여행	35.11635	129.03874	초량이바구길, 이바구	이야기로 피어난 어저	이바구 꽃이 피었습니	
254	절영해안산책로	영도구	도보여행	35.080116	129.04251	절영해안산책로, 중리	관기만 해도 힐링, 절	바다를 벗 삼아 걷는
282	스카이워크 시리즈	남구	이색여행	35.100853	129.1243	오륙도, 구름산책로, (부산 3대 스카이워크	바다 위를 걷는 즐거움
283	달맞이길/문탠로드	해운대구	도보여행	35.156742	129.1807	달맞이길/문탠로드	일출과 월출 모두를 즐	걷기 좋은 도심 속 숲
284	동백해안산책로	해운대구	도보여행	35.151962	129.15263	동백해안산책로, 누리	바다와 산, 그리고 부	부산을 담은 동백해안
303	피란수도길(한,영,중) 서구	도보여행	35.103672	129.01738	피란수도길, 비석문회	피란민들의 애환이 담		
305	강다니엘코스	영도구	이색여행	35.078865	129.04428	현여울문화마을, 태종	'강다니엘 투어' 따라	
306	부산곳곳, 벚꽃이여리	사상구	이색여행	35.170444	128.97269	낙동강변30리벚꽃길,	2024년 가장 먼저 달	
307	유채꽃명소(한,영,중) 강서구	이색여행	35.19906	128.97322	대저생태공원 팜파스	부산 유채꽃명소 추천		
309	원도심투어	동구	이색여행	35.11617	129.03847	초량이바구길,피란수	이야기가 있는 알짜배	

▲ [그림 etc03-2] 부산 공공데이터에서 제공되는 테마여행정보 SHEET

칼럼명	데이터 타입	설명
uc_seq	int	테마여행정보의 고유 식별 번호를 저장하는 칼럼입니다. uc_seq는 유일한 값이므로 주키(primary key)로 저장하였습니다.
main_title	varchar(255)	메인 제목을 저장하는 칼럼입니다.
gugun_nm	varchar(50)	구군명 (지역 정보)를 저장하는 칼럼입니다.
cate2_nm	varchar(100)	카테고리 (세부 테마)를 저장하는 칼럼입니다.
lat	varchar(100)	위도를 저장하는 칼럼입니다.
lng	double	경도를 저장하는 칼럼입니다.
place	double	장소 목록을 저장하는 칼럼입니다.
title	text	제목 (설명 포함)을 저장하는 칼럼입니다.
subtitle	text	부제목을 저장하는 칼럼입니다.
addr1	varchar(255)	기본 주소를 저장하는 칼럼입니다.
homepage_url	varchar(255)	홈페이지 URL을 저장하는 칼럼입니다.
trfc_info	text	교통 정보를 저장하는 칼럼입니다.

▲ [표 etc03-1] 테마여행정보 테이블 (tblBusanTheme)

앞의 명세표대로 다음과 같이 SQL문을 작성하고 F9 키를 누르면 10장에서 만들어 놓은 mydb 데이터베이스에 tblBusanTheme 테이블을 생성하겠습니다.



▲ [그림 etc03-3] HeidiSQL 테이블 만들기

02 _ 설계 및 구현

테마여행정보를 제공하는 웹 서비스를 단계적으로 구현하려 합니다. 전체 기능을 한 번에 개발하기 보다 흐름에 맞춰 점진적으로 완성하는 것이 기초 JSP 프로그래밍을 배우는 입장에서 더 효과적이라고 생각합니다.

이에 따라, 프로젝트를 두 단계로 나누어 진행하겠습니다.

1단계: 공공데이터 포털에서 여행 정보를 다운로드하여 JSON 파일을 데이터베이스에 저장하는 기능을 구현합니다.

2단계: 여행 정보 리스트를 제공하고, 모바일 웹에서 페이지징 처리를 적용하며, 상세보기 페이지를 구성하는 기능을 구현합니다.

이렇게 단계적으로 접근함으로써 각 과정의 이해도를 높이고, 안정적인 개발이 가능 하도록 하겠습니다.

TIP

이번 장에서는 필수적인 파일 위주로 설명하며, 앞 장에서 다룬 중복된 내용은 생략하겠습니다. 내용이 기억나지 않거나 이해하기 어려운 부분이 있다면 앞 장을 참고하시기 바랍니다.

설명이 생략되는 파일 리스트

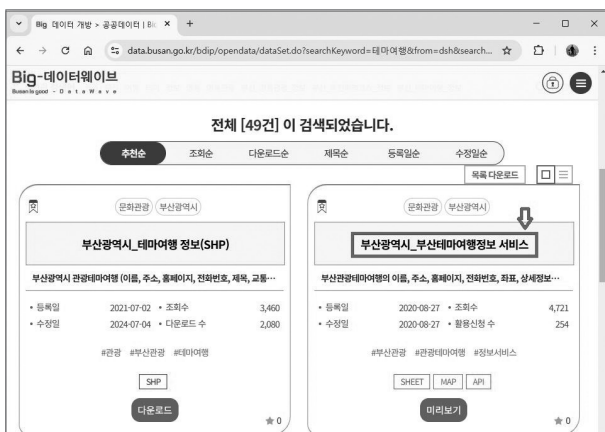
- reset.css
- common.css
- BusanThemeBean.java
- DBConnectionMgr.java

02-1 테마여행정보 다운로드 및 MySQL 데이터 삽입

부산공공데이터 홈페이지(<https://data.busan.go.kr>)에서 ‘테마여행’으로 검색



▲ [그림 etc03-4] Big-데이터웹사이트에서 검색

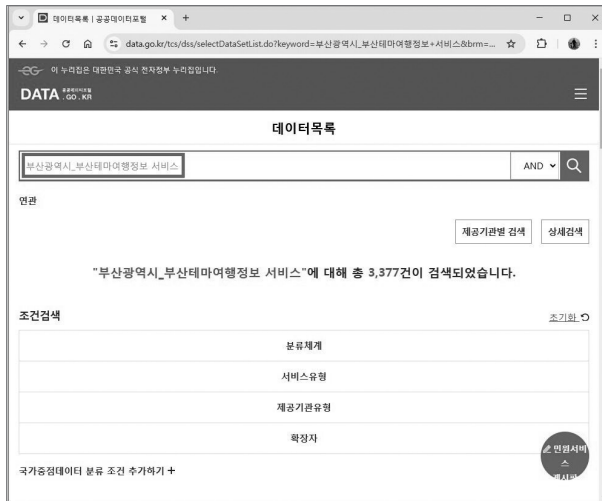


▲ [그림 etc03-5] 검색된 결과에서 부산광역시_부산테마여행정보 서비스 선택

자료확인에는 Big-데이터웹사이트에서 하고 자료를 다운로드 하기 위해서 인증키를 받기 위해서는 공공데이터 포털(<https://www.data.go.kr>)에서 회원가입을 하고 로그인 후에 다음과 같이 승인 요청을 해야 합니다.



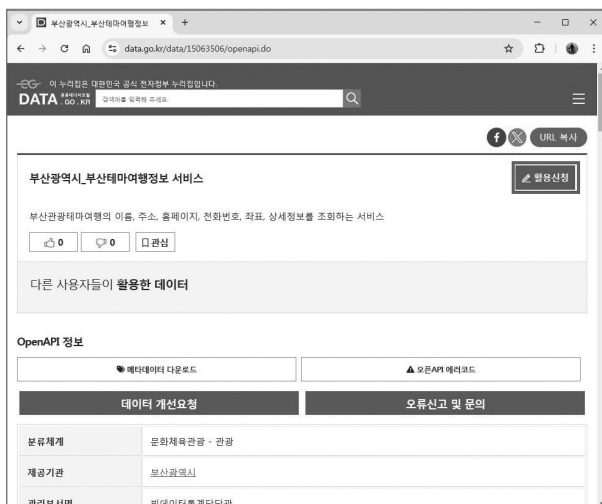
▲ [그림 etc03-6] 부산광역시_부산테마여행정보 Data 리스트 확인



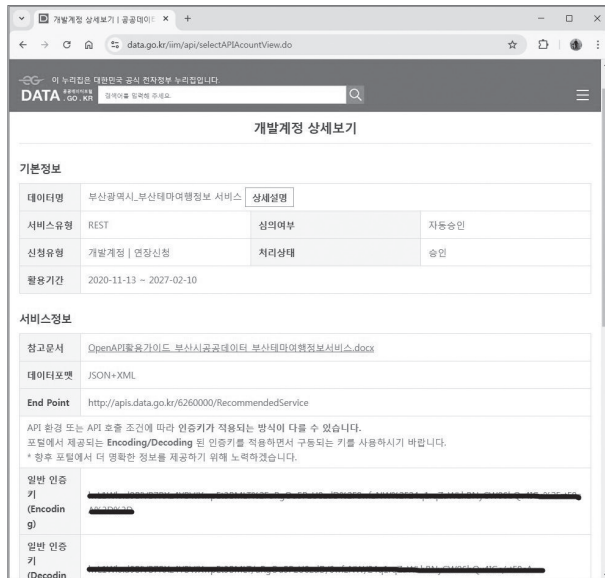
▲ [그림 etc03-7] 공공데이터 포털 로그인 후에 '부산광역시_부산테마여행정보 서비스' 검색



▲ [그림 etc03-8] 검색 후에 '부산광역시_부산테마여행정보 서비스' 선택



▲ [그림 etc03-9] 개인 인증키를 받기 위해서 '활용신청'



▲ [그림 etc03-10] 승인이 완료되면 일반인증키가 제공

여기서 잠깐!

공공데이터 포털 가입 및 활용 신청 절차가 복잡하거나 필요하지 않은 경우, 교재에서 제공하는 open_api.json 파일을 활용하여 데이터를 사용할 수 있습니다.

01 공공 Data에서 테마여행정보를 다운로드 하기 위해 다음과 같이 작성하고 저장합니다.

실습 파일 : source/etc03/ApiClient.java

```
01 : package etc03;
02 :
03 ~ 11: import는 생략합니다.
12 :
13 : public class ApiClient {
14 :
15 :     public static void main(String[] args) {
16 :         String baseUrl = "http://apis.data.go.kr/6260000/RecommendedService/getRecommendedKr";
17 :         String serviceKey = "본인의 일반인증키";
18 :         String outputFilePath = "open_api.json";
19 :         int pageNo = 1;
20 :         int numOfRows = 2200;
21 :
22 :         try {
23 :             StringBuilder urlBuilder = new StringBuilder(baseUrl);
24 :             urlBuilder.append("? " + URLEncoder.encode("ServiceKey", "UTF-8") + "=" +
URLEncoder.encode(serviceKey, "UTF-8"));
25 :             urlBuilder.append("& " + URLEncoder.encode("pageNo", "UTF-8") + "=" + URLEncoder.
encode(String.valueOf(pageNo), "UTF-8"));
26 :             urlBuilder.append("& " + URLEncoder.encode("numOfRows", "UTF-8") + "=" + URLEncoder.
encode(String.valueOf(numOfRows), "UTF-8"));
```

```

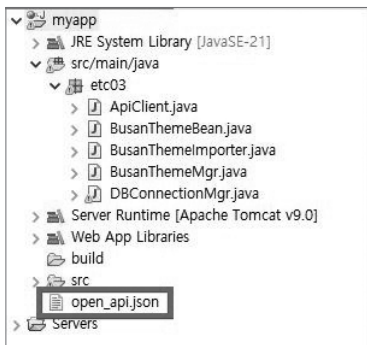
27 :             urlBuilder.append("&" + URLEncoder.encode("resultType", "UTF-8") + "=" +
URLEncoder.encode("json", "UTF-8"));
28 :
29 :             URI uri = new URI(urlBuilder.toString());
30 :             URL url = uri.toURL();
31 :             HttpURLConnection connection = (HttpURLConnection) url.openConnection();
32 :             connection.setRequestMethod("GET");
33 :             connection.setRequestProperty("Accept", "application/json");
34 :             connection.setRequestProperty("Content-Type", "application/json");
35 :             connection.setConnectTimeout(5000);
36 :             connection.setReadTimeout(5000);
37 :
38 :             int responseCode = connection.getResponseCode();
39 :             if (responseCode >= 200 && responseCode <= 300) {
40 :                 BufferedReader reader = new BufferedReader(new InputStreamReader(connection.
getInputStream()));
41 :                 StringBuilder response = new StringBuilder();
42 :                 String line;
43 :                 while ((line = reader.readLine()) != null) {
44 :                     response.append(line).append("\n");
45 :                 }
46 :                 reader.close();
47 :
48 :                 saveToFile(response.toString(), outputFilePath);
49 :                 System.out.println("JSON 데이터가 " + outputFilePath + " 파일에 저장됨.");
50 :             } else {
51 :                 BufferedReader reader = new BufferedReader(new InputStreamReader(connection.
getErrorStream()));
52 :                 StringBuilder errorResponse = new StringBuilder();
53 :                 String line;
54 :                 while ((line = reader.readLine()) != null) {
55 :                     errorResponse.append(line);
56 :                 }
57 :                 reader.close();
58 :                 System.out.println("HTTP 요청 실패: 응답 코드 " + responseCode);
59 :                 System.out.println("오류 메시지: " + errorResponse);
60 :             }
61 :
62 :             connection.disconnect();
63 :         } catch (Exception e) {
64 :             e.printStackTrace();
65 :         }
66 :     }
67 :
68 :     private static void saveToFile(String jsonData, String filePath) {
69 :         try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
70 :             writer.write(jsonData);
71 :         } catch (IOException e) {
72 :             System.out.println("파일 저장 중 오류 발생: " + e.getMessage());
73 :         }
74 :     }
75 : }

```

여기서 잠깐!

ApiClient.java 는 main 메소드가 있는 자바 파일이기 때문에 실행하기 위해서는 이클립스에서 Run As -> Java Application으로 실행해야 합니다.

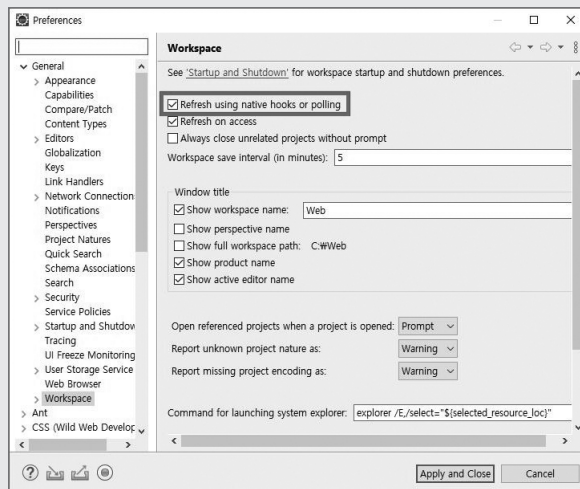
- 17~19 : baseUrl은 공공데이터 사이트에서 제공되는 API의 엔드포인트(URL)를 나타냅니다. serviceKey는 공공데이터 API를 사용하기 위한 개인 인증키이며, outputPath는 응답 데이터를 저장할 JSON 파일입니다.
- 22~66 : 공공데이터 API에서 제공하는 URL과 인증 키를 사용하여 데이터를 요청한 후, JSON 형식의 파일로 저장하는 코드입니다. 이 코드는 Java를 활용해 JSON 데이터를 저장하는 방식을 보여주며, 다른 데이터(API)에도 동일한 방식으로 적용할 수 있습니다.
- 68 ~74 : 매개변수로 받은 JSON 형식의 문자열을 지정된 파일에 저장하는 메소드입니다. 예외 처리를 통해 저장 과정에서 발생할 수 있는 오류를 방지합니다.



▲ [그림 etc03-11] ApiClient.java 실행 후에 생성된 JSON파일

여기서 잠깐!

이클립스와 Windows 탐색기가 동기화되지 않아 open_api.json 파일이 보이지 않을 수 있습니다. 이 경우, 다음과 같이 확인해 보세요.



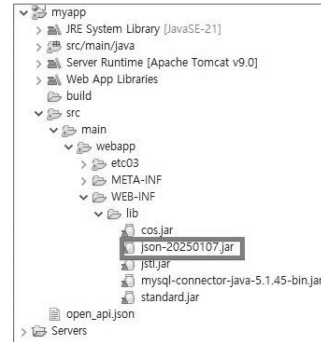
▲ [그림 etc03-12] 이클립스 & Windows 파일 동기화

02-2 JSON파일을 tblBusanTheme 테이블에 입력

공공데이터 포털에서 다운로드한 JSON 파일(open_api.json)을 mydb 데이터베이스의 tblBusanTheme 테이블에 저장하겠습니다.

먼저, JSON 형식의 데이터를 저장하기 위해 외부 라이브러리가 필요합니다.

교재에서 제공하는 json-20250107.jar 파일을 지정된 위치에 저장한 후 사용하시기 바랍니다.



▲ [그림 etc03-13] json-20250107.jar WEB-INF/lib에 저장

여기서 잠깐!

BusanThemeImporter.java를 통해 open_api.json에 있는 값을 tblBusanTheme 테이블에 삽입하려면, DBConnectionMgr.java의 42번째 줄에서 _url 문자열 변수를 다음과 같이 설정해야 합니다.

```
_url = "jdbc:mysql://127.0.0.1:3306/mydb?useUnicode=true&characterEncoding=EUC_KR&useSSL=false";
```

기존 설정에서 useSSL=false 옵션을 추가하여 SSL 연결을 비활성화해야 합니다.

useSSL=false는 데이터베이스 연결 설정에서 SSL(Secure Sockets Layer) 암호화 프로토콜을 사용하지 않도록 설정하는 옵션입니다. JDBC (Java Database Connectivity) URL에서 사용됩니다. 이 옵션을 사용하지 않고, 평문으로 데이터를 전송하도록 지정합니다.

결과
JSON 데이터가 open_api.json 파일에 저장됨.

01 JSON 파일을 지정한 테이블에 입력하기 위해 다음과 같이 작성하고 저장합니다.

실습 파일 : source/etc03/BusanThemeImporter.java

```
01 : package etc03;
02 :
03 ~ 11: import는 생략합니다.
12 : public class BusanThemeImporter {
13 :
14 :     public static void main(String[] args) {
15 :         DBConnectionMgr pool = DBConnectionMgr.getInstance();
16 :         String jsonFilePath = "open_api.json";
17 :         Connection con = null;
18 :         PreparedStatement pstmt = null;
19 :         try {
20 :             String jsonData = readJsonFile(jsonFilePath);
21 :             JSONObject jsonObject = new JSONObject(jsonData);
22 :             JSONArray items = jsonObject.getJSONObject("getRecommendedKr").getJSONArray("item");
23 :             con = pool.getConnection();
24 :             String sql = "INSERT tblBusanTheme (uc_seq, main_title, gugun_nm, "
25 :
```

JSON 파일의 내용을 문자열(String)로 읽어옵니다.

문자열을 JSON 객체로 변환합니다.

JSON 최상위 객체에서 "getRecommendedKr"을 가져와서 "getRecommendedKr" 내부의 "item" 배열을 추출합니다.

```

26 :             + "cate2_nm, lat, lng, place, title, subtitle, addr1, homepage_url, "
27 :             + "trfc_info, main_img_normal, main_img_thumb, itemcntnts) " +
28 :             "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
29 :
30 :         pstmt = con.prepareStatement(sql);
31 :
32 :         for (int i = 0; i < items.length(); i++) {
33 :             JSONObject item = items.getJSONObject(i);
34 :             pstmt.setInt(1, item.getInt("UC_SEQ"));
35 :             pstmt.setString(2, item.optString("MAIN_TITLE", ""));
36 :             pstmt.setString(3, item.optString("GUGUN_NM", ""));
37 :             pstmt.setString(4, item.optString("CATE2_NM", ""));
38 :             pstmt.setDouble(5, item.optDouble("LAT", 0.0));
39 :             pstmt.setDouble(6, item.optDouble("LNG", 0.0));
40 :             pstmt.setString(7, item.optString("PLACE", ""));
41 :             pstmt.setString(8, item.optString("TITLE", ""));
42 :             pstmt.setString(9, item.optString("SUBTITLE", ""));
43 :             pstmt.setString(10, item.optString("ADDR1", ""));
44 :             pstmt.setString(11, item.optString("HOMEPAGE_URL", ""));
45 :             pstmt.setString(12, item.optString("TRFC_INFO", ""));
46 :             pstmt.setString(13, item.optString("MAIN_IMG_NORMAL", ""));
47 :             pstmt.setString(14, item.optString("MAIN_IMG_THUMB", ""));
48 :             pstmt.setString(15, item.optString("ITEMCNTNTS", ""));
49 :
50 :             pstmt.addBatch();
51 :         }
52 :         pstmt.executeBatch();
53 :         System.out.println("데이터가 성공적으로 MySQL에 저장됨.");
54 :     } catch (Exception e) {
55 :         e.printStackTrace();
56 :     } finally {
57 :         pool.freeConnection(con, pstmt);
58 :     }
59 : }
60 :
61 : private static String readJsonFile(String filePath) {
62 :     StringBuilder content = new StringBuilder();
63 :     try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
64 :         String line;
65 :         while ((line = reader.readLine()) != null) {
66 :             content.append(line);
67 :         }
68 :     } catch (IOException e) {
69 :         e.printStackTrace();
70 :     }
71 :     return content.toString();
72 : }
73 : }

```

SQL 문을 한 번 실행하는 것이 아니라 여러 개의 SQL 문을 한 번에 모아둡니다.

addBatch()로 모아둔 모든 SQL 문을 한 번에 실행합니다. 개별 실행할 때보다 속도가 훨씬 빨라집니다.

```

open_api - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
{"getRecommendedKr":{"header":{"code":"00","message":"NORMAL_CODE"},"item":[{"UC_SEQ":5
"font-size28 colorDarkBlue medium">초량이바구길</p><p class="font-size20 medium">옛백제병원 - 남
'이 오르기 힘든 고갯길이라는 걸 알게 된다.\n초량이바구길은 피난민들이 그러했듯, 멈추지 않고 계속되는 우리네 삶의 여정이다.\n\n\n
광지로 자리매김했다. 산책로 담벼락을 따라 조성된 타일 벽화와 파도문양 바닥이 눈앞에 펼쳐진 바다와 어우러져 시원함을 더한다.\n절영해
다.\n"},"UC_SEQ":282,"MAIN_TITLE":"스카이워크 시리즈","GUGUN_NM":"남구","CATE2_NM":"이색여행","LA
송도구름산책로\n장애인 주차구역, 장애인 화장실, 구름산책로 휠체어 진입가능(거북섬까지)\n청사포 다릿돌 전망대\n장애인 주차구역",
어 바다 전망을 더욱 실감나게 즐길 수 있다.\n\n\n"},"UC_SEQ":283,"MAIN_TITLE":"달맞이길/문탠로드","GUGUN_NM
에는 달빛을 맞으며 걷는 길.\n파리에 몽마르트 언덕이 있다면 부산에는 달맞이길이 있다. \n달맞이길은 달맞이 언덕을 올라야하는 길이지
역한다면 확실히 다른 해운대의 숨은 매력을 만나 볼 차례이다.\n\n\n"},"UC_SEQ":284,"MAIN_TITLE":"동백해안산책로",
백 공원은 우거진 해송이 아름다운 곳이다. 동백섬 돌레를 따라 난 해안산책로는 950m 정도로 긴 편은 아니지만, 조용히 산책하며 하루의
도길, 비석문화마을,기차집에출세험장,석당박물관","TITLE":"피란민들의 애환이 담긴 임시수도 부산을 만나다","SUBTITLE":"","M
이 고스란히 남아 있다.\n\n\n\n비석문화마을에서 도보로 3분 정도면 `구름이 쉬어가는 전망대`에 닿을 수 있다. 이곳에선 아미동 일대

```

▲ [그림 etc03-14] open_api.json 파일을 메모장에서 열기

TIP

이클립스에서 open_api.json 파일을 열면 로딩 시간이 길어지고, 정상적인 출력 결과를 확인하기 어렵습니다. 따라서, 메모장 또는 기타 텍스트 편집기를 사용하여 열어보시길 권장합니다.

14~59 : main 메시드는 JSON 파일(open_api.json)을 읽어와 MySQL 데이터베이스의 tblBusanTheme 테이블에 저장하는 역할을 하는 메소드입니다. JSONObject를 사용해 JSON을 파싱하고, JSONArray에서 item 목록을 추출하여 tblBusanTheme 테이블에 데이터를 삽입합니다.

61~72 : 매개변수를 받은 JSON 파일을 문자열(String)로 읽어 반환하는 메소드입니다.

결과

데이터가 성공적으로 MySQL에 저장됨

localhost 5.x MySQL 12.10.0.7000

1 SELECT * FROM tblBusanTheme;

#	uc_seq	main_title	gugun_nm	cate2_nm	lat	lng	place
1	58	초량이바구길 (한,영,중간,중변,일)	동구	도보여행	35.11635	129.03874	초량이바구길, 이바구공작소, 168계단
2	254	절영해안산책로	영도구	도보여행	35.080116	129.04251	절영해안산책로, 종리광, 활여를 위한
3	282	스카이워크 시리즈	남구	이색여행	35.100853	129.1243	오름도, 구름산책로, 다릿돌 전망대
4	283	달맞이길/문탠로드	해운대구	도보여행	35.156742	129.1807	달맞이길/문탠로드
5	284	동백해안산책로	해운대구	도보여행	35.151962	129.15263	동백해안산책로, 누리마루 APEC 하우
6	303	피란수도길(한,영,중간,중변,일)	서구	도보여행	35.103672	129.01738	피란수도길, 비석문화마을,기차집예술
7	305	강다나일코스	영도구	이색여행	35.078865	129.04428	활여를문화마을, 태종대
8	306	부산곳곳, 봄꽃이여(한,영,중간,중변,일)	사상구	이색여행	35.170444	128.97269	낙동강변30리봄꽃길, 달맞이길, 개울
9	307	유채꽃명소(한,영,중간,중변,일)	강서구	이색여행	35.19906	128.97322	대저생태공원 팔파스, 삼학공원, 온천
10	309	원도실투어	동구	이색여행	35.11617	129.03847	초량이바구길, 피란수도길, 율무산길,국
11	310	무장여행	해운대구	이색여행	35.07614	129.0167	부산무장여행, 장애인추천여행,장예
12	323	이기대해안산책로	남구	도보여행	35.130497	129.12091	이기대해안산책로,
13	350	만화와 부산	해운대구	이색여행	35.172466	129.12541	부산책촌, 클로거리, 부산문화콘텐트

03 _ 테마여행정보 리스트 및 상세보기 페이지 구현, 페이징 처리

두 번째 단계에서는 list.jsp를 통해 여행 정보 리스트를 제공하며, 모바일 웹 화면에 최적화된 페이지 처리를 적용합니다. 또한, detail.jsp에서 세련된 디자인의 상세보기 페이지를 구현합니다.

03-1 테마여행정보 리스트 부분과 페이징 및 블록 처리

01 JSP 페이지 사용된 테마여행정보와 관련된 SQL 기능을 다음과 같이 작성하고 저장합니다.

실습 파일 : source/etc03/BusanThemeMgr.java

```
01 : package etc03;
02 :
03 ~ 06 : import는 생략합니다.
07 :
08 : public class BusanThemeMgr {
09 :
10 :     private DBConnectionMgr pool;
11 :
12 :     public BusanThemeMgr() {
13 :         pool = DBConnectionMgr.getInstance();
14 :     }
15 :
16 :     //테마여행정보 리스트를 가져오는 메소드입니다.
17 :     public Vector<BusanThemeBean> listBusanTheme(int page, int perPage) {
18 :         Connection con = null;
19 :         PreparedStatement pstmt = null;
20 :         ResultSet rs = null;
21 :         String sql = null;
22 :         Vector<BusanThemeBean> vlist = new Vector<BusanThemeBean>();
23 :         try {
24 :             con = pool.getConnection();
25 :             sql = "SELECT * FROM tblBusanTheme LIMIT ?, ?";
26 :             pstmt = con.prepareStatement(sql);
27 :             pstmt.setInt(1, (page - 1) * perPage);
28 :             pstmt.setInt(2, perPage);
29 :             rs = pstmt.executeQuery();
30 :             while (rs.next()) {
31 :                 BusanThemeBean theme = new BusanThemeBean();
32 :                 theme.setUcSeq(rs.getInt("uc_seq"));
33 :                 theme.setMainTitle(rs.getString("main_title"));
34 :                 theme.setGugunNm(rs.getString("gugun_nm"));
35 :                 theme.setCate2Nm(rs.getString("cate2_nm"));

```

```

36 :         theme.setPlace(rs.getString("place"));
37 :         theme.setMainImgThumb(rs.getString("main_img_thumb"));
38 :         vlist.addElement(theme);
39 :     }
40 : } catch (Exception e) {
41 :     e.printStackTrace();
42 : } finally {
43 :     pool.freeConnection(con, pstmt, rs);
44 : }
45 : return vlist;
46 : }
47 :
48 : //테마여행정보 테이블에 저장된 전체 레코드 개수를 리턴합니다.
49 : public int getTotalCount() {
50 :     Connection con = null;
51 :     PreparedStatement pstmt = null;
52 :     ResultSet rs = null;
53 :     int total = 0;
54 :     String sql = "SELECT COUNT(*) FROM tblBusanTheme";
55 :     try {
56 :         con = pool.getConnection();
57 :         pstmt = con.prepareStatement(sql);
58 :         rs = pstmt.executeQuery();
59 :         if (rs.next()) {
60 :             total = rs.getInt(1);
61 :         }
62 :     } catch (Exception e) {
63 :         e.printStackTrace();
64 :     } finally {
65 :         pool.freeConnection(con, pstmt, rs);
66 :     }
67 :     return total;
68 : }
69 :
70 : //테마여행정보를 한 개 가져오는 메소드입니다.
71 : public BusanThemeBean getBusanThemeDetail(int ucSeq) {
72 :     Connection con = null;
73 :     PreparedStatement pstmt = null;
74 :     ResultSet rs = null;
75 :     BusanThemeBean theme = null;
76 :     String sql = "SELECT * FROM tblBusanTheme WHERE uc_seq = ?";
77 :     try {
78 :         con = pool.getConnection();
79 :         pstmt = con.prepareStatement(sql);
80 :         pstmt.setInt(1, ucSeq);
81 :         rs = pstmt.executeQuery();

```



```

82 :         if (rs.next()) {
83 :             theme = new BusanThemeBean();
84 :             theme.setUcSeq(rs.getInt("uc_seq"));
85 :             theme.setMainTitle(rs.getString("main_title"));
86 :             theme.setGugunNm(rs.getString("gugun_nm"));
87 :             theme.setCate2Nm(rs.getString("cate2_nm"));
88 :             theme.setPlace(rs.getString("place"));
89 :             theme.setMainImgThumb(rs.getString("main_img_thumb"));
90 :             theme.setSubtitle(rs.getString("subtitle"));
91 :             theme.setAddr1(rs.getString("addr1"));
92 :             theme.setHomepageUrl(rs.getString("homepage_url"));
93 :             theme.setTrfcInfo(rs.getString("trfc_info"));
94 :             theme.setMainImgNormal(rs.getString("main_img_normal"));
95 :             theme.setItemcntnts(rs.getString("itemcntnts"));
96 :         }
97 :     } catch (Exception e) {
98 :         e.printStackTrace();
99 :     } finally {
100 :         pool.freeConnection(con, pstmt, rs);
101 :     }
102 :     return theme;
103 : }
104 : }

```

17~46 : listBusanTheme 메소드 (list.jsp의 42라인에서 사용)

테마여행정보 리스트를 가져오는 메서드입니다. page 매개변수는 페이지 번호를 의미하며, perPage는 한 페이지에 표시할 레코드 개수를 지정하는 값입니다. 현재 list.jsp에서는 이 값을 6개로 설정하고 있습니다.

49~68 : getTotalCount 메소드 (list.jsp의 68라인에서 사용)

테마여행정보 테이블에 저장된 전체 레코드 개수를 반환합니다. 이는 페이징 및 블록 처리를 위해 반드시 필요한 값입니다.

71~103 : getBusanThemeDetail 메소드 (detail.jsp의 33라인에서 사용)

매개변수로 받은 ucSeq 값을 조건에 맞는 테마여행정보를 리턴하는 메서드입니다.

02 테마여행정보 리스트를 기능을 구현하기 위해 다음과 같이 작성하고 저장합니다.

실습 파일 : :source/etc03/list.jsp

```

01 : <%@ page contentType="text/html; charset=EUC-KR" %>
02 : <%@ page import="java.util.Vector" %>
03 : <%@ page import="etc03.BusanThemeMgr" %>
04 : <%@ page import="etc03.BusanThemeBean" %>
05 : <jsp:useBean id="mgr" class="etc03.BusanThemeMgr"/>
06 : <!DOCTYPE html>
07 : <html>
08 : <head>
09 :     <meta charset="EUC-KR"/>

```

테마여행정보에 필요한 BusanThemeMgr 객체를 생성합니다.

```

10 : <meta name="viewport" content="width=device-width, initial-scale=1.0">
11 : <title>부산테마여행</title>
12 : <link rel="stylesheet" href="css/common.css">
13 : </head>
14 : <body>
15 : <!-- wrap -->
16 : <div id="wrap">
17 : <!-- header -->
18 : <header id="header">
19 : <!-- nav -->
20 : <nav id="nav">
21 : <h3 class="logo">
22 : <span>부산</span>
23 : </h3>
24 : </nav>
25 : </header>
26 : <!-- main -->
27 : <main class="common_wrap" id="theme_list_page">
28 : <div id="container">
29 : <section class="common_box content_box">
30 : <div class="common_title theme_title">
31 : <h1>테마 여행</h1>
32 : </div>
33 : <!-- 테마 여행 목록 -->
34 : <div class="theme_list_area">
35 : <%
36 :     int currentPage = 1;
37 :     if(request.getParameter("page") != null) {
38 :         currentPage = Integer.parseInt(request.getParameter("page"));
39 :     }
40 :     int perPage = 6;
41 :
42 :     Vector<BusanThemeBean> vlist = mgr.listBusanTheme(currentPage, perPage);
43 :     for (int i = 0; i < vlist.size(); i++) {
44 :         BusanThemeBean theme = vlist.get(i);
45 :     %>
46 : <ul class="theme_item">
47 : <ul class="theme_num">
48 : <li><%= theme.getCate2Nm() %></li>
49 : <li><%= theme.getGugunNm() %></li>
50 : </ul>
51 : <div class="theme_img">
52 : <a href="detail.jsp?ucSeq=<%= theme.getUcSeq() %>">
53 : ">
54 : </a>
55 : </div>

```

list.jsp 호출 시 page 값이 요청되지 않으면 기본값으로 1을 사용하며, 요청된 경우 해당 값을 currentPage에 리턴합니다.

Vector에 저장된 BusanThemeBean를 리턴 받고 theme 객체에는 현재 출력할 여행 정보(이름, 지역, 이미지, 설명 등)가 들어있습니다.

```

56 :         <div class="theme_article">
57 :             <div>
58 :                 <%= theme.getMainTitle() %> (<%= theme.getPlace() %>)
59 :             </div>
60 :         </div>
61 :     </ul>
62 :     <%
63 :         } // end for
64 :     %>
65 : </div>
66 : <!-- 동적 페이징 처리 시작 -->
67 : <%
68 :     int totalRecord = mgr.getTotalCount();
69 :     int totalPage = (int)Math.ceil((double)totalRecord / perPage);
70 :     int pageBlock = 5;
71 :     int nowBlock = (int)Math.ceil((double)currentPage / pageBlock);
72 :     int pageStart = (nowBlock - 1) * pageBlock + 1;
73 :     int pageEnd = pageStart + pageBlock - 1;
74 :     if(pageEnd > totalPage) {
75 :         pageEnd = totalPage;
76 :     }
77 : %>
78 : <ul class="theme_pagination">
79 :     <% if(currentPage > 1) { %>
80 :         <li class="prev">
81 :             <a href="list.jsp?page=<%= currentPage - 1 %>"></a>
82 :         </li>
83 :         <% } else { %>
84 :         <li class="prev"></li>
85 :         <% } %>
86 :         <% for(int j = pageStart; j <= pageEnd; j++) { %>
87 :             if(j == currentPage) { %>
88 :                 <li class="page_num active">
89 :                     <span><%= j %></span>
90 :                 </li>
91 :             } else { %>
92 :                 <li class="page_num">
93 :                     <a href="list.jsp?page=<%= j %>"><%= j %></a>
94 :                 </li>
95 :             }
96 :         } %>

```

총 데이터 개수(레코드 수)를 가져옵니다.

totalRecord를 perPage로 나누어 총 페이지 수를 계산합니다. 올림 방식으로 페이지 수를 구합니다.

페이징 처리 화면에 최대 5개의 페이지 번호가 한 블록에 표시됩니다. (예: [1][2][3][4][5])

현재 페이지(currentPage)가 몇 번째 블록에 속하는지 계산합니다. 예를 들어, currentPage = 7이고 pageBlock = 5이면 nowBlock=7÷5=1.4 올림을 하면 2가 됩니다.

현재 블록의 첫 번째 페이지 번호를 계산합니다.

pageEnd는 현재 블록의 마지막 페이지 번호를 계산합니다.

마지막 블록에서는 totalPage보다 큰 값을 가질 수 있으므로 제한을 걸어야 합니다.

현재 페이지(currentPage)가 1보다 클 경우, < 버튼을 클릭하면 이전 페이지로 이동합니다.

만약 currentPage = 1이면 이전 버튼을 클릭할 수 없도록 비활성화합니다.

pageStart부터 pageEnd까지 페이지 번호를 동적으로 생성합니다.

현재 페이지는 active 클래스로 강조를 하였습니다.

```

97 :      <% if(currentPage < totalPage) { %>
98 :      <li class="next">
99 :          <a href="list.jsp?page=<%= currentPage + 1 %>"></a>
100 :      </li>
101 :      <% } else { %>
102 :      <li class="next"></li>
103 :      <% } %>
104 :  </ul>
105 :      <!-- 동적 페이지징 처리 끝 -->
106 :  </section>
107 : </div>
108 : </main>
109 : </div>
110 : </body>
111 : </html>

```

만약 `currentPage = totalPage`이면 다음 버튼을 클릭할 수 없도록 비활성화 합니다.

현재 페이지(`currentPage`)가 마지막 페이지(`totalPage`)보다 작을 경우, > 버튼을 클릭하면 다음 페이지로 이동을 합니다.

TIP

meta viewport 태그는 애플이 아이폰, 아이패드 등 자사의 모바일 브라우저의 뷰포트(viewport) 크기 조절을 위해 만들었습니다. (뷰포트에 대한 설명은 본문 참조) meta viewport 태그는 W3C 명세에는 없기 때문에 표준은 아닙니다. 그러나 iOS 장치(아이폰 운영체제 브라우저 safari)가 널리 사용됨에 따라 사실상 표준처럼 사용되고 있고, 다른 브라우저들(Opera, Android, Mobile firefox(Fennec) 등)도 이 태그를 채택하게 됩니다.

- `width=device-width` : 웹 페이지의 크기가 모니터의 실제 크기를 따라가도록 만든 설정으로 모니터, 스마트폰 등의 화면에 맞는 비율로 화면이 뜨도록 돕습니다.
- `initial-scale = 1` : 화면의 줌 정도를 1배율로 한다는 뜻으로 이 값을 크게 키우면 화면이 줄어 크게 보입니다. 스마트폰에서만 효과가 있습니다.

- 36~41** : `currentPage`는 현재 페이지 번호를 나타내는 변수로, 기본값은 1입니다. `perPage`는 한 페이지에 표시할 테마 여행정보의 개수를 설정하는 값이며, 필요에 따라 유동적으로 조정할 수 있습니다. 현재는 6개로 지정을 하였습니다.
- 44~66** : `listBusanTheme` 메소드를 `currentPage`와 `perPage` 매개변수와 함께 호출하면, 테마 여행 정보 리스트를 Vector 형태로 리턴 받습니다. 테마여행정보를 DB에서 불러와서 동적으로 HTML로 출력하는 역할을 합니다. 이를 통해 사용자는 여러 페이지를 넘기면서 여행 목록을 확인할 수 있습니다.
- 68~76** : 페이지징 처리 및 블록 기능을 구현하기 위해, 전체 페이지 개수를 계산하고, 현재 블록을 결정하며, 블록 내에서 시작/끝 페이지를 지정하는 역할을 합니다.
- 78~104** : 페이지징 처리 및 블록 기능을 구현하여 사용자가 여러 페이지로 이동할 수 있도록 하는 것입니다. 이전 페이지 버튼 (`currentPage > 1`일 때만 활성화)과 현재 블록의 페이지 번호 버튼 (`active` 클래스로 현재 페이지 강조) 및 다음 페이지 버튼 (`currentPage < totalPage`일 때만 활성화)를 만드는 영역입니다.

03 테마여행정보 상세보기 페이지 기능을 구현하기 위해 다음과 같이 작성하고 저장합니다.

실습 파일 : source/etc03/detail.jsp

```
01 : <%@ page contentType="text/html; charset=EUC-KR"%>
02 : <%@ page import="etc03.BusanThemeBean" %>
03 : <jsp:useBean id="mgr" class="etc03.BusanThemeMgr"/>
04 : <!DOCTYPE html>
05 : <html>
06 : <head>
07 : <meta charset="EUC-KR"/>
08 : <meta name="viewport" content="width=device-width, initial-scale=1.0">
09 : <title>부산테마여행</title>
10 : <link rel="stylesheet" href="css/common.css">
11 : </head>
12 : <body>
13 :     <div id="wrap">
14 :         <!-- header -->
15 :         <header id="header">
16 :             <!-- nav -->
17 :             <nav id="nav">
18 :                 <h3 class="logo">
19 :                     <span>부산</span>
20 :                 </h3>
21 :             </nav>
22 :         </header>
23 :         <!-- main -->
24 :         <main class="common_wrap" id="theme_detail_page">
25 :             <div id="container">
26 :                 <section class="common_box content_box">
27 :                     <div class="common_title theme_title">
28 :                         <h1>테마 여행 상세보기</h1>
29 :                     </div>
30 :                     <div class="theme_field">
31 :                         <%
32 :                             int ucSeq = Integer.parseInt(request.getParameter("ucSeq"));
33 :                             BusanThemeBean theme = mgr.getBusanThemeDetail(ucSeq);
34 :                             if (theme != null) {
35 :                                 <%
36 :                                     <div class="theme_detail_content">
37 :                                         <div class="theme_detail_img">
38 :                                             
40 :                                         </div>
41 :                                         <div class="theme_detail_info">
42 :                                             <h2 class="theme_detail_title"><%= theme.getMainTitle() %></h2>
```

테마여행정보에 필요한 BusanThemeMgr 객체를 생성합니다.

list.jsp에서 요청한 ucSeq 값을 정수로 변환하여 받습니다.

mgr 객체의 getBusanThemeDetail 메소드를 통해 ucSeq에 해당하는 테마 여행의 상세 정보를 가져옵니다.

theme 객체가 null이 아니면, 즉, ucSeq에 해당하는 테마여행정보가 존재할 경우, 그 상세 정보를 화면에 표시합니다.

```

42 :                                <p class="theme_detail_subtitle">%= theme.getSubtitle() !=
null ? theme.getSubtitle() : "" %</p>
43 :                                <ul class="theme_detail_fields">
44 :                                    <li><span class="theme_detail_label">지역:</span> <%=
theme.getGugunNm() %></li>
45 :                                    <li><span class="theme_detail_label">카테고리:</span> <%=
theme.getCate2Nm() %></li>
46 :                                    <li><span class="theme_detail_label">위치:</span> <%=
theme.getPlace() %></li>
47 :                                    <li><span class="theme_detail_label">주소:</span> <%=
theme.getAddr1() %></li>
48 :                                    <li><span class="theme_detail_label">홈페이지:</span> <a
href="<%= theme.getHomepageUrl() != null ? theme.getHomepageUrl() : "#" %>" target="_blank">%=
theme.getHomepageUrl() != null ? theme.getHomepageUrl() : "홈페이지 없음" %></a></li>
49 :                                    <li><span class="theme_detail_label">교통 정보:</span>
<span class="theme_detail_textarea">%= theme.getTrfcInfo() != null ? theme.getTrfcInfo() : "교통 정
보 없음" %></span></li>
50 :                                    <li><span class="theme_detail_label">상세 설명:</span>
<span class="theme_detail_textarea">%= theme.getItemcntnts() != null ? theme.getItemcntnts() : "상
세 설명 없음" %></span></li>
51 :                                </ul>
52 :                                </div>
53 :                                </div>
54 :                                <div class="action_btn">
55 :                                    <button onclick="window.history.back()">목록으로 돌아가기</button>
56 :                                </div>
57 :                                <%> else {<%>
58 :                                    <p>해당 테마 여행 정보를 찾을 수 없습니다.</p>
59 :                                <%>%>
60 :                                </div>
61 :                                </section>
62 :                                </div>
63 :                                </main>
64 :                                </div>
65 : </body>
66 : </html>

```

목록으로 돌아가기 버튼을 누르면 이전 페이지로 돌아갑니다. JavaScript의 window.history.back()을 사용하여 이 동작을 처리합니다.

조건에 맞는 ucSeq 값에 테마여행정보가 없을 경우 출력되는 결과입니다.

34~56 : 요청된 테마여행정보가 존재할 경우, 해당 테마의 이미지, 제목, 설명, 위치, 홈페이지 등의 세부 정보를 출력합니다.

04 _ 테마여행정보 페이지 실행

앞에서 만든 테마여행정보를 실행하여, 모든 기능을 화면에서 확인해 보겠습니다.

여기서 잠깐!

교재에 표시된 화면은 실제 스마트폰에서 실행한 결과입니다. 만약 PC 브라우저에서 실행하려면 다음과 같은 방법을 따라주세요.



▲ [그림 etc03-16] PC 브라우저에서 list.jsp 실행화면

브라우저에서 F12 키를 누른 후, 오른쪽 상단에 있는 휴대폰 모양 아이콘(Toggle Device Toolbar)을 선택합니다. 그런 다음, 원하는 스마트폰 기종을 선택하면 해당 화면을 미리 볼 수 있습니다.

01 Eclipse에서 list.jsp를 실행하여 브라우저에서 모바일 웹으로 확인합니다.



02 list.jsp에서 스크롤을 내려 페이징 및 블록 처리 영역이 보이도록 합니다.



03 list.jsp에서 페이징 및 블록 처리를 직접 실행해 봅니다.



04 원하는 여행지를 클릭하여 detail.jsp를 실행하고 상세보기를 확인합니다.



이상으로 테마여행정보의 설명을 모두 마치도록 하겠습니다.