

JSP를 이용한 자바 웹 프로그램 만들기

코드 블록 4종류

<%	%>
<% =	%>
<% !	%>
<% @	%>

JSP 프로그래밍

Jasper가 작성할 서블릿 코드에
코드 블록을 적절히 끼워 달라고 지시하는 방식

글 목록 구현하기

- import 및 한글 깨짐 설정
 -

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

- db 설정 코드
 -

```

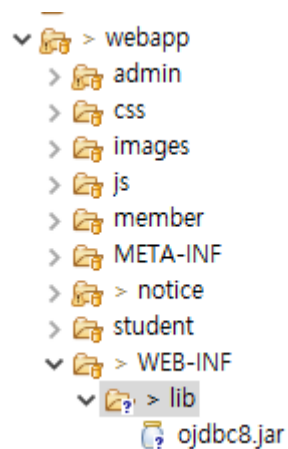
<%
String url = "jdbc:oracle:thin:@localhost:1521/xe";
String sql = "SELECT * FROM NOTICE";

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection(url, "scott", "tiger");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(sql);
%>

```

- ojdbc.jar 파일 wepapp 폴더 안 lib 에 넣기

○



- 데이터 create , insert

○

```

CREATE TABLE NOTICE
(
    ID          NUMBER,
    TITLE       NVARCHAR2(100),
    WRITER_ID   NVARCHAR2(50),
    CONTENT     CLOB,
    REGDATE     TIMESTAMP,
    HIT         NUMBER,
    FILES       NVARCHAR2(1000)
);

INSERT INTO NOTICE (ID, TITLE, WRITER_ID, CONTENT, REGDATE, HIT, FILES)
VALUES
(1, '새로운 공지사항', 'admin', '2024년 1분기 시스템 점검 안내', TIMESTAMP '2024-01-10 09:00:00', 0, NULL);

INSERT INTO NOTICE (ID, TITLE, WRITER_ID, CONTENT, REGDATE, HIT, FILES)
VALUES
(2, '휴무 일정 안내', 'hr_manager', '2024년 설 연휴 휴무 일정 안내입니다.', TIMESTAMP '2024-02-05 08:30:00', 0, NULL);

INSERT INTO NOTICE (ID, TITLE, WRITER_ID, CONTENT, REGDATE, HIT, FILES)
VALUES
(3, '보안 점검 결과', 'security_team', '최근 진행된 보안 점검 결과 및 조치 사항을 공유드립니다.', TIMESTAMP '2024-03-01 15:45:00', 0, NULL);

```

- while 문을 통해 글 목록 출력

```
<% while(rs.next()){ %>

<tr>
<td> <%= rs.getInt("ID")%> </td>
<td class="title indent text-align-left"><a href="detail.html"><%= rs.getString("TITLE") %></a></td>
<td><%=rs.getString("WRITER_ID") %></td>
<td>
<%=rs.getDate("REGDATE") %>
</td>
<td><%=rs.getInt("HIT") %></td>
</tr>
<% } %>
```

상세페이지 출력하기

- 제목을 눌렀을때 해당 페이지로 이동하기 위한 하이퍼링크 코드 수정

○

```
<tr>
<td> <%= rs.getInt("ID")%> </td>
<td class="title indent text-align-left"><a href="detail.jsp?id=<%=rs.getInt("ID") %>><%= rs.getString("TITLE") %>
```

- 상세페이지 내용 출력을 위한 쿼리문 코드 작성
 - getParameter를 통해 id를 넘겨받는다
 - 이는 위에서 하이퍼링크 클릭할때 url에있는 "id=" 에서 가져옴
 - id를 물음표에 꽂아 넣기 위해 PraparedStatement 활용
 - st객체에 SetInt를 통해 id값을 물음표에 넣는다

```
<%

int id = Integer.parseInt(request.getParameter("id"));

String url = "jdbc:oracle:thin:@localhost:1521/x";
String sql = "SELECT * FROM NOTICE WHERE ID=?"; //물음표에다가 값을 넣어줘야 한다.

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection(url, "scott", "tiger");
PreparedStatement st = con.prepareStatement(sql); //prepared는 는 쿼리문을 미리 준비한다
st.setInt(1, id); //첫번째 물음표에다가 id를 꽂아 넣겠다.
ResultSet rs = st.executeQuery();

rs.next();

%>
```


○

```

<div class="margin-top first">
  <h3 class="hidden">공지사항 내용</h3>
  <table class="table">
    <tbody>
      <tr>
        <th>제목</th>
        <td class="text-align-left text-indent text-strong text-orange" colspan="3"><%= rs.getString("TITLE") %></td>
      </tr>
      <tr>
        <th>작성일</th>
        <td class="text-align-left text-indent" colspan="3"><%= rs.getDate("REGDATE") %> </td>
      </tr>
      <tr>
        <th><%= rs.getString("WRITER_ID") %></th>
        <td>newlec</td>
        <th>조회수</th>
        <td><%= rs.getString("HIT") %></td>
      </tr>
      <tr>
        <th>첨부파일</th>
        <td colspan="3"><%= rs.getString("FILES") %></td>
      </tr>
      <tr class="content">
        <td colspan="4"><%= rs.getString("CONTENT") %></td>
      </tr>
    </tbody>
  </table>
</div>

```

- 완성

 공지사항

[home](#) > [고객센터](#) > [공지사항](#)

제목	휴무 일정 안내		
작성일	2024-02-05		
hr_manager	newlec	조회수	0
첨부파일	null		

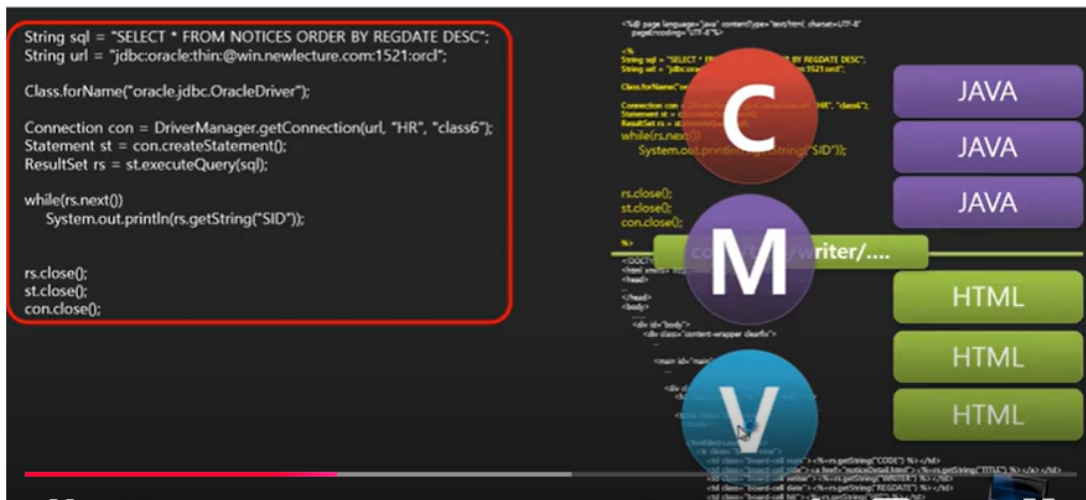
2024년 설 연휴 휴무 일정 안내입니다.

[목록](#)

자세한페이지 MVC1 모델로 변경하기

-

임시 변수를 이용한 코드 분리 : MVC



- 자바 코드를 한 곳에 몰아놓고 close되기 전 변수화시키기

```
<%
    int id = Integer.parseInt(request.getParameter("id")); //여기서 getParameter를 통해
    //가져오는 값은 list 페이지의 제목을 클릭했을때 하이퍼링크 url (detail.jsp?id=) 을 통해 가져오는 것

    String url = "jdbc:oracle:thin:@localhost:1521/xs";
    String sql = "SELECT * FROM NOTICE WHERE ID=?"; //물음표에다가 값을 넣어줘야 한다.

    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection(url, "scott", "tiger");
    PreparedStatement st = con.prepareStatement(sql); //prepared는 는 쿼리문을 미리 준비한다
    st.setInt(1, id); //첫번째 물음표에다가 id를 꽂아 넣겠다.
    ResultSet rs = st.executeQuery();

    rs.next();

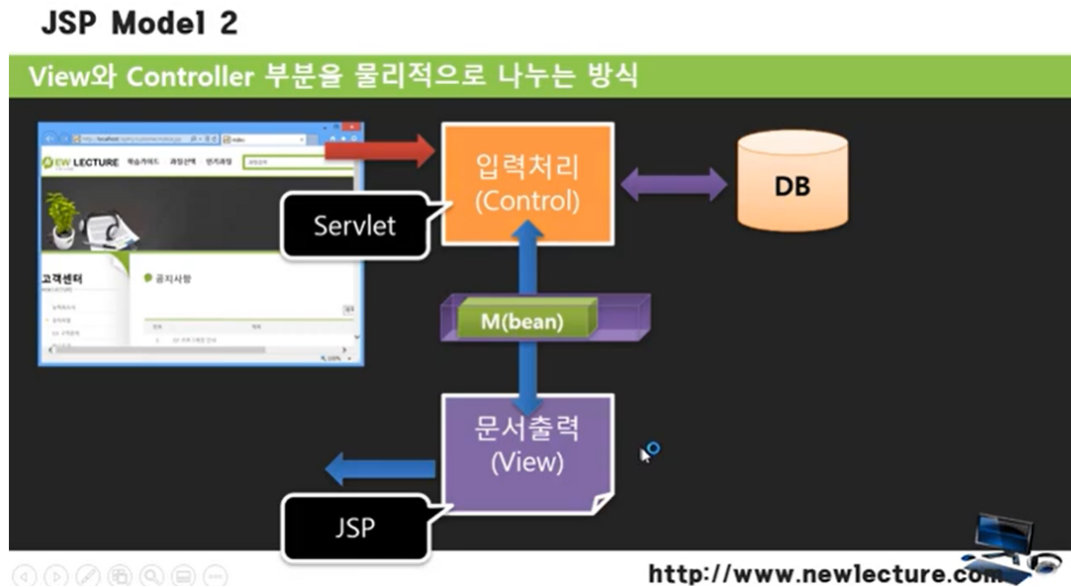
    String title = rs.getString("TITLE");
    Date regdate = rs.getDate("REGDATE");
    String writerId = rs.getString("WRITER_ID");
    String hit = rs.getString("HIT");
    String files = rs.getString("FILES");
    String content = rs.getString("CONTENT");

    rs.close();
    st.close();
    con.close();
%>
```

```
<div class="margin-top first">
    <h3 class="hidden">공지사항 내용</h3>
    <table class="table">
        <tbody>
            <tr>
                <th>제목</th>
                <td class="text-align-left text-indent text-strong text-orange" colspan="3"><%=title %></td>
            </tr>
            <tr>
                <th>작성일</th>
                <td class="text-align-left text-indent" colspan="3"><%=regdate %> </td>
            </tr>
            <tr>
                <th></th>
                <td><%=writerId %></td>
                <th>조회수</th>
                <td><%=hit %></td>
            </tr>
            <tr>
                <th>첨부파일</th>
                <td colspan="3"><%=files %></td>
            </tr>
            <tr class="content">
                <td colspan="4"><%=content %></td>
            </tr>
        </tbody>
    </table>
</div>
```

MVC 2 모델로 변경하기

-



- 나누었을때 두개가 된 SERVLET을 이어줄 수 있는 매개체가 필요하다
- Model에서의 상태 저장 방식이 필요하다. (pageContxt/session/ application) → request 사용이 바람직하다.
- detail.jsp의 자바코드를 별도의 controller 클래스를 만들어 옮긴다
 - NoticeDetailControlle.java

```
package com.newlecture.web.controller;

import java.io.IOException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
```

```

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/notice/detail")
public class NoticeDetailController extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        //가져오는 값은 list 페이지의 제목을 클릭했을때 하이퍼링크 url

        String url = "jdbc:oracle:thin:@localhost:1521/xepdb";
        String sql = "SELECT * FROM NOTICE WHERE ID=?";

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection(url, "scott", "tiger");
            PreparedStatement st = con.prepareStatement(sql);
            st.setInt(1, id); //첫번째 물음표에다가 id를 꽂아
            ResultSet rs = st.executeQuery();

            rs.next();

            String title = rs.getString("TITLE");
            Date regdate = rs.getDate("REGDATE");
            String writerId = rs.getString("WRITER_ID");
            String hit = rs.getString("HIT");
            String files = rs.getString("FILES");
            String content = rs.getString("CONTENT");

            request.setAttribute("title", title);
            request.setAttribute("regdate", regdate);
            request.setAttribute("writerId", writerId);
            request.setAttribute("hit", hit);
            request.setAttribute("files", files);
            request.setAttribute("content", content);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        rs.close();
        st.close();
        con.close();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    //redirect(페이지요청 시 아예 다른페이지로 보내버리는 방식)

    //forward (작업했던 내용을 이어받아 처리하는 방식)
    request
    .getRequestDispatcher("/notice/detail.jsp")
    .forward(request, response); //notice안에있는 detail.
    //현재 사용하고 있는 저장소 객체와 출력도구를 같이 공유함

}
}

```

- 이에 따라 detail.jsp 는 다음과 같이 수정한다

○

```

<div class="margin-top first">
    <h3 class="hidden">공지사항 내용</h3>
    <table class="table">
        <tbody>
            <tr>
                <th>제목</th>
                <td class="text-align-left text-indent text-strong text-orange" colspan="3"><%=request.getAttribute("title") %></td>
            </tr>
            <tr>
                <th>작성일</th>
                <td class="text-align-left text-indent" colspan="3"><%=request.getAttribute("regdate") %> </td>
            </tr>
            <tr>
                <th></th>
                <td><%=request.getAttribute("writerId") %></td>
                <th>조회수</th>
                <td><%=request.getAttribute("hit") %></td>
            </tr>
            <tr>
                <th>첨부파일</th>
                <td colspan="3"><%=request.getAttribute("files") %></td>
            </tr>
            <tr class="content">
                <td colspan="4"><%=request.getAttribute("content") %></td>
            </tr>
        </tbody>
    </table>
</div>

```


- mvc2 model로 변경했기 때문에 이제는 글 목록에서 상세페이지로 갈 때 detail.jsp로 가는 것이 아닌 위에서 만든 detail(Controller)로 url 맵핑 해야한다.

○

```
<tr>
  <td> <%= rs.getInt("ID")%> </td>
  <td class="title indent text-align-left"><a href="detail?id=<%=rs.getInt("ID") %>"><%= rs.getString("TITLE") %></a></td>
  <td><%=rs.getString("WRITER_ID") %></td>
  <td>
    <%=rs.getDate("REGDATE") %>
  </td>
  <td><%=rs.getInt("HIT") %></td>
</tr>
```

Model 데이터를 구조화하기

- 데이터를 속성이라는 이름으로 대체하고, 묶어서 표현해야한다.

Model 데이터를 위한 구조화의 필요성

구조화된 데이터를 이용하면?

```
request.setAttribute("title", rs.getString("title"));
request.setAttribute("writerId", rs.getString("writerId"));
request.setAttribute("regdate", rs.getDate("regdate"));
request.setAttribute("content", rs.getString("content"));
request.setAttribute("hit", rs.getInt("hit"));
```



```
request.setAttribute("notice", notice);
```

- notice 객체안에 있는 속성값을 꺼내기 위해서?

Model 데이터를 위한 구조화의 필요성

모델 데이터를 위한 클래스 정의와 사용방법의 변화

```
public class Notice {
    private int id;
    private String title;
    private String writer;
    private Date regdate;
    private String content;
    private int hit;

    public int getId() {
        return id;
    }
    ...
}
```

- getter를 호출해서 가져오는 방식이다.
- Entity - Notice.java
 -

```
package com.newlecture.web.entity;

import java.sql.Date;

public class Notice {
    private int id;
    private String title;
    private Date regdate;
    private String writerId;
    private String hit;
    private String files;
    private String content;

    public Notice() {
    }

    public Notice(int id, String title, Date regdate, Stri

        this.id = id;
        this.title = title;
        this.regdate = regdate;
```

```
        this.writerId = writerId;
        this.hit = hit;
        this.files = files;
        this.content = content;
    }

    public int getId() {
        return id;
    }

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

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public Date getRegdate() {
        return regdate;
    }

    public void setRegdate(Date regdate) {
        this.regdate = regdate;
    }

    public String getWriterId() {
        return writerId;
    }

    public void setWriterId(String writerId) {
        this.writerId = writerId;
    }
}
```

```

public String getHit() {
    return hit;
}

public void setHit(String hit) {
    this.hit = hit;
}

public String getFiles() {
    return files;
}

public void setFiles(String files) {
    this.files = files;
}

public String getContent() {
    return content;
}

public void setContent(String content) {
    this.content = content;
}

@Override
public String toString() {
    return "Notice [id=" + id + ", title=" + title + "
        + hit + ", files=" + files + ", content="
    }
}

```

- notice 엔티티 생성, 이를 Controller에서 객체 생성 후 setAttribute한다.

```

Notice notice = new Notice(
    id,
    title,
    regdate,
    writerId,
    hit,
    files,
    content
);

request.setAttribute("n", notice);

```

- 이를 detail.jsp파일에서 EL 표기법을 활용해 꺼내온다

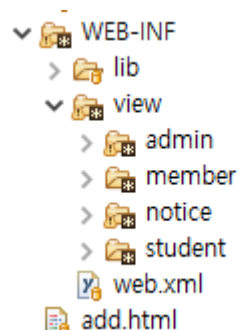
```

<div class="margin-top first">
  <h3 class="hidden">공지사항 내용</h3>
  <table class="table">
    <tbody>
      <tr>
        <th>제목</th>
        <td class="text-align-left text-indent text-strong text-orange" colspan="3">${n.title}</td>
      </tr>
      <tr>
        <th>작성일</th>
        <td class="text-align-left text-indent" colspan="3">${n.regdate}</td>
      </tr>
      <tr>
        <th></th>
        <td>${n.writerId}</td>
        <th>조회수</th>
        <td>${n.hit}</td>
      </tr>
      <tr>
        <th>첨부파일</th>
        <td colspan="3">${n.files}</td>
      </tr>
      <tr class="content">
        <td colspan="4">${n.content}</td>
      </tr>
    </tbody>
  </table>
</div>

```

view 페이지 은닉하기

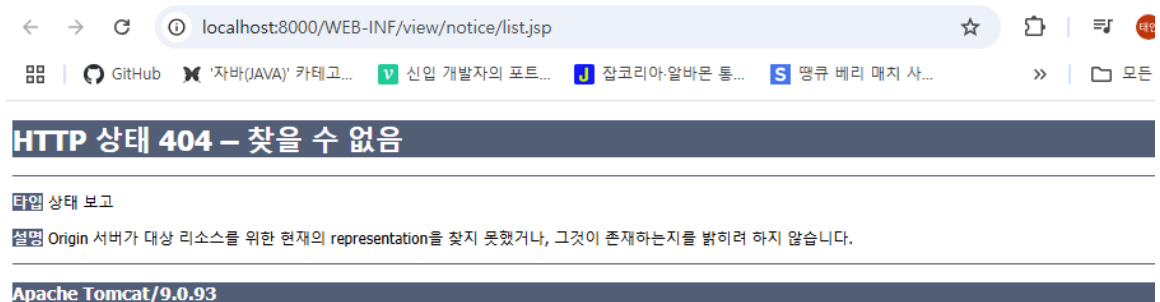
- 외부에서 요청할 수 없도록 jsp페이지를 은닉한다
-



-

```
request.setAttribute("list", list);  
//forward (작업했던 내용을 이어받아 처리하는 방식)  
request  
.getRequestDispatcher("/WEB-INF/view/notice/list.jsp")  
.forward(request, response);  
//notice안에있는 detail.jsp를 요청하면서  
//현재 사용하고 있는 저장소 객체와 출력도구를 같이 공유함
```

- 이로 인해 controller를 통해서만 jsp 페이지로 들어갈 수있다. 외부에서 .jsp 요청 시



view(list.jsp)에서 반복문 제거하기

-

View에서 사용하는 제어구조

자바의 반복문을 이용한 제어구조에서 태그를 이용한 제어구조로

```
<%for(Notice n : list) {%>
    <tr>
        <td>
        <td>
            <a href="noticeDetail.jsp?c=${n.code}">${n.title}</a>
        </td>
        <td>
        <td>
        <td>
    </tr>
<%} %>

<forEach>
    <tr class="board-row">
        <td class="board-cell num">${n.code}</td>
        <td class="board-cell title">
            <a href="noticeDetail.jsp?c=${n.code}">${n.title}</a>
        </td>
        <td class="board-cell writer">${n.writer}</td>
        <td class="board-cell date">${n.regdate}</td>
        <td class="board-cell hit">${n.hit}</td>
    </tr>
</forEach>
```

- jstl 다운받기

← → ↻ mvnrepository.com/artifact/javax.servlet/jstl/1.2

GitHub '자바(JAVA)' 카테고리... 신입 개발자의 포트... 잡코리아알바몬 통... 땡큐 베리 매치 사... >> 모든

MVN REPOSITORY

Search for groups, artifacts, categories Search Categories Popular Contact

Indexed Artifacts (44.9M)

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JVM Languages
- JSON Libraries
- Language Runtime
- Core Utilities

Home » javax.servlet » jstl » 1.2

JSTL » 1.2

JSTL

License	CDDL GPL 2.0
Categories	Java Specifications
Tags	standard servlet javax jstl specs
Date	Jun 23, 2011
Files	pom (356 bytes) jar (404 KB) View All
Repositories	Central GlareMasters Pub Gradle Plugins Less is More OneBusAway Pub Softmotions WSO2 Dist
Ranking	#248 in MvnRepository (See Top Artifacts) #17 in Java Specifications
Used By	2,181 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

- 태그 추가하기

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

- forEach문에서 item 속성을 통해 저장소에 담겨있는 것을 뽑아서 items에 담아서 꺼내기 위해 var 속성으로 꺼낸다 .

○

```
<%--
<%
List<Notice> list = (List<Notice>)request.getAttribute("list");
for(Notice n : list){
    pageContext.setAttribute("n", n);
}%> --%>

<c:forEach var="n" items="${list}">
<tr>
<td>${n.id}</td>
<td class="title indent text-align-left"><a href="detail?id=${n.id}">${n.title}</a></td>
<td>${n.writerId}</td>
<td>${n.regdate}</td>
<td>${n.hit}</td>
</tr>
</c:forEach>

<%-- <%! %> --%>
```

JSTL (JSP Standard Tag Library)

- view단에서 사용할 수 있는 태그들을 제어하고 있다
- JSTL Core

JSTL Core

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

<c:out >	<c:set >
<c:remove >	<c:if>
<c:choose>	<c:when>
<c:otherwise >	<c:import>
<c:forEach >	<c:forTokens>
<c:param>	<c:redirect >
<c:url>	<c:catch>

•

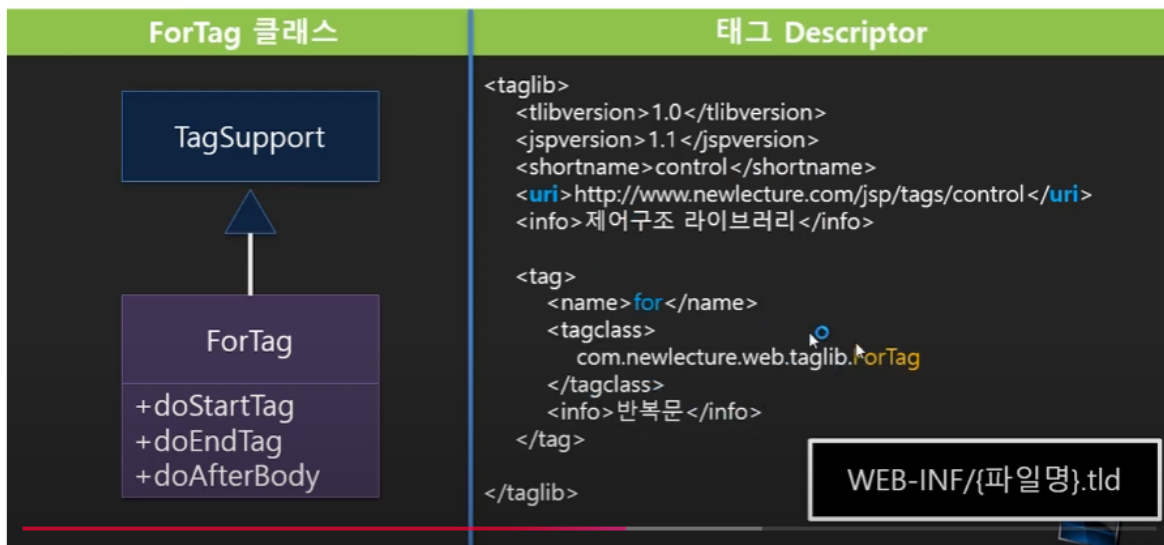
<ForEach /> 태그를 사용하여 공지사항 목록 작성하기

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:forEach var="n" items="${list}">
  <tr class="board-row">
    <td class="board-cell num">${n.code}</td>
    <td class="board-cell title">
      <a href="noticeDetail.jsp?c=${n.code}">${n.title}</a></td>
    <td class="board-cell writer">${n.writer}</td>
    <td class="board-cell date">${n.regdate}</td>
    <td class="board-cell hit">${n.hit}</td>
  </tr>
</c:forEach>
```

- Jspser에게 서버에서 처리할 taglib임을 알 수 있게 함
-

가장 간단한 for 태그 라이브러리 만들기 위한 두 가지 코드



ForTag 클래스 생성

TagSupport 클래스를 확장한 ForTag 클래스

```
public class ForTag extends TagSupport {

    @Override
    public int doStartTag() throws JspException {
        return EVAL_BODY_INCLUDE;
    }

    @Override
    public int doEndTag() throws JspException {
        return EVAL_PAGE;
    }

    @Override
    public int doAfterBody() throws JspException {
        return EVAL_BODY_AGAIN;
    }
}
```

Diagram illustrating the ForTag class and its interaction with the JSP tag:

- doStartTag()** returns **EVAL_BODY_INCLUDE**, which corresponds to the **include** action in the JSP tag.
- doEndTag()** returns **EVAL_PAGE**, which corresponds to the **loop** action in the JSP tag.
- doAfterBody()** returns **EVAL_BODY_AGAIN**, which corresponds to the **include** action in the JSP tag.

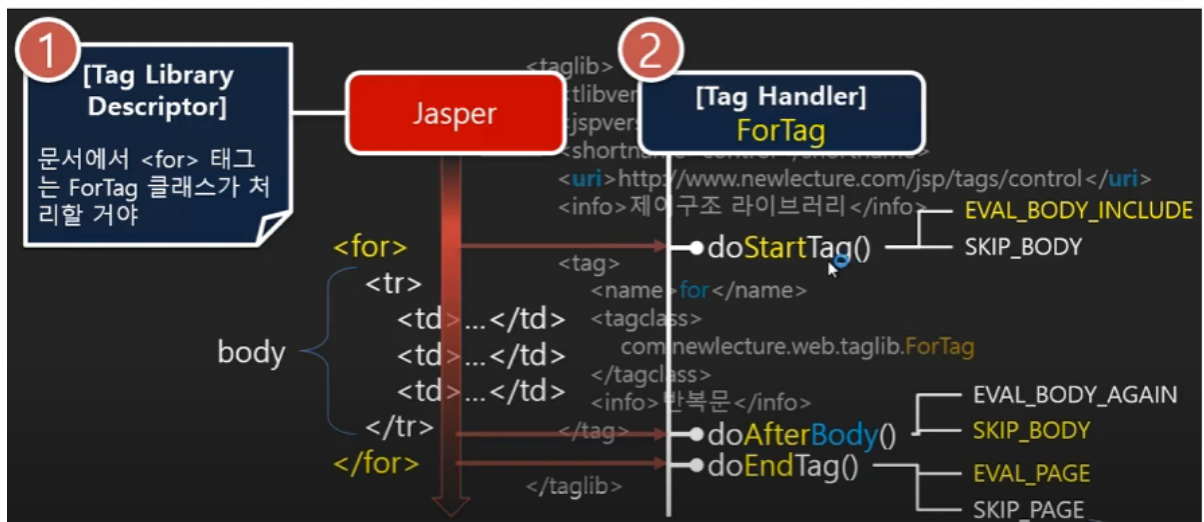
The JSP tag structure is as follows:

```
<for>
  <tr>
    <td>...</td>
    <td>...</td>
    <td>...</td>
  </tr>
</for>
</table>
</div>
```

Java Resources:

- src
 - com.newlecture.jsp
 - com.newlecture.jsp.taglib
 - ForTag.java

태그 라이브러리 만들어보기



중간 정리

- 서블릿
 - 자바 웹(서버) 프로그램
 - 입/출력 :
 - request (입력설정 / 입력값 읽기)

- response (출력 설정 / 출력 스트림)
 - 웹문서 출력 → 문서 기반(jsp)의 코드블록
 - 코드 블록이 문서 내에 너무 많아지자, 유지관리가 힘들어진다.. (스파게티 코드)
 - → MVC (코드 블록을 모으고, 출력 부분 분리) → View에서 코드블록이 꼭 필요한 경우 → EL, JSTL(제어 담당)