プログラミング Java 3

第5回:JSP入門。ユーザー入力およびユーザーに出力処理入門

１．JSP とは

JSP は Java Server Pages の略であり，Perl や C で作られる CGI や VBScript で作られる ASP(Active Server Pages)と機能的には似た動作をするものである。 HTML のフォームに入力したパラメータを解析して，DB 検索などのロジックを実行して，結果を表示することが典型的な動きである。

JSP ファイルはサーバーサイドでサーブレットソースに翻訳され，翻訳結果のサーブレットソースが更に Java クラスにコンパイルされて実行にかけられる。主にサーブレットの表示機能の代替として使われることが多く，本格的なコントロールはサーブレット，ロジックはビーンなどの Java クラスで記述する。

２.JSP 文法

コメント
　形式　<%-- ... --%>

Declaration(宣言)
　形式 <%! ... %>　例 <%! int i = 0; %>

Expression(式を評価して表示)
　形式 <%= ... %> 例 <%= SQLStatement %>

Scriptlet(Java コード)
　形式 <% ... %>　例 <% String ConnStr = "jdbc:oracle:oci8:@ORADB"; %>

Page Directive
　形式
　<%@ page
　　[language="java"]
　　[extends="package .class"]
　　[import="{package .class | package.*}, ..."]
　　[session="true|false"]
　　[buffer="none|8kb|sizekb"]
　　[autoFlush="true|false"]
　　[isThreadSafe="true|false"]
　　[info="text"]
　　[errorPage="relativeURL"]
　　[contentType="mimeType [;charset=characterSet]" | "text/html ; charset=ISO-8859-1"]
　　[isErrorPage="true|false"]
　%>

例：<%@ page import = "java.sql.*" %>

Include Directive（他の JSP を読み込んでからコンパイルする）
形式 <%@ include file="relativeURL" %> 例 <%@ include file = "logo.htm" %>

jsp:forward(クライアントのリクエストを、HTML、JSP、servlet に送る)
形式 <jsp:forward page="{relativeURL | <%= expression %>}" />
　　or
　<jsp:forward page="{relativeURL | <%= expression %>}" >
　<jsp:param name="parameterName"
　　value="{parameterValue | <%= expression %>}" />
　</jsp:forward>

jsp:include(他の JSP の実行結果を読み込む)
形式　<jsp:include page="{relativeURL | <%= expression %>}" flush="true" />
　　or
　<jsp:include page="{relativeURL | <%= expression %>}" flush="true" >
　<jsp:param name="parameterName"
　　value="{parameterValue | <%= expression %>}" />+
　</jsp:include>

jsp:plugin(applet またはビーンを実行する)
形式　<jsp:plugin
　　type="bean|applet"
　　code="classFileName"
　　codebase="classFileDirectoryName"
　　[name="instanceName"]
　　[archive="URIToArchive, ..."]
　　[align="bottom|top|middle|left|right"]
　　[height="displayPixels"]
　　[width="displayPixels"]

```
    [ hspace="leftRightPixels" ]
    [ vspace="topBottomPixels" ]
    [ jreversion="JREVersionNumber | 1.1" ]
    [ nspluginurl="URLToPlugin" ]
    [ iepluginurl="URLToPlugin" ] >
    [ <jsp:params>
    [ <jsp:param name="parameterName"
         value="{parameterValue | <%= expression %>}" /> ]+
      </jsp:params> ]
    [ <jsp:fallback> text message for user </jsp:fallback> ]
   </jsp:plugin>
```

jsp:useBean(ビーンのインスタンスを作成, または検索する)
形式

```
<jsp:useBean
  id="beanInstanceName"
  scope="page|request|session|application"
 { class="package.class" |
  type="package.class" |
  class="package.class" type="package.class" |
  beanName="{package.class | <%= expression %>}" type="package.class"
 }
 { /> |
   > other elements
   </jsp:useBean>
 }
```

jsp:getProperty(ビーンの getXXX メソッドを呼び出す)
形式

```
<jsp:getProperty name="beanInstanceName" property="propertyName" />
```

```
<jsp:useBean id="calendar" scope="page" class="employee.Calendar" />
<h2>
  Calendar of <jsp:getProperty name="calendar" property="username" />
</h2>
```

は, <% calendar.getUsername() %> と同じ。

jsp:setProperty(ビーンの setXXX メソッドを呼び出す)
特に prpperty="*" で, request の全パラメータをビーンにセットは便利。
形式

```
<jsp:setProperty
   name="beanInstanceName"
  { property="*" |
   property="propertyName" [ param="parameterName" ] |
   property="propertyName" value="{string | <%= expression %>}"
  }
/>
```

```
<jsp:setProperty name="mybean" property="*" />
<jsp:setProperty name="mybean" property="username" />
<jsp:setProperty name="mybean" property="username" value="Steve" />
```

最下段は, <% calendar.setUsername("Steve") %> と同じ。

予約語

JSP では、スクリプトレットや式内の暗黙的オブジェクトのための予約語が用意されている。
この暗黙的オブジェクトとは、JSP ページに有用なメソッドや情報を提供する Java
オブジェクトである。これらの暗黙的オブジェクトは、 スクリプトレットや 式の内部で
のみ宣言なしに使用できる。

request

request は、javax.servlet.HttpServletRequest オブジェクトである。
ブラウザからの要求についての情報が含まれ、クッキー、ヘッダ、およびセッション
データを取得するために便利なメソッドも用意されている。

```
<%
  String firstName = request.getParameter("fName");
  String lastName = request.getParameter("lName");
  out.println("Welcome " + firstName + " " + lastName);
%>
```

response

response は、javax.servlet.HttpServletResponse オブジェクトである。
JSP ページからブラウザに返送される応答の設定に便利な複数のメソッドを持つ。
この応答の例として、クッキーとその他のヘッダ情報がある。
response.getWriter() メソッドを JSP ページ内で使用することはできない。
out を利用する。

out

out は、javax.jsp.JspWriter のインスタンスで、ブラウザへの出力の返送に使える
メソッドを持っている。

出力ストリームを必要とするメソッドを使用している場合、JspWriter は動作しない。
この制限を回避するには、バッファ化ストリームを提供して、このストリームを out
に記述する。

pageContext

pageContext は、javax.servlet.jsp.PageContext オブジェクトである。
さまざまなスコープのネームスペースやサーブレット関連オブジェクトにアクセスする
ために便利な API で、一般的なサーブレット関連機能のためのラッパー メソッドを
提供する。

session

session は、要求に対する javax.servlet.http.HttpSession オブジェクト。
セッション ディレクティブは、デフォルトでは true に設定されている。
したがって、session はデフォルトで有効である。
JSP 1.1 仕様では、セッション ディレクティブが false に設定されている場合、
session キーワードを使用すると変換時に致命的なエラーが発生することが説明されている。
下の例では，session に String の firstName を登録し，取り出している。

```
<%
  session.setAttribute("fName", firstName);
  session.setAttribute("lName", lastName);
  String firstName = (String) session.getAttribute("fName");
  String lastName = (String) session.getAttribute("lName");
  out.println("Welcome " + firstName + " " + lastName);
%>
```

application

application は、javax.servlet.ServletContext オブジェクトを表す。
サーブレット エンジンやサーブレット環境に関する情報の検索に使用する。
リクエストを転送または取り込む場合は、ServletContext を使用してサーブレット
requestDispatcher にアクセスできる。
または、他のサーブレットへのリクエストの転送に JSP forward ディレクティブ、
他のサーブレットからの出力の取り込みに JSP include ディレクティブを使用すること
もできる。

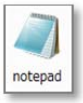
config

config は javax.servlet.ServletConfig オブジェクトを表し、サーブレット インスタンスの
初期化パラメータへのアクセスを提供する。

exception

java.lang.Throwable のインスタンス。
page ディレクティブの isErrorPage が "true"の JSP、つまりエラーページでのみ有効。
JSP で発生した例外にアクセスするためのオブジェクト。

3.実習

3.1  ＪＳＰの内容を入力　（Notepad, Notepad++, Eclipse 等）

JSP のファイル名：E:¥Temp¥Lecture_5_page.jsp

JSP の内容：

```
<HTML>
  <HEAD>
  <TITLE> Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html; charset=SHIFT_JIS" %>
  </HEAD>
  <BODY>
    <%@ include file  = "logo.htm" %>
    <% // Begin of Java code (scriptlet) in JSP
      String userName = request.getParameter("InputBox");
      if (userName != null) { // i.e., not the very first activation of JSP
        // Output via Java command out.println:
        out.println("Output to the user using Java command <B>out.println</B>:<BR>");
        out.println("User name: "+ userName);
        %> <!-- End of Java code (scriptlet) in JSP -->

        <BR><BR> <!-- Two breaks (new lines) -->

        <!--  Static HTML text:  -->

        Output to the user using <B>JSP expression:</B><BR>
        User name: <%= userName %>
    <% } %> <!-- End of the body of if statement -->
    <BR><BR>
    <FORM METHOD="GET" ACTION="Lecture_5_page.jsp" >
      Enter your name:<BR>
      <INPUT NAME = "InputBox">
      <INPUT TYPE = "Submit" VALUE="Submit"/>
    </FORM>
  </BODY>
</HTML>
```

図１．JSP Lecture_5_page.jsp の内容。ロケーション：E:¥TEMP¥

3.2.  Upload: FFFTP Client Setup

ホスト名：　isd-si.doshisha.ac.jp

ユーザ名：　guest

パスワード：　guest

Web-server のフォルダー（Upload の時）：/ 1XXXXXXX/

1XXXXXXX は学生番号（学生 ID）です。

3.3.  or  Web アクセス

JSP の URL (Web Browser のアドレス・バー)：

http://isd-si.doshisha.ac.jp/teaching/programming_3/1XXXXXXX/Lecture_5_page.jsp

1XXXXXXX は学生番号（学生 ID）です。

図 2. JSP Lecture_5_page.jsp の最初のアクセス

URL : http://isd-si.doshisha.ac.jp/teaching/programming_3/1XXXXXXX /Lecture_5_page.jsp

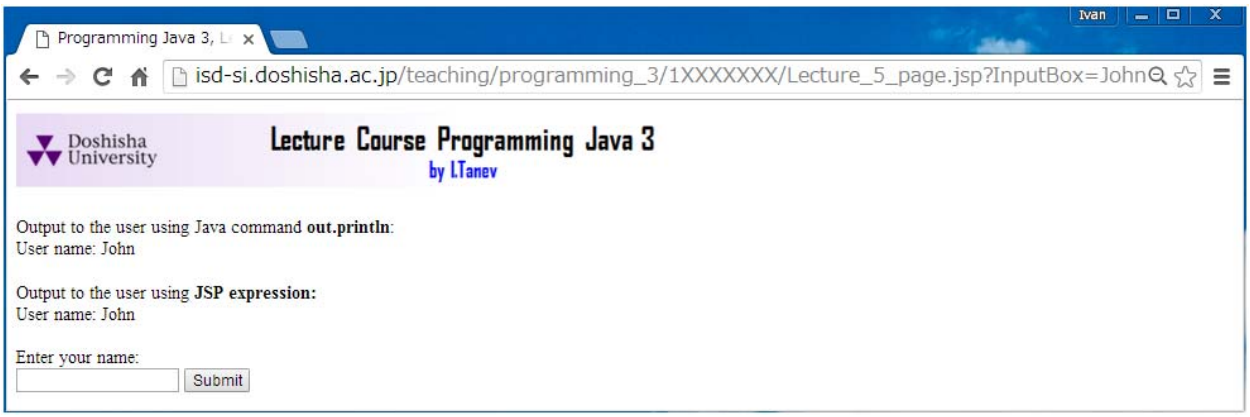


図 3. JSP Lecture_5_page.jsp の「John」を入力し、「Submit」ボタンの押してからの結果

URL : http://isd-si.doshisha.ac.jp/teaching/programming_3/1XXXXXXX/Lecture_5_page.jsp

3.4. JSP の自動パーズ(java)とコンパイル(class)の結果を確認する

a) FFFTP Client Setup

ホスト名： isd-si.doshisha.ac.jp

ユーザ名： pages

パスワード： pages

b) FFFTP Client が Server に接続してから、学籍番号のフォルダを開く

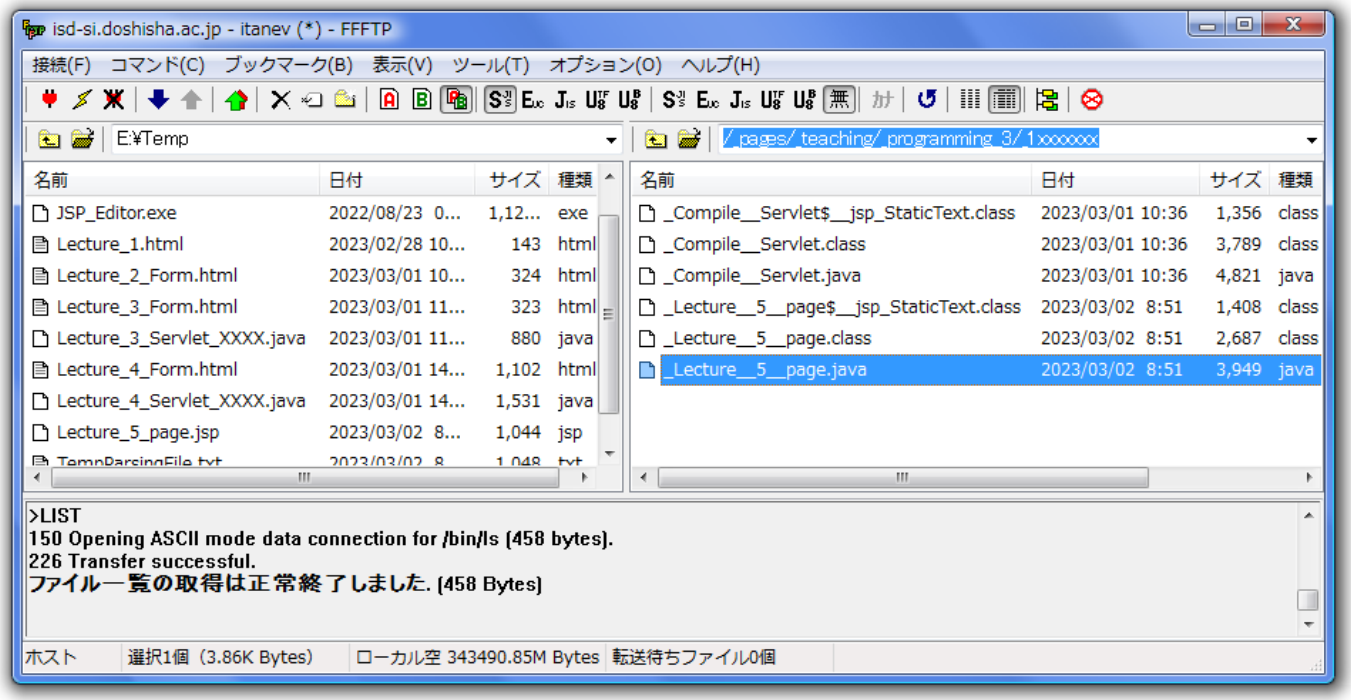


図 4. 自動的にパーズとコンパイルされた JSP Lecture_5_page.jsp（図 1）
「_Lecture__5__page.java」 と「_Lecture__5__page.class」のロケーション.

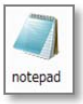
c) 

図 4 に示した java ファイルを開くと確認（図 5）。

```
package _teaching._programming__3._1xxxxxxx;
import oracle.jsp.runtime.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.io.*;
import java.util.*;
import java.lang.reflect.*;
import java.beans.*;

public class _Lecture__5__page extends oracle.jsp.runtime.HttpJsp {
    public final String _globalsClassName = null;

    // ** Begin Declarations
    // ** End Declarations

    public void _jspService(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        response.setContentType( "text/html");
        /* set up the intrinsic variables using the pageContext goober:
        ** session = HttpSession
        ** application = ServletContext
        ** out = JspWriter
        ** page = this
        ** config = ServletConfig
        ** all session/app beans declared in globals.jsa
        */
        JspFactory factory = JspFactory.getDefaultFactory();
        PageContext pageContext = factory.getPageContext( this, request, response, null, true, JspWriter.DEFAULT_BUFFER, true);
        // Note: this is not emitted if the session directive == false
        HttpSession session = pageContext.getSession();
        if (pageContext.getAttribute(OracleJspRuntime.JSP_REQUEST_REDIRECTED, PageContext.REQUEST_SCOPE) != null) {
            pageContext.setAttribute(OracleJspRuntime.JSP_PAGE_DONTNOTIFY, "true", PageContext.PAGE_SCOPE);
            factory.releasePageContext(pageContext);
            return;
}
        ServletContext application = pageContext.getServletContext();
        JspWriter out = pageContext.getOut();
        _Lecture__5__page page = this;
        ServletConfig config = pageContext.getServletConfig();
        try {
            // global beans
            // end global beans
            out.print(__jsp_StaticText.text[0]);
            out.print(__jsp_StaticText.text[1]);
            out.print(__jsp_StaticText.text[2]);
            out.print(__jsp_StaticText.text[3]);
            // Begin of Java code (scriptlet) in JSP
                    String userName = request.getParameter("InputBox");
                    if (userName != null) { // i.e., not the very first activation of JSP
                        // Output via Java command out.println:
                        out.println("Output to the user using Java command <B>out.println</B>:<BR>");
                        out.println("User name: "+ userName);
            out.print(__jsp_StaticText.text[4]);
            out.print( userName );
            out.print(__jsp_StaticText.text[5]);
            }
            out.print(__jsp_StaticText.text[6]);
            out.flush();
        }
        catch( Exception e) {
            try {
                if (out != null) out.clear();
            }
            catch( Exception clearException) {
            }
            pageContext.handlePageException( e);
        }
        finally {
            if (out != null) out.close();
            factory.releasePageContext(pageContext);
        }
    }
    private static class __jsp_StaticText {
        private static final char text[][]=new char[7][];
        static {
            text[0] = "<HTML>¥r¥n    <HEAD>¥r¥n    <TITLE> Programming Java 3, Lecture 5</TITLE>¥r¥n    ".toCharArray();
            text[1] = "¥r¥n    </HEAD>¥r¥n    <BODY>¥r¥n        ".toCharArray();
            text[2] = "<IMG SRC=¥"Logo_PJ3.png¥" ALIGN=TOP >¥r¥n<BR><BR>".toCharArray();
            text[3] = "¥r¥n        ".toCharArray();
            text[4] =" <!-- End of Java code (scriptlet) in JSP -->¥r¥n¥r¥n                <BR><BR> <!-- Two breaks (new lines) -->¥r¥n¥r¥n                <!--  Static HTML text:
-->¥r¥n¥r¥n                Output to the user using <B>JSP expression:</B><BR>¥r¥n                User name: ".toCharArray();
            text[5] = "¥r¥n        ".toCharArray();
            text[6] = " <!-- End of the body of if statement -->¥r¥n        <BR><BR>¥r¥n        <FORM METHOD=¥"GET¥" ACTION=¥"Lecture_5_page.jsp¥" >¥r¥n            Enter your
name:<BR>¥r¥n                    <INPUT    NAME    =    ¥"InputBox¥">¥r¥n                    <INPUT    TYPE    =    ¥"Submit¥"    VALUE=¥"Submit¥"/>¥r¥n                </FORM>¥r¥n
</BODY>¥r¥n</HTML>".toCharArray();
        }
    }
}
```

図 5. コンパイルされた JSP Lecture_5_page.jsp（図 1）の結果:ファイル「_Lecture__5__page.java」