プログラミング Java 3

第5回:

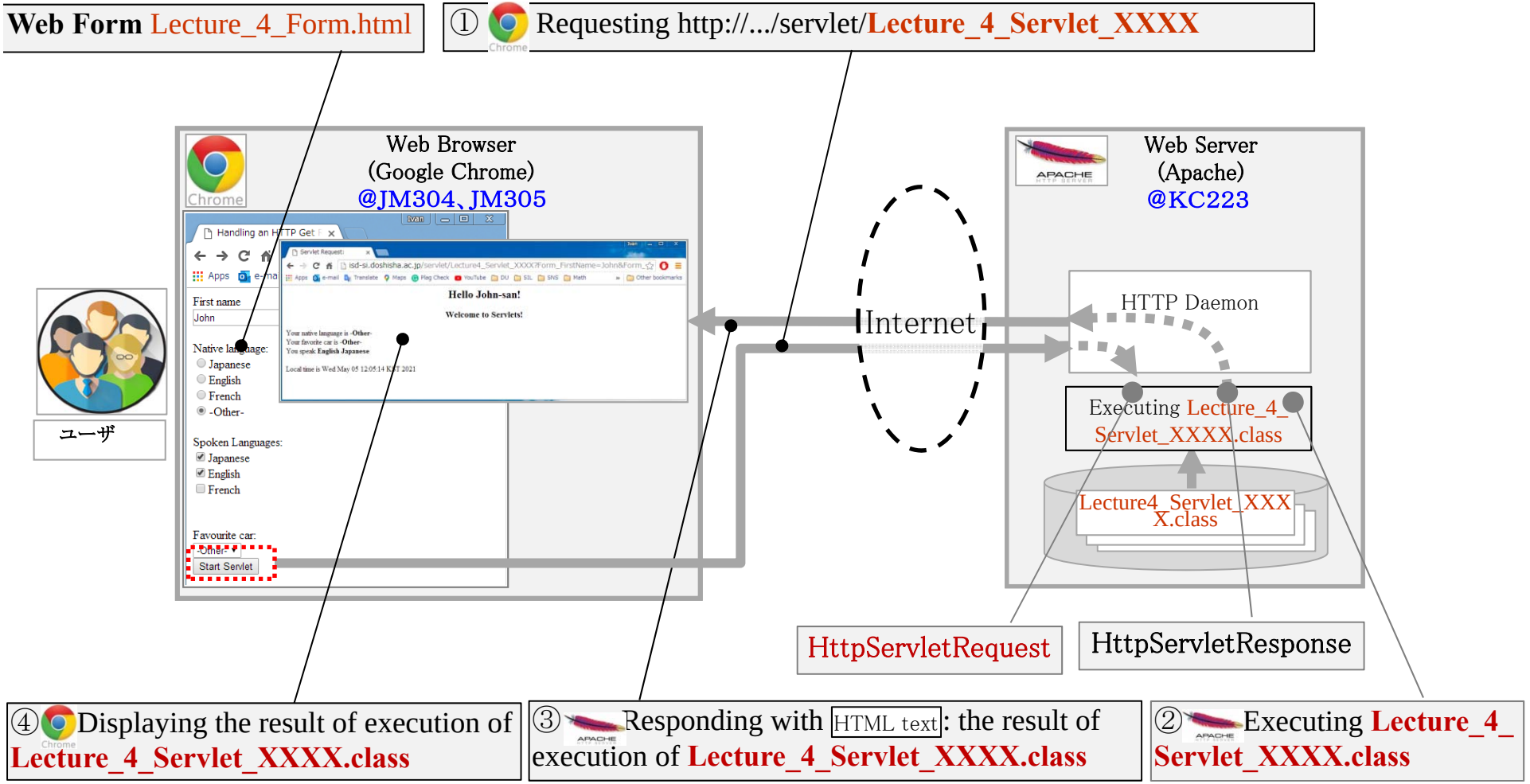
JSP入門.
ユーザ入力およびユーザへの出力の処理

Ivan Tanev

アウトライン

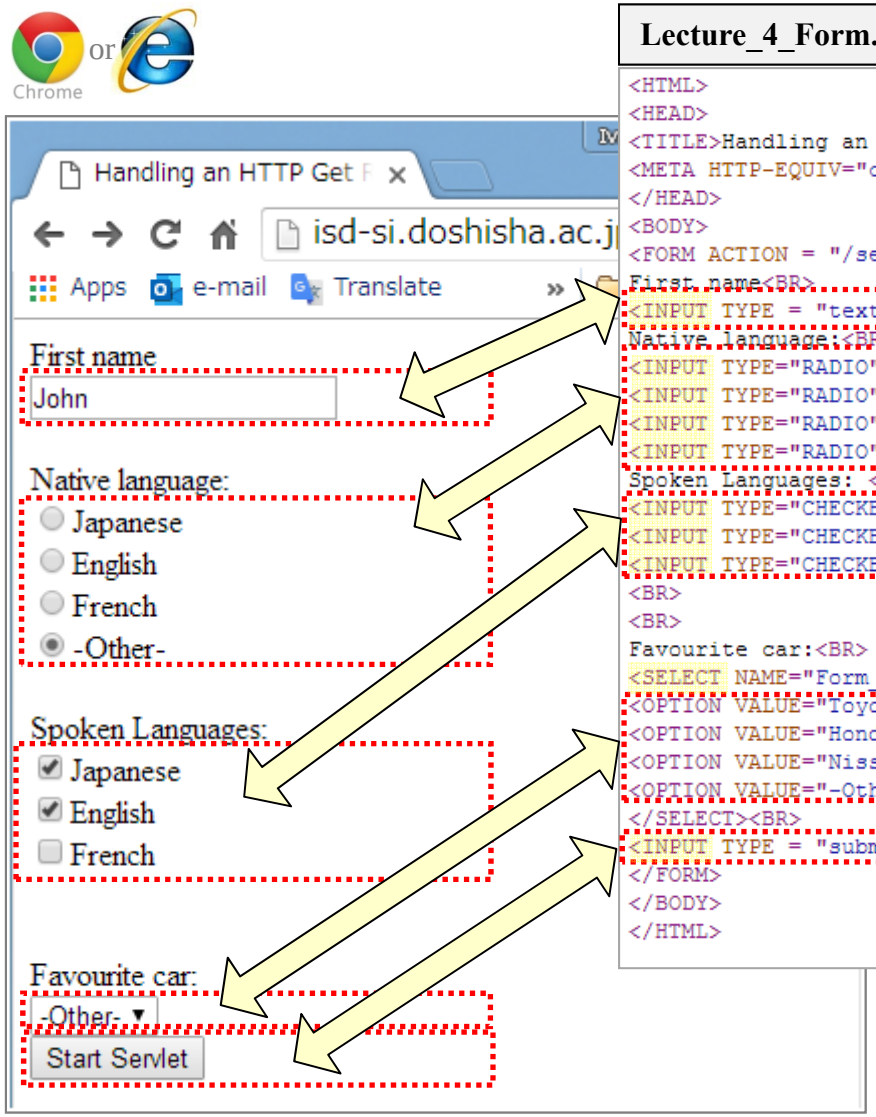
1. 第4回のまとめ
2. JSP入門
3. ユーザ入力(User Input)および
   ユーザに出力(User Output)の処理
4. JSP Directives
5. まとめ
6. 演習

1. 第4回のまとめ


Web Form Lecture_4_Form.html
① Requesting http://.../servlet/Lecture_4_Servlet_XXXX
Web Browser
(Google Chrome)
@JM304、JM305
Web Server
(Apache)
@KC223
Internet
HTTP Daemon
Executing Lecture_4_Servlet_XXXX.class
Lecture4_Servlet_XXXX.class
ユーザ
HttpServletRequest
HttpServletResponse
④ Displaying the result of execution of Lecture_4_Servlet_XXXX.class
③ Responding with HTML text: the result of execution of Lecture_4_Servlet_XXXX.class
② Executing Lecture_4_Servlet_XXXX.class

1. 第4回のまとめ

User's inputコンポネント @ Web Form:
<INPUT>及び<SELECT>HTMLタグ



Lecture_4_Form.htmlのソース

```
<HTML>
<HEAD>
<TITLE>Handling an HTTP Get Request</TITLE>
<META HTTP-EQUIV="content-type" CONTENT="text/html">
</HEAD>
<BODY>
<FORM ACTION = "/servlet/Lecture4_Servlet_XXXX" METHOD = "get">
First name<BR>
<INPUT TYPE = "text" NAME = "Form_FirstName" /><BR><BR>
Native language:<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="Japanese" CHECKED="CHECKED">Japanese<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="English">English<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="French">French<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="-Other-">-Other-<BR><BR>
Spoken Languages: <BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakJapanese" CHECKED="CHECKED">Japanese<BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakEnglish">English<BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakFrench">French<BR>
<BR>
<BR>
Favourite car:<BR>
<SELECT NAME="Form_Cars">
<OPTION VALUE="Toyota">Toyota
<OPTION VALUE="Honda">Honda
<OPTION VALUE="Nissan">Nissan
<OPTION VALUE="-Other-">-Other-
</SELECT><BR>
<INPUT TYPE = "submit" VALUE = "Start Servlet"/>
</FORM>
</BODY>
</HTML>
```

1. 第4回のまとめ

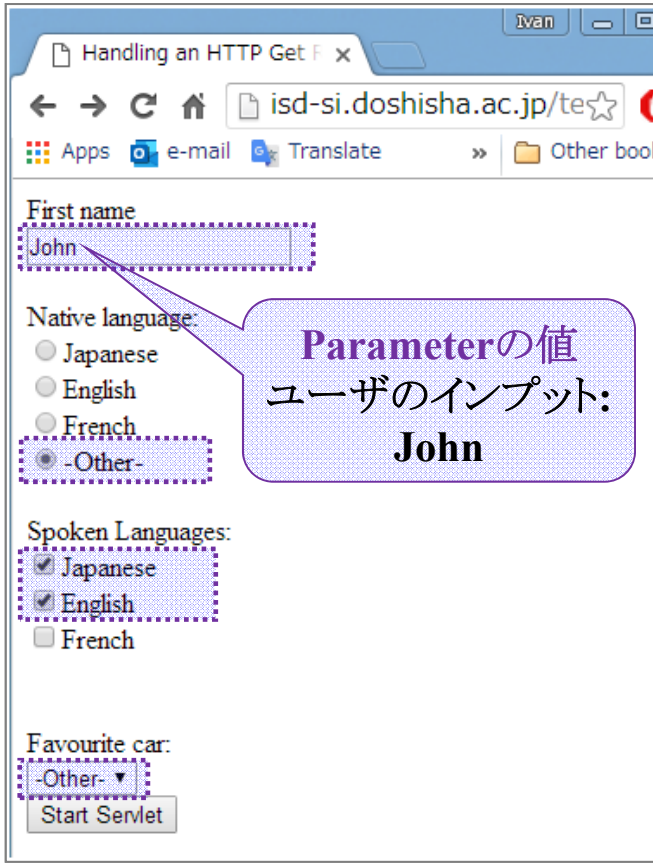
HTTP Requestの構造

Web form　Lecture4_Form.html の　ソース

```
<HTML>
<HEAD>
<TITLE>Handling an HTTP Get Request</TITLE>
<META HTTP-EQUIV="content-type" CONTENT="text/html">
</HEAD>
<BODY>
<FORM ACTION = "/servlet/Lecture4_Servlet_XXXX" METHOD = >
First name<BR>
<INPUT TYPE = "text" NAME = "Form_FirstName" /><BR><BR>
Native language:<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="Japanese" CHECKED="CHECKED">Japanese<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="English">English<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="French">French<BR>
<INPUT TYPE="RADIO" NAME = "Form_Language" VALUE="-Other-">-Other-<BR><BR>
Spoken Languages: <BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakJapanese" CHECKED="CHECKED">Japanese<BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakEnglish">English<BR>
<INPUT TYPE="CHECKBOX" NAME="Form_SpeakFrench">French<BR>
<BR>
<BR>
Favourite car:<BR>
<SELECT NAME="Form_Cars">
<OPTION VALUE="Toyota">Toyota
<OPTION VALUE="Honda">Honda
<OPTION VALUE="Nissan">Nissan
<OPTION VALUE="-Other-">-Other-
</SELECT><BR>
<INPUT TYPE = "submit" VALUE = "Start Servlet"/>
</FORM>
</BODY>
</HTML>
```

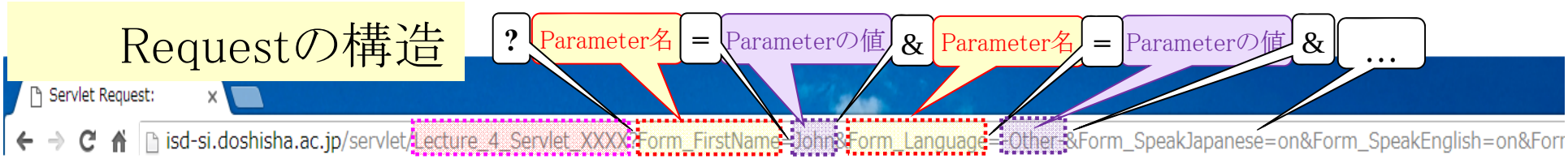
Parameter名
(NAMEアトリビュートの値)

Chrome or



Parameterの値
ユーザのインプット:
John

Requestの構造



1. 第4回のまとめ

HTTP Requestの処理 & HTTP Responseの作成

Servlet Lecture_4_Servlet_XXXX.java の内容

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  public class Lecture_4_Servlet_XXXX extends HttpServlet {
5  public void service (HttpServletRequest request, HttpServletResponse response)
6  throws ServletException, IOException
7  {
8   response.setContentType("text/html");
9  // Getting the values of parameters of Web Form:
10  String firstName   = request.getParameter("Form_FirstName");
11  String nativeLang  = request.getParameter("Form_Language");
12  String car         = request.getParameter("Form_Cars");
13  String spokenLang  = "";
14  if (request.getParameter("Form_SpeakEnglish")  !=null) spokenLang="English";
15  if (request.getParameter("Form_SpeakJapanese") !=null) spokenLang=spokenLang + " Japanese";
16  if (request.getParameter("Form_SpeakFrench")   !=null) spokenLang=spokenLang + " French";
17  // Getting the communication channel with the requesting client
18  PrintWriter out = response.getWriter();
19  // Writing the response to be sent to Web Browser
20  out.println("<HTML><HEAD><TITLE>Servlet Request: </TITLE>"
21      + " <META HTTP-EQUIV='Content-Type' Content='text/html';"
22      + " charset='SHIFT_JIS'></META></HEAD>");
23  out.println(" <CENTER>"
24      + "<H2>Hello "+firstName+"-san!</H2>"
25      + "<H3>Welcome to Servlets!</H3></CENTER>"
26      + " Your native language is <B>"+nativeLang+"</B><BR>"
27      + " You speak <B>"+ spokenLang +"</B><BR>"
28      + " Your favorite car is <B>"+car+"</B><BR><BR>");
29  out.println("Local time is " + new java.util.Date());
30  out.println("</BODY></HTML>");
31  }
32  }
```

注意：class名＝ファイル名　(.java)
例:　class名: **Lecture_4_Servlet_1116221001**
　　ファイル名: **Lecture_4_Servlet_1116221001.java**

HTTP Requestの処理

HTTP Responseの作成

1. 第4回のまとめ

Servlet Lecture_4_Servlet_XXXX.java の内容

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  public class Lecture_4_Servlet_XXXX extends HttpServlet {
5  public void service (HttpServletRequest request, HttpServletResponse response)
6  throws ServletException, IOException
7  {
8   response.setContentType("text/html");
9  // Getting the values of parameters of Web Form:
10  String firstName   = request.getParameter("Form_FirstName");
11  String nativeLang  = request.getParameter("Form_Language");
12  String car         = request.getParameter("Form_Cars");
13  String spokenLang  = "";
14  if (request.getParameter("Form_SpeakEnglish")  !=null) spokenLang="English";
15  if (request.getParameter("Form_SpeakJapanese") !=null) spokenLang=spokenLang + " Japanese";
16  if (request.getParameter("Form_SpeakFrench")   !=null) spokenLang=spokenLang + " French";
17  // Getting the communication channel with the requesting client
18  PrintWriter out = response.getWriter();
19  // Writing the response to be sent to Web Browser
20  out.println("<HTML><HEAD><TITLE>Servlet Request: </TITLE>"
21      + " <META HTTP-EQUIV='Content-Type' Content='text/html';"
22      + " charset='SHIFT_JIS'></META></HEAD>");
23  out.println(" <CENTER>"
24      + "<H2>Hello "+firstName+"-san!</H2>"
25      + "<H3>Welcome to Servlets!</H3></CENTER>"
26      + " Your native language is <B>"+nativeLang+"</B><BR>"
27      + " You speak <B>"+ spokenLang +"</B><BR>"
28      + " Your favorite car is <B>"+car+"</B><BR><BR>");
29  out.println("Local time is " + new java.util.Date());
30  out.println("</BODY></HTML>");
31  }
32  }
```

Static Content

Dynamic Content

Static Content

HTTP Responseの作成

Q: サーブレットはStatic Contentをどのように作成しますか？
A: Dynamic Contentと同じように、**Java プログラミング**の **out.print( )** の命令で.

2. Java Server Pages (JSP)入門

Lecture_5_page.jsp

```
<HTML>
  <HEAD>
  <TITLE>Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html" %>
  </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>
```

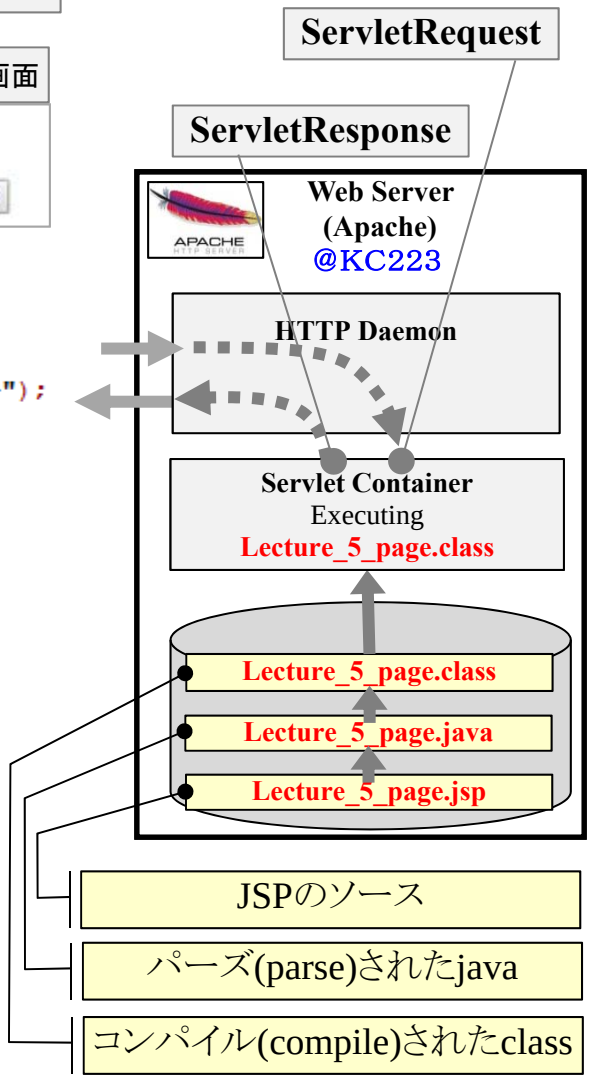
Static content: HTMLタグ

Javaコード: <% ... %>

Static content: Web Form

Web Browserの画面

Enter your name:
John Submit



Q: JSPはStatic Contentをどのように作成しますか？
A: 直接、**HTMLタグがついているテキスト**で. Static Web page (htmlファイル)と同じです(参考:第1回の授業)。

2. JSP入門

JSPの特徴

① Web Serverに保存されています(ファイル　*.jsp)。

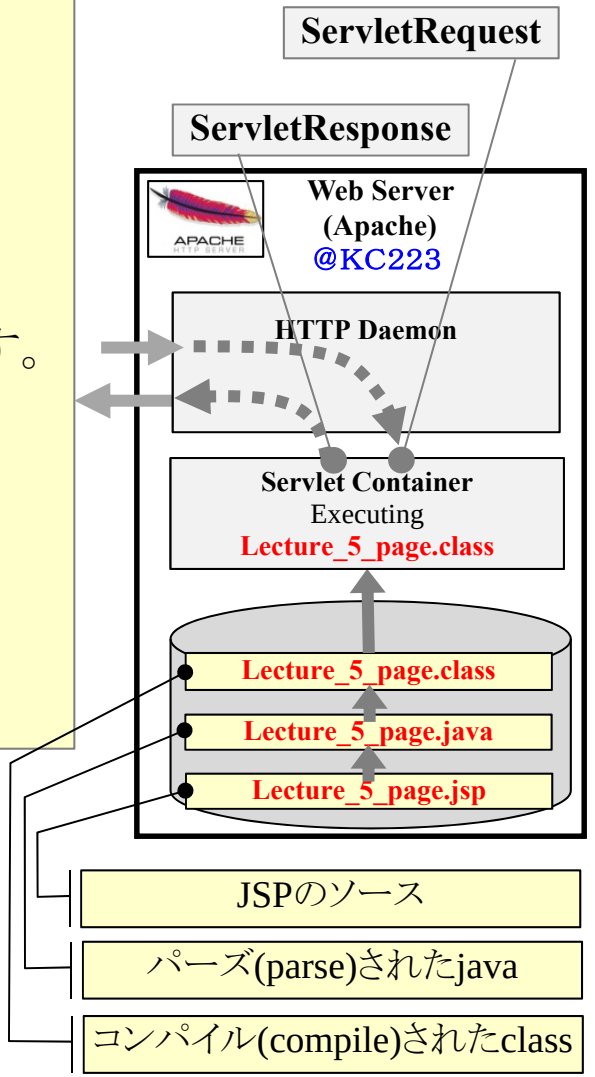
② Static ContentはHTMLタグで表現されています。

③ Dynamic Contentは<% … %>の間のJavaで表現されています。

④ Web Serverに実行されています。

⑤ コンパイルされたclassファイルが実行されています。

⑥ **パーズ(*.jsp→*.java)とコンパイル(*.java →*.class)は自動的に行います。**




```
Output to the user using <B>JSP expression:</B><BR>
User name: <%= userName %>
```

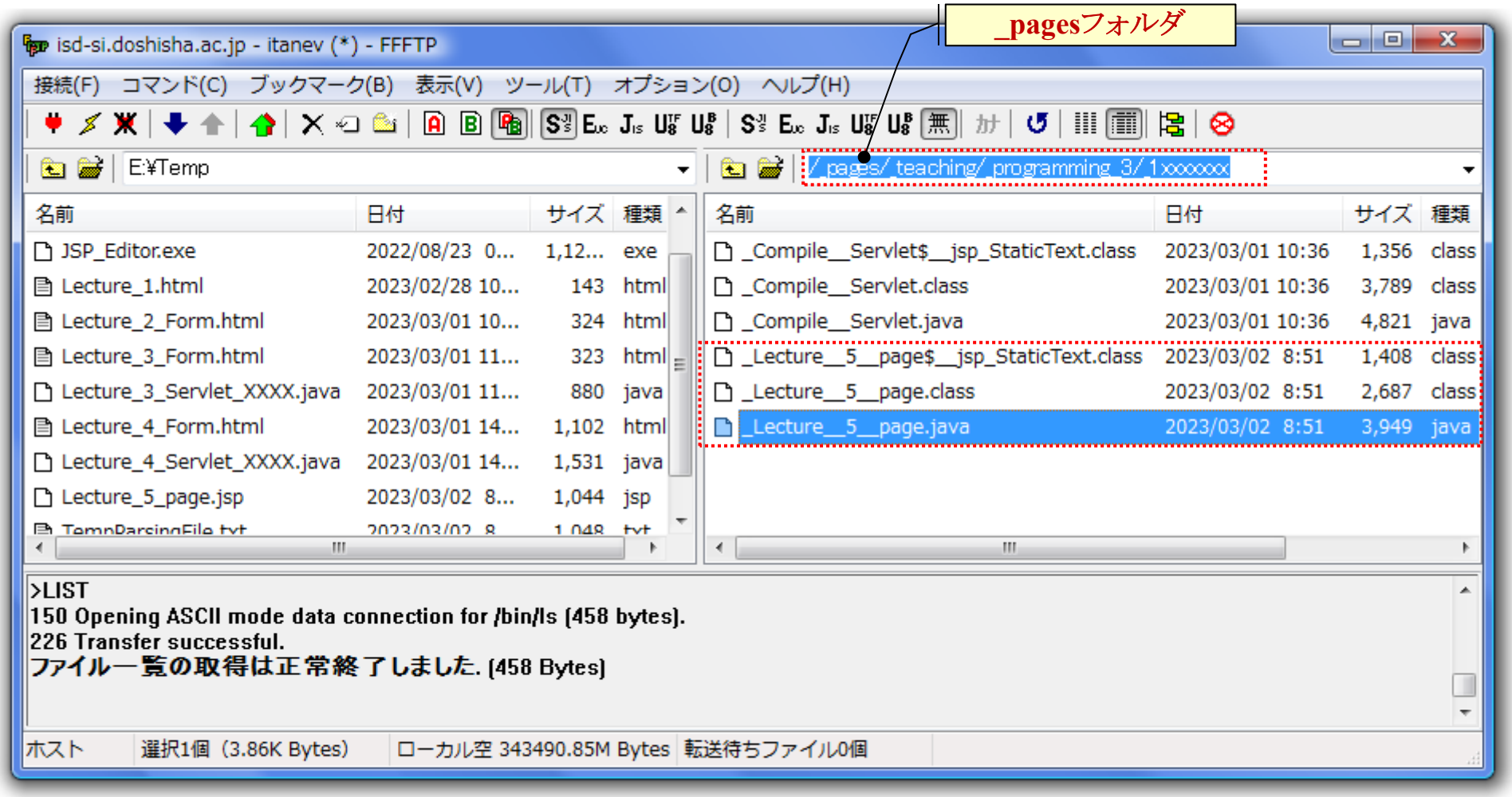
サーブレットと同じ特徴：　①　④　⑤

```
Enter your name:<BR>
<INPUT NAME = "InputBox">
<INPUT TYPE = "Submit" VALUE="Submit"/>
```

サーブレットと違う特徴：　②　③　⑥

2. JSP入門

Q: JSPからコンパイルされたjavaとclassファイルの保存されていす場所は何所ですか？
A: Web Serverの**_pagesフォルダ**です。



https://docs.oracle.com/cd/A87860_01/doc/java.817/a83726/toc.htm

http://isd-si.doshisha.ac.jp/teaching/programming_3/Lectures/Lecture_5_Appendix.pdf

2. JSP入門

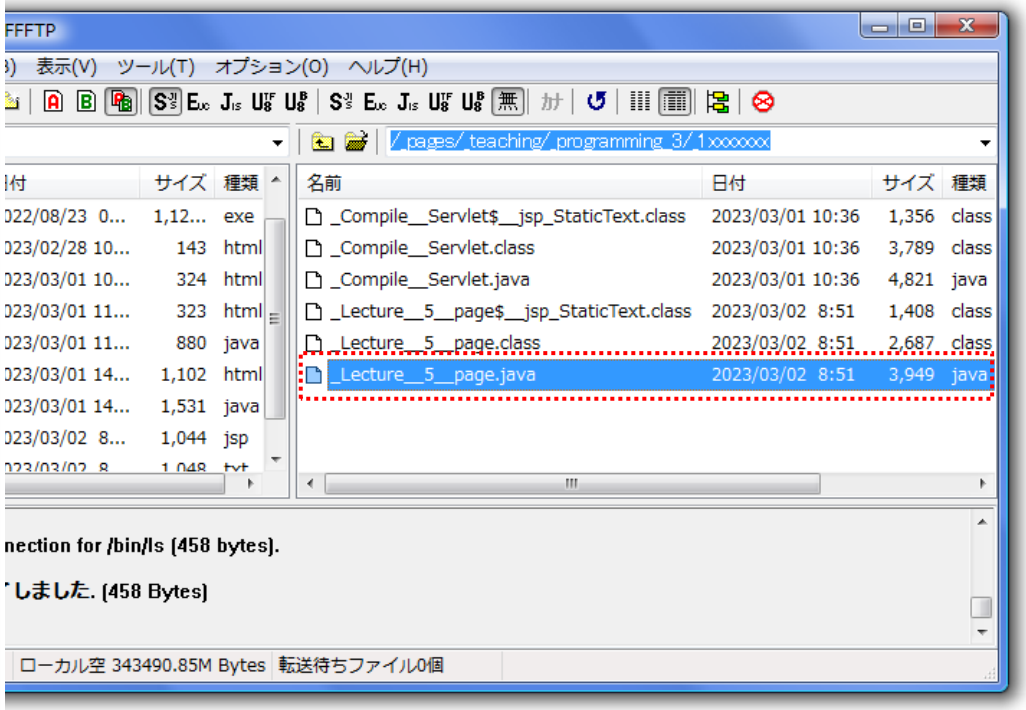
Q: JSPからコンパイルされたjavaとclassファイルの保存されていす場所は何所ですか？
A: Web Serverの**_pagesフォリダ**です。

JSPからコンパイルされたjava (一部)

_Lecture__5__page - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
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;
}
```

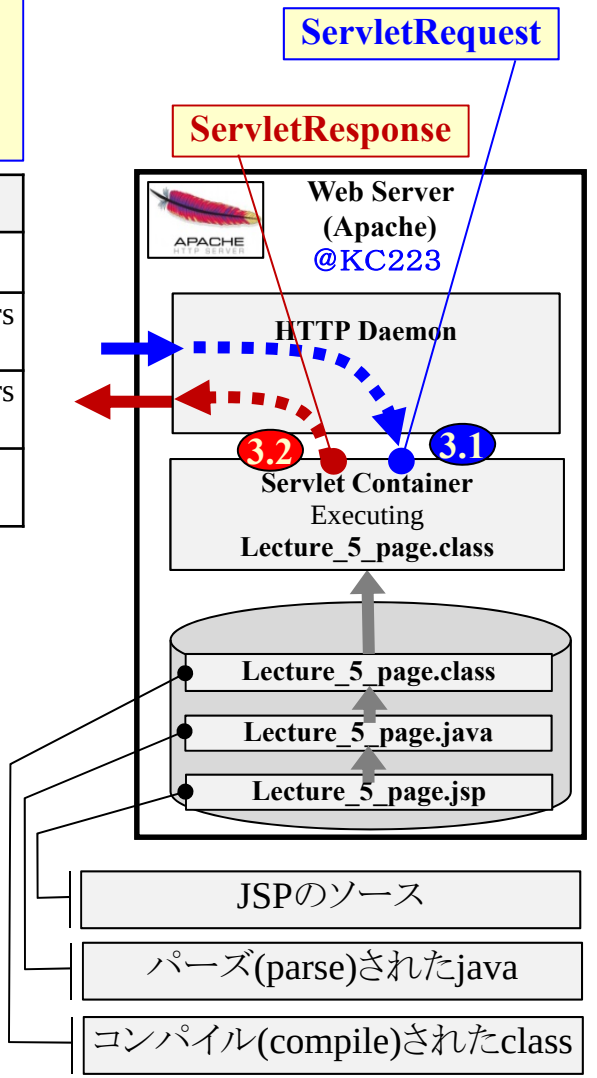


3. ユーザ入力およびユーザに出力の処理

3.1 ユーザー入力の処理 （ServletRequest）

サーブレットと同じ、第4回の授業に参考

Method	Defined In	Job Performed
getRequest	javax.servlet.jsp.PageContext	Returns the current request object
getParameterNames	javax.servlet.ServletRequest	Returns the names of the parameters request currently contains
getParameterValues	javax.servlet.ServletRequest	Returns the values of the parameters request currently contains
getParameter	javax.servlet.ServletRequest	Returns the value of a parameter if the name is provided



3.2 ユーザー出力の処理 （ServletResponse）

方法1: **PrintWriterの利用**（サーブレットと同じ、第3回の授業に参考）

<% out.println(StringVariable) %>

out はサーブレットのPrintWriterです。
out.println は サーブレットのout.println です。

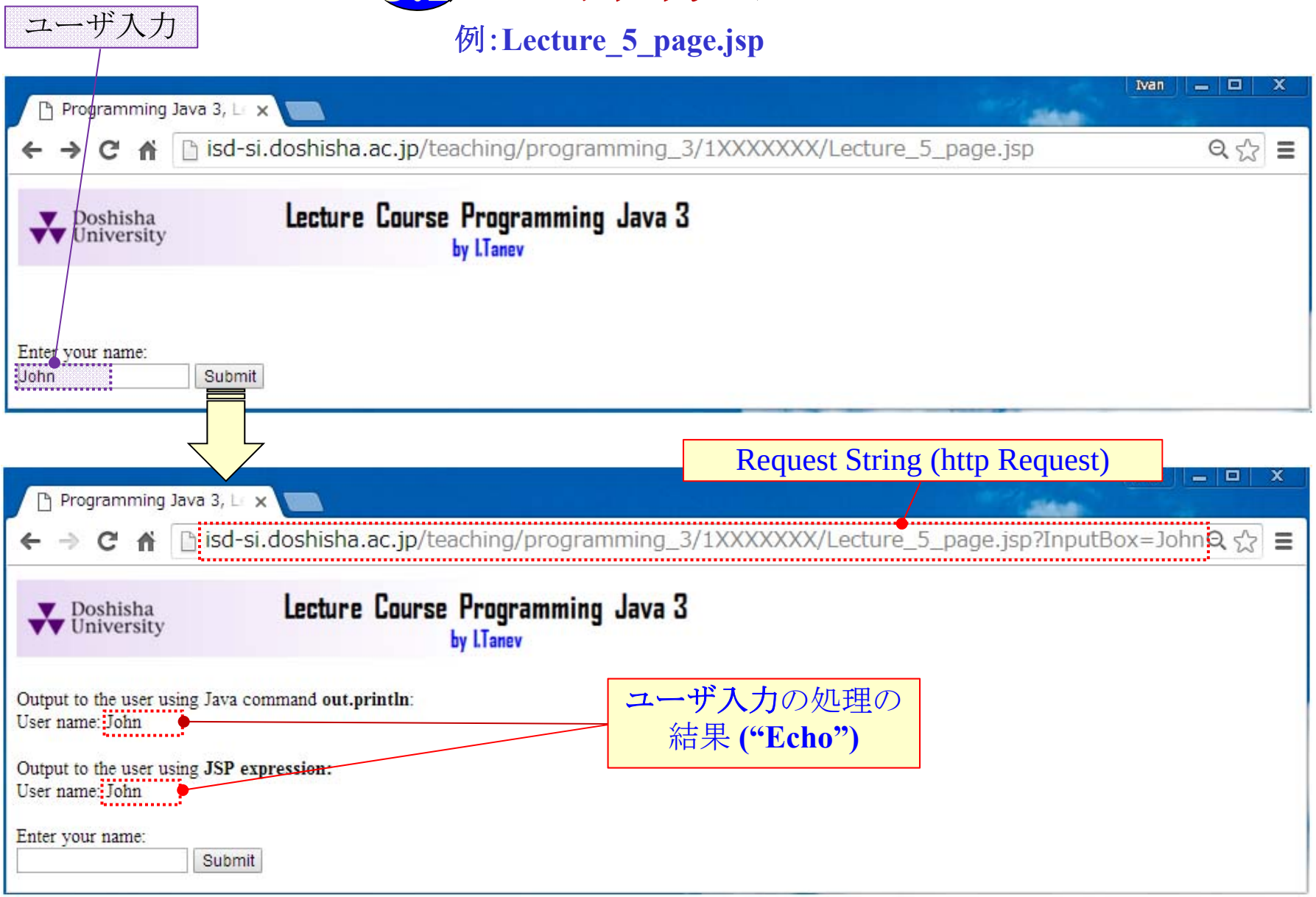
New 方法2： **JSP Expressionの利用**

<%= StringVariable %>

3. ユーザ入力およびユーザに出力の処理

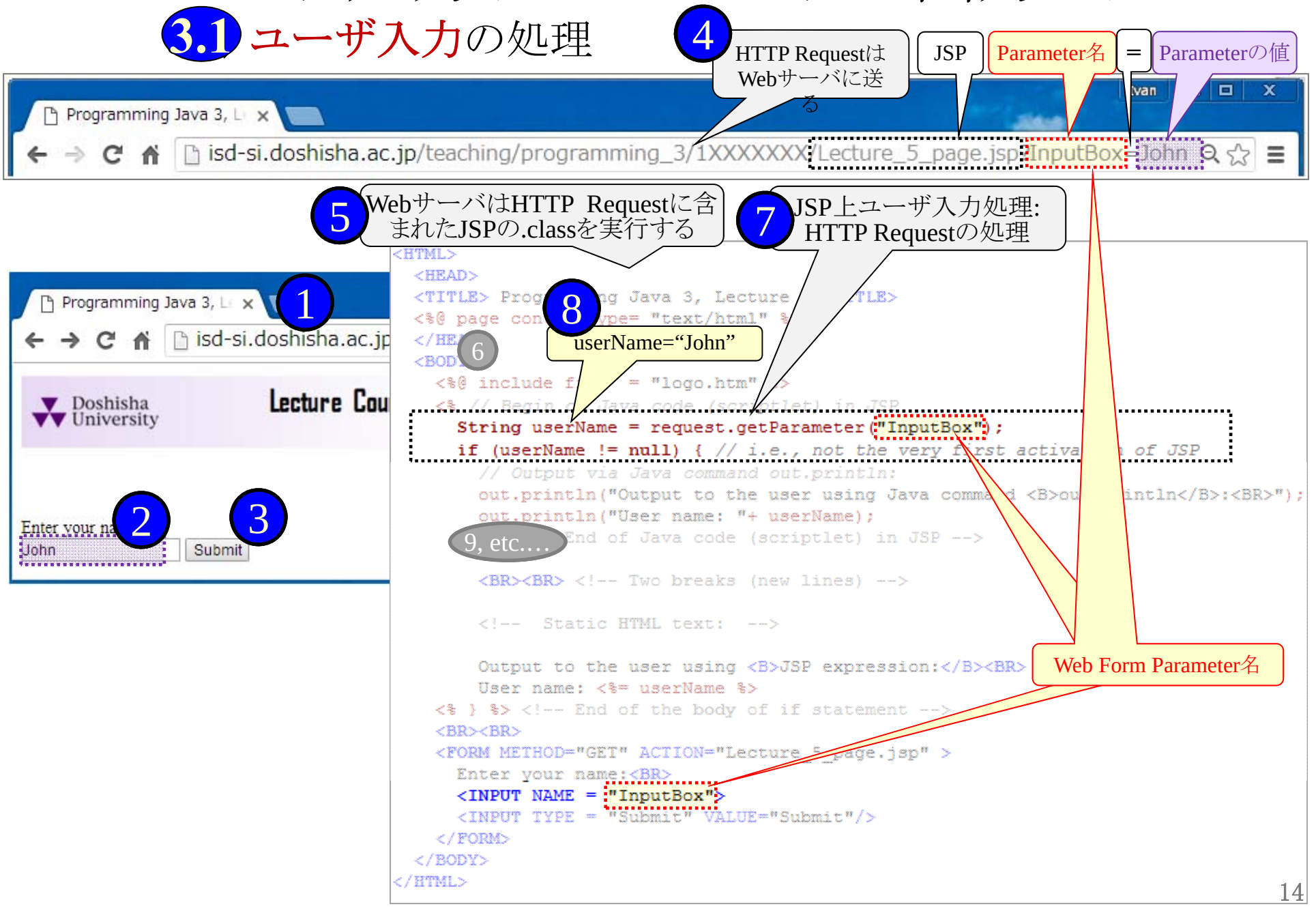
3.1 ユーザ入力の処理

例：**Lecture_5_page.jsp**



3. ユーザ入力およびユーザに出力の処理

3.1 ユーザ入力の処理




```
<HTML>
  <HEAD>
  <TITLE> Prog ng Java 3, Lecture TLE>
  <%@ page con ype= "text/html" %
  </HE
  <BOD
    <%@ include f = "logo.htm" >
    <% // Begin J Java code (scriptlet) in JSP
      String userName = request.getParameter("InputBox");
      if (userName != null) { // i.e., not the very first activa of JSP
        // Output via Java command out.println:
        out.println("Output to the user using Java comma d <B>ou intln</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>
```

3. ユーザ入力およびユーザに出力の処理

3.2 出力処理の処理

（例：JSPの文字列変数userNameの値をユーザに出力）

```
<HTML>
  <HEAD>
  <TITLE> Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html" %>
  </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
        // 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 line

        <!--  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>
```

変数

定数

方法1: **PrintWriter**、お呼**println**の命令（メソード）利用
（サーブレットと同じ。参考：第3回＆第4回の授業）

New
方法2：
a) 直接のテキスト（constant, 定数の場合）
b) JSP expression **<%= … %>**の
利用 (variable, 変数の値の場合）

New
JSP expression **<%= … %>** （.jspファイル）は自動的に**out.print**（.javaファイル）にパースされています。
例： **<%= userName %>** は **out.print(userName)** にパースされています。

3. ユーザ入力およびユーザに出力の処理

3.2) 出力処理の処理

New 方法2： JSP Expressionの利用

http://docs.oracle.com/javaee/5/tutorial/doc/bnaov.html

The Java EE 5 Tutorial

Home | Download | PDF | FAQ | Feedback

JSP Expressions

A **JSP expression** is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client. When the scripting language is the Java programming language, an expression is transformed into a statement that converts the value of the expression into a `String` object and inserts it into the implicit `out` object.

The syntax for an expression is as follows:

```
<%= scripting-language-expression %>
```

Note that a semicolon is not allowed within a JSP expression, even if the same expression has a semicolon when you use it within a scriptlet.

In the web service version of the `hello1` application, `response.jsp` contains the following scriptlet, which gets the proxy that implements the service endpoint interface. It then invokes the `sayHello` method on the proxy, passing the user name retrieved from a request parameter:

```
<%
    String resp = null;
    try {
        Hello hello = new HelloService().getHelloPort();
        resp = hello.sayHello(request.getParameter("username"));
    } catch (Exception ex) {
        resp = ex.toString();
    }
%>
```

A scripting expression is then used to insert the value of `resp` into the output stream:

```
<h2><font color="black"><%= resp %>!</font></h2>
```

4. JSP Directives

Lecture_5_page.jsp

```
<HTML>
  <HEAD>
  <TITLE> Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html" %>
  </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 -->
```

Include Directive

Include Directive

Includes a static file in a JSP page, parsing the file's JSP elements.

JSP Syntax

```
<%@ include file="relativeURL" %>
```

```
<BR>
  <FORM METHOD="GET" ACTION="Lecture_5_page.jsp" >
```

https://www.oracle.com/technetwork/java/syntaxref12-149806.pdf (Pages 13~14)
http://isd-si.doshisha.ac.jp/teaching/programming_3/Lectures/Lecture_5_Appendix.pdf (Pages 1~3)

4. JSP Directives

Lecture_5_page.jsp

```
<HTML>
  <HEAD>
  <TITLE> Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html" %>
  </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>
```

Page Directive

Page Directive

Defines attributes that apply to an entire JSP page.

JSP Syntax

```
<%@ 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" ]
   [ pageEncoding="characterSet | ISO-8859-1" ]
%>
```

https://www.oracle.com/technetwork/java/syntaxref12-149806.pdf (Pages 15~19)
http://isd-si.doshisha.ac.jp/teaching/programming_3/Lectures/Lecture_5_Appendix.pdf (Pages 1~3)

5. まとめ:JSP vs Servlets

Lecture_5_page.jsp

```
<HTML>
  <HEAD>
  <TITLE> Programming Java 3, Lecture 5</TITLE>
  <%@ page contentType= "text/html" %>
  </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>
        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>
```

何時**JSP**を利用するのか？

- A lot of Static HTML
- Little data processing

何時**Servlet**を利用するのか？

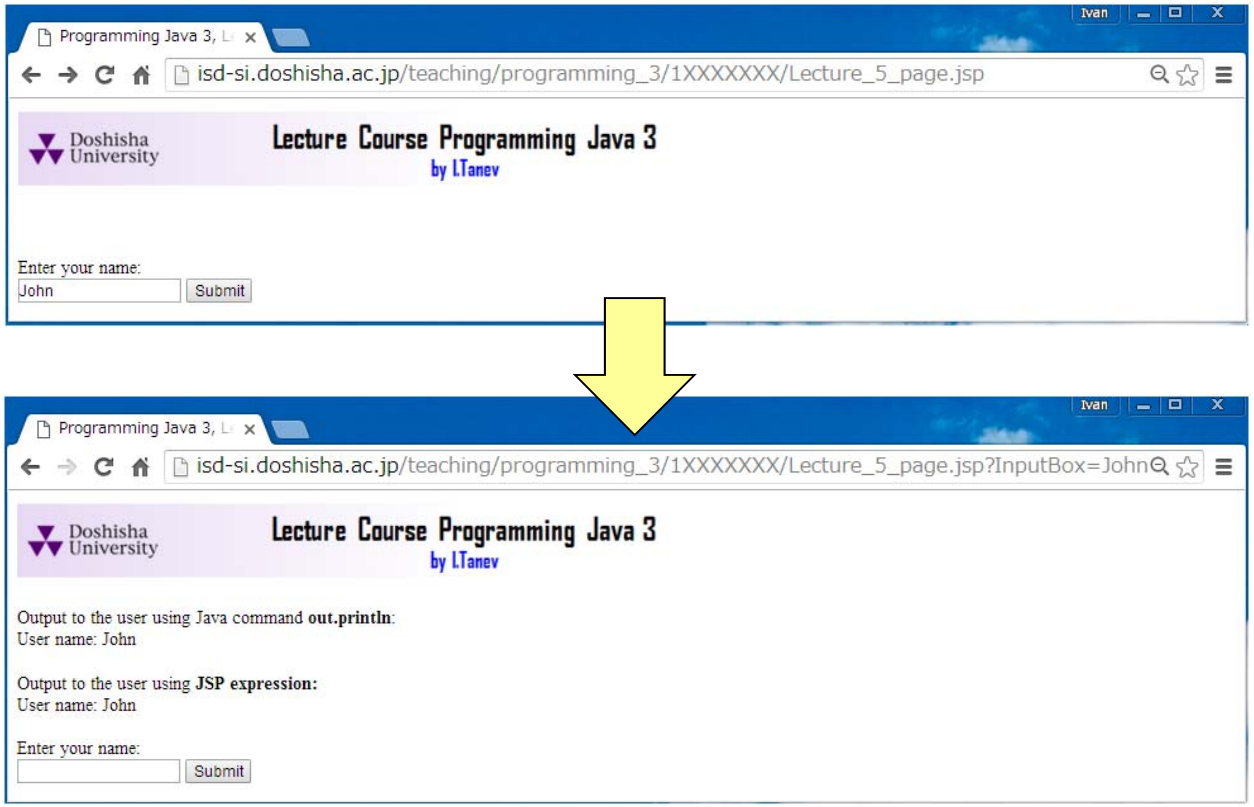
- Little static HTML
- A lot of data processing

Servlet Lecture_4_Servlet_XXXX.javaのソース

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  public class Lecture_4_Servlet_XXXX extends HttpServlet {
5  public void service (HttpServletRequest request, HttpServletResponse response)
6  throws ServletException, IOException
7  {
8   response.setContentType("text/html");
9  // Getting the values of parameters of Web Form:
10  String firstName   = request.getParameter("Form_FirstName");
11  String nativeLang  = request.getParameter("Form_Language");
12  String car         = request.getParameter("Form_Cars");
13  String spokenLang  = "";
14  if (request.getParameter("Form_SpeakEnglish")  !=null) spokenLang="English";
15  if (request.getParameter("Form_SpeakJapanese") !=null) spokenLang=spokenLang + " Japanese";
16  if (request.getParameter("Form_SpeakFrench")   !=null) spokenLang=spokenLang + " French";
17  // Getting the communication channel with the requesting client
18  PrintWriter out = response.getWriter();
19  // Writing the response to be sent to Web Browser
20  out.println("<HTML><HEAD><TITLE>Servlet Request: </TITLE>"
21      + " <META HTTP-EQUIV='Content-Type' Content='text/html';"
22      + " charset='SHIFT_JIS'></META></HEAD>");
23  out.println(" <CENTER>"
24      + "<H2>Hello "+firstName+"-san!</H2>"
25      + "<H3>Welcome to Servlets!</H3></CENTER>"
26      + " Your native language is <B>"+nativeLang+"</B><BR>"
27      + " You speak <B>"+ spokenLang +"</B><BR>"
28      + " Your favorite car is <B>"+car+"</B><BR><BR>");
29  out.println("Local time is " + new java.util.Date());
30  out.println("</BODY></HTML>");
31  }
32  }
```

6. 演習

http://isd-si.doshisha.ac.jp/teaching/programming_3/Lectures/Lecture_5_Appendix.pdf
(Pages 4~8)



Lecture_5.jspの開発の締め切り：

5月21日（木曜日）23:59

The End