

上級プログラミング2(第9回)

工学部 情報工学科
木村昌臣
中島毅

今日のテーマ

□ 先週の復習

- Webアプリケーションの仕組み
- 動かすための設定

□ Webアプリケーションプログラム

- Servlet
- JSP

先週の復習

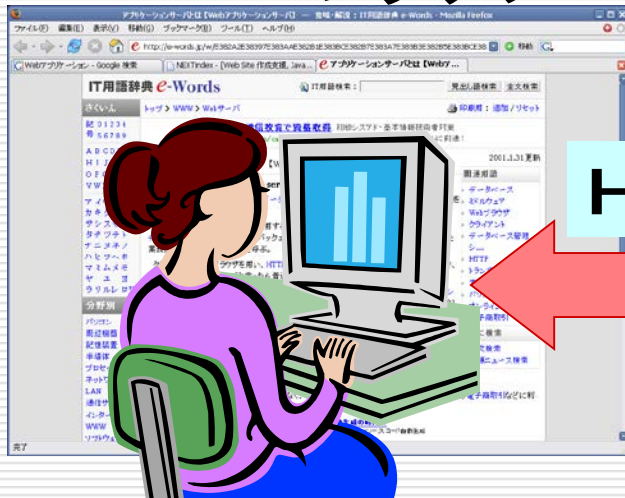
Webアプリケーションの仕組み

ユーザ
インタフェース

HTTP
-アプリケーション間
インタフェース

アプリケーション
が動作
(Javaプログラム等)

Webブラウザ



HTTP

Web
サーバ

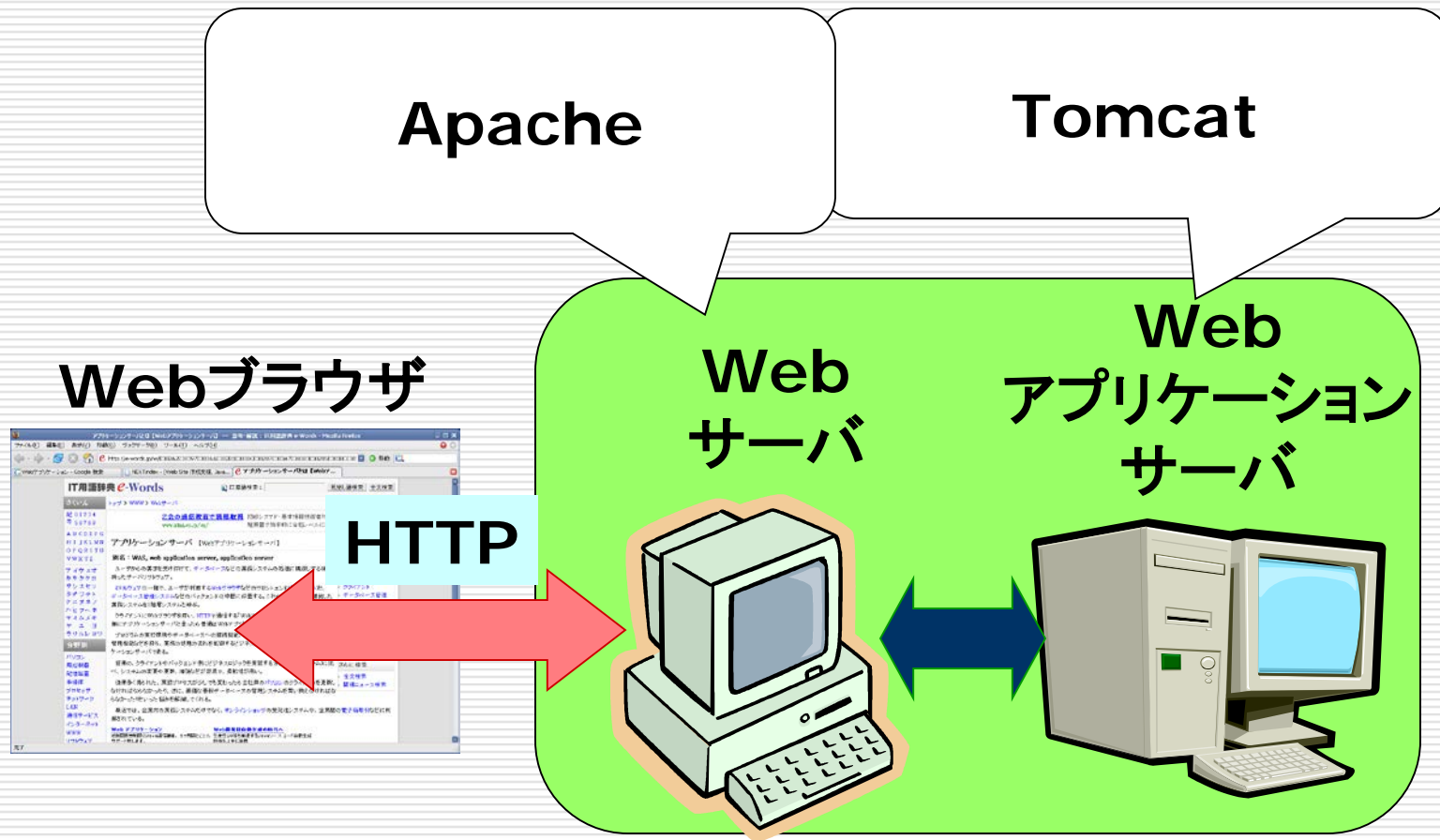


Web
アプリケーション
サーバ



接続には専用の
モジュールが必要

Webアプリケーションの実行環境例



CLASSPATH

- Servlet (Javaプログラム)をコンパイル可能にするために**CLASSPATH**に以下を含める.
 - Tomcatのインストールディレクトリ下
 - servlet-api.jar
 - jsp-api.jar
 - カレントディレクトリ(.)

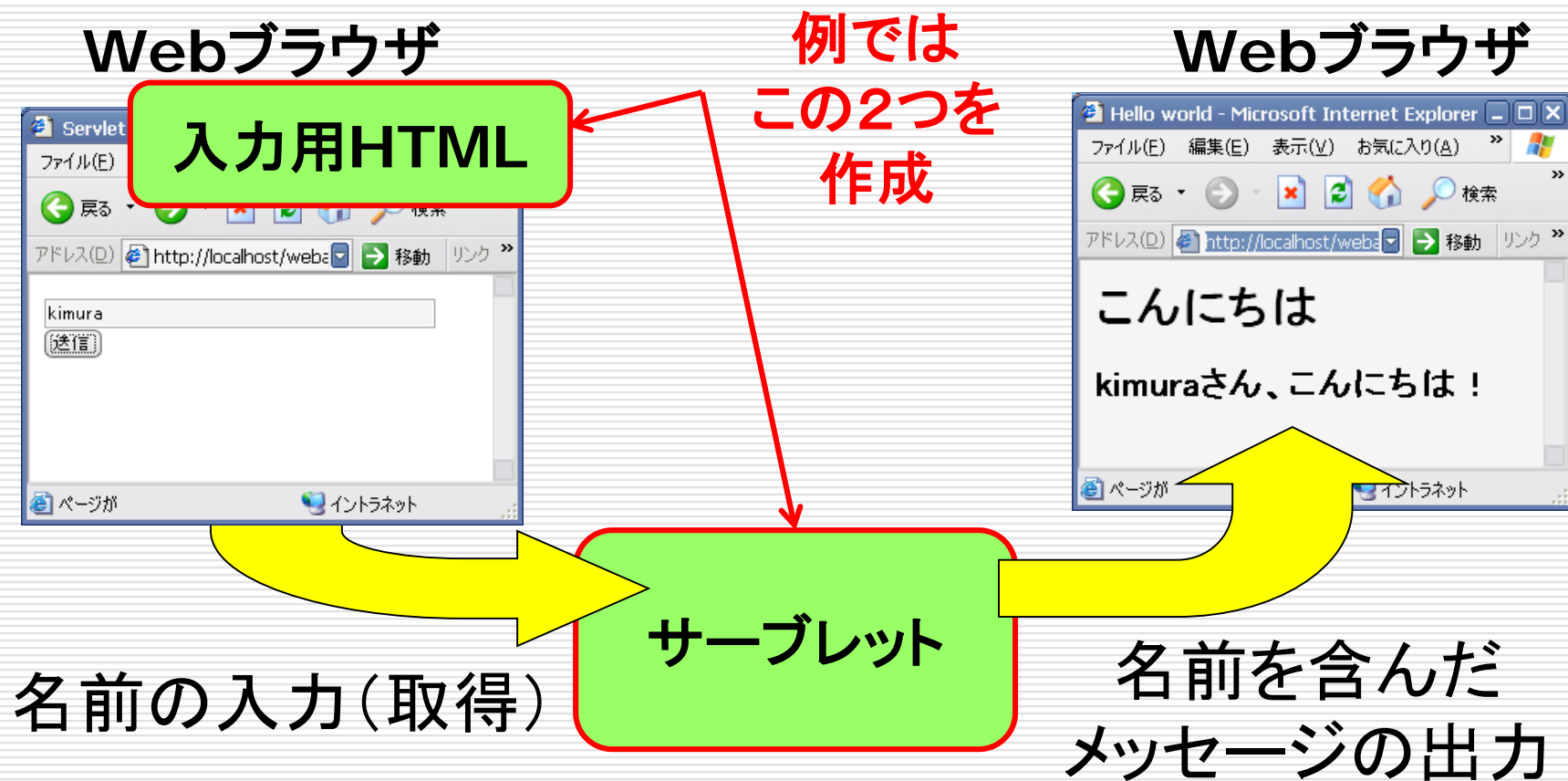
```
set CLASSPATH=.;C:¥temp¥lib¥servlet-api.jar;  
C:¥temp¥lib¥jsp-api.jar; %CLASSPATH%
```

Webアプリケーションプログラミング

【復習】サーブレット(Servlet)

- Webアプリケーションサーバで動作するJavaプログラム
 - HttpServletクラスを継承して作る
 - 次の三つのパッケージはインポートする必要あり
 - java.io.*
 - javax.servlet.*
 - javax.servlet.http.*
 - HttpServletRequestで入力、
HttpServletResponseで出力を行う
-

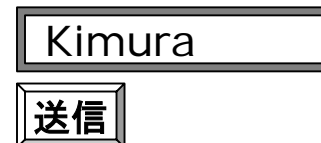
サンプルアプリケーション



サンプル(入力用HTML)

```
<html>
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=Shift_JIS">
  <title>Servlet Examples</title>
</head>
<body bgcolor="#FFFFFF">

<FORM method="GET" action="servlet/Simple">
<input type="text" name="MYNAME"
        size="50" maxlength="45" />
<input type="submit" value="送信" />
</FORM>
</body>
</html>
```



Kimura

送信

サンプル(サーブレット)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SimpleServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException
    {

        String name=request.getParameter("MYNAME");

        response.setContentType("text/html;charset=Shift
_JIS");
```

サーブレットのつづき

```
PrintWriter out = response.getWriter();
```

```
out.println("<html>");  
out.println("<head>");  
out.println("<title>Hello world</title>");  
out.println("</head>");  
out.println("<body>");  
out.println("<h1> こんにちは </h1>");  
out.println("<h2>" + name + "さん、こんにちは！ </h2>");  
out.println("</body>");  
out.println("</html>");
```

```
}
```

```
}
```

HttpServletRequest クラス

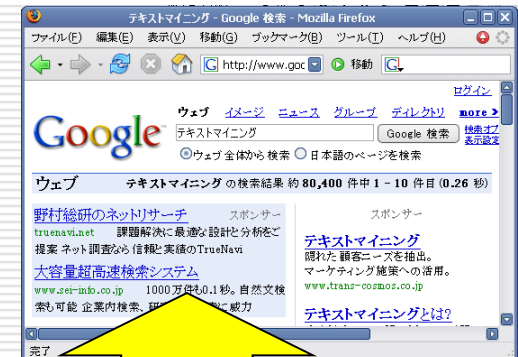
HttpServletResponse クラス

Webブラウザ



サーバレット

Webブラウザ



HttpServletRequest
ブラウザに入力された
情報を格納

HttpServletResponse
ブラウザに出力する
情報を格納

サーブレットの2つのメソッド

□ doGetメソッド

- ブラウザから直接にURL指定で呼び出された場合
- HTMLフォーム(<form>タグ)からGETメソッドでページが要求された場合
- デフォルトで呼び出される

□ doPostメソッド

- HTMLフォーム(<form>タグ)からPOSTメソッドでページが要求された場合

□ 両方とも、引数に以下のオブジェクトをもつ

- HttpServletRequest
- HttpServletResponse

Webブラウザから値を渡す方法: POST/GET

ブラウザからWebアプリケーションサーバに値を渡す方法として、POSTとGETの2種類がある

Webブラウザ

URLにパラメータ
を埋め込む方法

GET

標準入力へパラメータ
を渡す方法

POST

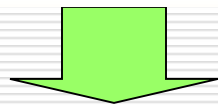
Webサーバ/
Webアプリケーション
サーバ



GET での URLエンコーディング

```
<input type="text" name="MYNAME" />
```

木村	送信
----	----



日本語はそのまま送れない

```
http://localhost/webapps/servlet/Simple/?  
MYNAME=%E6%9C%A8%E6%9D%91
```

- URLの最後に?をつけ、その後にパラメータとその値を「パラメータ名=値」という形式で列挙。複数ある場合は&で連結
- 日本語文字列は'%' + 二桁の16進数の列に変換する
- 検索エンジンではこの方法で値をサーバに渡していることが多い(例えばgoogleなど)

サーブレットでの値の取得方法

- HttpServletRequestオブジェクトの `getParameter` メソッドを利用する
 - 引数には、HTMLのINPUTタグの `name="..."` で指定した名前を書く
 - 戻り値はStringオブジェクト

```
String name=request.getParameter("MYNAME");
```

ブラウザへのHTMLの出力方法

- `HttpServletResponse`の`getWriter`メソッドを利用.
 - 戻り値: `PrintWriter`オブジェクト
 - `PrintWriter`の`println`メソッドを使ってHTMLを書き込む
 - あらかじめ`HttpServletResponse`オブジェクトの`setContentType`で文字コードなどを設定しておくとう文字化けしない

```
response.setContentType  
    ("text/html; charset=Shift_JIS");  
PrintWriter out = response.getWriter();  
out.println("<html>");  
out.println("<head>");  
.....
```

web.xml ファイル

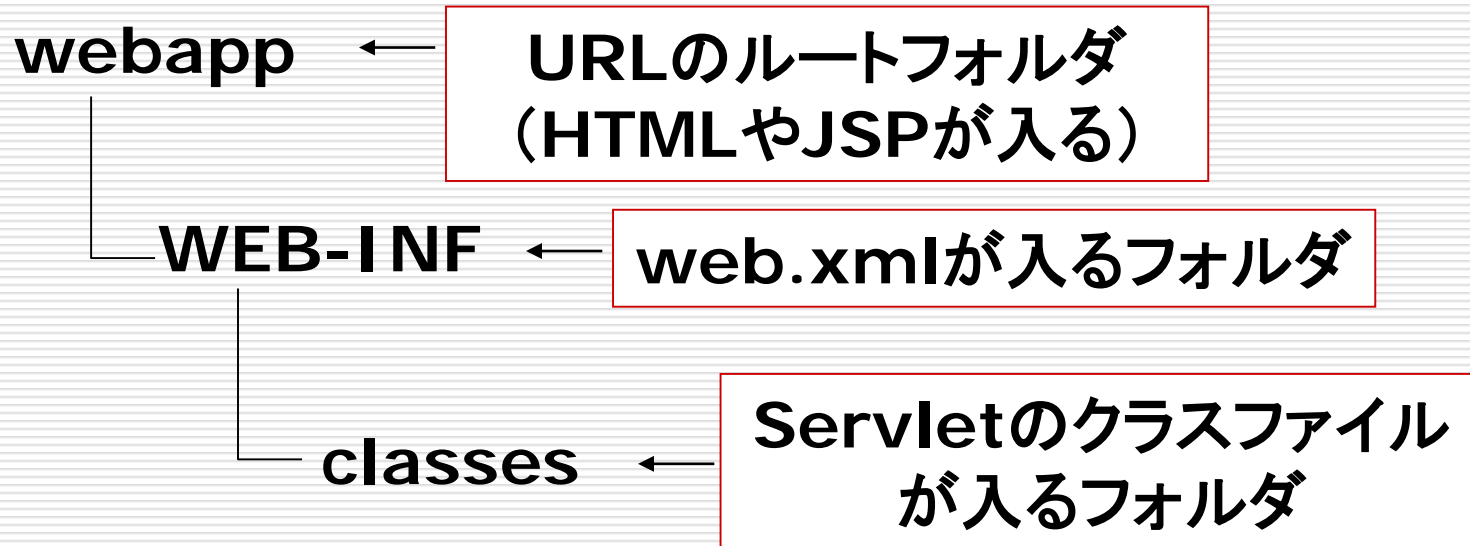
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>Simple</servlet-name>
    <servlet-class>SimpleServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Simple</servlet-name>
    <url-pattern>/servlet/Simple</url-pattern>
  </servlet-mapping>
</web-app>
```

クラスと
サーブレット名
の対応付け

サーブレット名
とURL表示の
対応付け

ファイルの置き場所



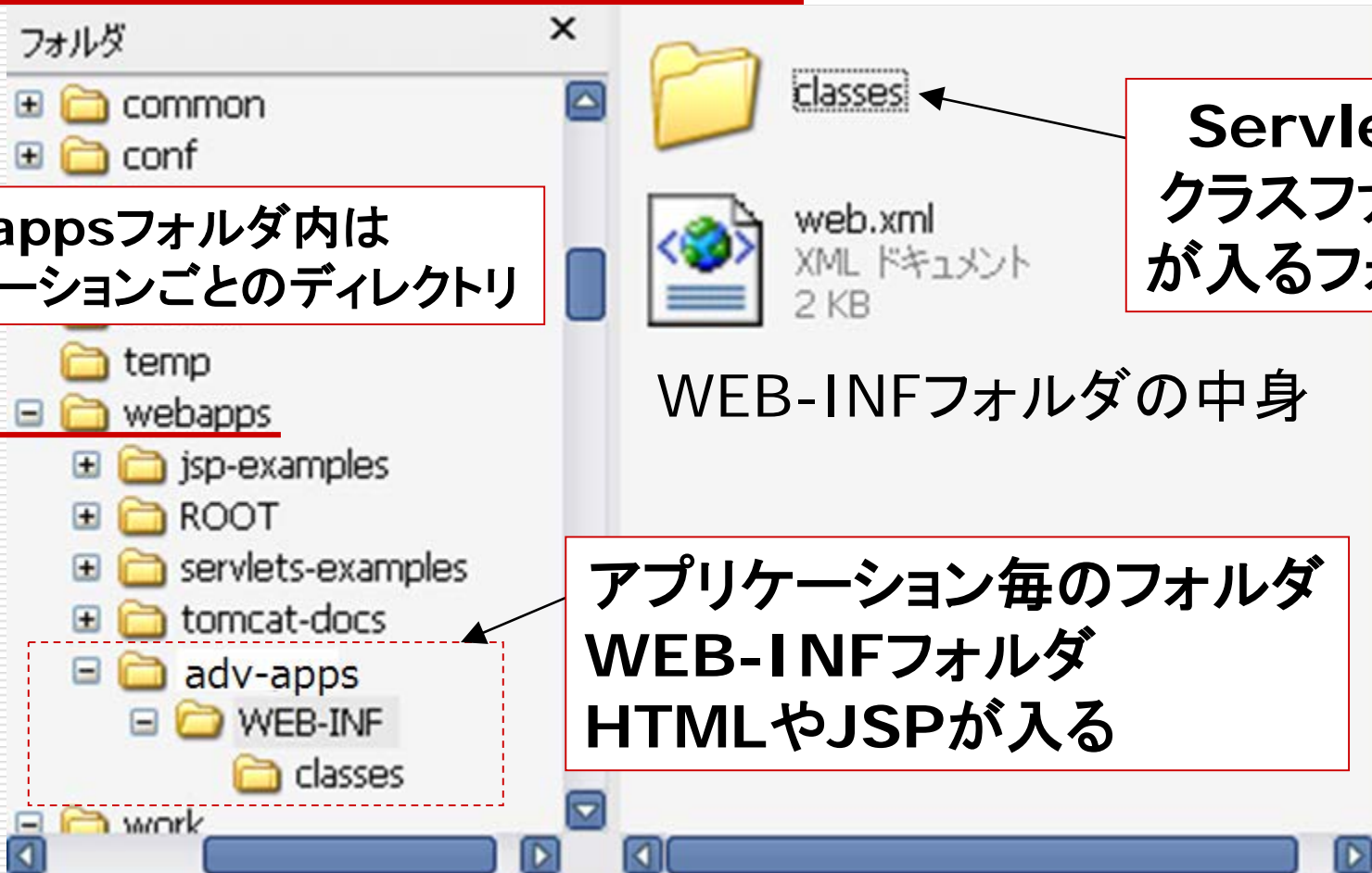
【参考】 ファイルの置き場所

※webappsフォルダ内は
アプリケーションごとのディレクトリ

Servletの
クラスファイル
が入るフォルダ

WEB-INFフォルダの中身

アプリケーション毎のフォルダ
WEB-INFフォルダ
HTMLやJSPが入る



【復習】JSP

- Java Server Page
- HTMLに Javaのプログラム処理を付加したもの
 - サーブレットは最終的にHTML形式で出力
→ページのレイアウトの調整を詳細に行うことは困難
 - JSPでは, Webページのレイアウトを決めて(HTMLベース)、必要な処理(Javaプログラム)を付加する

サンプルプログラムを, JSPを使うように改造 ～まずは出力用のHTMLを作る～

```
<html>
<head>
<meta http-equiv="Content-Type"
      content="text/html; charset=Shift_JIS">
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
<h2> ○○○さん、こんにちは！ </h2>
</body>
```

※ファイルの拡張子をjspとする(この例ではoutput.jspとする)

次のように処理を追加する(赤字部分)

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<html>
<head>
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
    <%
        String name=request.getParameter("MYNAME");
    %>
<h2>ようこそ！ <%= name %>さん、こんにちは！ </h2>
</body>
</html>
```


pageディレクティブ

- JSPをどのように処理したらいいのか指示
 - importしたいパッケージを指定
 - バッファサイズを指定
 - 文字コードなどを指定 (contentType)

```
<%@ page contentType=  
    "text/html; charset=Shift_JIS" %>
```

※ちなみに、JSP内部の<% ~ %>はJavaによる処理と関係する部分である

pageディレクティブの構文

```
<%@ 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" ]
    [ isELIgnored="true|false" ]
%>
```

スクリプトレット：Javaプログラムの埋込み

- `<%` と `%>` で囲まれた部分にJavaのコードを書くことが出来る

`<%`

```
String name=request.getParameter("MYNAME");
```

`%>`

- HTMLをまたいで書くことも出来る

```
<% for(int i=0; i<10; i++){ %>
```

```
    <H1>10回書くぞ！！！！</H1><HR>
```

```
<% } %>
```

代入

□ HTMLの中に**変数**の値を埋め込みたいときは
 `<%= 変数 %>`とする。

※ %と=の間にスペースなどを入れないこと

□ 処理結果を埋め込むときによく利用される

`<h2>ようこそ！ <%= name %>さん、こんにちは！ </h2>`

サーブレットの変更

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SimpleServlet2 extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    { //JSPのURL
        String url="/servlet/output.jsp";
        RequestDispatcher dispatcher
            =getServletContext().getRequestDispatcher(url);
        dispatcher.forward(request, response);
    }
}
```

RequestDispatcherクラス

- サーブレットからJSPへブラウザからのリクエストを転送し、残りの処理の実行を指示
- `getServletContext().getRequestDispatcher(url)`で生成
(urlはJSPのURL文字列)
- `forward`メソッドで転送

`dispatcher.forward(request, response);`

実行結果

