

## <JSP を使ったプログラミング>

---

### <GET 通信と POST 通信の違い>

GET 通信 :

`http://www.yahoo.co.jp/?user=taro & password=123`

※URL に値がそのまま表示される。

→手入力して画面を開くことも可能。

※URL の文字列は 1024 文字以内に限定される。

POST 通信 :

`http://www.yahoo.co.jp/`

※URL に値は表示されない。(プログラム内でやりとりされる)

※文字列の制約なし

---

### <JSP と Servlet の違い>

JSP (= Java Server Pages) :

※HTML ベースのプログラムに Java をプログラムしてゆく。

※完成したプログラムはサーバーに保存して実行する。

Servlet (Server+let(小さな、細かな)) :

※Java ベースのプログラムに HTML、CSS、JavaScriptなどをプログラムしてゆく。

※完成したプログラムはサーバーに保存して実行する。

---

＜動的 Web プロジェクトを作成する＞

---

※Web サイトは「動的 Web プロジェクト」で作成してゆく。

---

### [Eclipse の起動]

Eclipse を起動する。

画面表示を「JavaEE」 パースペクティブにする。

### [新規動的 Web プロジェクトの作成]

(手順)

プロジェクト・エクスプローラー・ペインの空白部分を右クリックする。

新規 -> その他 -> Web -> 動的 Web プロジェクトの順でメニュー選択する。

「次へ」 ボタンを押下する。

「プロジェクト名」を入力する。

※今回、プロジェクト名は HelloJSP と入力する。

「デフォルト・ロケーションを使用」をチェックする。

「ターゲット・ランタイム」にて Tomcat v. 8.0 を選択する。

(確認)

(1) 「動的 web モジュールバージョン」が自動設定されていること。

(2) 「構成」がデフォルト構成にて設定されていること。

「次へ」 ボタンを押下する。

(3) 「ビルド・パス上のソース・フォルダ」にて src が表示されていること。

(4) 「デフォルト出力フォルダー」にて build\classes が表示されていること。

「次へ」 ボタンを押下する。

(5) 「コンテキスト・ルート」にて作成した動的 Web プロジェクト名 (今回は HelloJSP) が表示されていること。

(6) 「コンテンツ・ディレクトリー」にて WebContent が表示されていること。

(7) 「web.xml デプロイメント記述子の生成」を必ずチェックする。

「完了」を押下する。

(8) プロジェクト・エクスプローラー・ペインに今回作成した動的 Web プロジェクトのフォルダが作成されていること。

(9) メニュー -> プロジェクト -> プロパティの順に選択する。

リソース を選択する。

テキスト・ファイルのエンコードを確認する。

※UTF-8 にて設定されていること。

### <web.xml の記述>

web.xml は画面を表示する為の設定ファイル。

※必要に応じて画面中央あたりの「ソース」タブをクリックする。

---

WebContent¥WEB-INF のweb.xml をプログラミングしましょう。

---

```
<?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_3_1.xsd" id="WebApp_ID" version="3.1">

<display-name>HelloJSP</display-name>
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

---

※<welcome-file>タグにプログラムしたファイルが、上から順に検索される。

↓

該当ファイルが見つかったら画面に表示される。

---

## <JSP ファイルの作成>

---

JSP ファイルを作成しましょう。

---

WebContent フォルダを右クリック -> 新規 -> JSP ファイルの順に選択する。

「ファイル名」欄に index.jsp と記述する。  
「完了」ボタンを押下する。

---

JSP をプログラミングしましょう。

---

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset=" utf-8" >
    <title>TEST</title>
  </head>
  <body>
    <p>こんにちは！</p>
    <% out.println(new java.util.Date()); %>
  </body>
</html>
```

---

JSP を実行しましょう。

---

プロジェクトフォルダ「HelloJSP」を右クリックする。  
実行をクリックする。  
「サーバーで実行」をクリックする。  
実行するサーバーを選択する。（今回は Tomcat8.0）  
「次へ」をクリックする。

使用可能欄→Eclipse 内で作成された動的 Web プロジェクトのリスト  
構成済み →実際に動かしたい動的 Web プロジェクト

「追加」「削除」「すべて追加」「すべて除去」ボタンを使って、実際に動かしたい動的 Web プロジェクトを構成済みに含めるように調整しておく。

「完了」ボタンをクリック。

---

JSP で Java を書くには、<body>タグ内に以下のようなプログラミングをします。

---

```
<%  
ここに Java のプログラムを記述する。  
%>
```

---

JSP で Java をプログラミングする為に、以下のようなタグも準備されています。

**<%! 宣言文; %>**

変数、メソッドを宣言します。変数、メソッドの宣言の際は必ず ; (セミコロン) が必要です。

**<% スクリプトレット; %>**

JSP のタグでは記述できない処理を Java コードを記述して自由な処理を実行する場合に使用します。Java のコードのため、各コードには必ず ; (セミコロン) が必要です。

**<%= 式 %>**

Java コードを記述しその実行結果を表示します。

void のメソッドや、変数の宣言のみを式に記述することはできません。

---

---

(演習)

JSP をプログラミングしましょう。

<body>タグ内を以下のように書き換えてみましょう。

---

```
<%!  
static int add(int a, int b) {  
    return a+b;  
}  
%>
```

```
<p>1+2=<%=add(1, 2) %></p>  
<p>1+2=<%=add(3, 4) %></p>
```

---

JSP を実行してみましょう。

---

(演習)

JSP をプログラミングしましょう。

<body>タグ内を以下のように書き換えてみましょう。

---

```
<%! static int countA=0; %>  
<%  
    int countB=0;  
    countA++;  
    countB++;  
%>
```

```
<p>宣言による変数 countA=<%=countA %></p>  
<p>スクリプトレットによる変数 countB=<%=countB %></p>
```

---

JSP を実行してみましょう。

実行できたら、再表示ボタン (F5) を押して数字が変化することを確認してみましょう。

---

(演習)

JSP をプログラミングしましょう。  
<body>タグ内を以下のように書き換えてみましょう。

---

```
<p><% out.println(Math.random()); %></p>
<p><%=Math.random() %></p>
```

---

JSP を実行してみましょう。

---

(演習)

JSP をプログラミングしましょう。  
<body>タグ内を以下のように書き換えてみましょう。

---

```
<p>お名前を入力してください。</p>
<form method="post" action="greeting-out.jsp">
<input type="text" name="user">
<input type="submit" value="確定">
</form>
```

---

同じプロジェクトの WebContent 内に greeting-out.jsp を作成しましょう。

---

(greeting-out.jsp)  
JSP をプログラミングしましょう。  
<body>タグ内を以下のように書き換えてみましょう。

---

```
<% request.setCharacterEncoding("UTF-8"); %>
<p>こんにちは、<%=request.getParameter("user") %>さん！</p>
```

---

JSP を実行してみましょう。

---

---

(演習)

JSP をプログラミングしましょう。

<body>タグ内を以下のように書き換えてみましょう。

---

```
<form method="post" action="total-out.jsp">
<input type="text" name="price">
円 x
<input type="text" name="count">
個+送料
<input type="text" name="delivery">
円=
<input type="submit" value="計算">
</form>
```

---

同じプロジェクトの WebContent 内に total-out.jsp を作成しましょう。

---

(total-out.jsp)

JSP をプログラミングしましょう。

<body>タグ内を以下のように書き換えてみましょう。

---

```
<%@page errorPage="total-error.jsp" %>
<%
request.setCharacterEncoding("UTF-8");
int price=Integer.parseInt(request.getParameter("price"));
int count=Integer.parseInt(request.getParameter("count"));
int delivery=Integer.parseInt(request.getParameter("delivery"));
%>
<%=price %>円 x<%=count %>個+送料<%=delivery %>円=<a></a>
<%=price*count+delivery %>円
```

---

同じプロジェクトの WebContent 内に total-error.jsp を作成しましょう。

---



---

(total-error.jsp)

JSP をプログラミングしましょう。

<body>タグ内を以下のように書き換えてみましょう。

---

```
<%@page isErrorPage="true" %>
<p>数値を入力してください。</p>
<button onclick="history.back()">戻る</button>
<br>
<p><%=exception %></p>
<table border=1>
<tr>
  <td><strong>エラーメッセージ</strong></td>
  <td><%= exception.getMessage() %></td>
</tr>
<tr>
  <td><strong>例外を文字列に変換</strong></td>
  <td><%= exception.toString() %></td>
</tr>
<tr>
  <td><strong>スタックトレース</strong></td>
  <td>
<%
    exception.printStackTrace(new java.io.PrintWriter(out));
%>
</td></tr>
</table>
```

---

JSP を実行してみましょう。

---