サーバーサイドJava

1.サーブレットとJSP

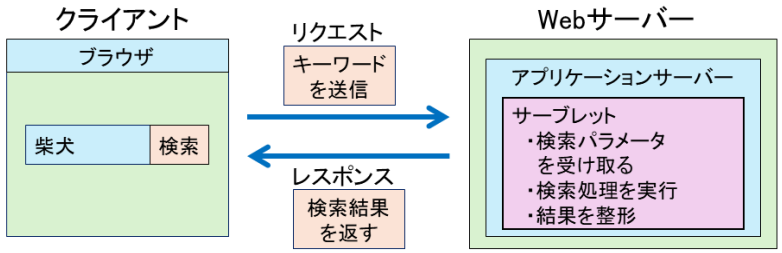
1

サーブレット

- サーブレット(Servlet)は「サーバー上で動作する Javaプログラム」を指す。
- JavaScriptといったブラウザで実行されるプログラムとは違い、サーブレットはサーバー上で動作する。
- サーブレットには次のような特徴がある:
  - 動的な処理
    サーブレットはプログラムの処理を行い動的な結果を返す
  - Webを対象
    サーブレットは通常、ブラウザなどのWebクライアントに対するサービスを提供する
  - アプリケーション・サーバー
    サーブレットはアプリケーションサーバー上で動作する

2

サーブレット



3

JSP

- JSP(Jakarta Server Pages)は、HTMLページを出力するためのサーブレットの拡張技術。
- サーブレットは動的なHTMLを作成できるが、サーブレットのソースコード中にHTMLのタグを記述していくのは面倒なことも多い。
- JSPはHTMLの中にJavaのコードを埋め込んで記述するもので、HTMLページの出力がサーブレットよりも容易にできる。次頁にHTMLページ表示のコード例を示す。
- HTMLページの表示に関してはJSPのほうが簡単に行うことができるが、複雑な処理はサーブレットのほうが向いている。

4

JSP

サーブレットのコード例

```
public class MyServlet
                extends HttpServlet{
  protected void doGet(…)…{
    PrintWriter out =
                  response.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("時刻:" + new Date());
    out.println("</body>");
    out.println("</html>");
}
```

JSPのコード例

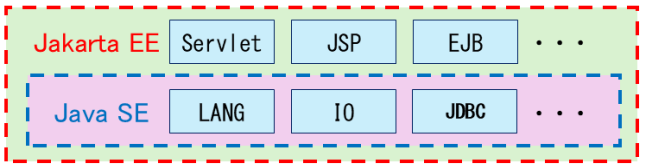
```
<html>
<body>
時刻:<%= new Date() %>
</body>
</html>
```


JSP

- サーブレットは実行するまでに「コンパイルを行い、クラスファイルを決まった場所に配置する」という作業が必要となる。
- 一方、JSPの場合はコンパイルを行う必要がない。正確には裏で自動的にコンパイルが行われる。
- サーブレットとJSPは別々に利用することができるが、組み合わせて利用することもできる。組み合わせる場合には処理(ロジック)はサーブレット、表示(レイアウト)はJSPという具合に役割分担をする。

Jakarta EE

- Jakarta EE(旧Java EE)は、企業向けのJavaのエディション(種類)のこと。主にサーバーサイドで、高信頼やパフォーマンスが求められる場面での利用を想定したAPIが提供されている。
- サーブレットとJSPは、Jakarta EEの一部で、Java SE拡張機能の形で提供される。



- Jakarta EEのアプリケーションの開発、実行にはアプリケーション・サーバーを使う必要がある。

サーバーサイドJava

2. Servletの利用

Servletの作成

●Servletを使うには、ソースファイルをコンパイルしてクラスファイルを作成し、サーバーで設定した場所におく必要がある。

●ソースファイルの作成には次のことが必要になる。

① 必要なインポート
- java.ioパッケージ
- jakarta.servletパッケージ
- jakarta.servlet.httpパッケージ

② HttpServletクラスの継承
HttpServletはサーブレットクラスの基本的な機能(クライアントからの要求受信・応答送信・処理振り分けなど)を提供する。

③ 例外処理
IOException, ServletException など

Servletの作成

● Servletの雛形は次のようになる。

```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
public class クラス名 extends HttpServlet {
  protected void doXXX(HttpServletRequest request,
                       HttpServletResponse response)
      throws ServletException,IOException {
       //処理
  }
}
```

インポート宣言

HttpServletの継承

Servletの作成

●HttpServletクラスを継承したクラスでは、通常、次のメソッドの少なくとも1つをオーバーライドする。

➢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
- ■ページ内のリンク(<a>タグ),<img>タグなどを介して呼び出された場合
- ■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
(methodオプションはデフォルトでGETと見なされる)

➢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

Servletの作成

●doGet(),doPost()メソッドでは引数にHttpServletRequestとHttpServletResponseをとる。

```
doXXX(HttpServletRequest request,
                  HttpServletResponse response)
        throws ServletException, IOException
```

➢HttpServletRequest
クライアントのリクエストに含まれている情報を Servlet に提供するオブジェクトを定義

➢HttpServletResponse
Servlet がクライアントに送り返すレスポンスをラップするオブジェクトを定義

Tomcatプラグイン

- サーブレットはアプリケーションサーバー上で動作する。
- ここではアプリケーションサーバーとしてオープンソースのTomcatを使用する。
- EclipseにはTomcatプラグインがあり、EclipseからTomcatを利用して開発を行うことができる。
  → 今回使用しているEclipseにはインストール済み

Servletの作成

- Webで作成した動的WebプロジェクトにServletを作成する。
- ここでは、ページに「こんにちは」と表示するだけのものを作成する。EclipseでServletを新規作成すると、通常のJavaのクラス作成時と同じように、雛形を作成するので、必要な処理を追加していけばよい。
- URL指定のWebページ要求は、HTTPプロトコルのGETリクエストとなる。このためServlet内では doGet()メソッドをオーバライドして必要な処理を記述する。

Servletの作成

- doGet()メソッドでは次のような処理を行う
  ① ContentTypeにデータの種類と文字コードを設定する。

```
response.setContentType("text/html; charset=UTF-8");
```

  ② getWriter()メソッドから、ページの出力用のPrintWriterストリームを取得する。

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

  ③ ②で取得したストリームにページソースを出力する。

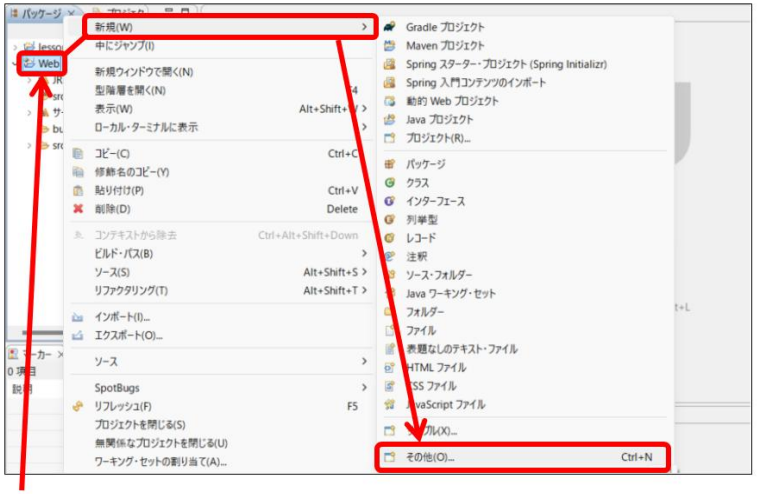
```
out.println("<html><body>…</body></html>");
```


Servletの作成

- Servletの雛形は作成されているので必要なコードを付け足す。
  → java.io.PrintWriter をインポートする必要がある！

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                  throws ServletException, IOException {
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html>"
              +  "<body>"
              +  "<h1>こんにちは</h1>"
              +  "</body>"
              +  "</html>");
}
```

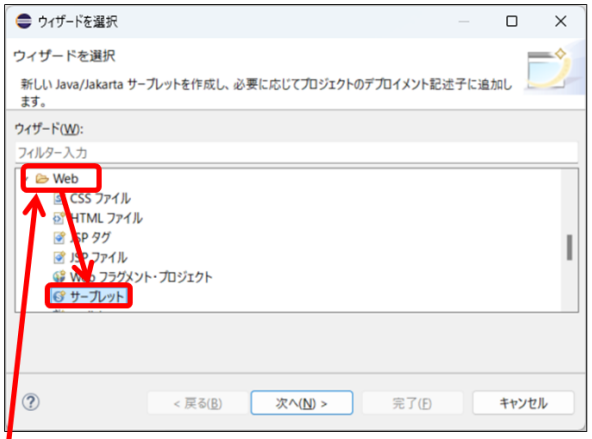
Servletの作成



プロジェクト名のWebを右クリックして「新規 → その他」を選択

17

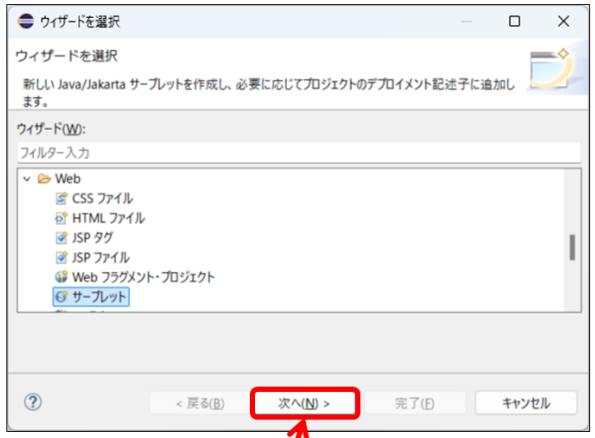
Servletの作成



Webを展開して「サーブレット」を選択

18

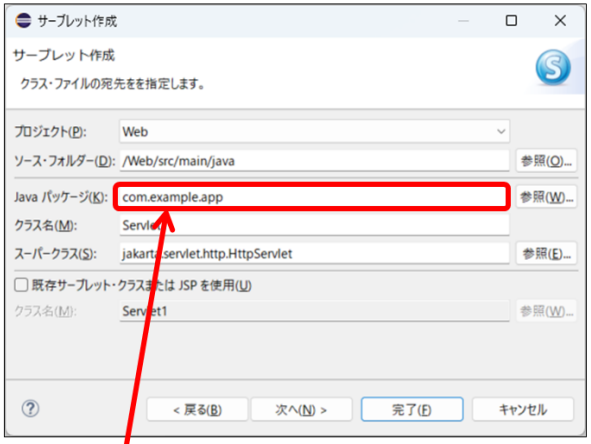
Servletの作成



「次へ」をクリック

19

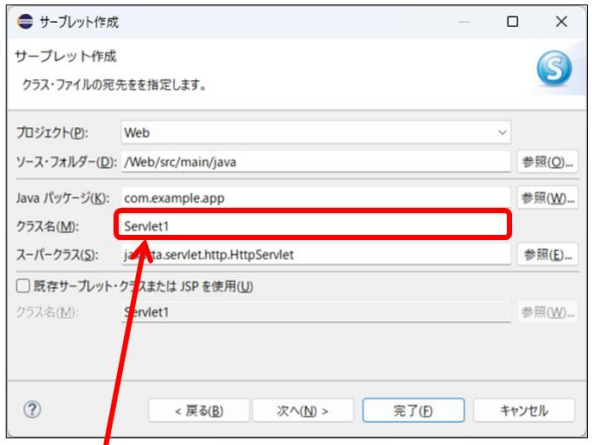
Servletの作成



Javaパッケージに「com.example.app」と入力

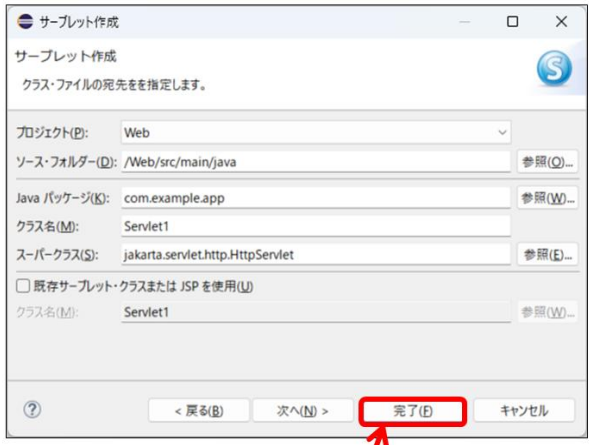
20

Servletの作成



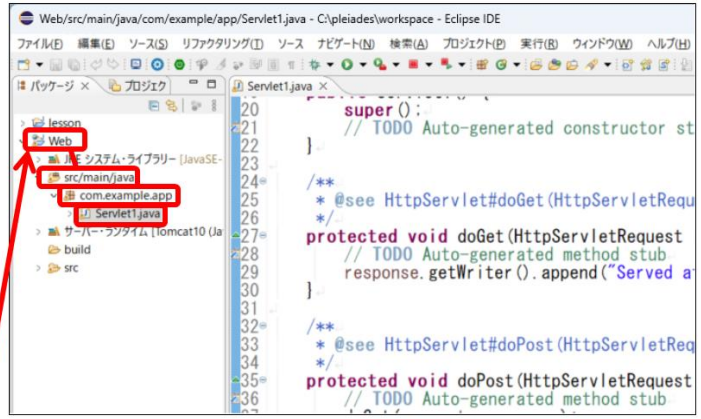
クラス名に「Servlet1」と入力

Servletの作成



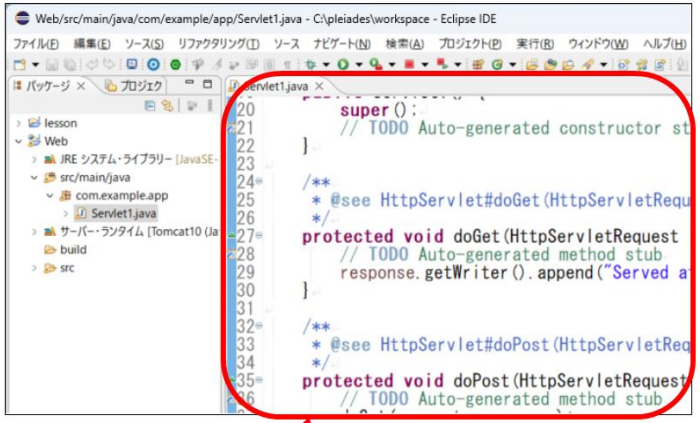
「完了」をクリック

Servletの作成



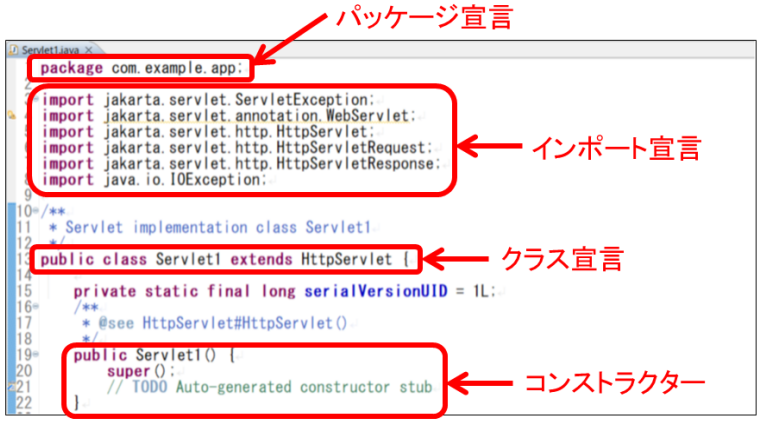
「Web → src/main/java → com.example.app」に
Servlet1.javaが作成される

Servletの作成



作成されたファイルにはソースの雛形が入力されている

Servletの作成[生成されたソースの内容1/2]




```
package com.example.app;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Servlet implementation class Servlet1
 */
public class Servlet1 extends HttpServlet {

    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servlet1() {
        super();
        // TODO Auto-generated constructor stub
    }
```


Servletの作成[生成されたソースの内容2/2]


```
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```


Servletの作成[doGet()メソッドを変更]

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, Ht
 */
protected void doGet(HttpServletRequest request, HttpSe
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html>"
              + "<body>"
              + "<h1>こんにちは</h1>"
              + "</body>"
              + "</html>");
}
```

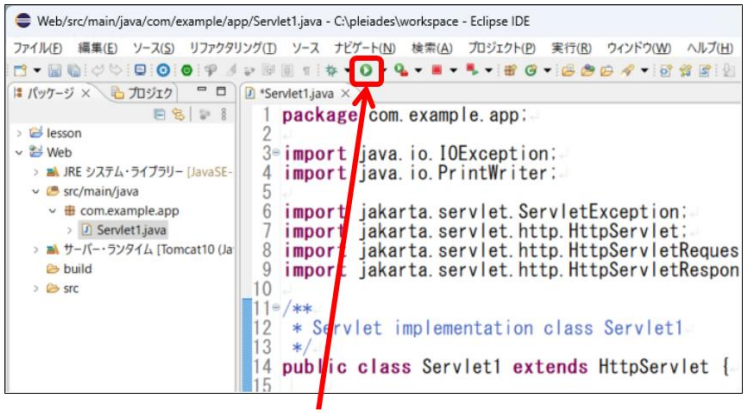
doGet()メソッドの中身を変更
→ URL指定のアクセスではリクエスト方法にGETが使われ、doGet()メソッドがコールされるので、doGet()メソッドに処理を記述

Servletの作成[import文を追加]

```
import java.io.PrintWriter;
```

PrintWriterクラスを使うためにimport文が必要となるので
[Ctrl]+[Shift]+[o]キーを同時に押して追加する！

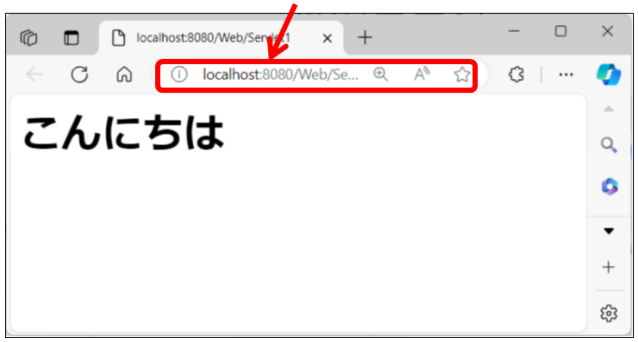
Servletの作成



実行ボタン「 」をクリックして実行

Servletの実行

http://localhost:8080/Web/Servlet1



Servletで書き込んだソースにより「こんにちは」と表示される

Servletの変更

- Tomcatは一度動作させたServlet/JSPをキャッシュに保存しており、二度目以降の起動ではそれを利用する(設定やバージョンにもよる)。ソースを更新して新しいものを動かすにはTomcatを再起動する。
- EclipseはServletやJSPを変更した場合に、それを反映させるためにtomcatの再起動が必要となることを判断して、その都度警告ダイアログを表示する。この場合は、「OK」をクリックして再起動する。

フォームの利用

- フォームから送信された値をServletで処理をすることを考える。
- Servletでコントロールの値を取得するには(doGet()やdoPost()の第1引数の)HttpServletRequestオブジェクトのgetParameter()メソッドを使う。引数には、フォーム内のコントロール名(name属性で指定した名前)を指定する。戻り値は、コントロールで指定された値(String型)となる。もし、該当するコントロール名がない場合はnullを返す。

```
String param = request.getParameter(コントロール名)
```

(チェックボックスなど)複数の値を返すコントロールの場合にはgetParameterValues()メソッドを使う。

```
String[] params =
        request.getParameterValues(コントロール名)
```

フォームの利用

HTML

```
<form action="送信先URL" method="GET">
  名前<input type="text" name="abc">
  <input type="submit" value="送信">
</form>
```

HTMLの表示

名前 山田 送信

コントロール名がabcなので、getParameter()の引数にはabcを指定する

methodがGETなのでdoGet()をコール

Servlet

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                        throws ServletException, IOException{
    String value = request.getParameter("abc");
```

「山田」が代入される

フォームの利用

●次の例では、最初のページで名前を入力して送信し、その応答として挨拶ページを表示する。ここではフォームのmethodにはGETを指定している。このためサーブレット内の処理はdoGetメソッドで定義する。

名前と年齢を入力して「送信」ボタンを押す

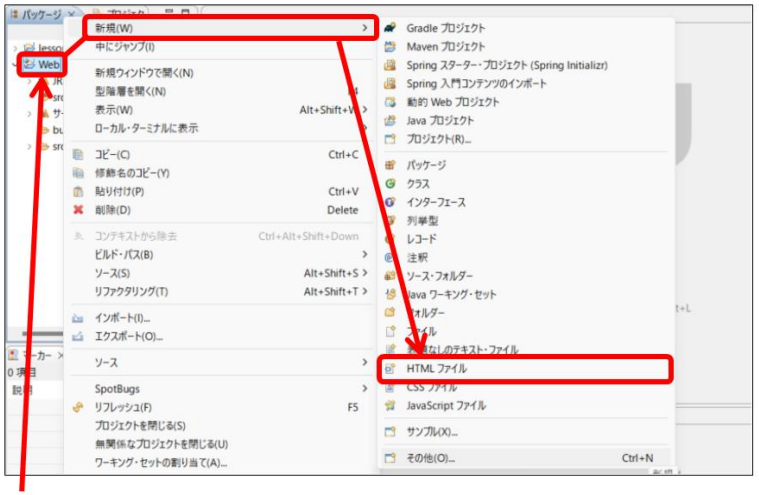
入力された名前と年齢を表示

HTMLの作成

Servlet2.html

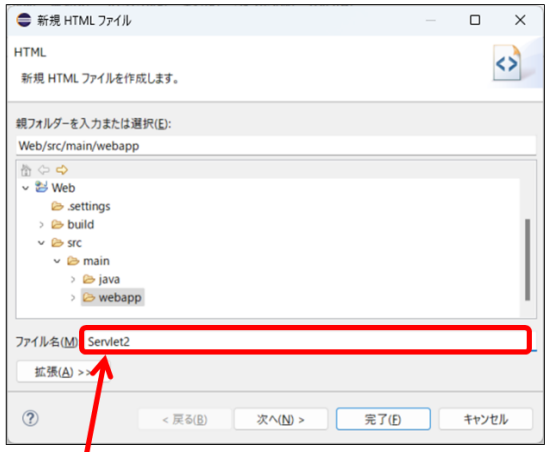
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Servlet2</title>
</head>
<body>
    <form action="Servlet2" method="GET">
        名前<input type="text" name="name"><br>
        年齢<input type="number" name="age"><br>
        <input type="submit" value="送信">
    </form>
</body>
</html>
```


HTMLの作成



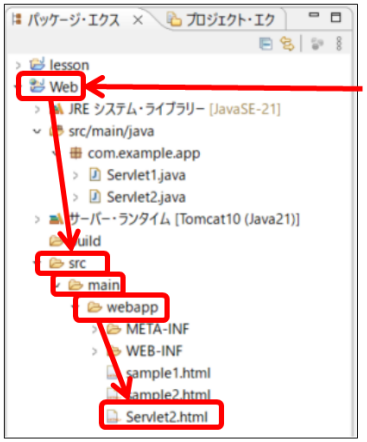
プロジェクト名のWebを右クリックして「新規 → HTMLファイル」を選択

HTMLの作成



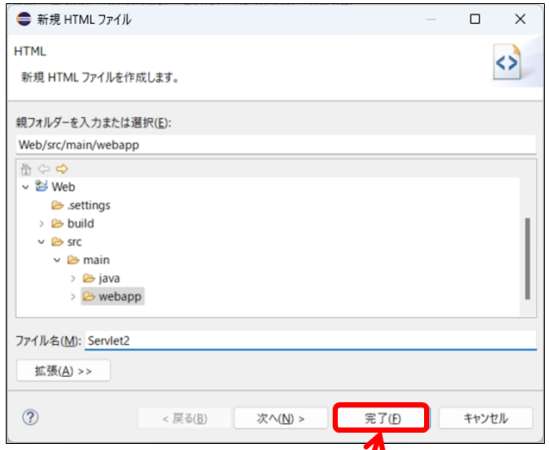
ファイル名に「Servlet2」と入力

HTMLの作成



「Web → src
→ main → webapp」に
Servlet2.htmlが作成される

HTMLの作成



「完了」をクリック

HTMLの作成

```
Servlet2.html
 1 <!DOCTYPE html>
 2 <html>
 3 <head>
 4     <meta charset="UTF-8">
 5     <title>Servlet2</title>
 6 </head>
 7
 8 <body>
 9     <form action="Servlet2" method="GET">
10         名前<input type="text" name="name"><br>
11         年齢<input type="number" name="age"><br>
12         <input type="submit" value="送信">
13     </form>
14 </body>
15 </html>
```

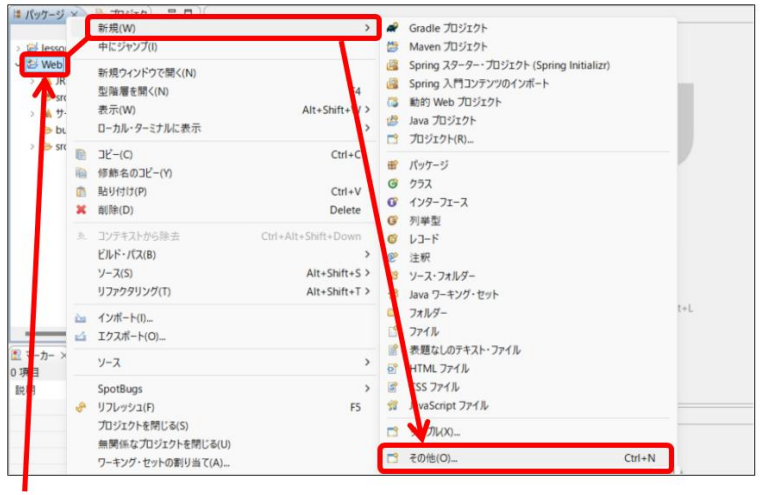
ソースの雛形が作成されるのでコードを入力する！

Servletの作成

Servlet2.java[doGet内を編集]

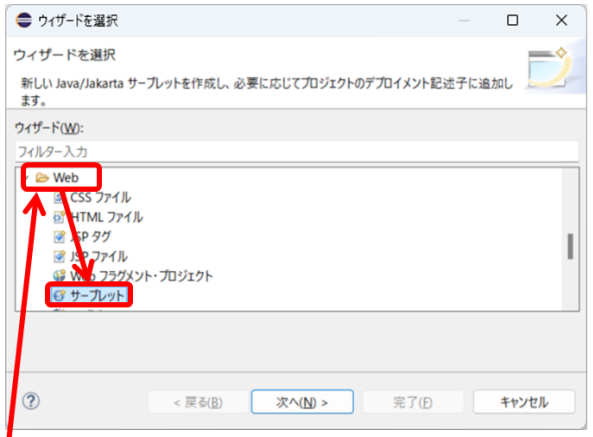
```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                       throws ServletException, IOException{
      String name = request.getParameter("name");
      String age  = request.getParameter("age");
      response.setContentType("text/html; charset=UTF-8");
      PrintWriter out = response.getWriter();
      out.println("<html><body>"
                + "名前：" + name
                + "<br>"
                + "年齢：" + age
                + "</body></html>");
}
```


Servletの作成



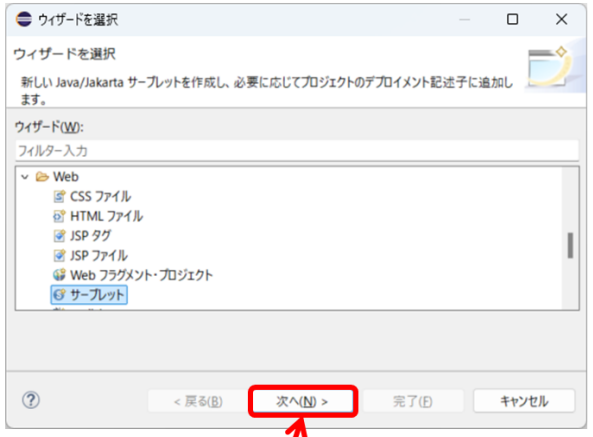
プロジェクト名のWebを右クリックして「新規 → その他」を選択

Servletの作成



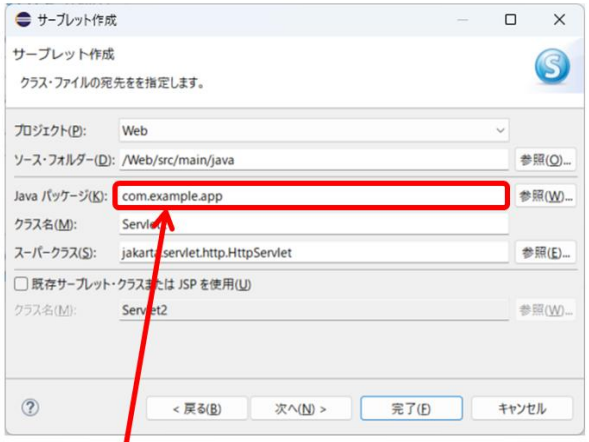
Webを展開して「サーブレット」を選択

Servletの作成



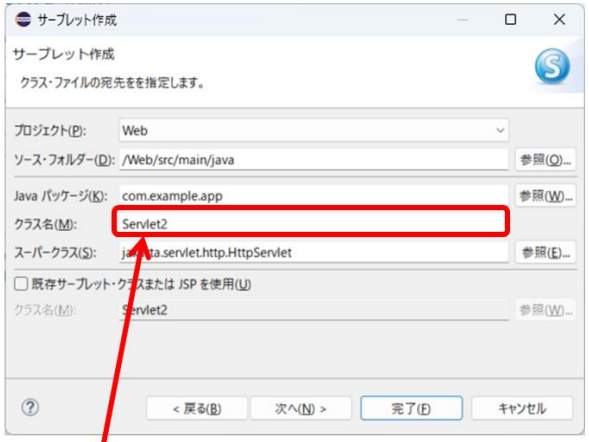
「次へ」をクリック

Servletの作成



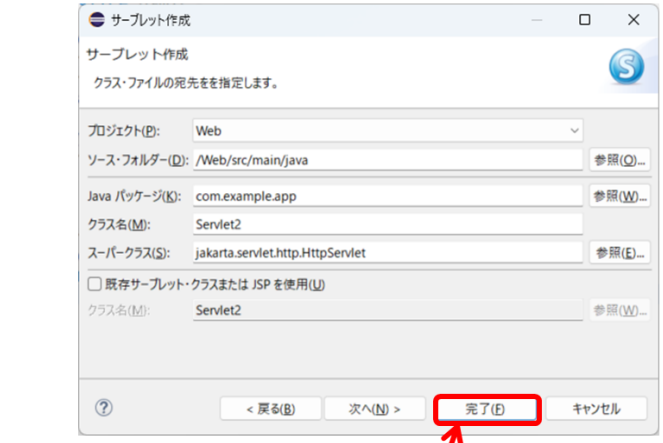
Javaパッケージに「com.example.app」と入力

Servletの作成



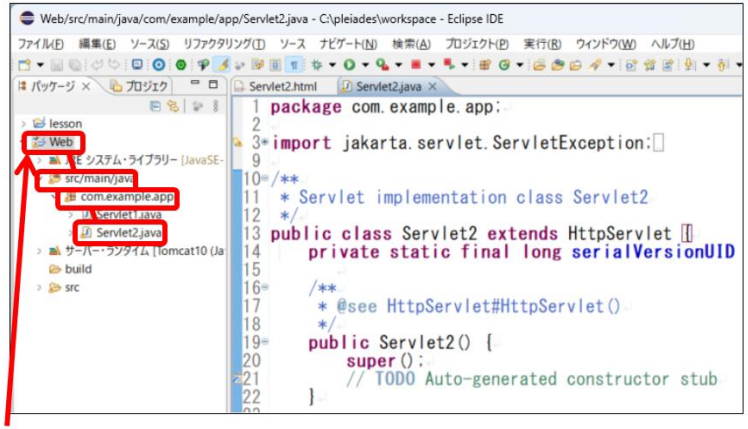
クラス名に「Servlet2」と入力

Servletの作成



「完了」をクリック

Servletの作成



「Web → src/main/java → com.example.app」に
Servlet2.javaが作成される

Servletの作成[doGet()メソッドを変更]

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpS
 */
protected void doGet(HttpServletRequest request, HttpServl
    String name = request.getParameter("name");
    String age  = request.getParameter("age");
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html><body>"
              + "名前：" + name
              + "<br>"
              + "年齢：" + age
              + "</body></html>");
}
```

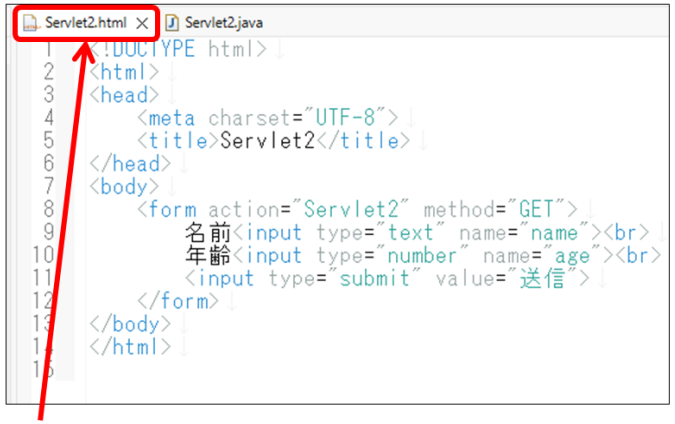
doGet()メソッドの中身を変更
→ URL指定のアクセスではリクエスト方法に
GETが使われ、doGet()メソッドがコールされる
ので、doGet()メソッドに処理を記述

Servletの作成[import文を追加]

```
import java.io.PrintWriter;
```

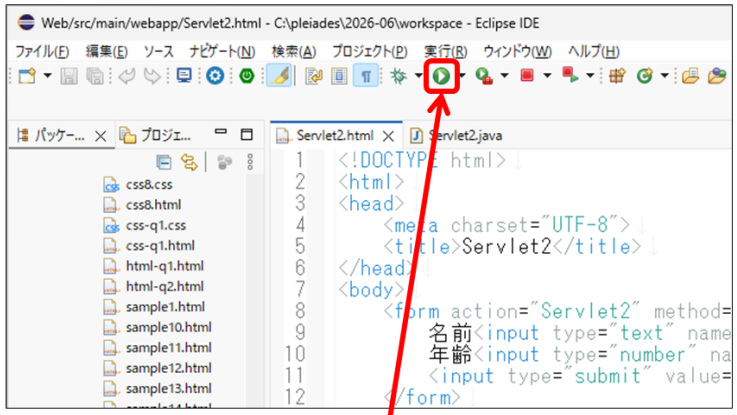
PrintWriterクラスを使うためにimport文が必要となるので
[Ctrl]＋[Shift]＋[o] キーを同時に押して追加する！

実行



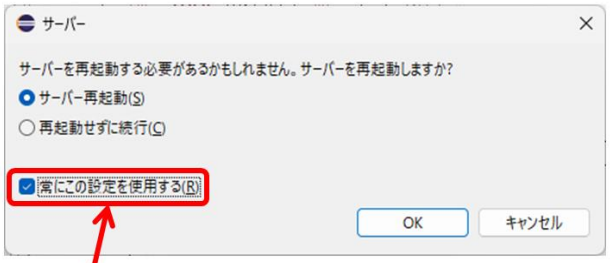
Servlet2.htmlをクリック

実行



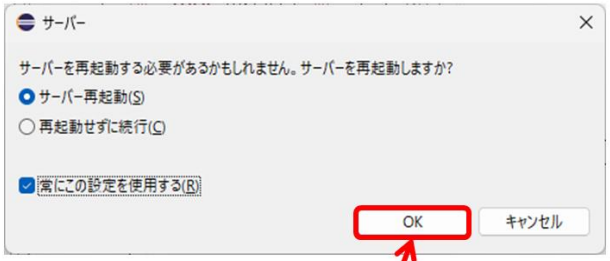
実行ボタン「 」をクリックして実行

実行



「常にこの設定を使用する」にチェック

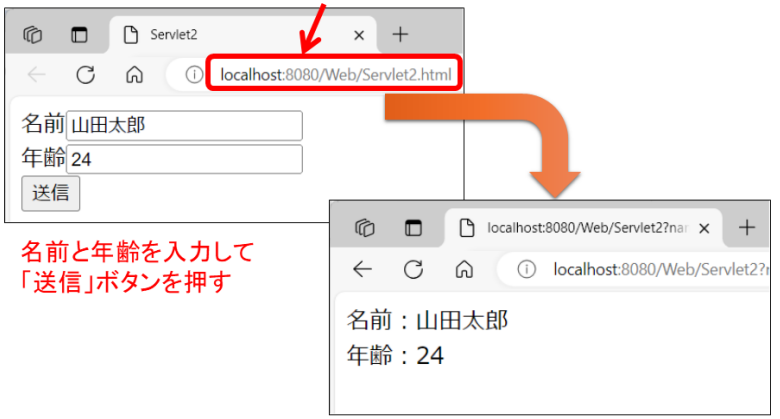
実行



「OK」をクリック

実行

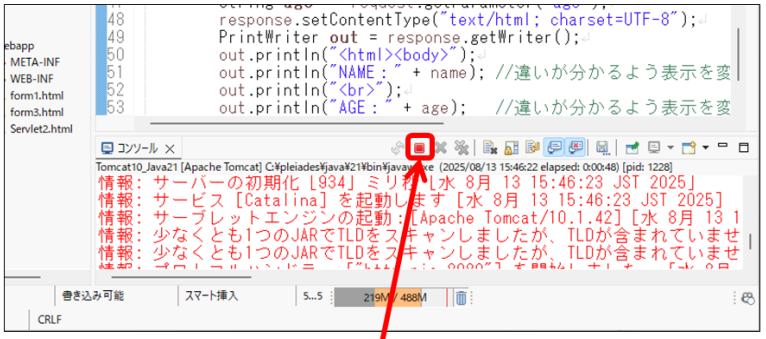
http://localhost:8080/Web/Servlet2.html



名前と年齢を入力して
「送信」ボタンを押す

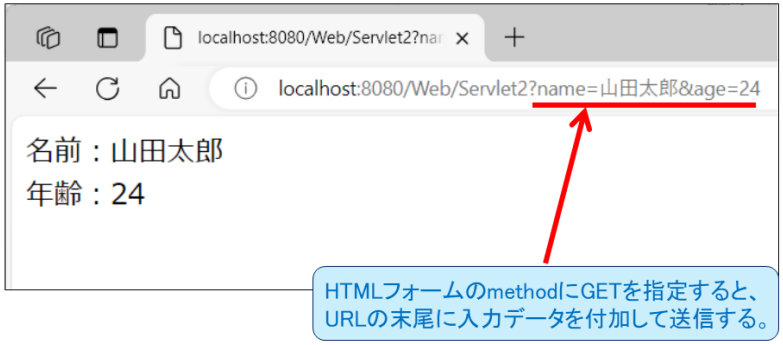
入力された名前と年齢を表示

(参考)接続先が見つからないときは サーバーを一旦停止して実行する



コンソールの停止ボタン「■」でサーバーを止める

実行



参考.日本語などのデータは実際にはパーセントエンコーディングを使って
符号化されて送信される

POSTとGET

- HTTPはクライアントとサーバ間で行われる通信の規約を定めている。クライアントからサーバに送るデータをリクエストと言い、サーバからクライアントに送るデータをレスポンスと言う。
- 代表的なリクエスト方法にはGETとPOSTがある。これらは次の場合に使われる。
  - GET
    - ブラウザから直接にURL指定で呼び出された場合
    - ページ内のリンクで呼び出された場合
    - HTMLフォームでGETメソッド指定された場合
  - POST
    - HTMLフォームでPOSTメソッド指定された場合

POSTとGET

- リクエスト方法のGETとPOSTの処理に関して次のような違いがある。
  - GET
    フォームのデータをURLの末尾に追加して送信する。送信できるデータ量には制限がある。
  - POST
    フォームのデータのみを送信する。一度に 大量のデータを送信することができる。
- HTMLでフォームを作るformタグでは、属性methodを使ってGETまたはPOSTのリクエスト方法を指定できる(指定しない場合はGETになる)。

GETからPOSTへの変更

- フォームのサンプルにおいて送信方法をGETからPOSTに変更する。
- Servlet2.htmlにおいて、formタグ内のmethod属性にPOSTを指定する。
- また、サーブレットServlet2のdoGet()メソッドの中身をdoPost()メソッドの中にコピーして作成する。

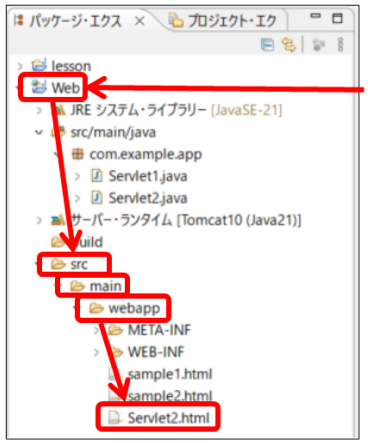
GETからPOSTへの変更（Servlet2.htmlを編集）

Servlet2.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Servlet2</title>
</head>
<body>
    <form action="Servlet2" method="POST">
        名前<input type="text" name="name"><br>
        年齢<input type="number" name="age"><br>
        <input type="submit" value="送信">
    </form>
</body>
</html>
```

GETをPOSTに変更

GETからPOSTへの変更（Servlet2.htmlを編集）



「Web → src → main → webapp」にあるServlet2.htmlをダブルクリックして開く

GETからPOSTへの変更（Servlet2.htmlを編集）

Servlet2.htmlを編集

GETからPOSTへの変更（Servlet2.javaを編集）

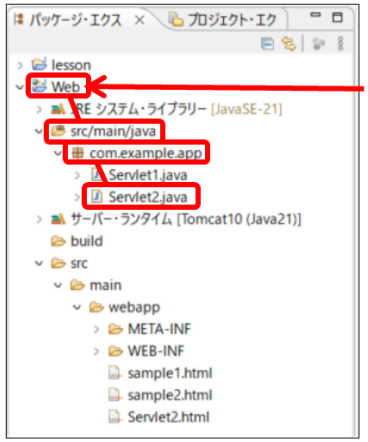
Servlet2.java[doPostメソッド内を編集]

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                         throws ServletException, IOException{

    String name = request.getParameter("name");
    String age  = request.getParameter("age");
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html><body>"
              + "NAME：" + name   //違いが分かるよう表示を変更
              + "<br>"
              + "AGE：" + age     //違いが分かるよう表示を変更
              + "</body></html>");
}
```

doGet()メソッドの内容をコピー

GETからPOSTへの変更（Servlet2.javaを編集）



「Web → src/main/java
→ com.example.app」
にあるServlet2.javaを
ダブルクリックして開く

GETからPOSTへの変更（Servlet2.javaを編集）

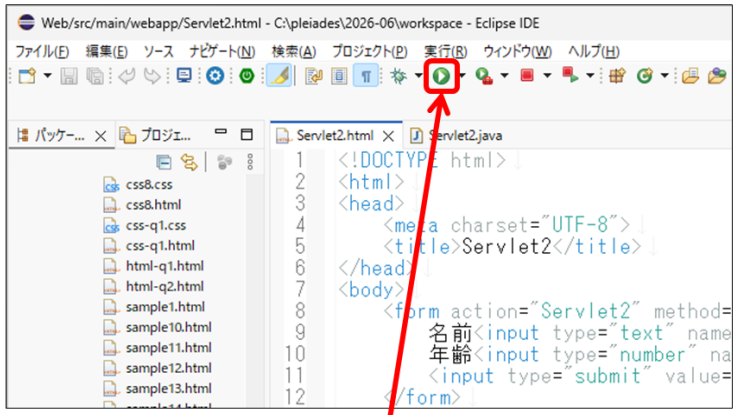
```
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServ
 */
protected void doPost(HttpServletRequest request, HttpServletR
    String name = request.getParameter("name");
     String age  = request.getParameter("age");
     response.setContentType("text/html; charset=UTF-8");
     PrintWriter out = response.getWriter();
     out.println("<html><body>"
                + "NAME : " + name  //違いが分かるよう表示を変更
                + "<br>"
                + "AGE : " + age    //違いが分かるよう表示を変更
                + "</body></html>");
}
```

Servlet2.javaを編集

実行

Servlet2.htmlをクリック

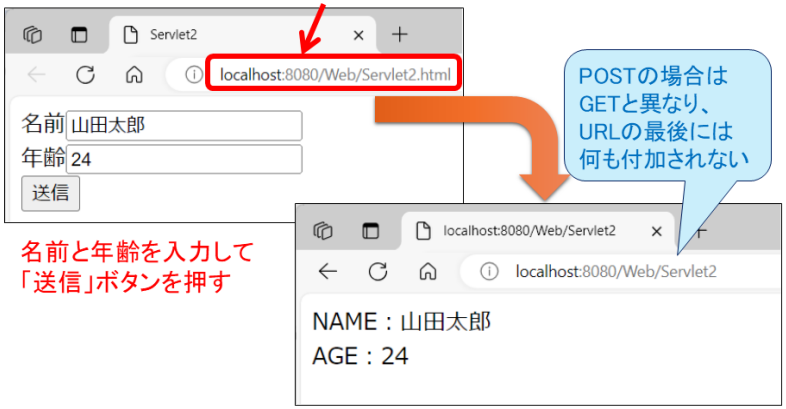
実行



実行ボタン「 」をクリックして実行

実行

http://localhost:8080/Web/Servlet2.html



名前と年齢を入力して
「送信」ボタンを押す

入力された名前と年齢を表示

サーバーサイドJava

3. JSPの利用

JSP

- Servletが「サーバサイドのプログラム」であったのに対し、JSP(Jakarta Server Pages)はサーバで動作するページを表す。
- JSPはHTMLのソースをベースとして<%～%>でJavaの断片的なコードを埋め込むことができる。
- JSPはコンパイルを必要としない。JSPは「jsp」ファイルを、サーバー指定の場所に配置すれば動作する。
- JSPの作成は通常のHTMLを書く作業とほとんど変わらないが、実際にはJSPがサーブレットにコンパイルされて動作する。

JSPの式

- JSPの式は、式の評価結果(=値)を出力中に挿入するために使う。式を指定するJSPの構文は次のようになる。

```
<%= 式 %>
```

式が評価され結果が文字列に変換されてページに挿入される。

- 例えば、次のコードは、現在の日時を表示する。

```
<%= new java.util.Date() %>
```

JSPのスクリプトレット

●式ではなく、もっと複雑なことをするときにはJSPのスクリプトレットを使って、任意のコードを記述する。構文は次のようになる。

```
<% Javaのコード %>
```

●例えば、次のコードは＊を三角形状に表示する。

```
<% for(int row = 1 ; row <= 5 ; row++ ){
      for(int col = 1 ; col <= row ; col++ ){
         out.print("* ");
      }
      out.print("<br>");
   }
 %>
```


JSPの宣言

●JSPの宣言を使ってメソッドやフィールドを定義できる。構文は次のようになる。

```
<%! Javaのコード %>
```

宣言そのものは出力を何もしないので、宣言は通常、JSPの式やスクリプトレットと一緒に使われる。

●次の例は、変数maxを宣言し、その後で表示に使っている。

```
<%! int max = 5; %>
<h2>サイズ<%= max %>の三角形を描きます。</h2>
```


JSPの定義済み変数

●JSPには定義済み変数があり、すぐにオブジェクトが利用できる(このオブジェクトは暗黙オブジェクトと呼ばれる)。代表的なものに次がある。

- request
  リクエストを表すHttpServletRequestオブジェクト(Servletのdoget(),doPost()の引数requestと同じ情報)
- response
  レスポンスを表すHttpServletResponseオブジェクト(Servletのdoget(),doPost()の引数responseと同じ情報)
- out
  クライアントにテキストを送るためのJspWriterオブジェクト(出力のためのprint(), println()メソッドが使える)

JSPのディレクティブ

●JSPでは <%@ ～ %> で括られたディレクティブ(制御文)を用いて、JSPページから作られるサーブレットの全体的な構造を制御する。

●EclipseでJSPを作成すると次のディレクティブが自動挿入される。

```
<%@ page language="java"
         contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
```

ここで、language属性は、式・スクリプトレット・宣言を記述する言語を指定し、contentType属性はクライアントに送るドキュメントの種類と文字コードを指定し、pageEncoding属性は、JSPファイルを記述している文字コードを指定する。

JSPのディレクティブ

- ディレクティブを用いて、Javaクラスのインポートができる。また、ワイルドカードインポートも可能である。

```
<%@ page import="パッケージ.クラス" %>
```

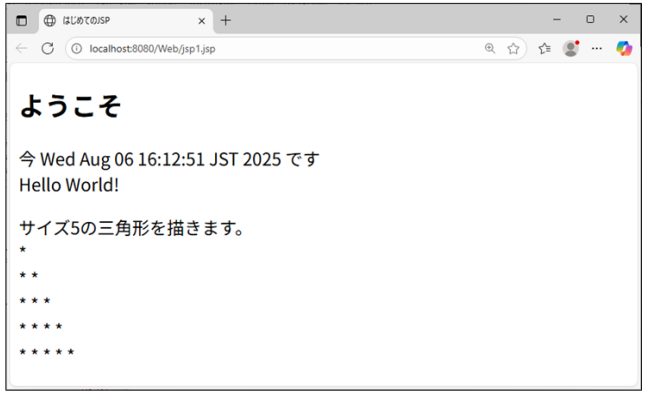
```
<%@ page
  import="パッケージ1.クラス1,パッケージ2.*,…" %>
```

- includeディレクティブは、これを記述した位置に他のHTMLファイルやJSPファイルなどのテキストファイルを読み込む。

```
<%@ include file="ファイル名"%>
```


JSPの作成

- JSPの例を示す。ここでは、JSPの式、スクリプトレット、宣言を使っている。



JSPの作成

jsp1.jsp(1/2)

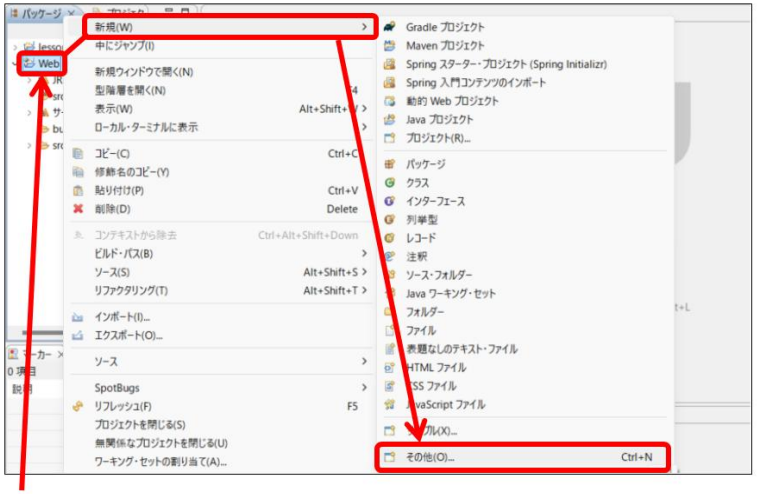
```
<%@ page language="java" contentType="text/html;charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.Date" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>はじめてのJSP</title>
</head>
<body>
<h2>ようこそ</h2>
<p>
   今 <%= new Date() %> です<br>
   <%= "Hello World!" %>
</p>
```


JSPの作成

jsp1.jsp(2/2)

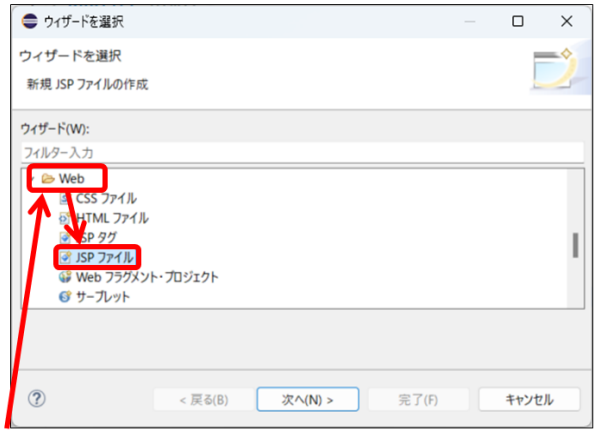
```
<p>
   <%! int max = 5; %>
   サイズ<%= max %>の三角形を描きます。<br>
   <% for(int row = 1 ; row <= max ; row++ ){
         for(int col = 1 ; col <= row ; col++ ){
            out.print("* ");
         }
         out.print("<br>");
      }
   %>
</p>
</body>
</html>
```

JSPの作成



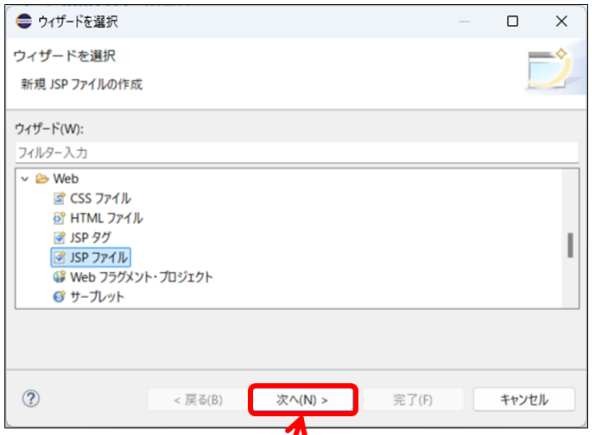
プロジェクト名のWebを右クリックして「新規 → その他」を選択

JSPの作成



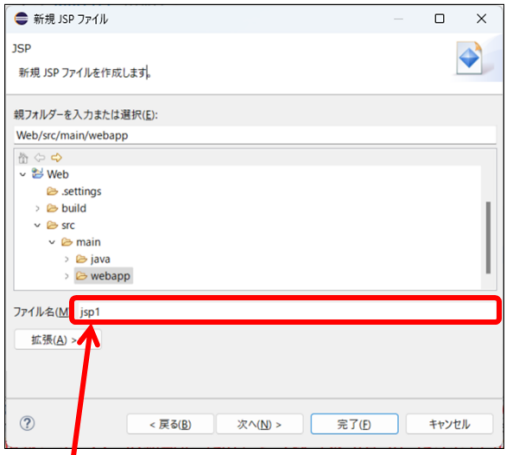
Webを展開して「JSPファイル」を選択
→ もし「JSPファイル」がない場合には資料最後の(参考)を実施する
(Pleiades 2026-06の不具合？)

JSPの作成



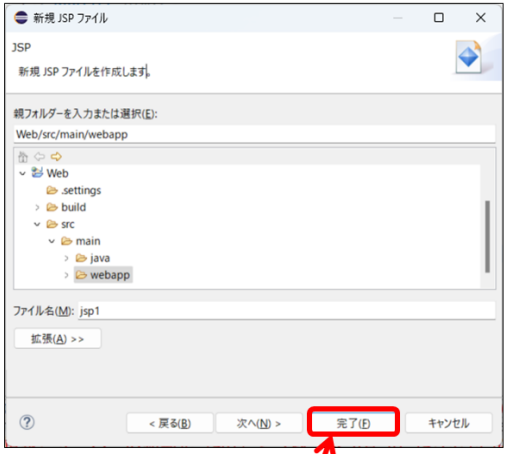
「次へ」をクリック

JSPの作成



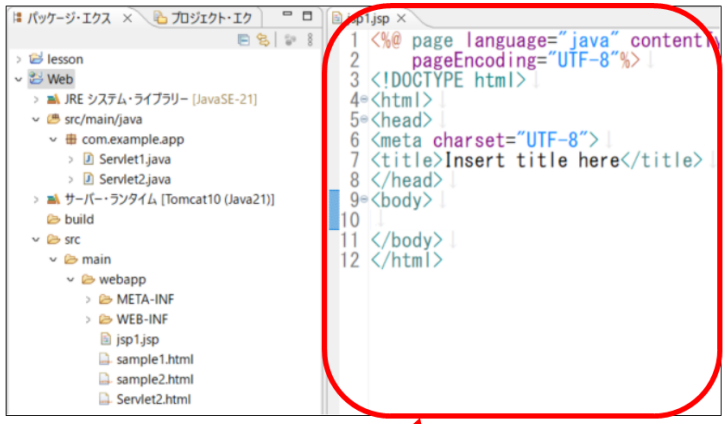
ファイル名に「jsp1」と入力

JSPの作成



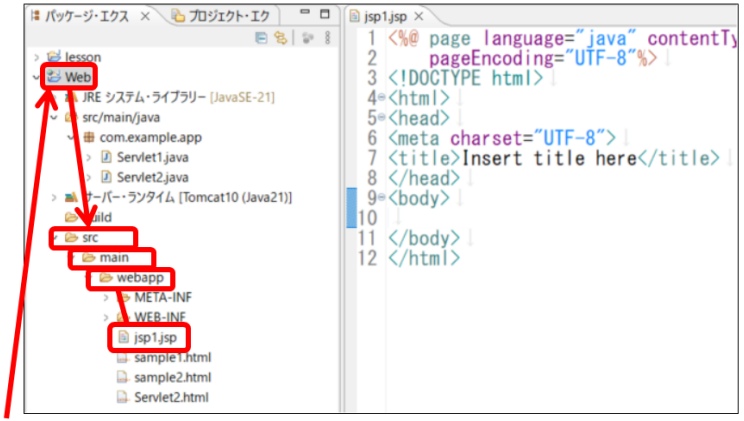
「完了」をクリック

JSPの作成



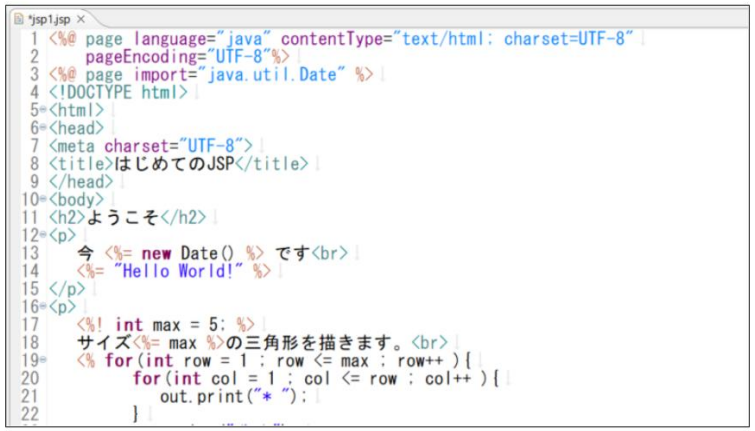
作成されたファイルにはソースの雛形が入力されている

JSPの作成



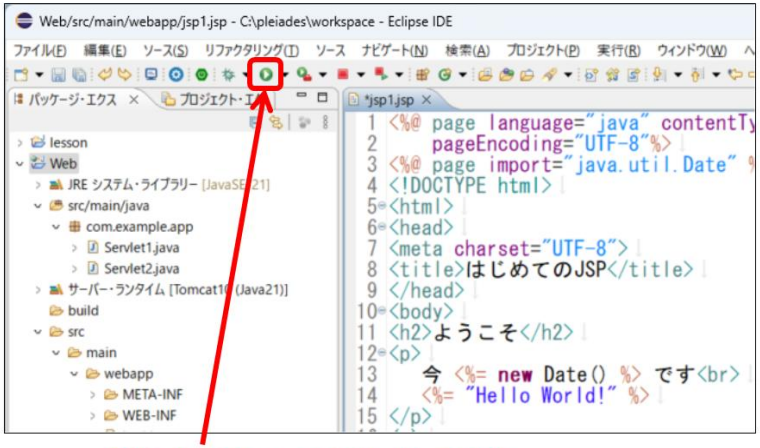
「Web → src → main → webapp」に jsp1.jsp が作成される

JSPの作成



コードを入力

JSPの実行



JSPの実行

http://localhost:8080/Web/jsp1.jsp

サーバーサイドJava

4. ServletとJSPの連携

ServletとJSPの連携

- サーブレットとJSPは別々に利用することができるが、組み合わせて利用することもできる。
- 組み合わせる場合には処理はサーブレット、表示はJSPという具合に役割分担をする。
- ここでは、サンプルとして作成した Servlet2 の表示を JSP に切り替えて動かしてみることにする。

Servletの変更

●サーブレットの処理終了後にページを遷移させるには次のように指定する。このとき、doGet() や doPost() の引数 request と response も JSP に引き渡される。

```
request.getRequestDispatcher("遷移先URL")
       .forward(request, response);
```

●サーブレットから遷移先のJSPにオブジェクト(値)を渡すことができる。その場合にはサーブレットの引数 request に setAttribute() メソッドを使ってセットする。第1引数にはデータを区別する名前(String型)を指定し、第2引数は値(Object型)を指定する。

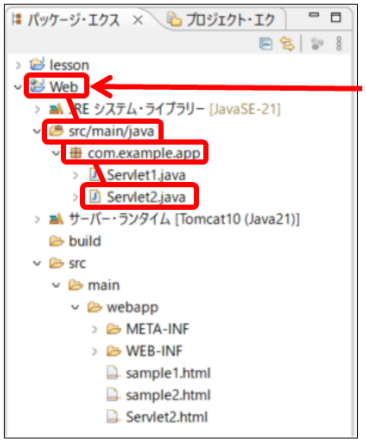
```
request.setAttribute(名前, オブジェクト)
```


Servletの変更

●JSPでの表示に切り替えるためServlet2.javaのdoPost()メソッドを次のように変更する。ここでは、名前と年齢をrequestにセットして show.jsp にフォワードしている。

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                        throws ServletException, IOException{
    String name = request.getParameter("name");
    String age = request.getParameter("age");
    request.setAttribute("value1", name + "さん");
    request.setAttribute("value2", age + "歳");
    request.getRequestDispatcher("/show.jsp")
           .forward(request, response);
}
```


Servletの変更(Servlet2.javaを編集)



「Web → src/main/java
→ com.example.app」
にあるServlet2.javaを
ダブルクリックして開く

Servletの変更(Servlet2.javaを編集)

変更部分

```
protected void doPost(HttpServletRequest request, Htt
    String name = request.getParameter("name");
    String age  = request.getParameter("age");
    request.setAttribute("value1", name + "さん");
    request.setAttribute("value2", age + "歳");
    request.getRequestDispatcher("/show.jsp")
           .forward(request, response);
}
```

Servlet2.javaのdoPost()メソッドを変更

表示用のJSP

- サーブレットからrequest.setAttribute(名前,値)メソッドで渡された値をJSPで取得するにはgetAttribute()メソッドを使って、次のようにする。ここで、値のデータ型はObjectとなっているので、必要に応じてキャストする必要がある。

```
Object value = request.getAttribute(名前)
```

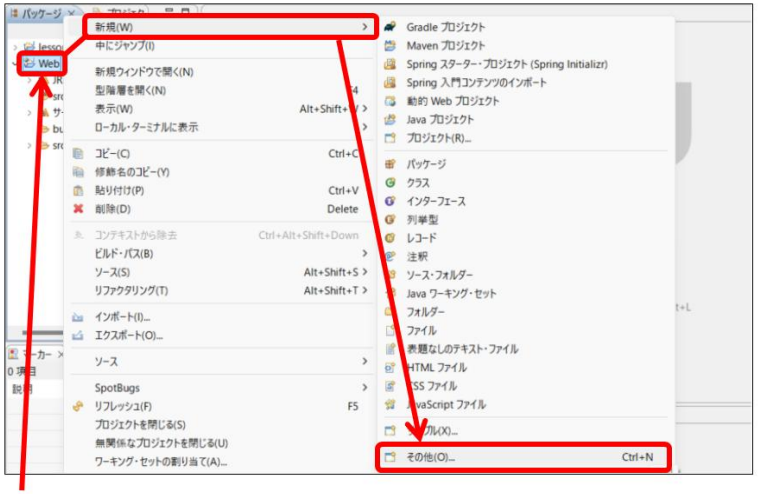
- 今回のサンプルでは、Servlet2から渡された値をそのまま表示するJSPを作成する。

表示用JSPの作成

show.jsp

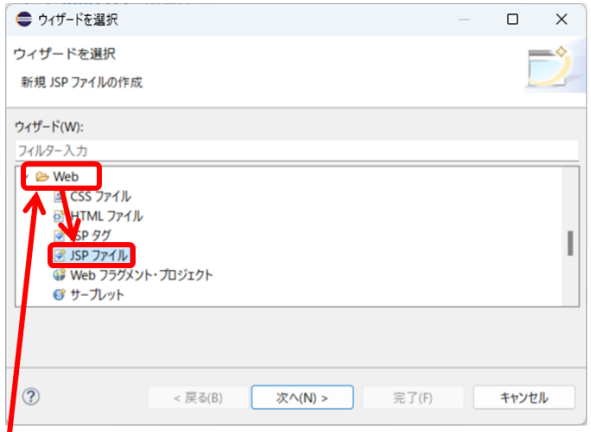
```
<%@ page language="java" contentType="text/html;charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>JSPで表示</title>
</head>
<body>
        名前：<%= request.getAttribute("value1") %>
        <br>
        年齢：<%= request.getAttribute("value2") %>
</body>
</html>
```


表示用JSPの作成



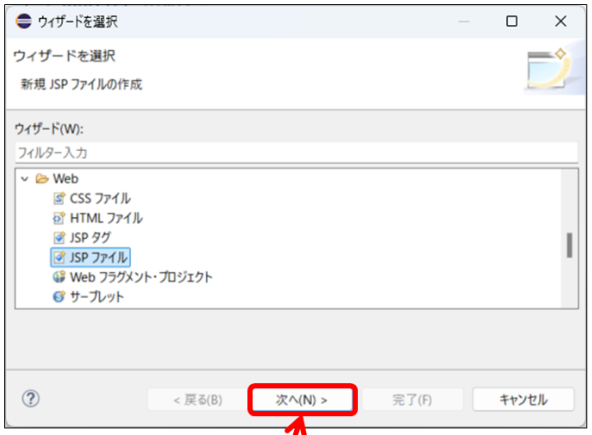
プロジェクト名のWebを右クリックして「新規 → その他」を選択

表示用JSPの作成



Webを展開して「JSP ファイル」を選択

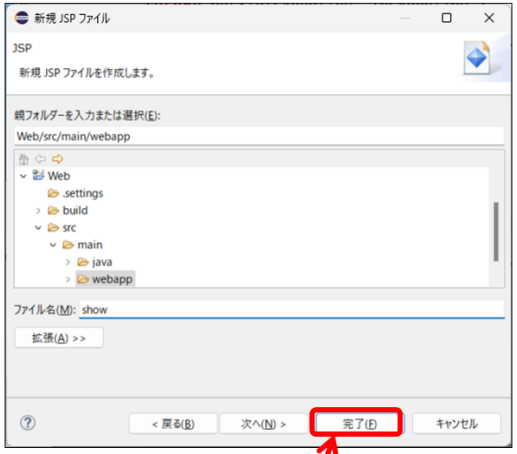
表示用JSPの作成



「次へ」をクリック

101

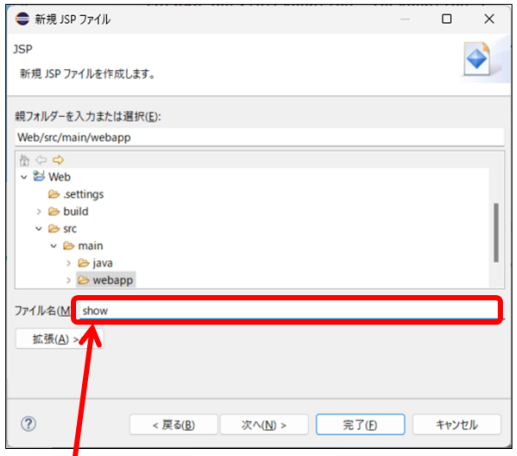
表示用JSPの作成



「完了」をクリック

103

表示用JSPの作成



ファイル名に「show」と入力

102

表示用JSPの作成

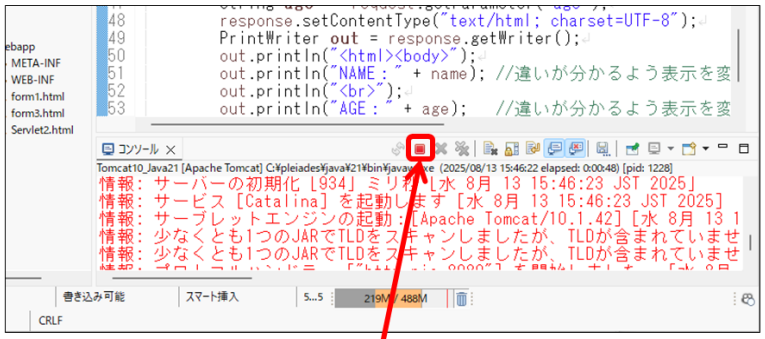
show.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSPで表示</title>
</head>
<body>
    名前：<%= request.getAttribute("value1") %>
    <br>
    年齢：<%= request.getAttribute("value2") %>
</body>
</html>
```

コードを入力

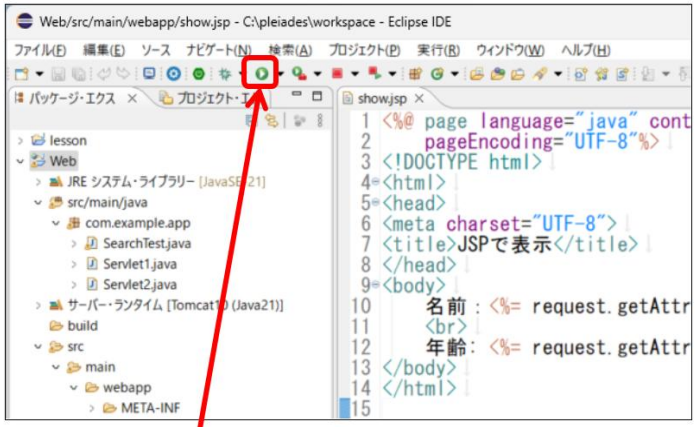
104

実行



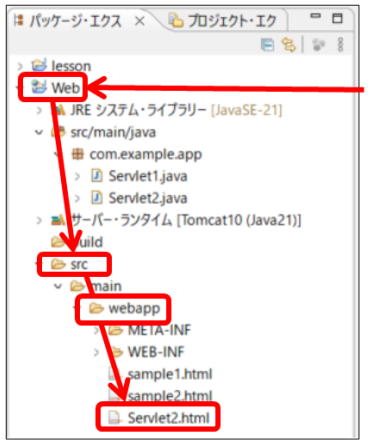
コンソールの停止ボタン「■」でサーバーを止める

実行



実行ボタン「 」をクリックして実行

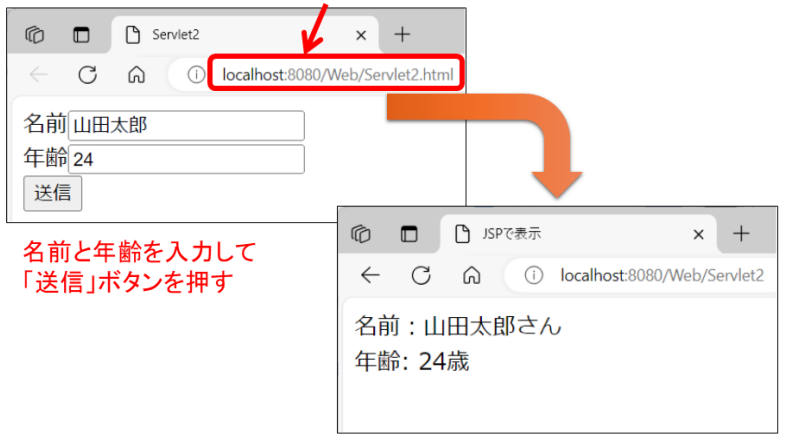
実行



「Web → src → webapp」にあるServlet2.htmlをダブルクリックして開く

実行

http://localhost:8080/Web/Servlet2.html



名前と年齢を入力して「送信」ボタンを押す

入力された名前と年齢をJSPで表示

JSPファイルが選択できないとき（参考）

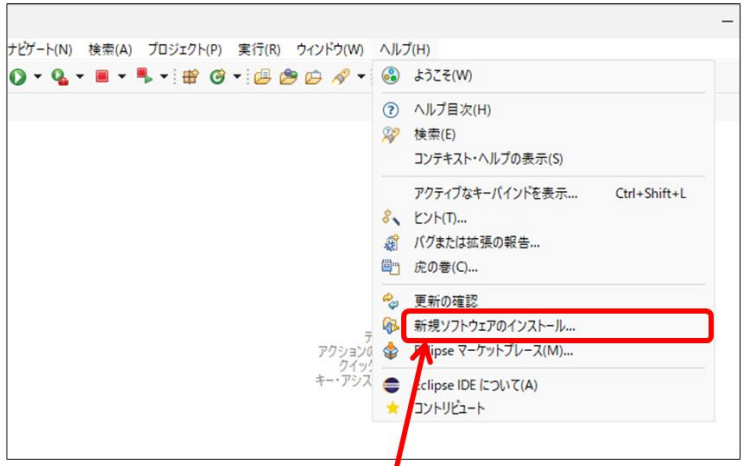
Eclipseのバージョンやプラグインによりファイルの新規作成ダイアログで「JSPファイル」が表示されないときがある。そのときにはこちらの手順を試す。

109

JSPファイルが選択できないとき

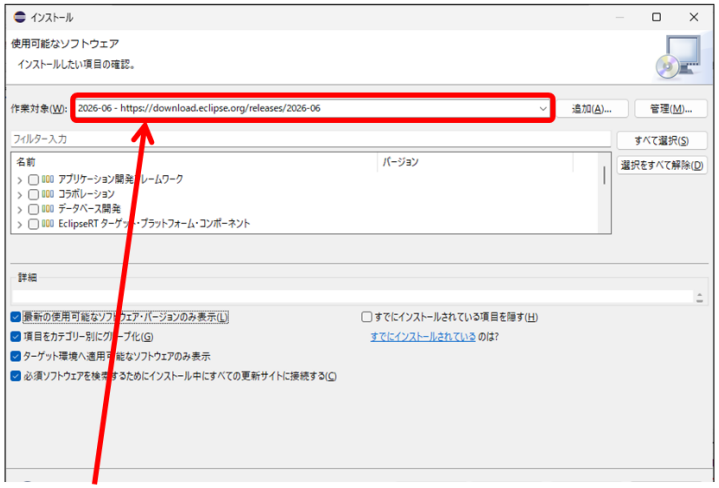
- ファイルの新規作成で「JSPファイル」が表示されないとき、使用しているEclipseが「Java標準開発向け」のパッケージになっており、Web開発用の機能（WTP: Web Tools Platform）が含まれてい場合がある。
- このときにはEclipse マーケットプレイスから「Eclipse Enterprise Java and Web Developer Tools」を探し、インストールする。

110



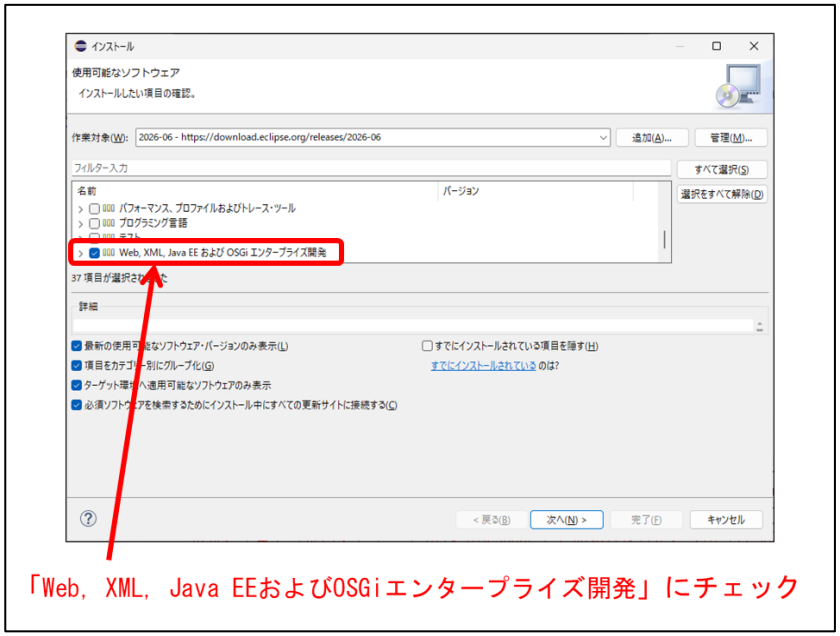
「新規ソフトウェアのインストール」をクリック

111

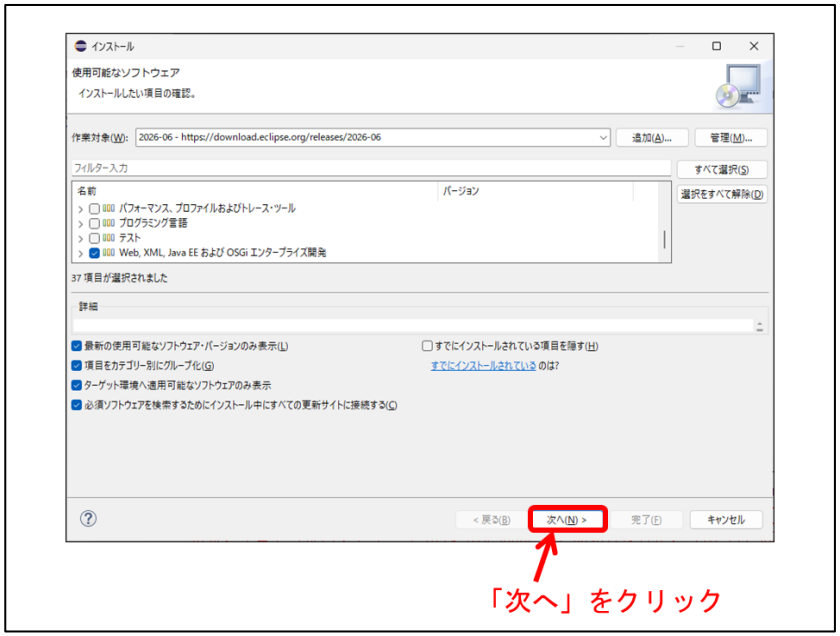


「検索対象」で
「2026-06 - https://download.eclipse.org/releases/2026-06」
を選択

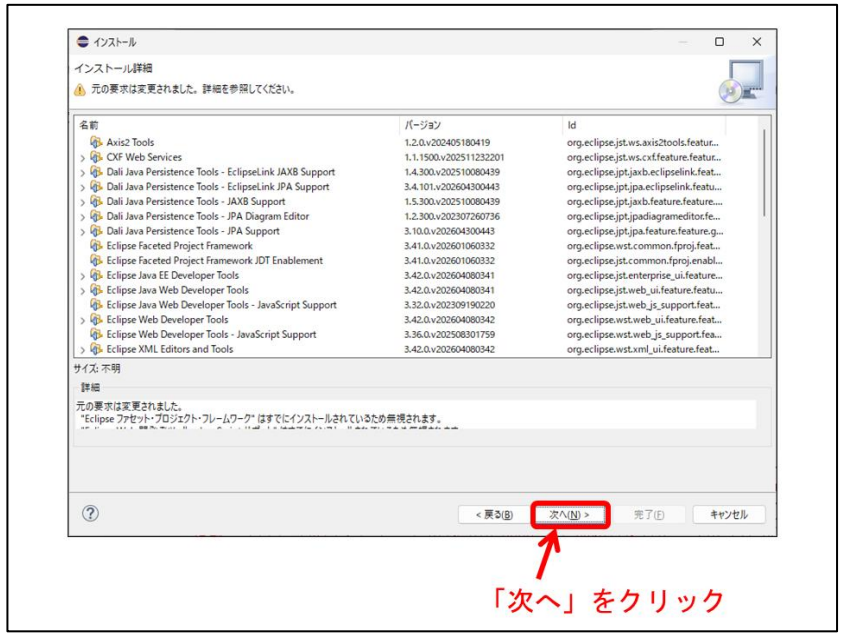
112



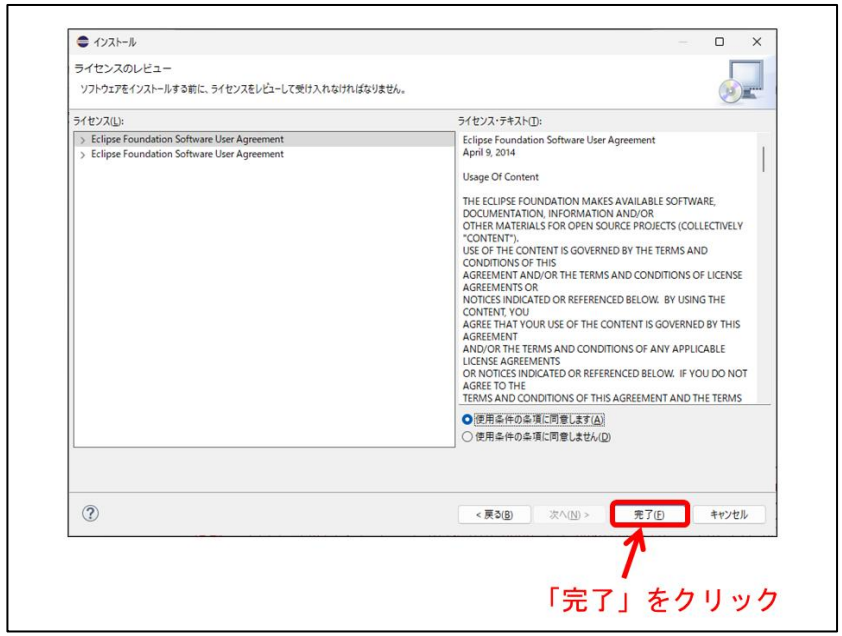
113



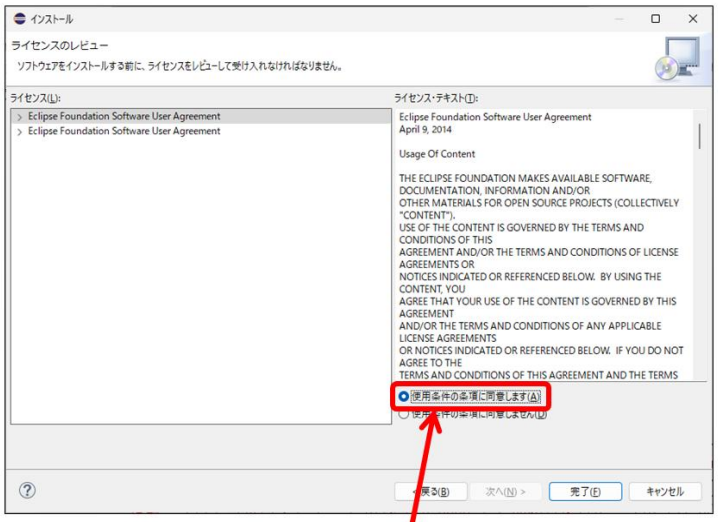
114



115

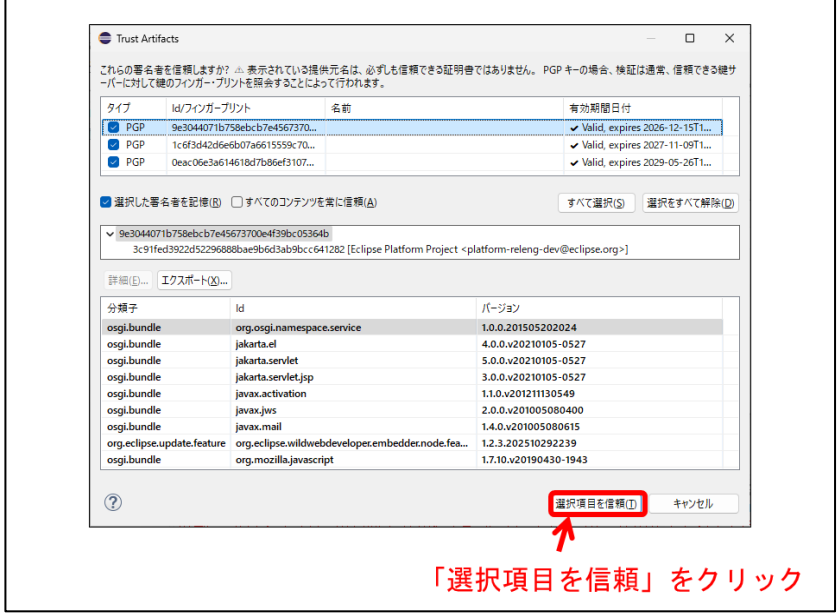


116



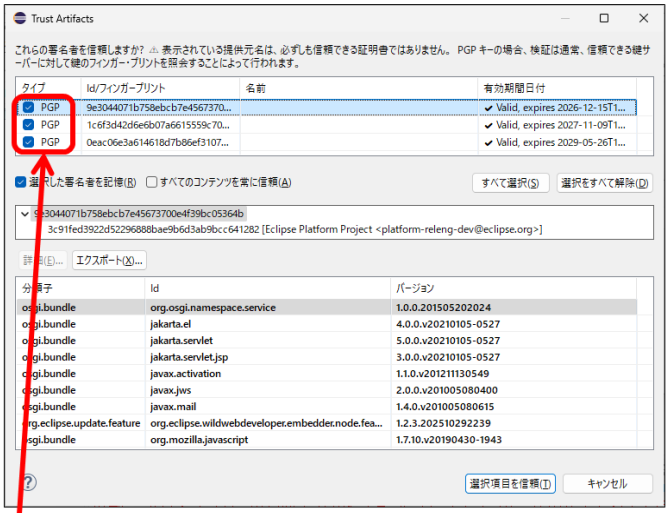
「使用条件の条項に同意します」を選択

117



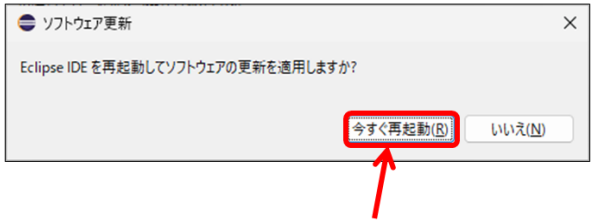
「選択項目を信頼」をクリック

119



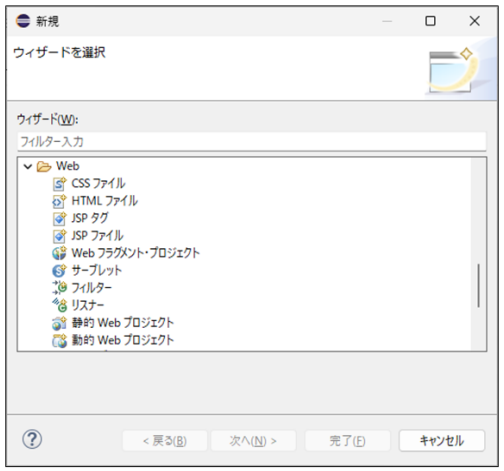
表示された項目にチェック

118



「今すぐ再起動」をクリック

120



ファイルの新規作成ダイアログに「JSPファイル」などが表示される