

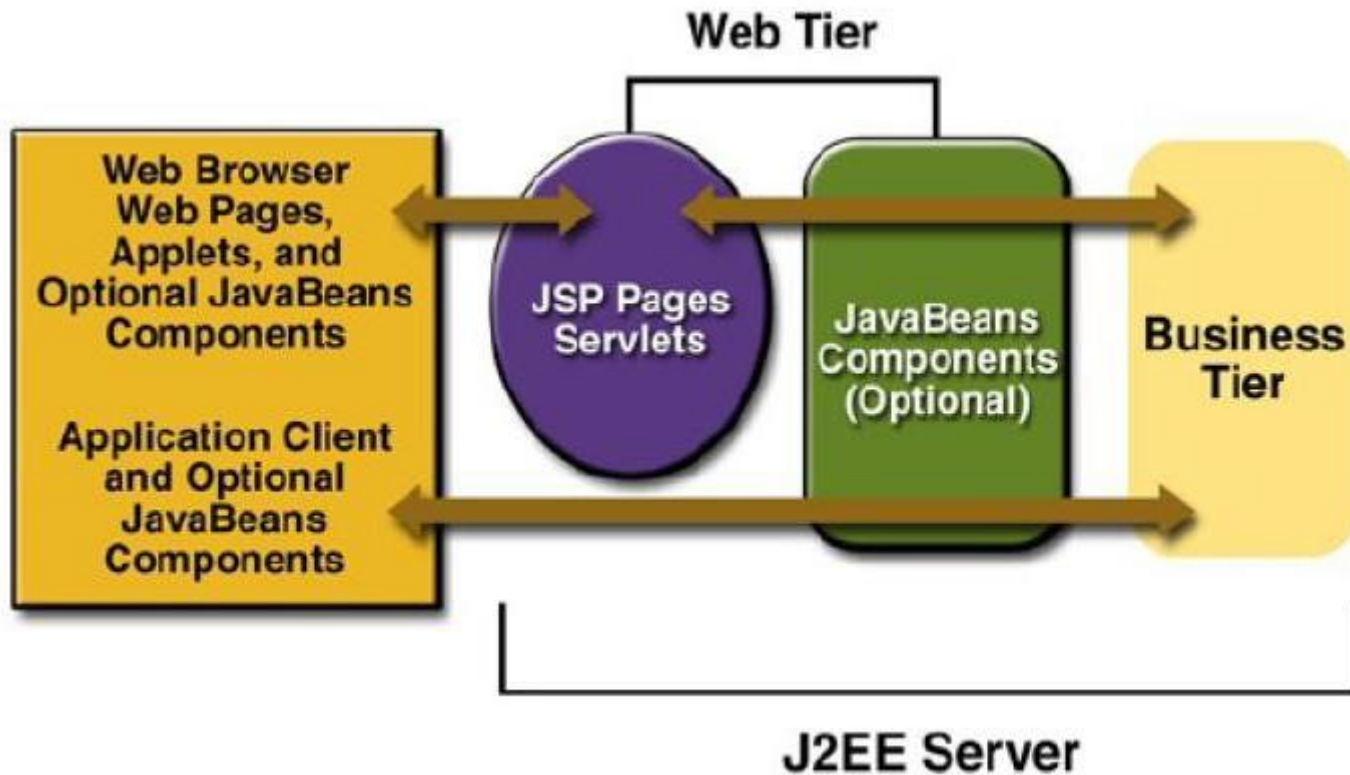
# **J2EE**

## **Java 2 Enterprise Edition**

# J2EE Web Tier

- Java 2 Enterprise Edition (J2EE) merupakan platform yang digunakan untuk pengembangan aplikasi enterprise yang berbasis komponen.
- Model aplikasi yang digunakan oleh platform ini disebut juga **distributed multi-tier**.
  - distributed : Aplikasi yang didesain dan dibangun di atas platform ini memiliki komponen-komponen yang berada pada berbagai mesin yang berbeda.
  - multi-tier : aplikasi didesain dengan level pemisahan sesuai dengan mayoritas komponen dari aplikasi.
- Aplikasi web merupakan implementasi dari multi-tier
  - Presentation layer -> client browser
  - Business logic layer -> program yang berada di Server
  - Storage layer -> database yang menangani data aplikasi

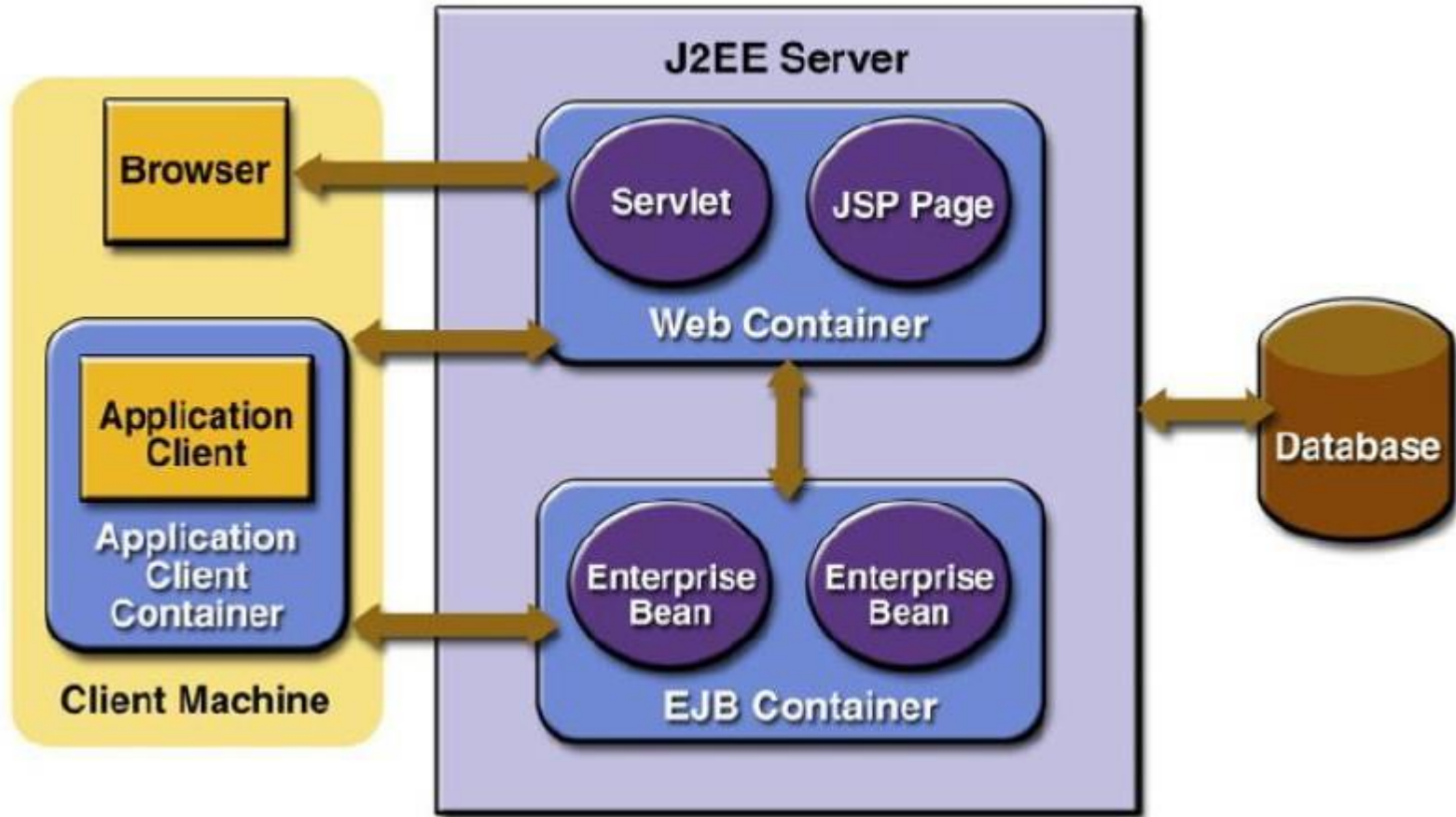
# J2EE Web Tier



# J2EE Web Tier

- Servlet
  - Technology Servlet digunakan untuk membaca data yang berada pada Request yang dikirim ke Server dan kemudian menghasilkan Response dinamis berdasar pada data tersebut.
- JavaServerPage(JSP)
  - JSP terlihat seperti HTML pada umumnya, dan memiliki kemampuan dinamis melalui penggunaan script dan expression.

# Container



# **JSP**

## **(JAVA SERVER PAGES)**

# Pendahuluan Teknologi JSP

- Para pengembang Java yang hanya menggunakan teknologi servlet dalam membangun konten dinamik HTML pada site, menghabiskan banyak kode dan hal ini berdampak pada *maintenance*.
- Para pengembang yang hanya menggunakan teknologi servlet dalam membangun konten dinamik HTML diasumsikan telah familiar dalam Java dan HTML. Memberikan pelatihan Java kepada designer site dan pelatihan HTML kepada pengembang Java sangat menyita waktu.

# Definisi JSP

- Java Server Pages (JSP) adalah teknologi servlet-based yang digunakan dalam web, menyediakan konten dinamik dan statik.
- JSP adalah teknologi text-based dan sebagian besar berisi teks template HTML yang digabungkan dengan konten teks spesifik dinamik.



# Kenapa Memakai JSP ?

- Menghindarkan pengembang dari memanipulasi String yang sangat panjang karena JSP adalah teks dokumen dan mirip dengan HTML, konten dari HTML sekarang tidak dipasang dalam kode Java sehingga ini memudahkan dalam hal *maintenance*.
- JSP menjadi familiar di kalangan semua orang yang mempunyai pengetahuan HTML, dengan hanya mempelajari markup dynamic. Ini memungkinkan designer site untuk membuat template HTML dari sebuah site, dengan memproses JSP untuk kemudian menyertakan konten dinamik dari tags. Ini memudahkan bagi pengembangan web page.

# Kenapa memakai JSP?

- JSP mempunyai support built-in untuk penggunaan komponen-komponen dari *reusable software* (JavaBeans). Ini tidak hanya menghindari pengembang untuk menemukan kembali roda dari setiap aplikasi, memiliki support pada komponen software yang terpisah untuk menangani pemisahan presentasi logic dan *business logic*.
- JSP sebagian bagian dari solusi Java untuk pengembangan aplikasi web adalah inheritas multi-program dan dapat berjalan dalam kompatibilitas kontainer servlet, berdasar vendor system operasi.

# Kenapa memakai JSP ?

- Berdasar cara kerja JSP, dia tidak membutuhkan *explicit compilation* oleh pengembang. Kompilasi ini dilakukan oleh kontainer servlet. Modifikasi pada JSP secara otomatis dideteksi dan dihasilkan pada proses kompilasi ulang. Ini membuat JSP relatif mudah bagi pengembang.

# Contoh #1

- Buat project aplikasi web baru
- Lalu ubah kode program index.html menjadi :

```
<html>
  <body>
    <form method='post' action='result.jsp'>
      <fieldset>
        Angka 1:
        <input type='text' name='angka1' />
        <br><br>
        Angka 2:
        <input type='text' name='angka2' />
        <br><br>
        <input type='submit' value='Hitung' />
      </fieldset>
    </form>
  </body>
</html>
```

# Elemen Skrip JSP

- Terdiri dari 3: *Scriptlets*, *Expressions*, dan *Declarations*.
  - o ***Scriptlets* ( <% ... %> )**  
media untuk memasukkan kode bit Java diantara potongan template data, mempunyai bentuk:  
`<% Java code; %>`
  - o ***Expressions* ( <%= %> )**  
Digunakan untuk menginputkan nilai Java langsung kedalam output.  
Bentuk ekspresinya  
`<%= Java Expression %>`
  - o ***Declarations* ( <%! %> )**  
Digunakan untuk mendefinisikan method atau variabel. Bentuk ekspresinya  
`<%! Java Code %>`

# Contoh #2

- Tambahkan file jsp dengan nama result.jsp
- Pada project klik kanan pada Web Pages → JSP, lalu ubah kode menjadi :

```
<html>
  <body>
    <%@page errorPage="/error.jsp" %>
    <%
      String angka1str = request.getParameter("angka1");
      String angka2str = request.getParameter("angka2");
      int angka1 = Integer.parseInt(angka1str);
      int angka2 = Integer.parseInt(angka2str);
      int hasil = angka1 + angka2;
    %>
    <h3>Hasil dari <%=angka1%> + <%=angka2%> adalah <%=hasil%></h3>
  </body>
</html>
```

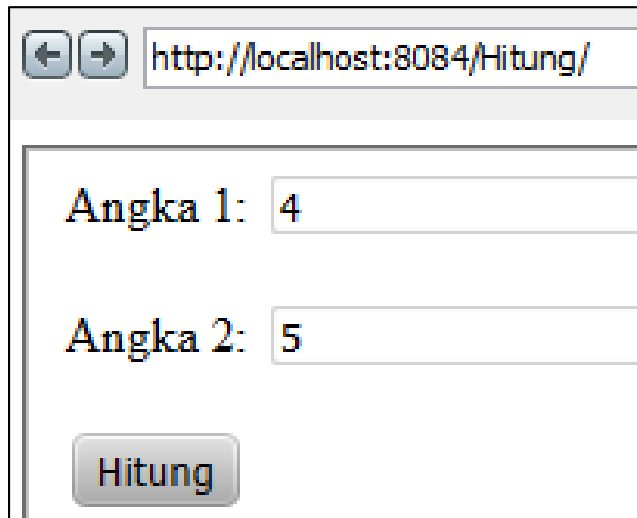
# Contoh #3

- Tambahkan file jsp dengan nama error.jsp
- Pada project klik kanan pada Web Pages → JSP, lalu ubah kode menjadi :

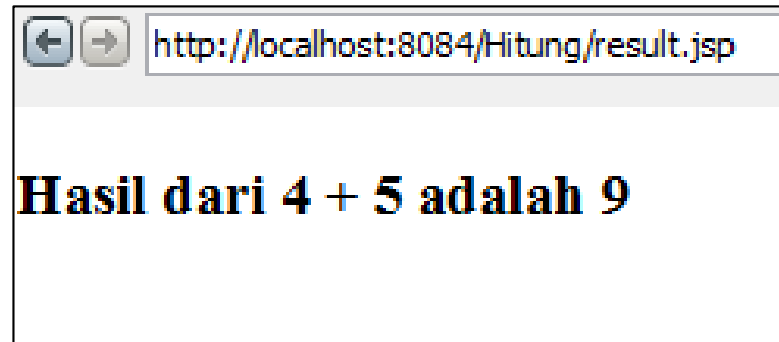
```
<html>
  <body>
    <%@page isErrorPage="true" %>
    <h1>Telah Terjadi ERROR!!</h1>
  </body>
</html>
```

# Contoh #4

- Tambahkan file jsp dengan nama error.jsp
- Pada project klik kanan pada Web Pages → JSP, lalu ubah kode menjadi :



A screenshot of a web browser window. The address bar shows `http://localhost:8084/Hitung/`. The page content includes two input fields: "Angka 1:" with the value "4" and "Angka 2:" with the value "5". Below these fields is a button labeled "Hitung".



A screenshot of a web browser window. The address bar shows `http://localhost:8084/Hitung/result.jsp`. The page content displays the text "Hasil dari 4 + 5 adalah 9" in a large, bold, black serif font.



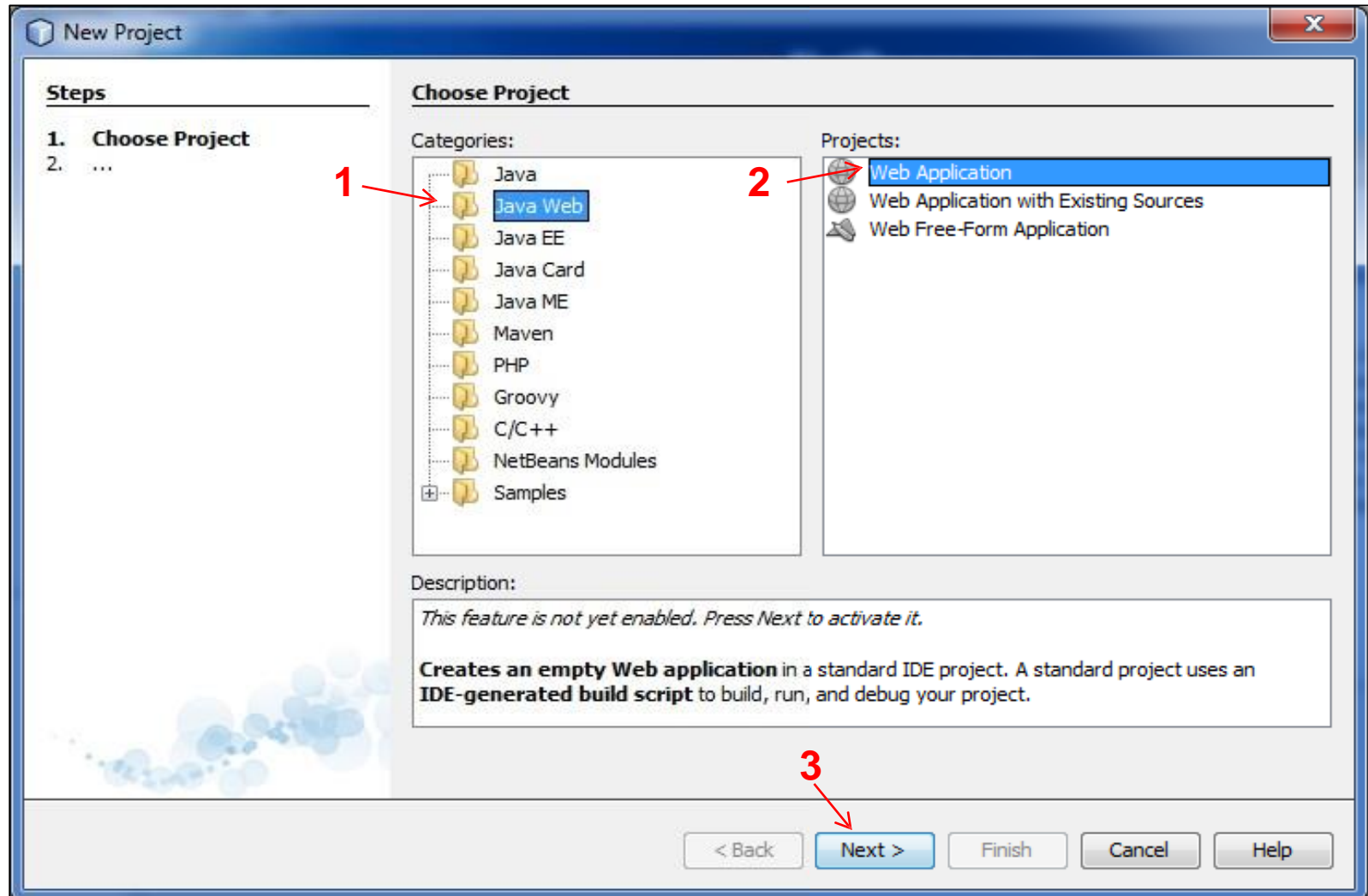
A screenshot of a web browser window. The address bar shows `http://localhost:8084/Hitung/result.jsp`. The page content displays the text "Telah Terjadi ERROR!!" in a large, bold, black serif font.



# Contoh Membuat JSP Form Otentikasi<sup>#1</sup>

- Menggunakan Java editor NetBeans, buat project aplikasi web baru
  - Pada Menubar, pilih **File -> New Project**
  - Pilih kategori **Java Web**
  - Pada bagian kanan, pilih **Web Application**

# Membuat JSP Form Otentikasi #2



# Membuat JSP Form Otentikasi #3

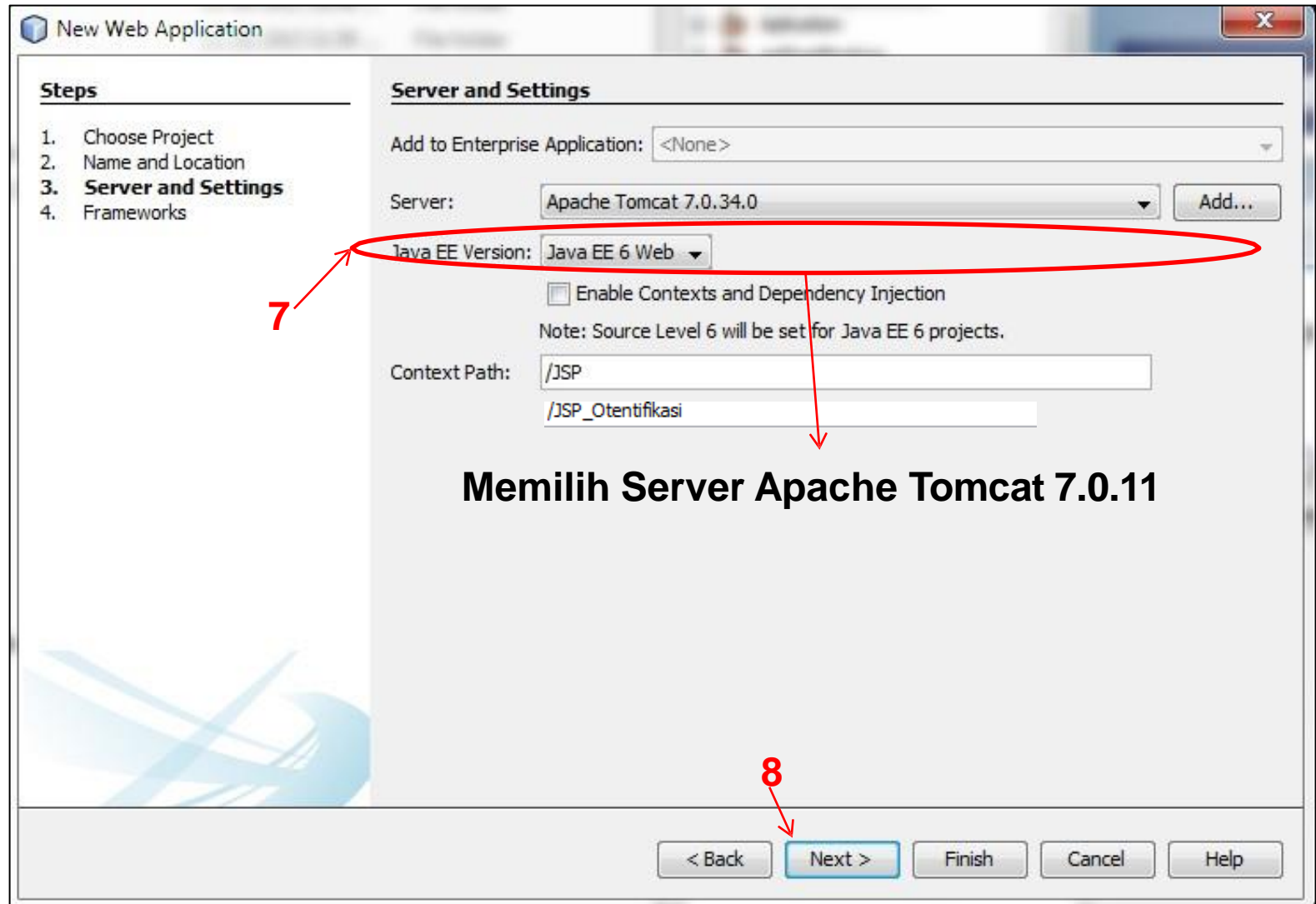
The screenshot shows the 'New Web Application' dialog box with the following components:

- Steps:**
  1. Choose Project
  2. **Name and Location**
  3. Server and Settings
  4. Frameworks
- Name and Location:**
  - Project Name: JSP\_Otentifikasi
  - Project Location: D:\S.C.H.O.O.L\S.T.M.I.K\KULIAH\M.K\Java\2013 (highlighted with a red oval and a red arrow labeled '5')
  - Project Folder: D:\S.C.H.O.O.L\S.T.M.I.K\KULIAH\M.K\Java\2013\JSP\_Otentifikasi
  - ☐ Use Dedicated Folder for Storing Libraries
  - Libraries Folder: (empty field)
- Buttons:** < Back, Next > (highlighted with a red arrow labeled '6'), Finish, Cancel, Help

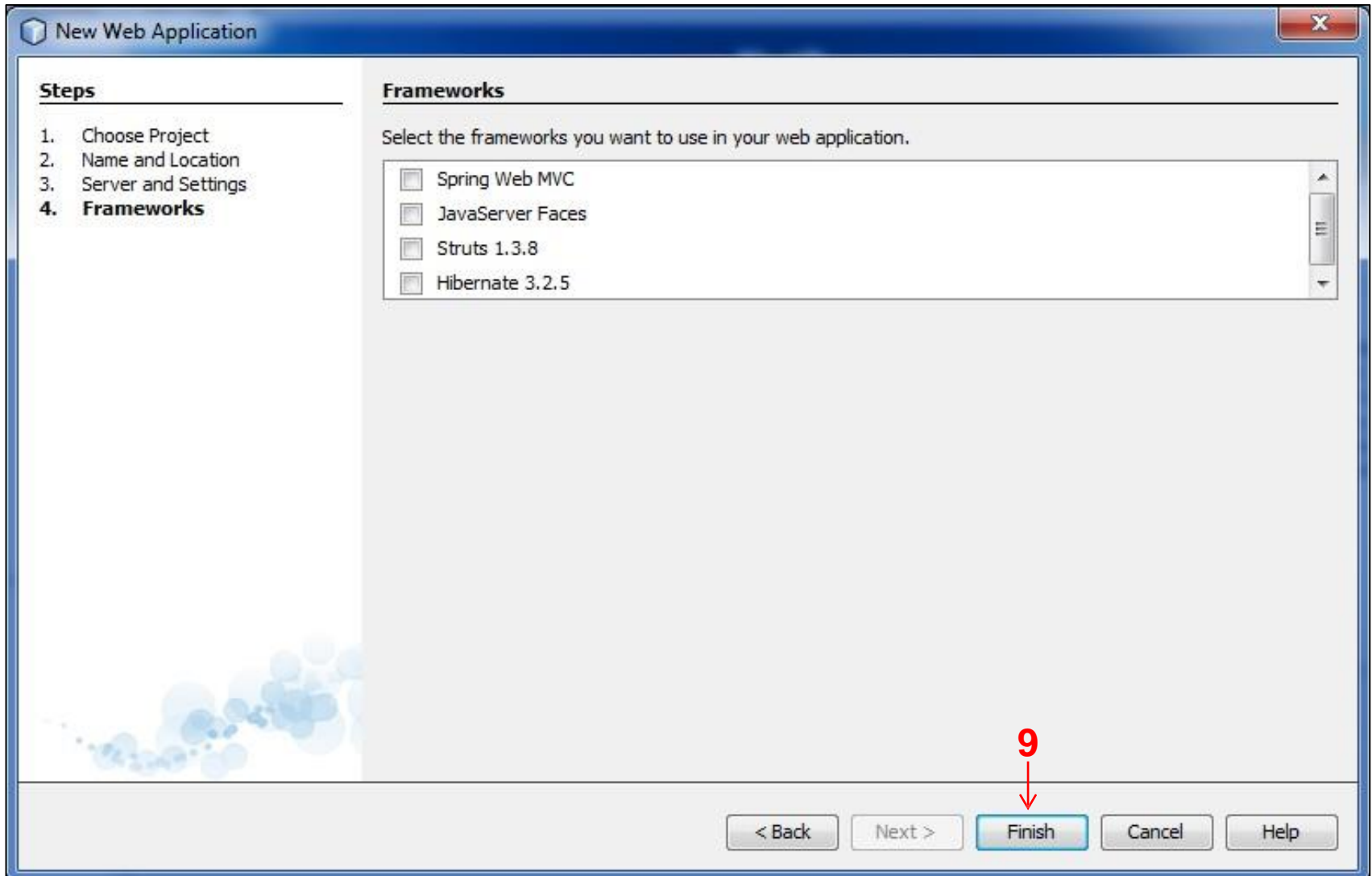
Additional annotations include a red arrow labeled '4' pointing to the 'Project Name' field and a red arrow pointing from the 'Project Location' field to the text 'Menentukan direktori project'.

**Menentukan direktori project**

# Membuat JSP Form Otentikasi #4

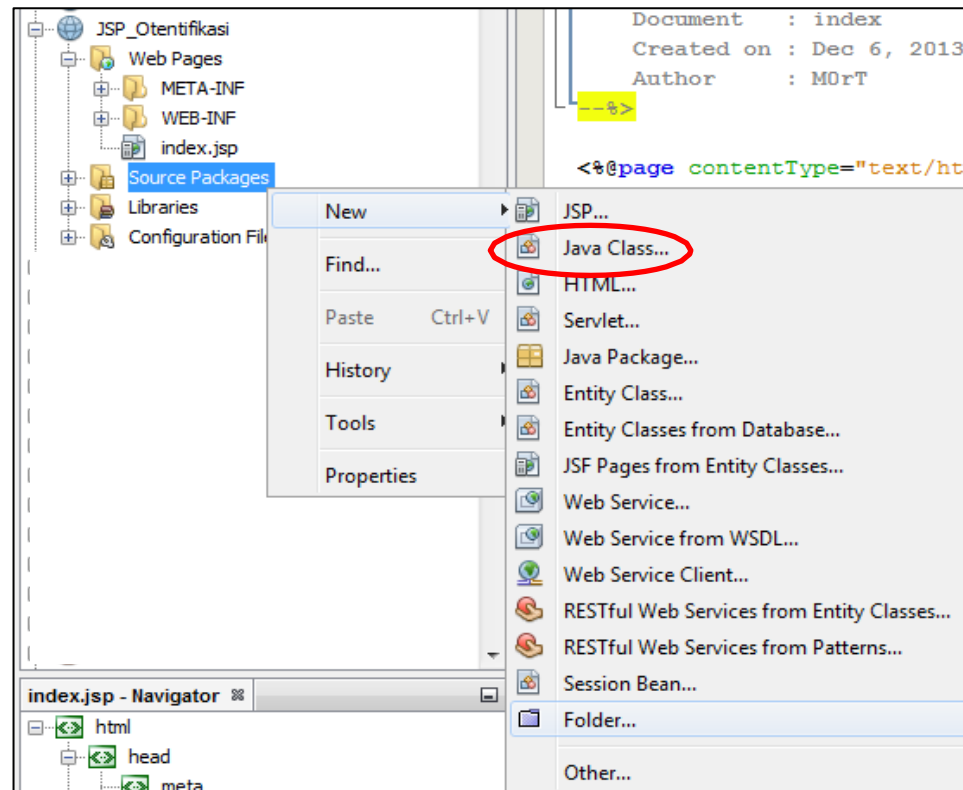


# Membuat JSP Form Otentikasi #5



# Membuat JSP Form Otentikasi #6

1. Selanjutnya membuat class java :
2. Klik kanan pada source package → New → Java Class



# Membuat JSP Form Otentikasi #7

3. Tulis Class Name → UserData dan Package → com.info, lalu tekan finis.

**New Java Class**

**Steps**

1. Choose File Type
- 2. Name and Location**

**Name and Location**

Class Name: UserData

Project: JSP\_Otentifikasi

Location: Source Packages

Package: com.info

Created File: LIAH\M.K\Java\2013\JSP\_Otentifikasi\JSP\_Otentifikasi\src\java\com\info\UserData.java

< Back   Next >   **Finish**   Cancel   Help

# Membuat JSP Form Otentikasi #8

4. Tambahkan kode program dibawah pada UserData.java :

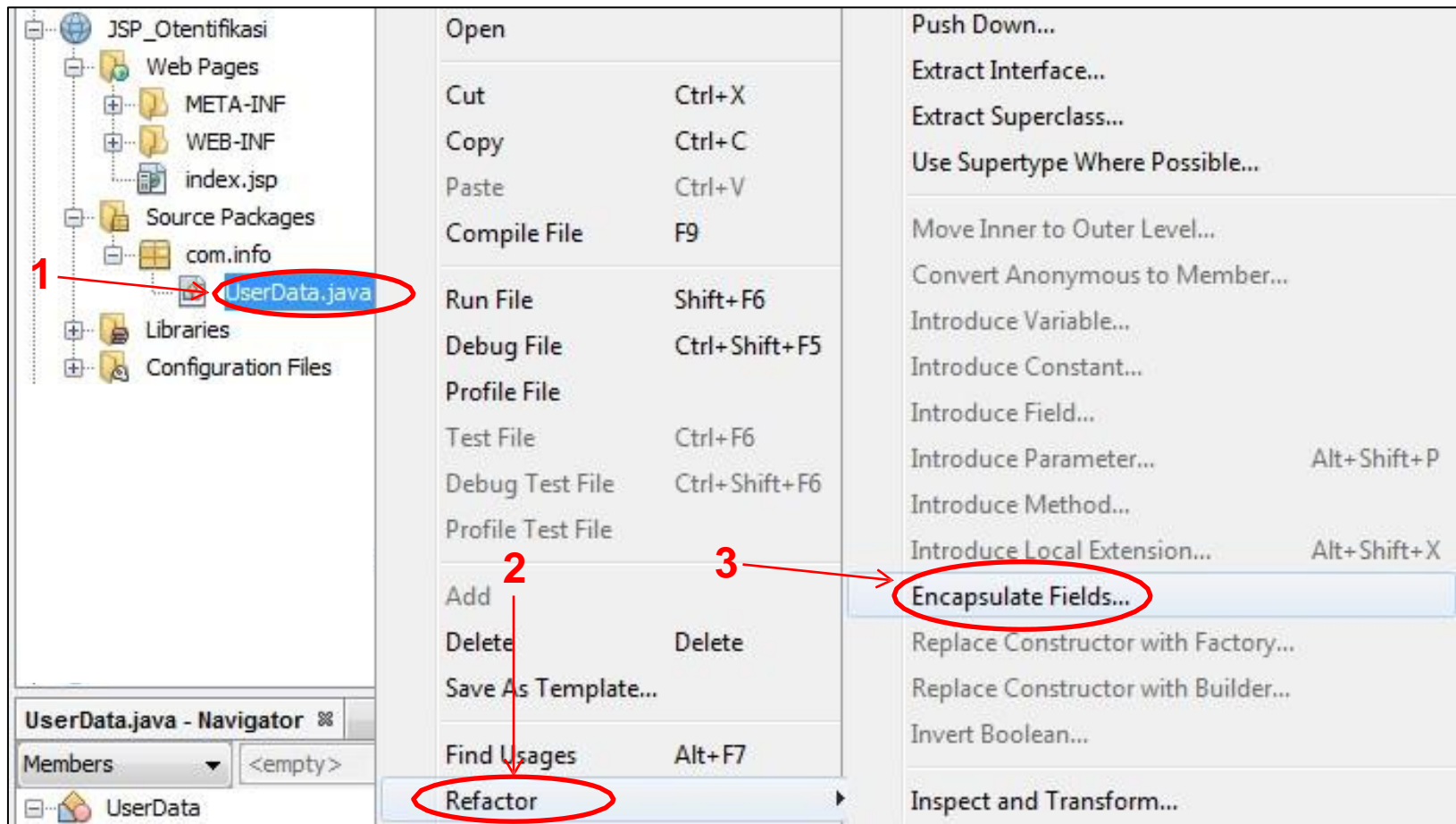
```
package com.info;

public class UserData {
    String namaUser;
    int umur;
}
```



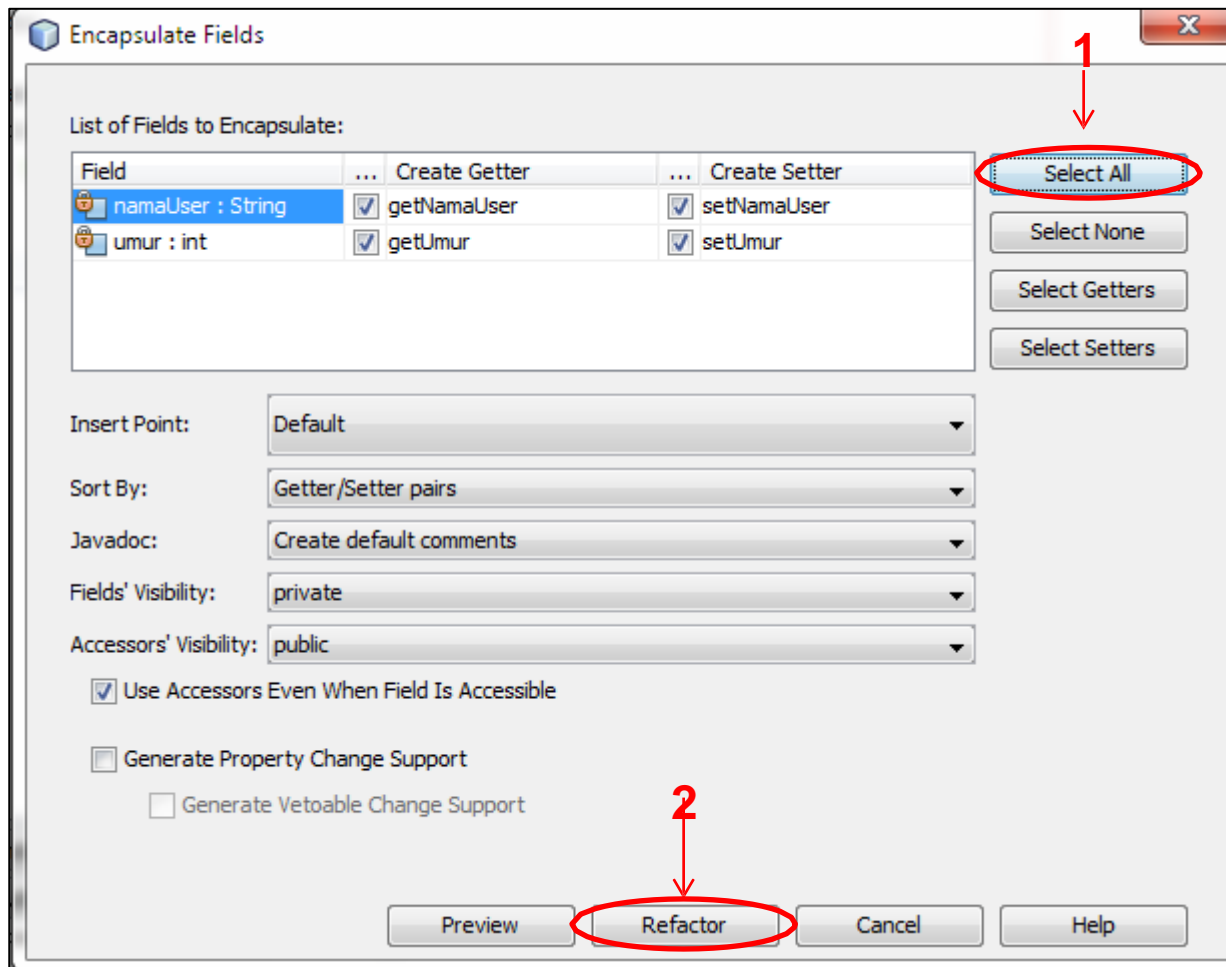
# Membuat JSP Form Otentikasi #9

5. Klik kanan pada UserData.java → Refactor → Encapsule Fields



# Membuat JSP Form Otentikasi #10

6. Tekan Select All → Refactor



# Membuat JSP Form Otentikasi #11

## 7. Hasilnya

```
package com.info;

public class UserData {
    String namaUser;
    int umur;

    /**...*/
    public String getNamaUser() {
        return namaUser;
    }

    /**...*/
    public void setNamaUser(String namaUser) {
        this.namaUser = namaUser;
    }

    /**...*/
    public int getUmur() {
        return umur;
    }

    /**...*/
    public void setUmur(int umur) {
        this.umur = umur;
    }
}
```

# Membuat JSP Form Otentikasi #12

8. Edit kode program pada Web Pages → index.jsp menjadi :

```
<%@ page import="com.info.UserData" %>
<html>
  <head>
  <body>
    <h1>WEB JSP OTENTIFIKASI</h1>

    <%
      UserData user = (UserData) session.getAttribute("userData");
      if (user == null) {

        <h3>Anda belum terdaftar. Silakan
        <a href="login.html">Daftar</a></h3>

      <%}
      else {

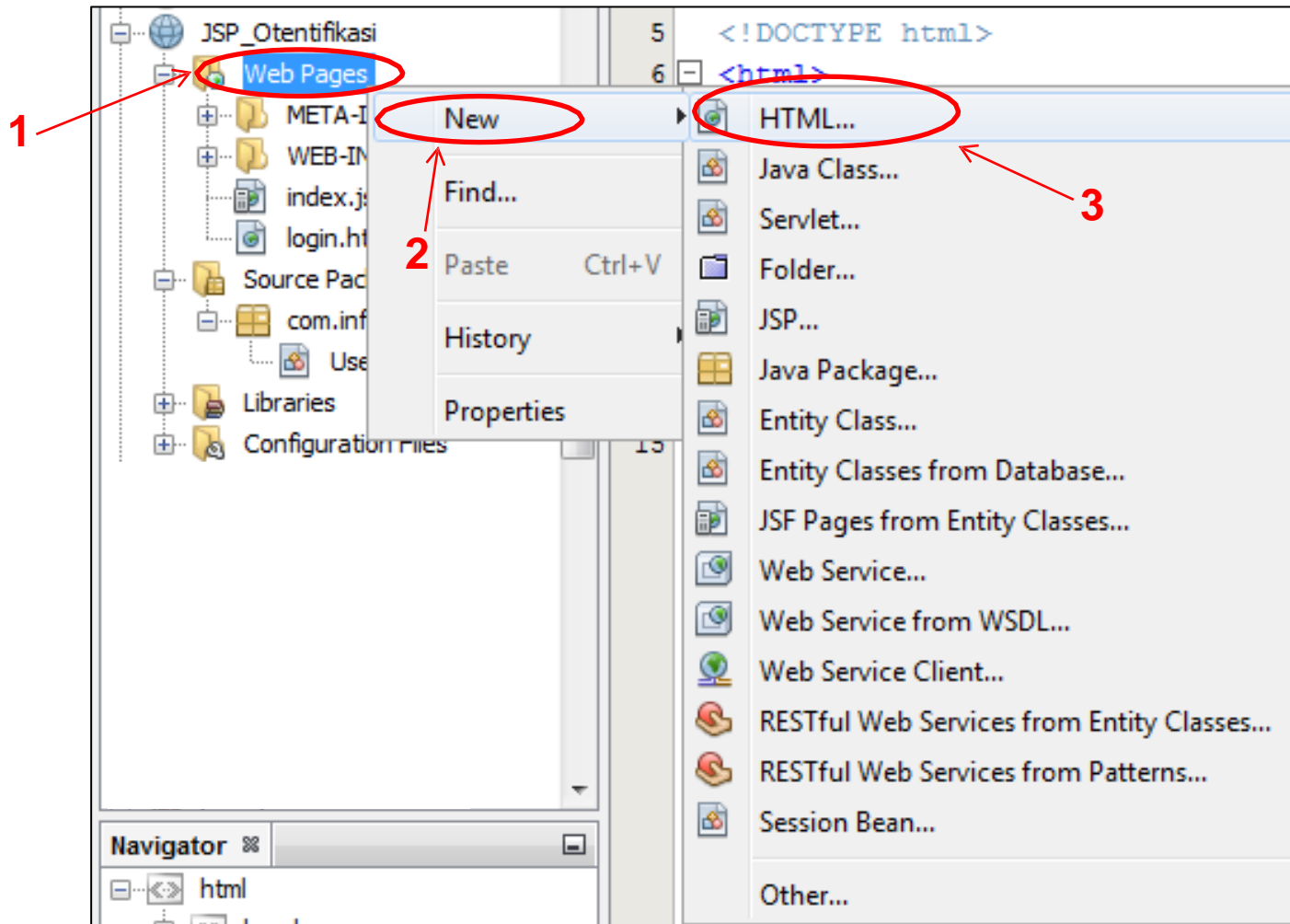
        <P>ANDA MENGAKSES HALAMAN WEB KAMI
        <h3> <a href="utama.jsp">Kembali...</a></h3>

      <% }
    <%>

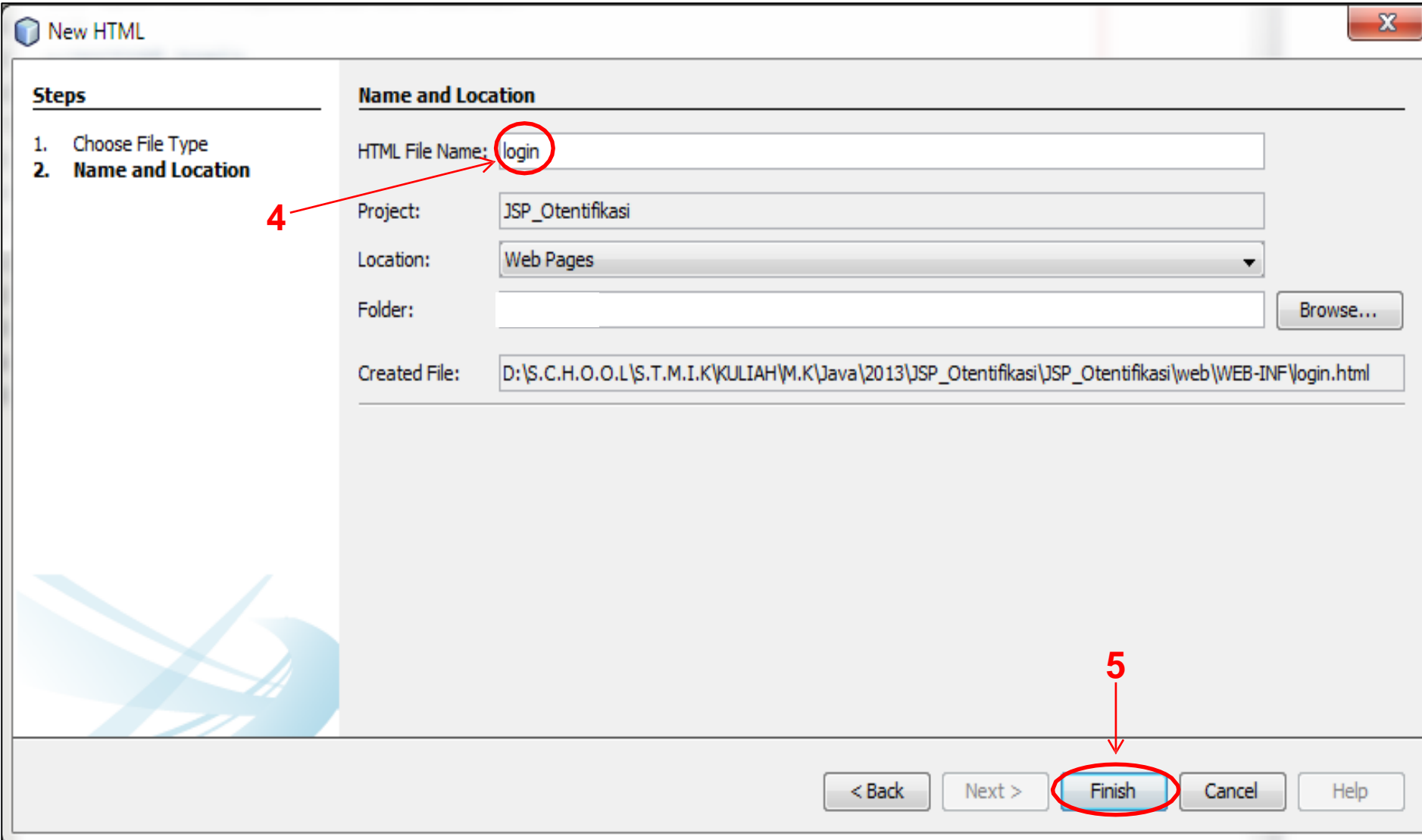
  </body>
</html>
```

# Membuat JSP Form Otentikasi #13

9. Buat file login.html dengan cara klik kanan pada Web Pages→New→HTML



# Membuat JSP Form Otentikasi #14



**New HTML**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

HTML File Name:

Project:

Location:

Folder:

Created File:

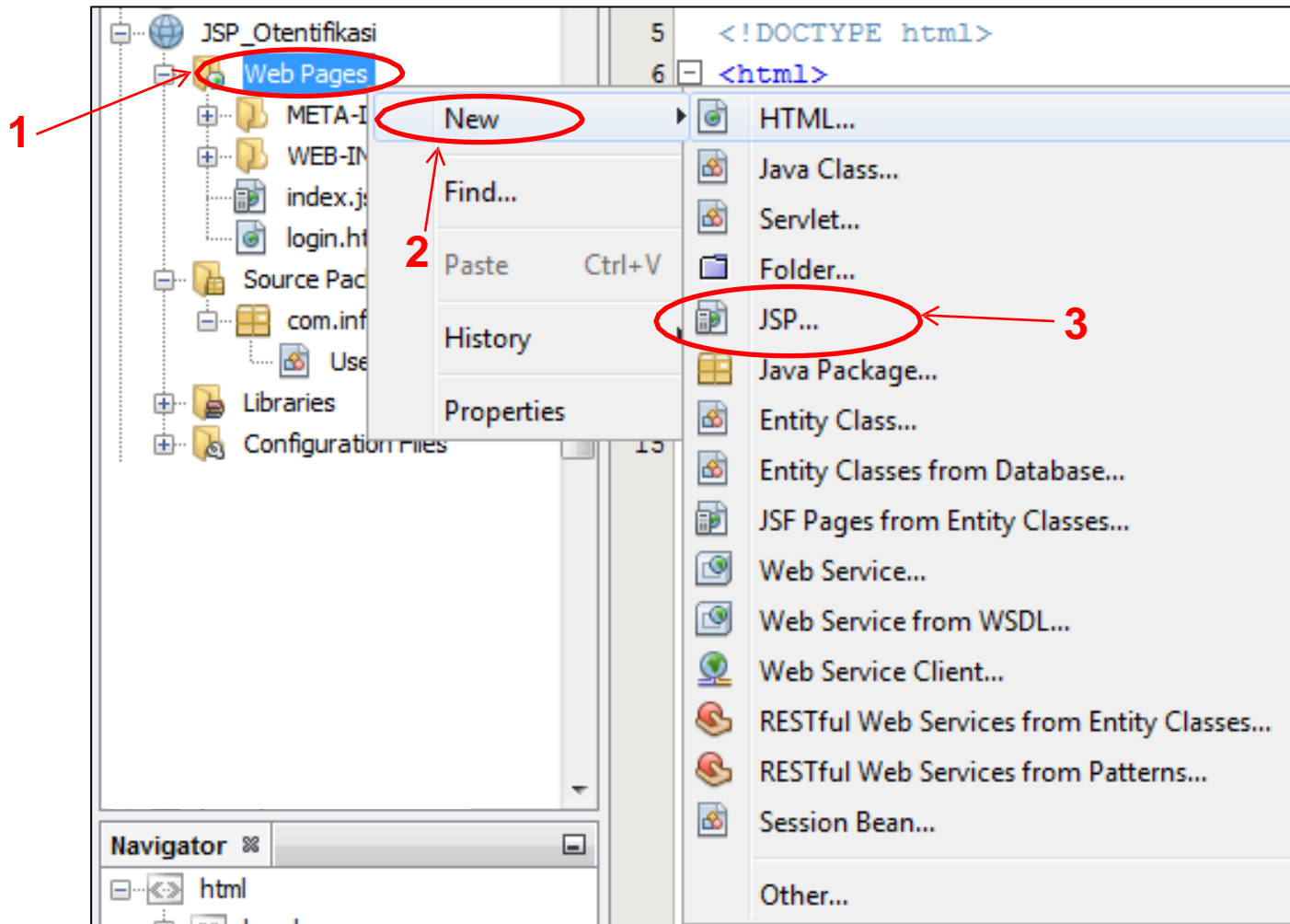
# Membuat JSP Form Otentikasi #15

10. Kode login.html :

```
<html>
  <body>
    <h2>Isi Data Anda:</h2>
    <form name="Login" action="utama.jsp" method="POST">
      <table border="0" cellspacing="2">
        <tr>
          <td align="right">Nama:</td>
          <td align="left">
            <input type="text" name="namaUser" width="36" />
          </td>
        </tr>
        <tr>
          <td align="right">Umur:</td>
          <td align="left">
            <input type="text" name="umur" width="8" />
          </td>
        </tr>
        <tr>
          <td></td>
          <td align="right">
            <input type="submit" value="KIRIM" />
          </td>
        </tr>
      </table>
    </form><hr>
  </body>
</html>
```

# Membuat JSP Form Otentikasi #16

11. Buat file utama.jsp dengan cara klik kanan pada Web Pages → New → JSP





# Membuat JSP Form Otentikasi #17

**New JSP**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

File Name: utama

Project: JSP\_Otentifikasi

Location: Web Pages

Folder:  Browse...

Created File: D:\S.C.H.O.O.L\S.T.M.I.K\KULIAH\M.K\Java\2013\JSP\_Otentifikasi\JSP\_Otentifikasi\web\utama.jsp

**Options:**

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

**Description:**

A JSP file using JSP standard syntax.

< Back Next > **Finish** Cancel Help

# Membuat JSP Form Otentikasi #18

12. Edit kode program pada Web Pages → utama.jsp menjadi :

```
<html>
  <head>
  </head>
  <body>
    <h1>HALAMAN KONFIRMASI</h1>
    <jsp:useBean id="userData" scope="session" class="com.info.UserData"/>
    <jsp:setProperty name="userData" property="*" />
    <p>Selamat datang, <%= userData.getNamaUser() %>.<br>
      Usia Anda = <jsp:getProperty name="userData" property="umur"/>.<br>
      <a href="index.jsp">KEMBALI</a></p>
  </body>
</html>
```

13. Run project

# Membuat JSP Form Otentikasi #9

Tampilan pada browser :



http://localhost:8084/JSP\_Otentifikasi/

## WEB JSP OTENTIFIKASI

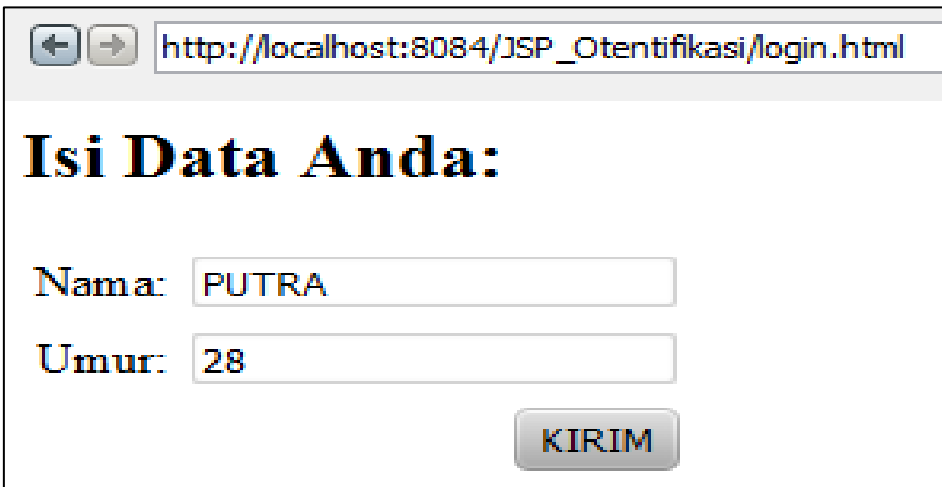
Anda belum terdaftar. Silakan [Daftar](#)



http://localhost:8084/JSP\_Otentifikasi/utama.jsp

## HALAMAN KONFIRMASI

Selamat datang, PUTRA.  
Usia Anda = 28.  
[KEMBALI](#)



http://localhost:8084/JSP\_Otentifikasi/login.html

## Isi Data Anda:

Nama:

Umur:



http://localhost:8084/JSP\_Otentifikasi/index.jsp

## WEB JSP OTENTIFIKASI

ANDA MENGAkses HALAMAN WEB KAMI

[Kembali...](#)

# Xampp – Apache Tomcat

Buka tomcat-users.xml. di : \xampp\tomcat\conf

Tambahkan script di bawah:

```
<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status,admin-gui,admin-script"/>
```

**SERVLET**  
SERVLET

# Pengenalan

Servlet merupakan Class dari bahasa pemrograman Java yang digunakan untuk memperluas kemampuan dari server yang diakses aplikasi host melalui model pemrograman Request-Response.

Servlet mampu untuk :

- Mengimplementasi interface Servlet
- Menerima Request yang datang dari Class Java, Web Client atau Servlet lainnya
- Menghasilkan Response

# Pengenalan

Untuk membuat Servlet, ada Class-class pada package berikut yang harus di-import :

- **javax.servlet** - memuat framework servlet dasar
- **javax.servlet.http.** - digunakan sebagai extension dari framework Servlet bagi Servlet untuk menjawab HTTP Request

# Common Gateway Interface (CGI)

CGI memfasilitasi Server dengan sebuah interface bagi program eksternal, yang dapat dipanggil Server untuk menangani request dari Client. Setiap panggilan terhadap resource CGI akan menciptakan proses baru pada Server (Informasi dalam hal ini adalah program yang dilewatkan ke proses ini menggunakan input standar dan environment variable).

Masalah dengan penggunaan CGI :

- Membutuhkan beban dalam jumlah banyak pada system resource
- Terbatasnya jumlah user yang dapat ditangani oleh aplikasi pada saat bersamaan



# Servlet vs CGI

Servlet didesain untuk mengatasi masalah yang ada pada CGI.

Pada Servlet, hanya dibutuhkan satu proses yang dapat menangani semua Request (proses yang dibutuhkan oleh servlet container supaya berjalan).

Servlet hanya sekali di-load ke memory

- Container me-load Servlet ke memory pada saat server startup
- atau pada saat servlet pertama kali dibutuhkan untuk melayani Client

# Servlet vs CGI

Perbedaan antara Servlet dan CGI:

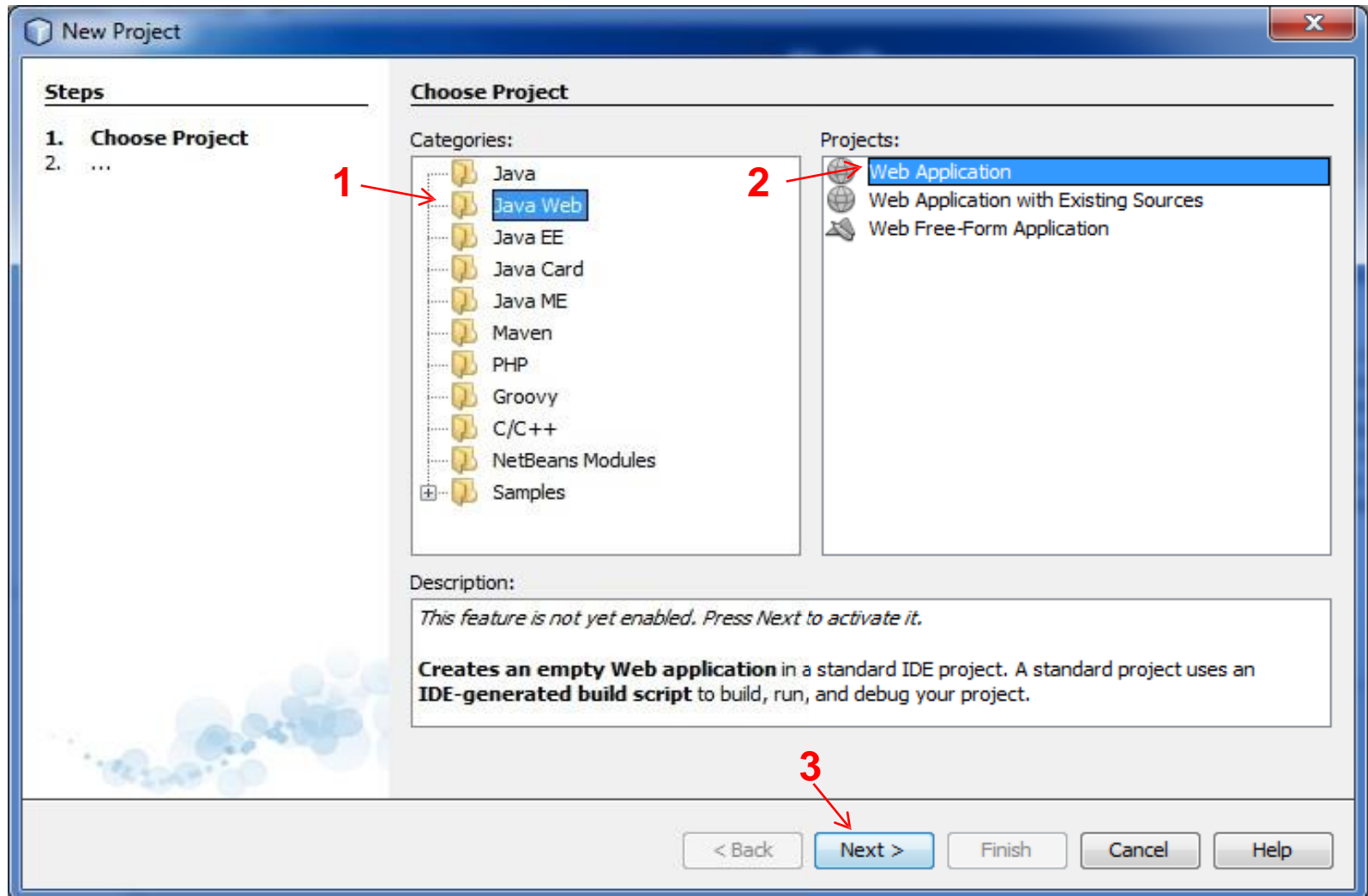
Servlet hanya di-load sekali ke memori, akan tetap berada disana, dan siap untuk menangani setiap request dari client.

CGI akan melakukan proses load dan unload program dari dan ke dalam memori, setiap kali request dari Client datang.

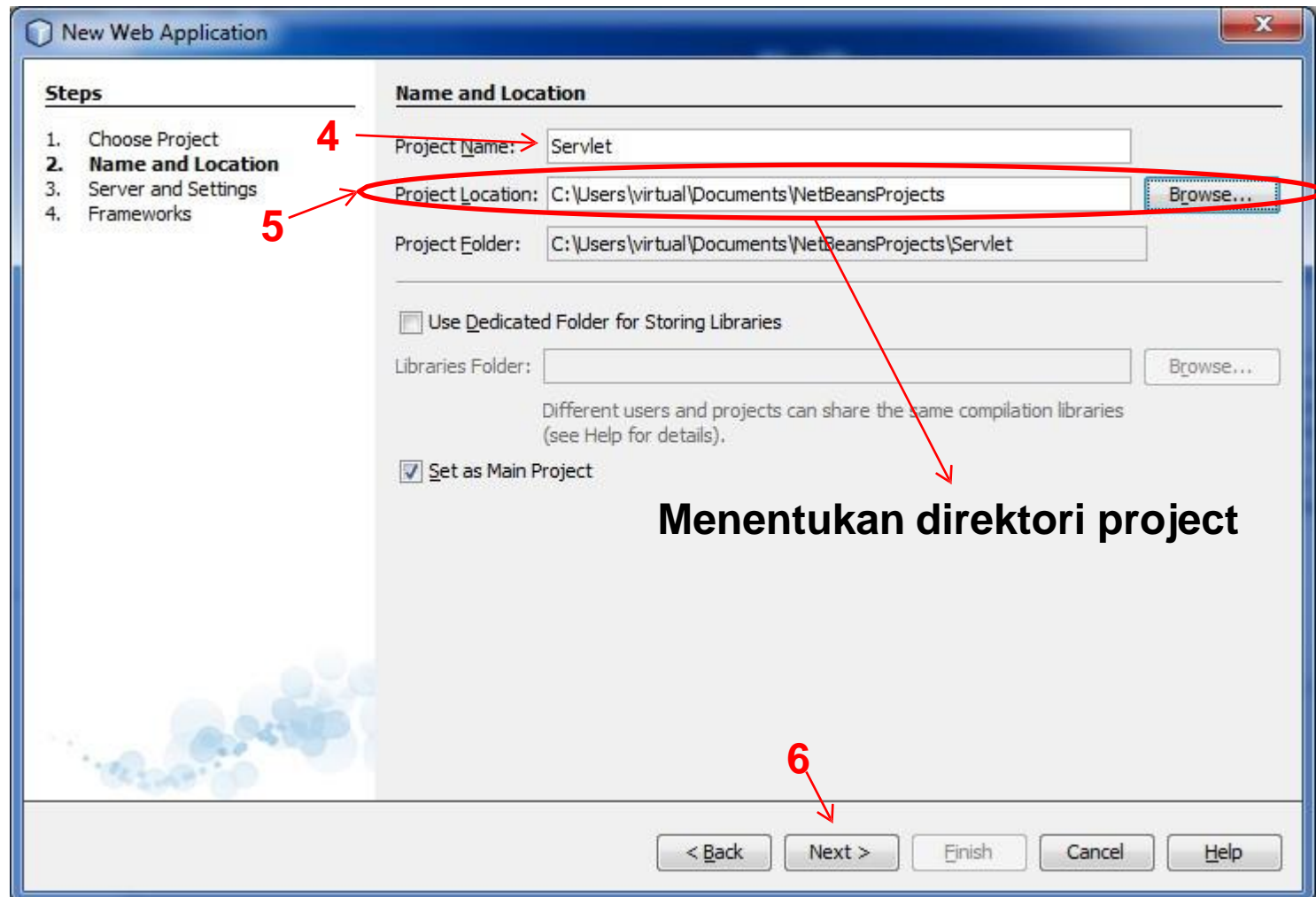
# Memulai Servlet Pertama #1

- Menggunakan Java editor NetBeans, buat project aplikasi web baru
  - Pada Menubar, pilih **File -> New Project**
  - Pilih kategori **Java Web**
  - Pada bagian kanan, pilih **Web Application**

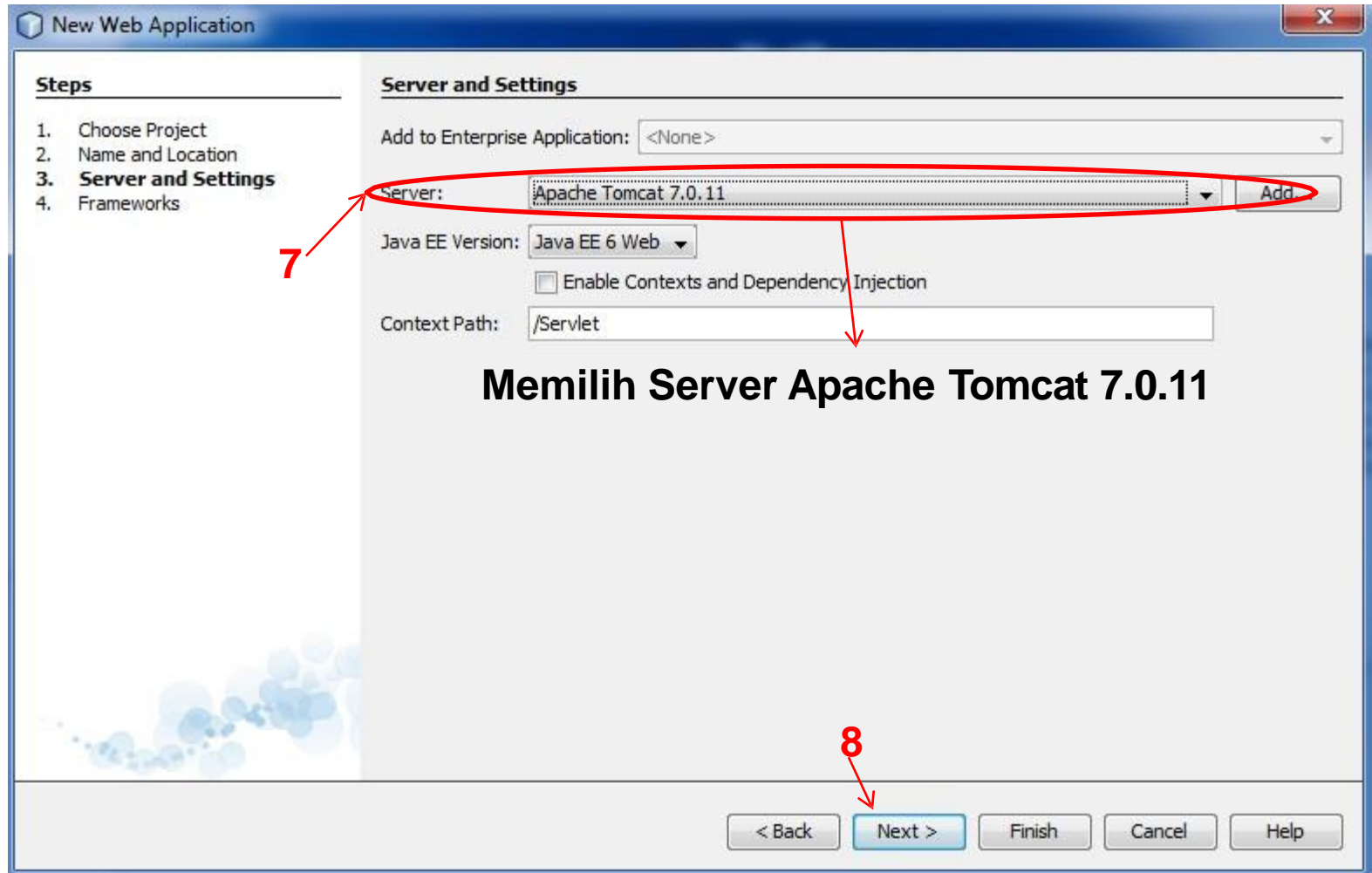
# Memulai Servlet Pertama #2



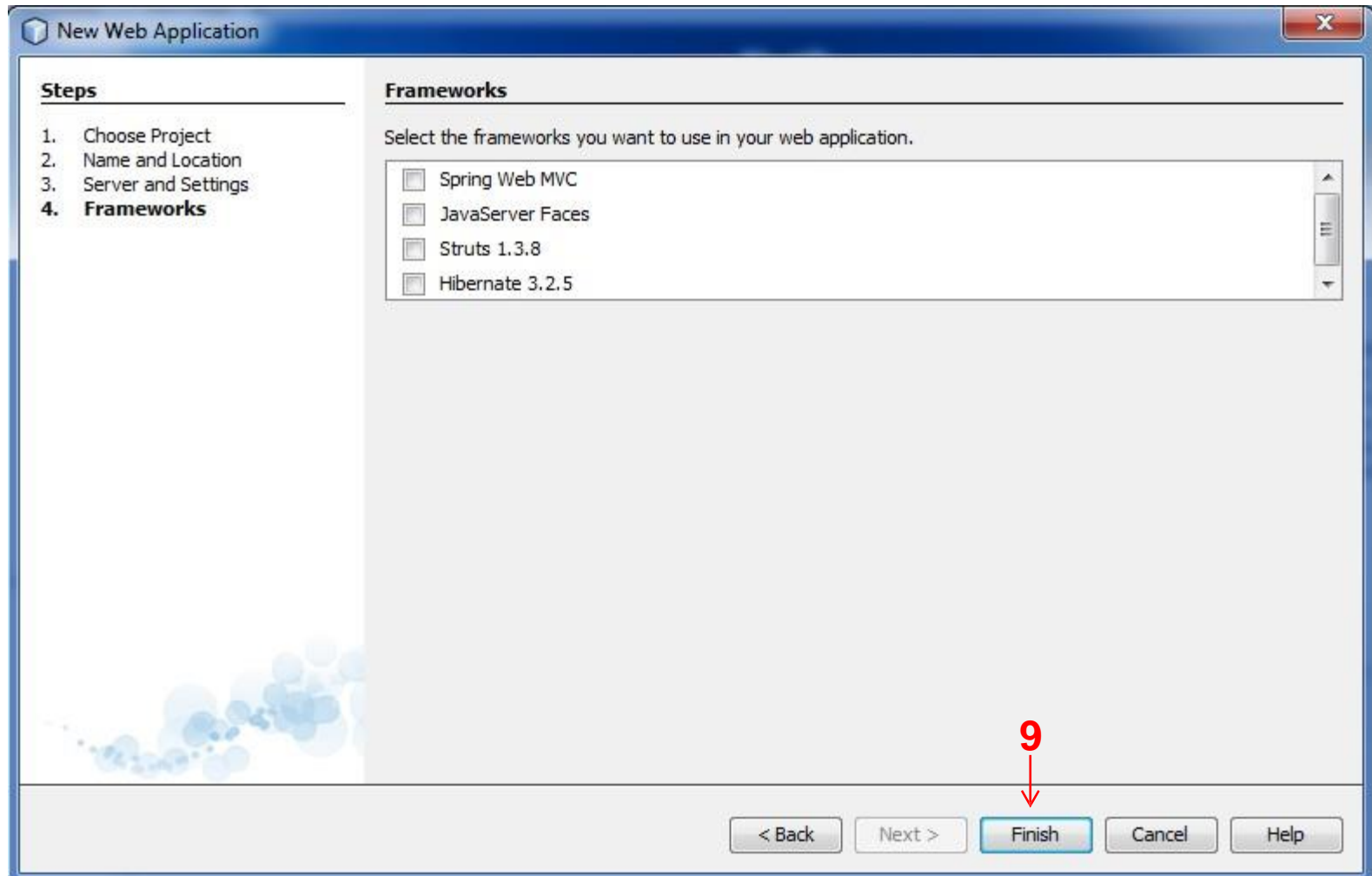
# Memulai Servlet Pertama #3



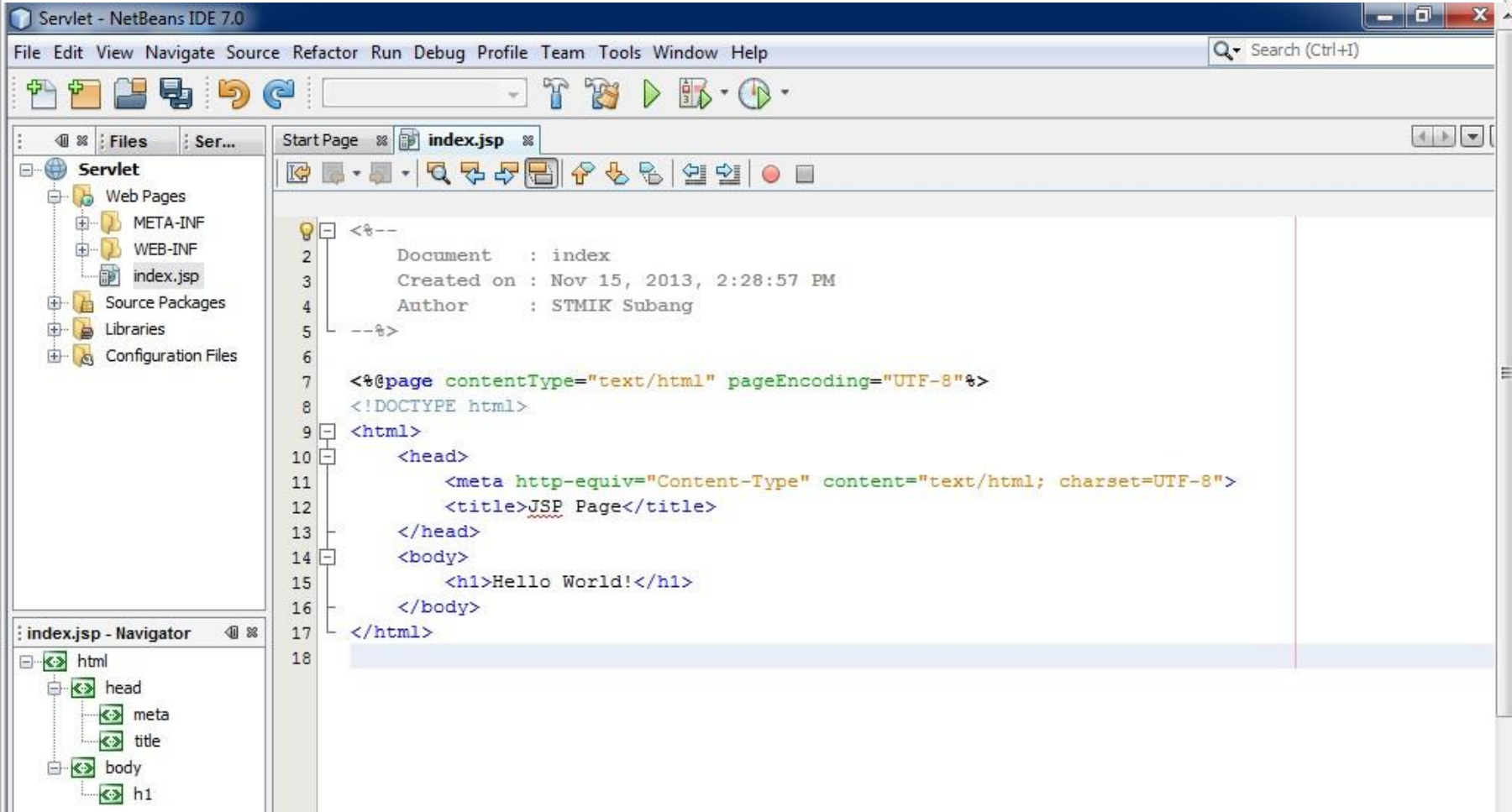
# Memulai Servlet Pertama #4



# Memulai Servlet Pertama #5



# Memulai Servlet Pertama #6





# Memulai Servlet Pertama #7

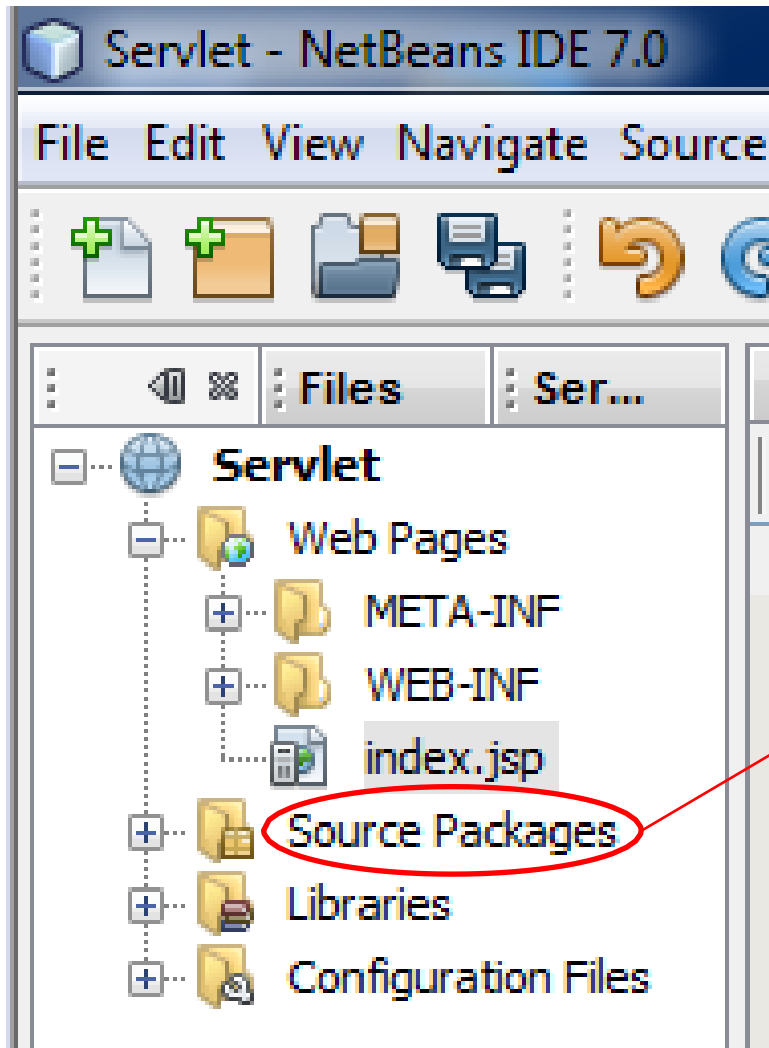
Untuk menambah Servlet ke aplikasi,

- klik kanan pada **Source Package**, pilih **New -> Servlet**

atau

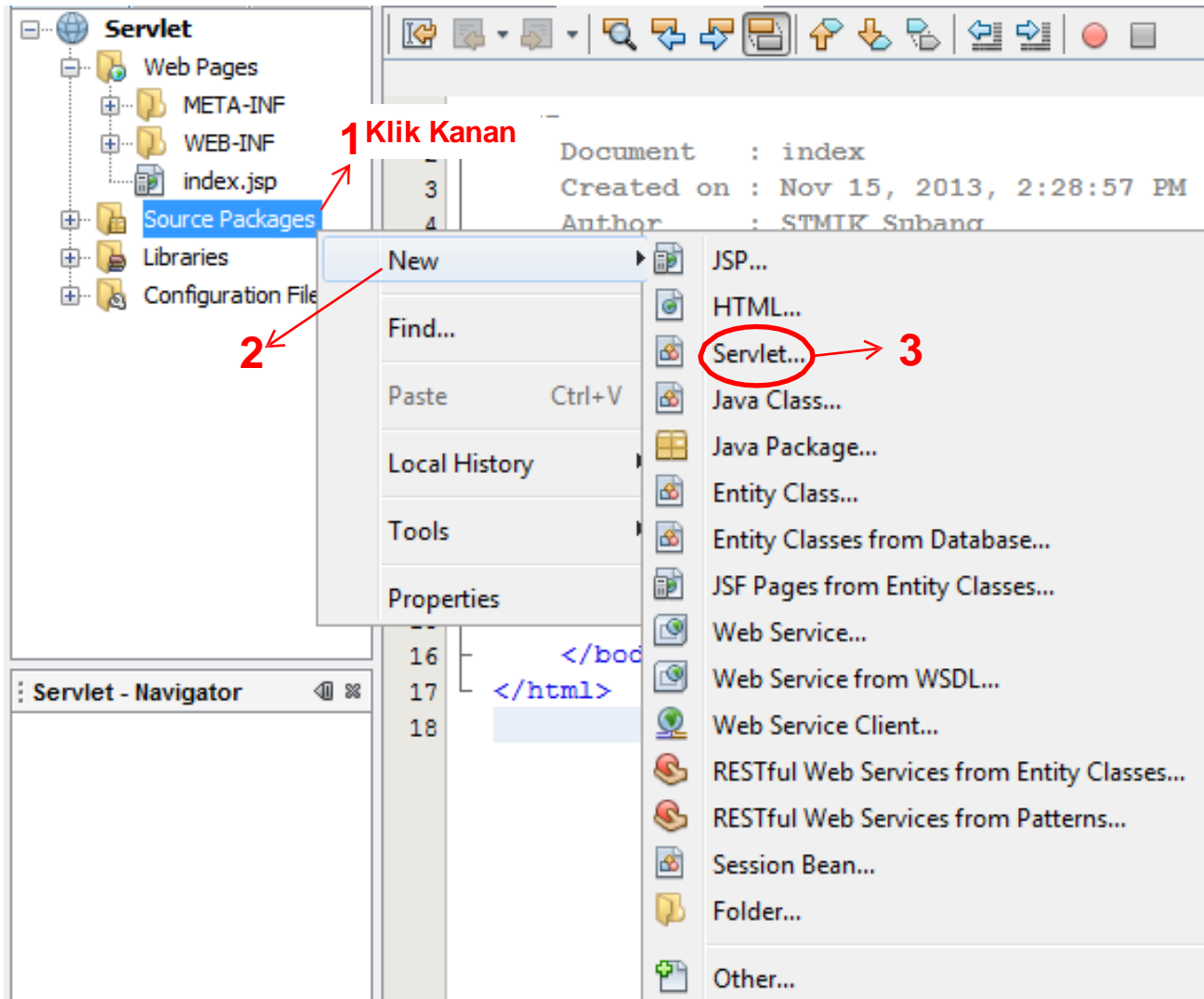
- Pada menubar, pilih **File -> New File**. Pilih kategori **Web**, kemudian pilih **Servlet**

# Memulai Servlet Pertama #8



Klik Kanan lalu pilih servlet

# Memulai Servlet Pertama #9



# Memulai Servlet Pertama #10

**New Servlet**

**Steps**

1. Choose File Type
- 2. Name and Location**
3. Configure Servlet Deployment

**Name and Location**

Class Name: tesServlet → 4

Project: Servlet

Location: Source Packages

5 → Package: com.servlet

Created File: J:\Users\virtual\Documents\NetBeansProjects\Servlet\src\java\com\servlet\tesServlet.java

< Back   **Next >**   Finish   Cancel   Help

6 ↓

# Memulai Servlet Pertama #11

**New Servlet**

**Steps**

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

**Configure Servlet Deployment**

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☒ Add information to deployment descriptor (web.xml)

Class Name:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

**8** ↓

< Back Next > **Finish** Cancel Help

# Memulai Servlet Pertama #12

The screenshot shows an IDE window with a project named 'Servlet'. The left sidebar displays the project structure, including 'Web Pages', 'META-INF', 'WEB-INF', 'index.jsp', 'Source Packages', 'com.servlet', 'tesServlet.java', 'Libraries', and 'Configuration Files'. The 'Members View' for 'tesServlet.java' is expanded, showing methods: 'doGet(HttpServletReq', 'doPost(HttpServletReq', 'getServletInfo() : Strin', and 'processRequest(HttpS'. The main editor window shows the code for 'tesServlet.java', which is a new class extending 'HttpServlet'. The code includes a 'processRequest' method that sets the content type to 'text/html; charset=UTF-8' and prints an HTML response. A comment at the bottom of the code block says: 'HttpServlet methods. Click on the + sign on the left to edit the code.'

```
13
14  /**...*/
18  public class tesServlet extends HttpServlet {
19
20  /**...*/
27  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28      throws ServletException, IOException {
29      response.setContentType("text/html; charset=UTF-8");
30      PrintWriter out = response.getWriter();
31      try {
32          /* TODO output your page here
33          out.println("<html>");
34          out.println("<head>");
35          out.println("<title>Servlet tesServlet</title>");
36          out.println("</head>");
37          out.println("<body>");
38          out.println("<h1>Servlet tesServlet at " + request.getContextPath () + "</h1>");
39          out.println("</body>");
40          out.println("</html>");
41          */
42      } finally {
43          out.close();
44      }
45  }
46
47  HttpServlet methods. Click on the + sign on the left to edit the code.
82 }
```

# Memulai Servlet Pertama #13

- IDE telah menciptakan dan mengimplementasikan sebagian method yang bernama **processRequest**.
- Ada tanda "+" di bagian kiri bawah pada coding, jika di-klik, dapat dilihat bahwa ternyata method `processRequest` adalah method yang akan dipanggil oleh **doGet** dan **doPost**.
- Content dari method `processRequest` membentuk fungsionalitas dasar dari Servlet.

# Memulai Servlet Pertama #14

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**...*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}
```



# Memulai Servlet Pertama #15

- Edit kode yang ada di dalam blok method processMethod seperti berikut :

```
out.println("<html>");
out.println("<head>");
out.println("<title>STMIK Subang</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>SELAMAT BELAJAR SERVLET</h1>");
out.println("</body>");
out.println("</html>");
```

# Memulai Servlet Pertama #16

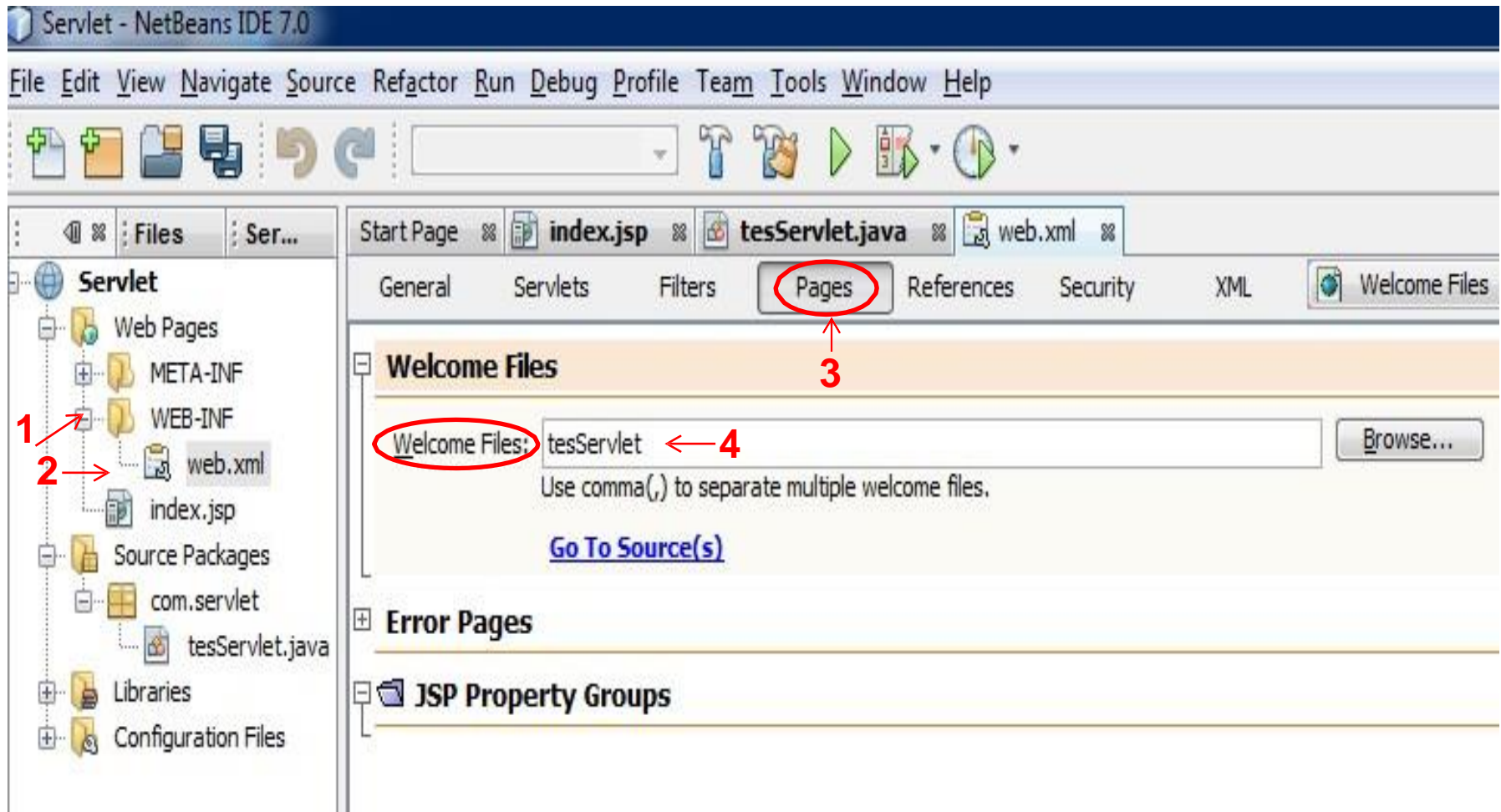
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>STMIK Subang</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>SELAMAT BELAJAR SERVLET </h1>");
    out.println("</body>");
    out.println("</html>");
}
```

# Memulai Servlet Pertama #17

Agar servlet yang kita buat bisa muncul di web browser, maka atur pada :

web.xml → Pages → Welcome Files → tulis tesServlet

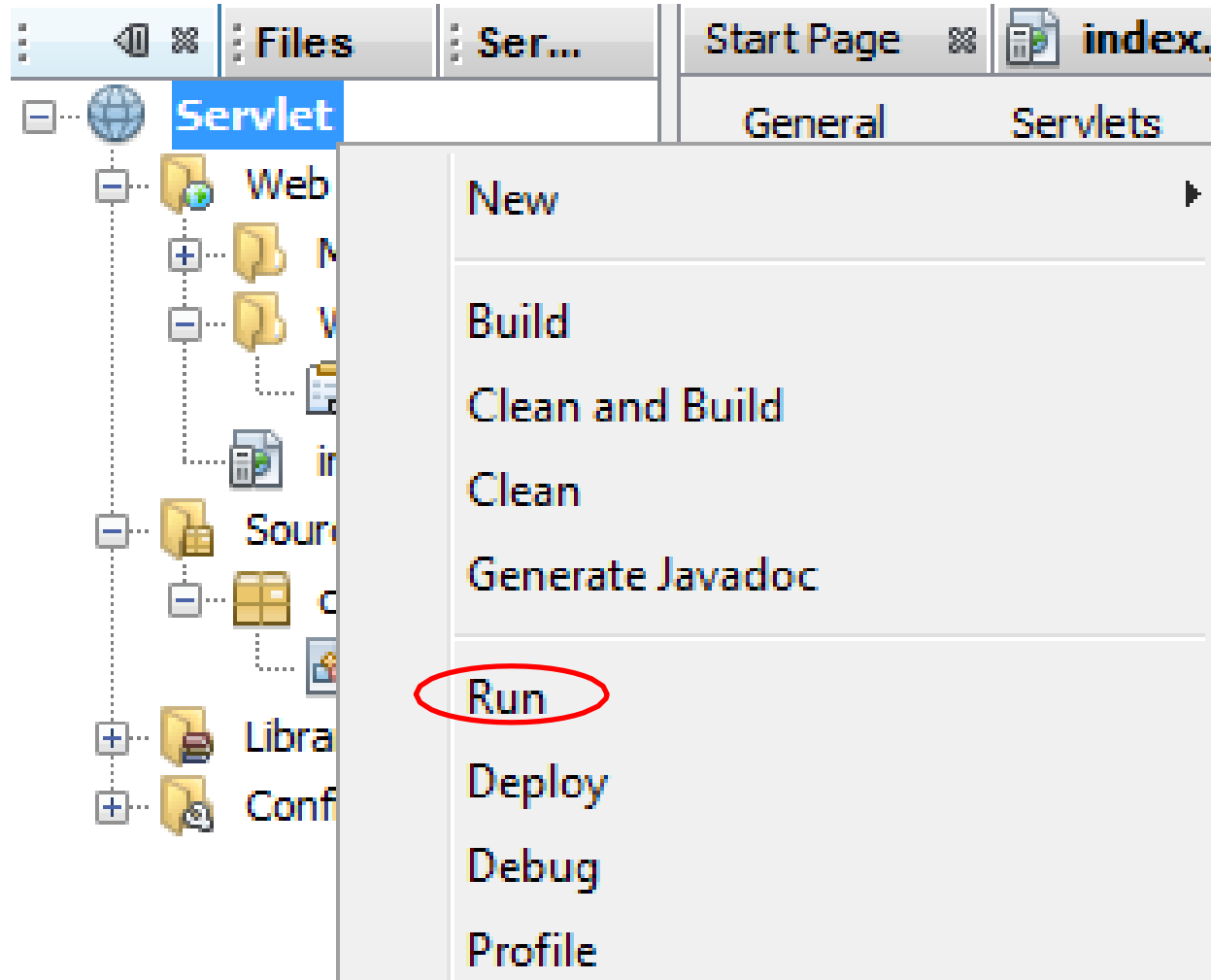
# Memulai Servlet Pertama #18



# Memulai Servlet Pertama #19

Untuk menampilkan di web browser, Klik kanan pada nama project, kemudian pilih **Run**

# Memulai Servlet Pertama #20



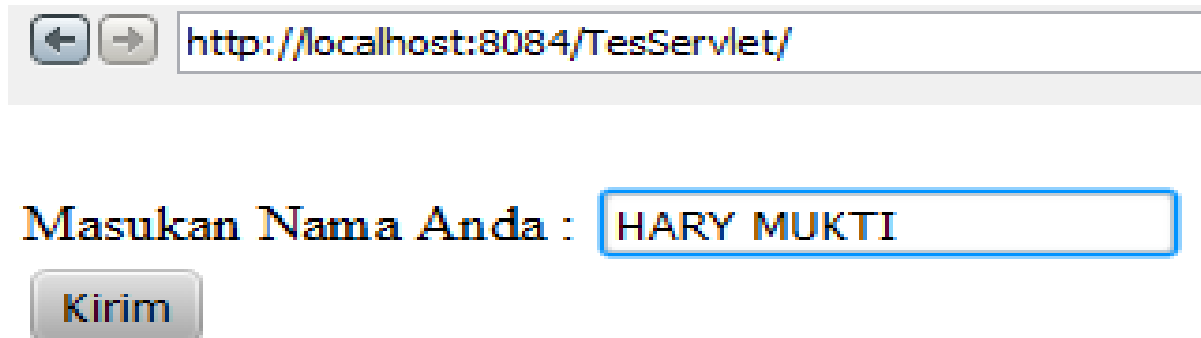
# Memulai Servlet Pertama #21



# Form Data dan Parameter

- `getParameter`

Contoh aplikasi yang me-respon kepada user dipresentasikan dalam bentuk form, adalah sebagai berikut ini :



The screenshot shows a web browser window. The address bar contains the URL `http://localhost:8084/TesServlet/`. Below the address bar, there is a form with the label "Masukan Nama Anda :". The input field contains the text "HARY MUKTI". Below the input field is a button labeled "Kirim".

Diberikan form diatas, yang mengambil nama user, dan kita ingin untuk memberikan response tertentu kepada nama user tersebut.



# Method `getParameter`

- Java telah menyediakan method `getParameter` didalam object `HttpServletRequest`.
- Fungsi method `getParameter` : mengembalikan sebuah value dari elemen pertama dari nama yang diberikan.
- Method ini akan mengambil sebuah parameter `String` (nama dari element form dimana valuenya dapat diambil kembali)
- Method ini akan mengembalikan response sebuah `String`(sebuah value dari form elemen spesifik dari form).

# Method `getParameterValues`

- Fungsi : untuk mendapatkan semua value.
- Method ini akan mengambil value dalam bentuk String sebagai nama dari elemen form, akan tetapi akan mempunyai nilai kembalian String array.

# Method `getParameterNames`

- Fungsi : mengembalikan sebuah enumeration dari nama yang berasal dari elemen-elemen form yang telah digabungkan pada saat user me-request.

# Form dengan Parameter #1

- Buat project baru
- Tambahkan sebuah servlet dengan nama Servlet

# Kode Servlet.java

- Edit dan tuliskan kode berikut:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String Nama = request.getParameter("nama");
    out.println("<HTML><BODY>");
    out.println("Selamat Datang, " + Nama + " !");
    out.println("</BODY></HTML>");
}
```

# Kode HTML (index.html)

```
<!--
```

```
To change this template, choose Tools | Templates  
and open the template in the editor.
```

```
-->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title></title>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
  </head>
```

```
  <body>
```

```
    <form action="parameter" method="post">
```

```
    Masukan Nama Anda : <input type="text" name="nama"/>
```

```
    <input type="submit" value="Kirim"/>
```

```
  </form>
```

```
  </body>
```

```
</html>
```

# Form dengan Parameter #3

- Atur web.xml → Servlets :

General **Servlets** Filters Pages References Security XML Servlet ▼

**Servlets**

**Servlet -> /parameter**

Servlet Name:  Startup Order:

Description:

☒ Servlet Class:   [Go to Source](#)

☐ JSP File:   [Go to Source](#)

URL Pattern(s):  Use comma (,) to separate multiple patterns.

# Form dengan Parameter #3

- Atur web.xml → Servlets :

General **Servlets** Filters Pages References Security XML Servlet ▼

**Servlets**

**Servlet -> /parameter**

Servlet Name:  Startup Order:

Description:

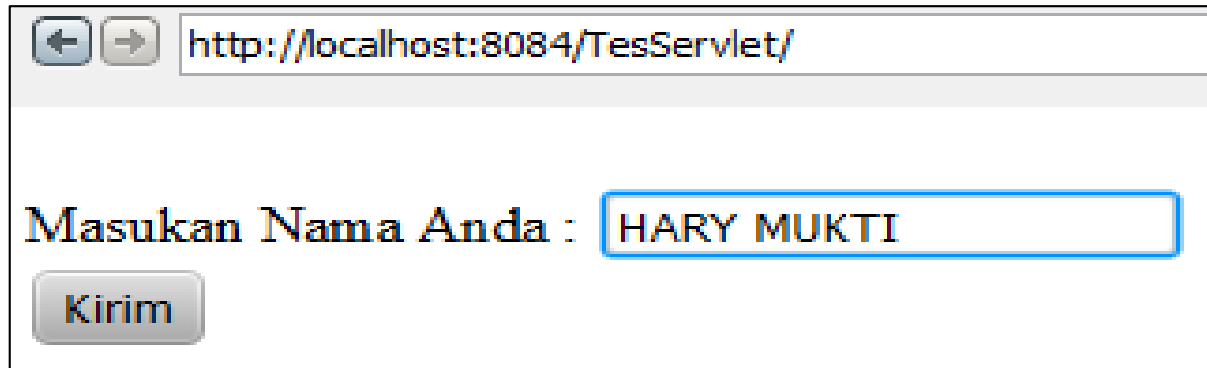
☒ Servlet Class:   [Go to Source](#)

☐ JSP File:   [Go to Source](#)

URL Pattern(s):  Use comma (,) to separate multiple patterns.

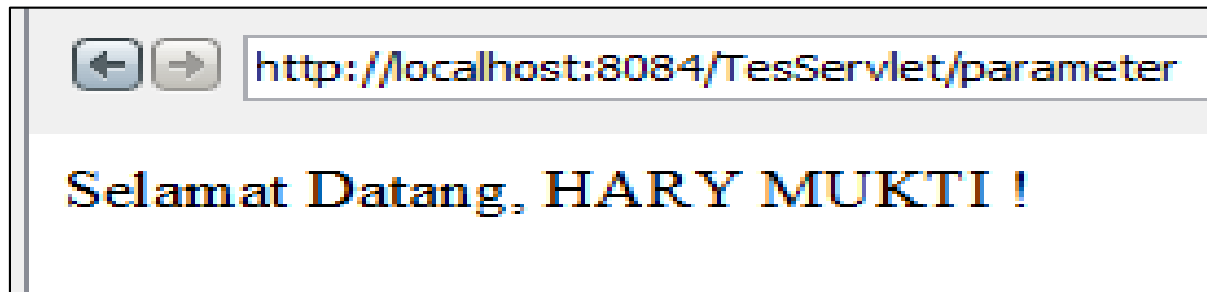


# Form dengan Parameter #4



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8084/TesServlet/`. The main content area displays the text "Masukan Nama Anda :" followed by a text input field containing the name "HARY MUKTI". Below the input field is a button labeled "Kirim".

Gambar 1 Halaman 1



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8084/TesServlet/parameter`. The main content area displays the text "Selamat Datang, HARY MUKTI !".

Gambar 2 Halaman 2