

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/316999541>

Modul Java Web Programming

Book · January 2014

CITATIONS

0

READS

11,086

1 author:



Indra Indra

Universitas Budi Luhur

9 PUBLICATIONS 4 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Skripsi 2017/2018 GASAL [View project](#)

MODUL PRAKTIKUM JAVA WEB PROGRAMMING



**Oleh :
Indra, S. Kom, M.T.I**

**FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR
JAKARTA
2014**

KATA PENGANTAR

Dengan mengucapkan puji dan syukur kehadirat Alloh SWT yang Maha Pengasih dan Maha Penyayang yang telah memberikan rahmat serta karunia-Nya sehingga penulis dapat menyelesaikan modul praktikum java web programming ini dengan maksimal.

Modul praktikum ini diharapkan dapat menjadi sarana dan media pembelajaran bagi mahasiswa pada umumnya dan khususnya mahasiswa dengan jurusan Teknik Informatika. Mahasiswa diharapkan dapat lebih mudah dalam memahami dan menerapkan konsep dan contoh program java web programming dalam kasus yang dihadapi dalam kehidupan sehari-hari. Modul secara ringkas berisi konsep dari beberapa bagian penting dalam pembelajaran java web programming dan contoh program untuk lebih memahami penerapan konsep java web programming tersebut.

Tak ada gading yang tak retak, penulis menyadari modul ini masih memiliki banyak kekurangan. Kritik dan saran yang bersifat membangun bagi pengembangan penulis sangatlah kami harapkan. Semoga modul ini memberikan manfaat bagi perkembangan dunia teknologi dan informasi terutama dalam mensosialisasikan pemrograman java kepada insan pembelajar.

Jakarta, 03 Januari 2014

Indra,S.Kom, M.T.I

**LEMBAR PENGESAHANMODUL PRAKTIKUM
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR**

Nama Mata Kuliah : Java Web Programming
Nama Modul : Modul Praktikum Java Web Programming
Penyusun : Indra, S.Kom, M.T.I
Kerangka Modul :

Pertemuan 1 : Pengenalan Java Web Programming dan Instalasi
Netbeans dengan web server Tomcat
Pertemuan 2 : Dasar-Dasar Servlet
Pertemuan 3 : Response Redirecting
Pertemuan 4 : Cookies dan Filtering
Pertemuan 5 : Session Tracking
Pertemuan 6 : Dasar-Dasar JSP (Java Server Pages)
Pertemuan 7 : Pra UTS
Pertemuan 8 : Ujian Tengah Semester
Pertemuan 9 : Pembuatan Template
Pertemuan 10: Pembuatan Template Tingkat Lanjut
Pertemuan 11: Konsep Pembuatan Login di JSP
Pertemuan 12: Aplikasi Menyimpan, Menrubah dan Menghapus dengan
database MySQL
Pertemuan 13: Aplikasi ke Database Tingkat Lanjut
Pertemuan 14: Hosting dan Domain Java Web
Pertemuan 15: Pra UAS
Pertemuan 16: Ujian Akhir Semester

Daftar Pustaka

Jakarta, 03 Januari 2014

Ketua Program Studi
Teknik Informatika

(Muhammad Ainur Rony, S. Kom, M.T.I)

DAFTAR ISI

	Halaman
Kata Pengantar -----	i
Lembar Pengesahan -----	ii
Daftar Isi -----	ii
Pertemuan1 -----	5
Pertemuan2 -----	11
Pertemuan3 -----	17
Pertemuan4 -----	30
Pertemuan5 -----	50
Pertemuan6 -----	65
Pertemuan 7 -----	74
Pertemuan8 -----	75
Pertemuan 9 -----	76
Pertemuan 10 -----	80
Pertemuan 11 -----	82
Pertemuan 12 -----	85
Pertemuan 13 -----	90
Pertemuan 14 -----	91
Pertemuan 15 -----	96
Pertemuan 16 -----	97

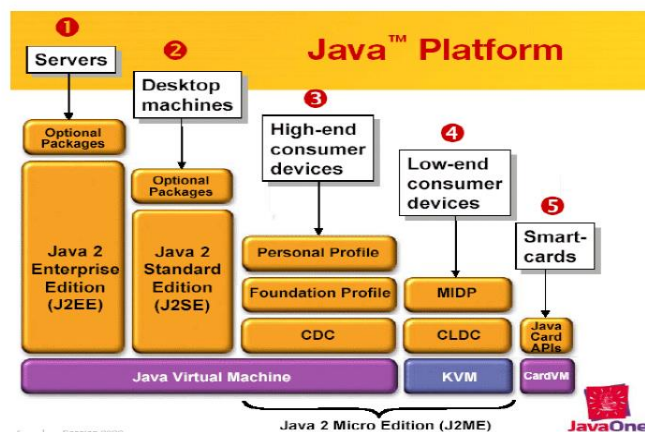
Pertemuan 1

Pengenalan Java Web Programming, dan Konfigurasi IDE Netbeans dengan web server Tomcat

1. Java Web Programming

Java Web adalah suatu device dari java yang digunakan untuk membuat aplikasi skala besar. Web Component adalah komponen-komponen java yang digunakan untuk membuat aplikasi web seperti Servlet dan JSP. Web Component dapat berkomunikasi dengan komponen java lain serta memanfaatkannya, seperti menggunakan komponen JDBC untuk mengakses database, komponen JMS & JavaMail untuk mengirim email.

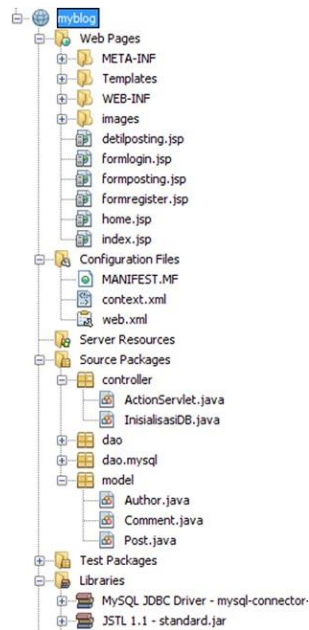
Gambaran arsitektur Java



Web Container adalah web server berbasis java yang menampung web-component serta menyediakan layanan bagi web component seperti request dispatching, security, serta lifecycle management. Beberapa pembuat Web Container antara lain :

- Free software : Apache Tomcat, JResin, Jetty, Sun Glassfish.
- Sun SJSAS, JBoss, IBM Websphere, Oracle Weblogic.

Salah satu free software ERP yang menggunakan web container Apache Tomcat adalah Compiere dan Adampiere.



Gambar : Struktur Web Application

Web Application adalah sebuah paket aplikasi yang siap di deploy (install) ke web container. Web application terdiri dari :

- Web component (Servlet, JSP, dsb).
- File-file lain seperti; HTML, gambar.
- Library : Driver database, JSTL, dsb.
- Deployment descriptor (web.xml).

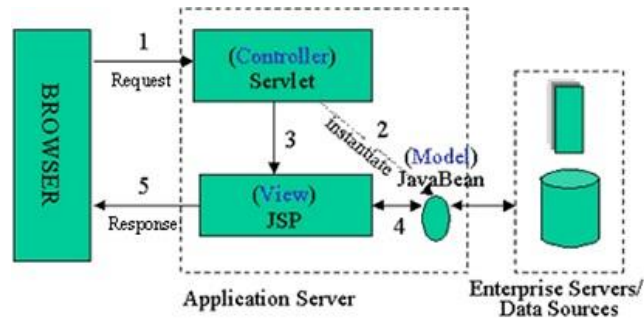
Paket aplikasi berbentuk 1 file berekstensi *.war (web archive), setelah dideploy akan terekstrak menjadi file-file dengan struktur khusus seperti gambar di atas.

2. MVC (Model-View-Controller)

MVC adalah model pembuatan sistem/program yang memisahkannya menjadi Model, View, dan Controller. Contoh MVC adalah struts, hibernate, spring, dll. Konsep MVC (model-view-controller) diciptakan agar tanggung jawab tiap individu di dalam tim pengembang software menjadi jelas. Konsep MVC membagi tugas sebagai berikut :

- Programmer berfokus pada Controller yang mengatur DFD (Data Flow Diagram) atau proses bisnis dari suatu aplikasi web.

- DBA berfokus pada model yang menyusun ERD (Entity Relationship Diagram) dalam bentuk ORM (Object Relationship Mapping).
- Designer berfokus pada View (estetika) dari tampilan web saja.



Gambar : Alur MVC pada aplikasi Java Web

Alur MVC pada aplikasi Java Web adalah :

1. Web browser mengirim request ke web container dan diterima oleh Servlet sebagai Controller.
2. Controller bertugas mengecek request lalu diproses. Output yang dihasilkan controller bisa berbentuk objek/java bean. Model bertugas merepresentasi data-data pada database dalam bentuk objek-objek yang saling berhubungan atau biasa disebut ORM (Object Relationship Mapping).
3. Controller kemudian menyerahkan tugas View (file JSP) untuk menampilkan objek.
4. View mengambil data berbentuk objek/java bean lalu memprosesnya.
5. View mengirim response ke web browser untuk menampilkan data dengan syntax EL dan library JSTL.

3. Framework

Framework adalah kerangka kerja. Framework juga dapat diartikan sebagai kumpulan script (terutama class dan function) yang dapat membantu developer/programmer dalam menangani berbagai masalah-

masalah dalam pemrograman seperti koneksi ke database, pemanggilan variabel, file, dll sehingga developer lebih fokus dan lebih cepat membangun aplikasi.

Bisa juga dikatakan framework sebagai modul-modul program yang dibuat untuk memudahkan pembuatan program J2EE. Framework ini berisi guideline bagaimana membuat program J2EE yang sudah tersusun dengan rapi dan didalamnya sudah berisi MVC. Secara sederhana bisa dijelaskan bahwa framework adalah kumpulan fungsi (libraries), maka seorang programmer tidak perlu lagi membuat fungsi-fungsi (biasanya disebut kumpulan library) dari awal, programmer tinggal memanggil kumpulan library atau fungsi yang sudah ada didalam framework, tentunya cara menggunakan fungsi-fungsi ini sudah ditentukan oleh framework. Beberapa contoh fungsi-fungsi standar yang telah tersedia dalam suatu framework adalah fungsi paging, enkripsi, email, SEO, session, security, kalender, bahasa, manipulasi gambar, grafik, tabel bergaya zebra, validasi, upload, captcha, proteksi terhadap XSS (XSS filtering), template, kompresi, XML, dan lain-lain.

4. Spring

Spring atau lebih tepatnya SpringMVC adalah teknologi yang memisahkan data, business logic, dan presentation layer. SpringMVC yang terdiri dari IOC(Injection Of Control) atau Dependency Injection telah membuat framework MVC dapat saling dipertukarkan. Teknologi lain yang berkembang sebelum SpringMVC adalah Struts, WebWork, Tapestry, JSF, dan Stripes.

Spring merupakan sebuah framework (kerangka kerja) yang digunakan untuk membangun sebuah aplikasi Enterprise. Spring termasuk framework yang lightweight (ringan) untuk mendukung secara penuh dalam pengembangan aplikasi Enterprise siap pakai. Spring dapat digunakan untuk melakukan pengaturan deklarasi manajemen transaksi, remote access dengan menggunakan RMI atau layanan web lainnya, fasilitas mailing, dan beragam opsi untuk pengaturan data ke database.

Spring juga memungkinkan kita menggunakan hanya modul-modul tertentu sehingga kita tidak usah menggunakan semua modul spring dalam aplikasi apabila tidak ditemukan.

Fitur-fitur dari Spring Framework :

- Transaction Management : Spring framework menyediakan sebuah layer abstrak yang generik untuk manajemen transaksi, sehingga memudahkan para developer dalam melakukan manajemen transaksi.
- JDBC Exception Handling : Layer abstrak JDBC menawarkan exception yang bersifat hierarki sehingga memudahkan penanganan error.
- Integration with Hibernate, JDO, and iBatis : Spring menawarkan layanan integrasi terbaik dengan Hibernate, JDO, dan iBatis.
- AOP Framework : Spring merupakan framework AOP terbaik yang pernah ada.
- MVC Framework : Spring hadir dengan framework aplikasi web MVC, yang dibangun di atas inti Spring. Spring merupakan framework yang sangat fleksibel dalam pengaturan strategi interface, dan mengakomodasi beberapa teknologi view seperti JSP, Velocity, Tiles, iText, dan POI.

5. Software Yang Dibutuhkan Untuk Pengembangan Aplikasi

Untuk melakukan pengembangan sebuah aplikasi berbasis java web programming dibutuhkan beberapa software penunjang, antara lain :

- JDK 1.6
- NetBeans 7.3 (include Apache Tomcat)
- XAMPP
- MySQL Front

6. WAR Files

Sebuah aplikasi web merupakan sekelompok halaman HTML, halaman JSP, servlet, source file, yang dapat dikelola sebagai satu

kesatuan unit. Web archive (WAR) file merupakan sebuah package dari web application. File WAR dapat digunakan untuk mengimpor aplikasi web ke web server.

Selain project resources, WAR Files berisikan file descriptor dari deployment. Web deployment descriptor adalah sebuah file XML yang berisikan informasi deployment, tipe MIME, rincian konfigurasi session, dan pengaturan lainnya pada aplikasi web. Web deployment descriptor file (web.xml) memberikan informasi mengenai WAR file yang telah dibagikan oleh para pengembang aplikasi, perakitan, dan pengembang dalam lingkungan Java EE.

Pertemuan 2

Dasar-dasar Servlet

1. Definisi Servlet

Servlet adalah sebuah API dalam pemrograman java yang mempermudah para developer untuk menambahkan konten-konten dinamis dalam sebuah web server yang mengimplementasikan platform java. Spesifikasi Servlet dibuat oleh Sun Microsystems. Hingga 10 Mei 2006, versi dari spesifikasi Servlet sudah sampai versi 2.5.

Konten yang dihasilkan dari Servlet API ini pada umumnya berupa HTML, tetapi tidak menutup kemungkinan untuk menghasilkan konten XML dan konten-konten yang lain. Servlet sendiri dibuat untuk menandingi teknologi-teknologi konten web yang dinamis lainnya seperti PHP, CGI, ASP.NET.

Servlet API terdapat didalam javax.servlet package. Dalam sebuah Web Server, Servlet berinteraksi dengan sebuah Web Container, dimana Web Container ini bertanggung jawab penuh terhadap daur hidup, pemetaan URL terhadap Servlet tertentu lainnya. Servlet sendiri sebenarnya adalah sebuah object (dari class interface) yang menerima request, kemudian menggenerate response berdasarkan request yang diterima tadi. Sedangkan untuk spesifik HTTP, disediakan sub-class dari Servlet, yaitu HttpServlet yang didalamnya terdapat juga obyek untuk manajemen session. Sedangkan daur hidup dari sebuah Servlet sendiri ada 4 state yaitu:

- a. Pada saat startup, class-class Servlet di load dalam Web Container. Web Container kemudian memanggil method `init()`. Method ini yang bertugas menginisialisasi Servlet dan harus dipanggil sebelum Servlet melayani requests. Method `init()` ini hanya dipanggil sekali. Setelah method `init()` dipanggil, barulah Servlet bisa melayani requests.
- b. Setiap request yang masuk dilayani dan ditangani oleh thread yang berbeda. Web Container memanggil method `service()` untuk setiap request yang masuk. Method ini menentukan jenis

request yang masuk, kemudian menentukan method mana yang akan handle request tersebut.

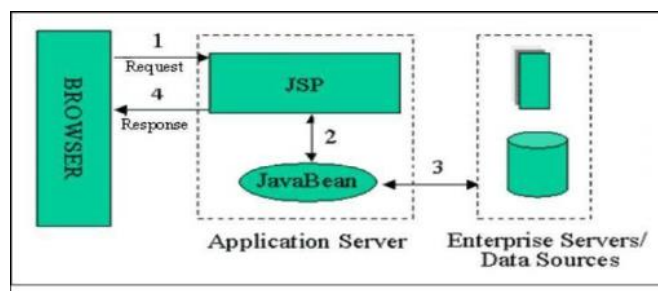
- c. State terakhir adalah ketika Web Container memanggil method `destroy()`. Method ini menyerupai method `init()`, hanya dipanggil sekali, dan menandakan bahwa sudah tidak ada lagi layanan dari Servlet.
- d. `ServletContext` hanya ada satu di setiap aplikasi, dan dapat digunakan oleh semua Servlet yang terlibat di aplikasi tersebut. Di sisi lain, masing-masing Servlet tadi memiliki `ServletConfig` sendiri-sendiri. `ServletConfig` ini menyediakan inisialisasi parameter untuk Servlet tersebut.

2. Gambaran Arsitektur Servlet

Di dalam pemrograman web menggunakan JSP dan Servlet terdapat 2 macam desain untuk implementasi pola MCV (Modeller-View-Controller).

1. Page-centric Architecture

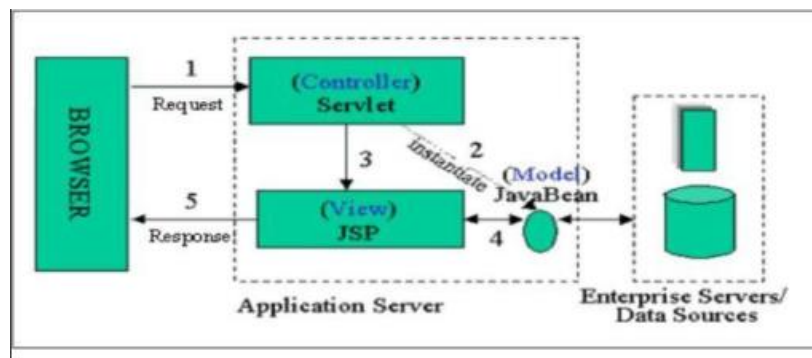
Di dalam arsitektur ini, sebenarnya tidak ada perbedaan antara Viewer dengan Controller karena keduanya ditangani oleh sebuah file JSP yang sama. Hal ini dapat dilakukan karena pada dasarnya sebuah file JSP adalah sebuah servlet. Oleh karena itu JSP memiliki atribut-atribut yang sama dengan sebuah Servlet dan memiliki banyak kemudahan. Model diimplementasikan dengan bentuk `JavaBean`. Arsitektur ini nantinya akan sangat mirip dengan jika kita melakukan pengembangan pada bahasa PHP.



Gambar : Page-centric Architecture

2. Servlet-centric Architecture

Pada arsitektur ini sebuah servlet bertugas untuk menerima request dari web browser kemudian melakukan pemrosesan model berdasarkan request tersebut. Setelah itu servlet akan meminta file JSP untuk menangani tampilan dari data-data. Dengan arsitektur ini kita dapat melakukan pengembangan seperti layaknya kita melakukan pengembangan pada aplikasi Java seperti biasanya. JSP digunakan hanya untuk memudahkan desain tampilan.



Gambar :Servlet-centric Architecture

3. **HttpServletRequest**

HttpServletRequest adalah sebuah object yang memberikan akses bagi segala informasi terhadap client request, termasuk apa saja bentuk parameter value yang dapat diletakkan pada Http request header, Http request method yang telah mereka gunakan, dan sebagainya.

Beberapa method yang sering digunakan, yang berada di dalam interface HttpServletRequest :

1. `getMethod()`

Fungsi `getMethod()` adalah untuk mengambil jenis method HTTP yang digunakan. Nilainya bisa berupa get, post, put, dll.

2. `getRequestURI()`

Fungsi `getRequestURI()` adalah untuk mengambil URL yang dimulai setelah `namahost:port/` sampai sebelum nama parameter.

3. `getRequestURL()`

Fungsi `getRequestURL()` adalah untuk mengambil URL.

4. `getProtocol()`

Fungsi `getProtocol()` adalah untuk mengambil nama dan versi protokol yang digunakan.

5. `getRemoteAddr()`

Fungsi `getRemoteAddr()` adalah untuk mengambil alamat IP client.

6. `getRemotePort()`

Fungsi `getRemotePort()` adalah untuk mengambil nomor port client.

7. `getServerName()`

Fungsi `getServerName()` adalah untuk mengambil nama host dari server.

8. `getServerPort()`

Fungsi `getServerPort()` adalah untuk mengambil nomor port server.

9. `getServletPath()`

Fungsi `getServletPath()` adalah untuk mengambil URL yang memanggil servlet, dimulai dari karakter `'/'`.

10. `getContentType()`

Fungsi `getContentType()` adalah untuk mengambil jenis MIME (Multipurpose Internet Mail Extension).

11. `getHeaderNames()`

Fungsi `getHeaderNames()` adalah untuk mengambil nama header yang berada di object request.

12. `getHeader()`

Fungsi `getHeader()` adalah untuk mengambil nilai dari suatu header.

13. `getParameter()`

Fungsi `getParameter()` adalah untuk mengambil nilai parameter yang dikirimkan oleh client ke servlet.

14. `getParameterName()`

Fungsi `getParameterName()` adalah untuk mengambil nama dari semua parameter yang dikirimkan oleh client.

15. `getParameterValues`

Fungsi `getParameterValues()` adalah untuk mengambil nilai dari parameter yang memiliki multiple-value (misalnya komponen checkbox).

4. `HttpServletResponse`

`HttpServletResponse` adalah object yang terdiri dari semua method yang dibutuhkan oleh developer untuk memproduksi sebuah response yang akan dikirimkan kembali kepada client. Meliputi method-method yang harus di-set pada HTTP response header, untuk mendeklarasikan tipe MIME dari response, sebaik method yang digunakan untuk mengambil instance dari class Java I/O yang akan kita gunakan secara langsung untuk memproduksi output.

Beberapa method `HttpServletResponse` yang sering digunakan:

a. `encodeRedirectURL()`

Fungsi `encodeRedirectURL()` adalah untuk Menkodekan URL untuk penggunaan yang terbatas dalam metode `sendRedirect` atau, jika pengkodean tidak diperlukan, Mengembalikan URL yang tidak diubah.

b. `containsHeader()`

Fungsi `containsHeader()` adalah untuk mengembalikan boolean yang menunjukkan apakah named response header sudah di set.

c. `isCommitted()`

Fungsi `isCommitted()` adalah untuk mengembalikan boolean yang mengindikasikan jika response telah committed.

d. `addCookie()`

Fungsi `addCookie()` adalah untuk menambahkan spesifikasi cookie pada response.

e. `addDateHeader()`

Fungsi `addDateHeader()` adalah untuk menambahkan response header dengan nama dan nilai tanggal yang telah diberikan.

f. `reset()`

Fungsi `reset()` adalah untuk membersihkan semua data yang ada pada buffer termasuk status code dan headers.

g. `resetBuffer()`

Fungsi `resetBuffer()` adalah untuk membersihkan isi konten buffer tanpa membersihkan status code dan header.

h. `sendError()`

Fungsi `sendError()` adalah untuk mengirim error response ke client menggunakan spesifikasi tertentu

(pertemuan 2 belum selesai pak)

Pertemuan 3

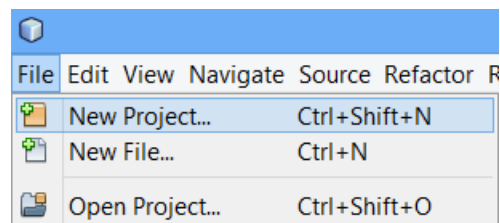
Response Redirecting

1. Objek RequestDispatcher

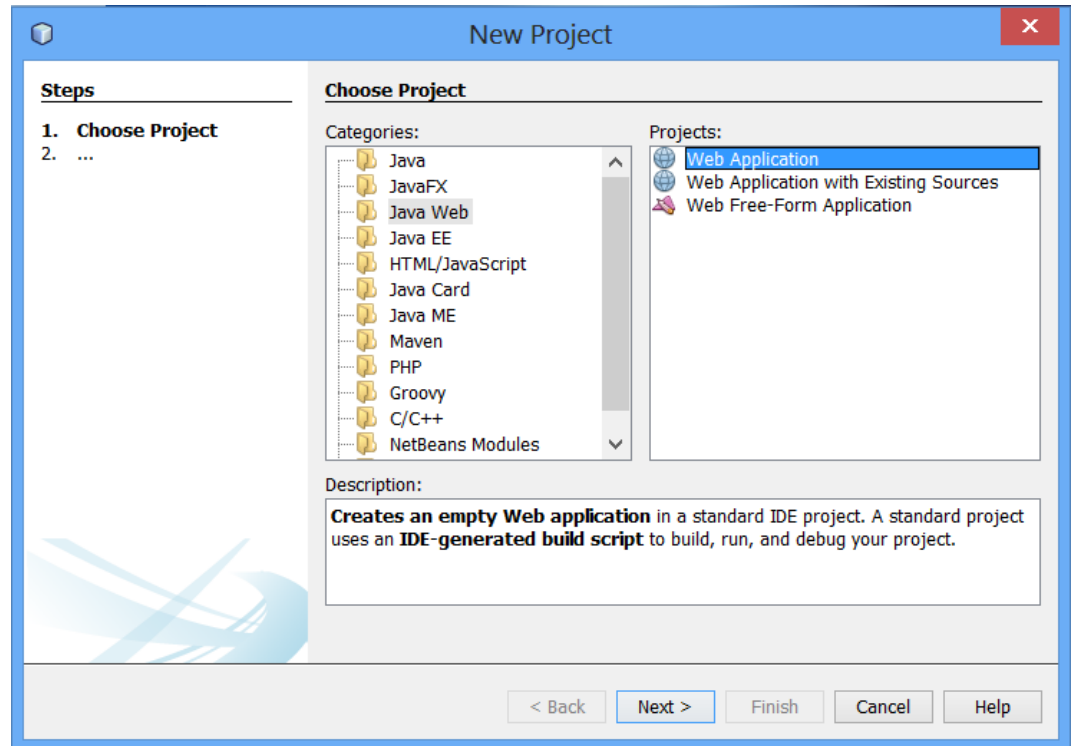
Sebuah RequestDispatcher merupakan Class JavaSW sangat penting yang memungkinkan untuk 'including' konten dalam permintaan / respon atau 'forwarding' permintaan / respon terhadap sumber daya. Sebagai contoh yang khas, servletW dapat menggunakan RequestDispatcher untuk menyertakan atau meneruskan permintaan / respons terhadap JSPW. Dalam pemrograman Model-View-Controller di Java, servlet biasanya berfungsi sebagai 'Controller'. Controller pada dasarnya berisi atau referensi kode untuk melakukan tindakan tertentu, dan memutuskan yang 'melihat' untuk mengirim pengguna untuk. Di Java, pandangan biasanya JSP. Dengan demikian, RequestDispatcher melakukan peran yang sangat penting dalam arsitektur Java MVCW karena dapat berfungsi sebagai mekanisme untuk 'pengendali' (servlet) untuk memperbolehkan pengguna untuk 'melihat' (JSP).

Pada contoh ini akan ditunjukkan bagaimana RequestDispatcher digunakan untuk memforward atau menginclude response dari servlet. Disini kira menggunakan index.html untuk mendapatkan username dan password dari user, Validate servlet akan memvalidasi password yang dimasukan oleh user, jika password yang dimasukan oleh user "mahasiswa" maka dia akan ke Welcome servlet jika tidak idex.html akan muncul kembali.

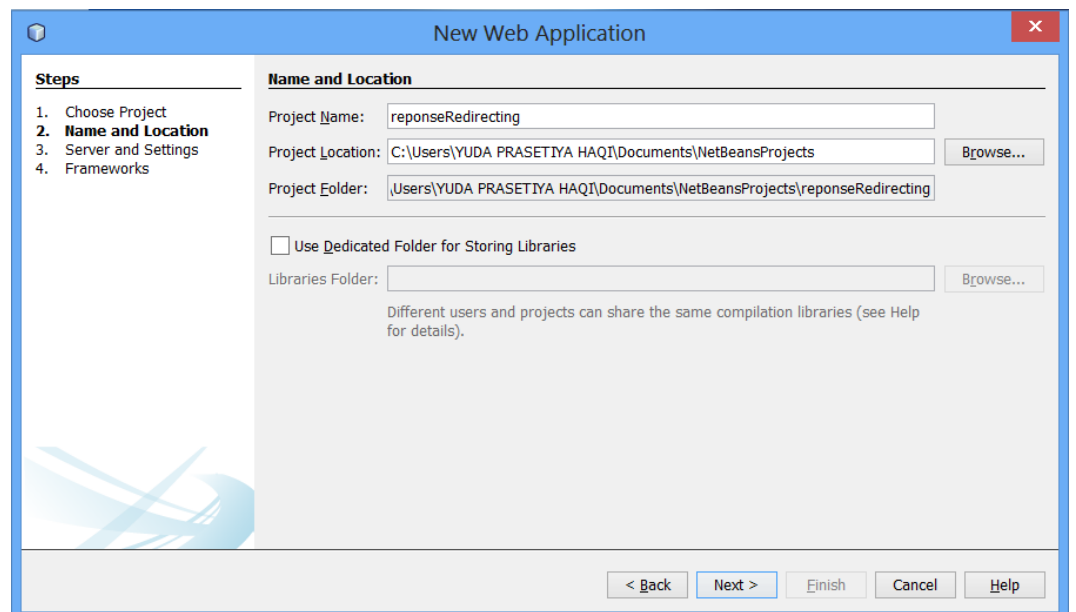
Pertama bikin project baru :



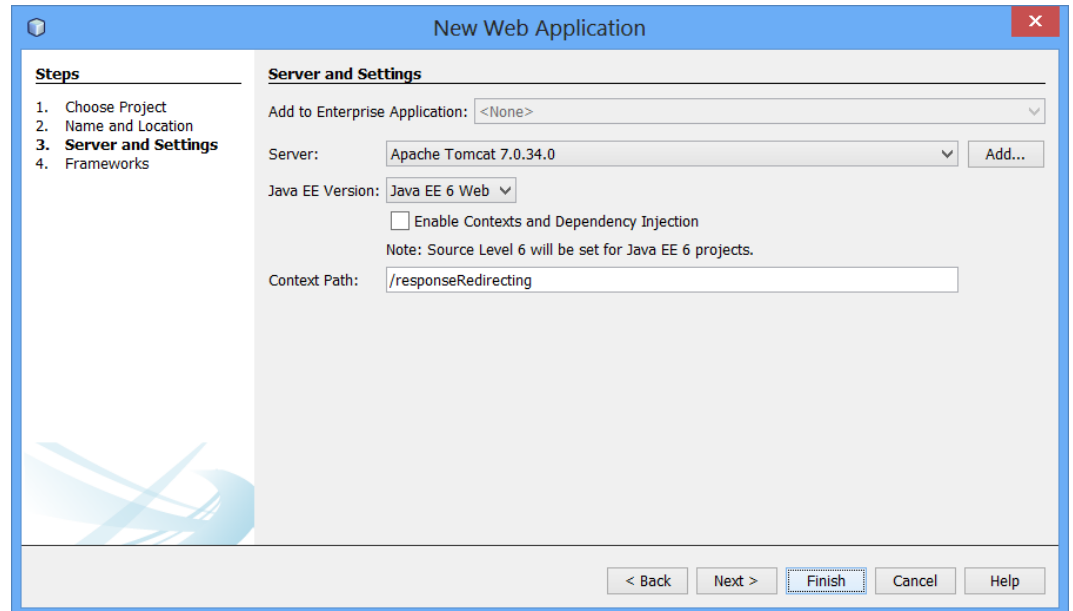
Pilih Java Web > Web Application lalu pilih **Next**



Beri nama project responseRedirecting lalu pilih **Next**



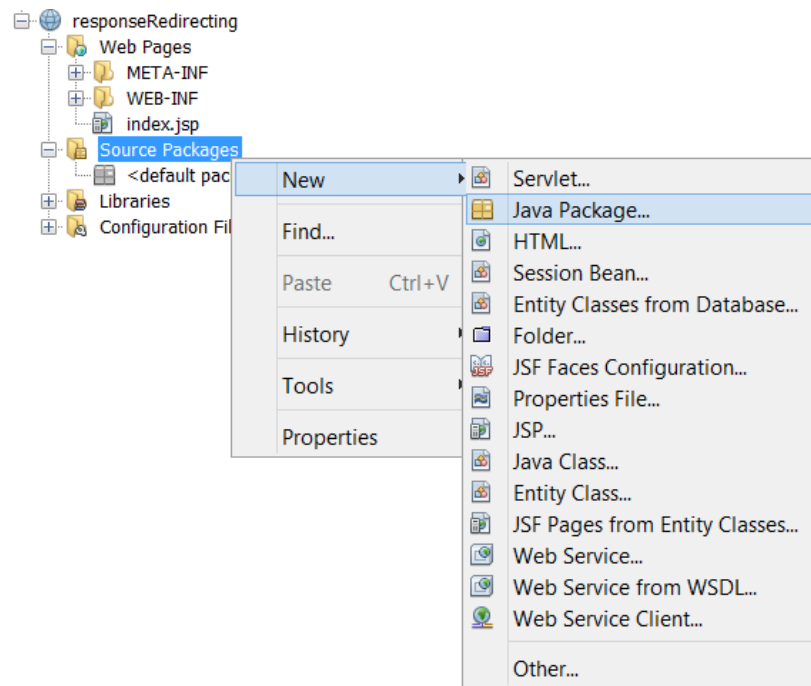
Pilih Apache Tomcat sebagai server lalu klik **Finish**



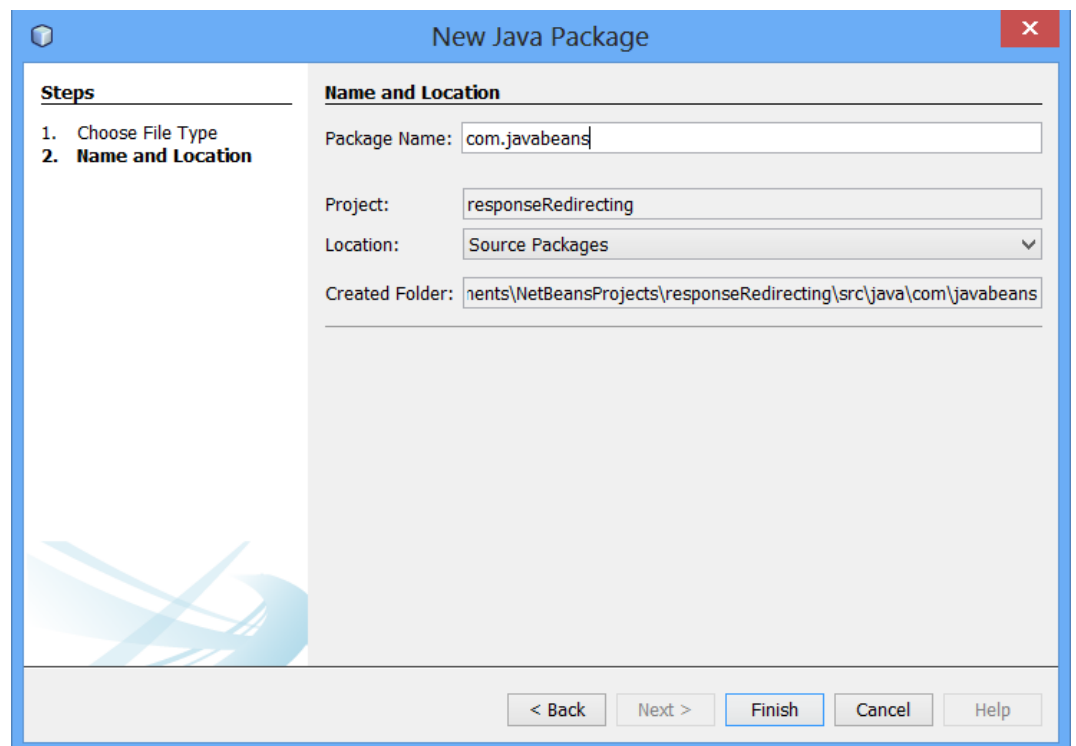
Ubah kodingan index.jsp menjadi

```
<form method ="post" action="Validate">
    Name : <input type="text" name="user"/> <br/>
    Password : <input type="password" name="pass"/> <br/>
    <input type="submit" value="submit"/>
</form>
```

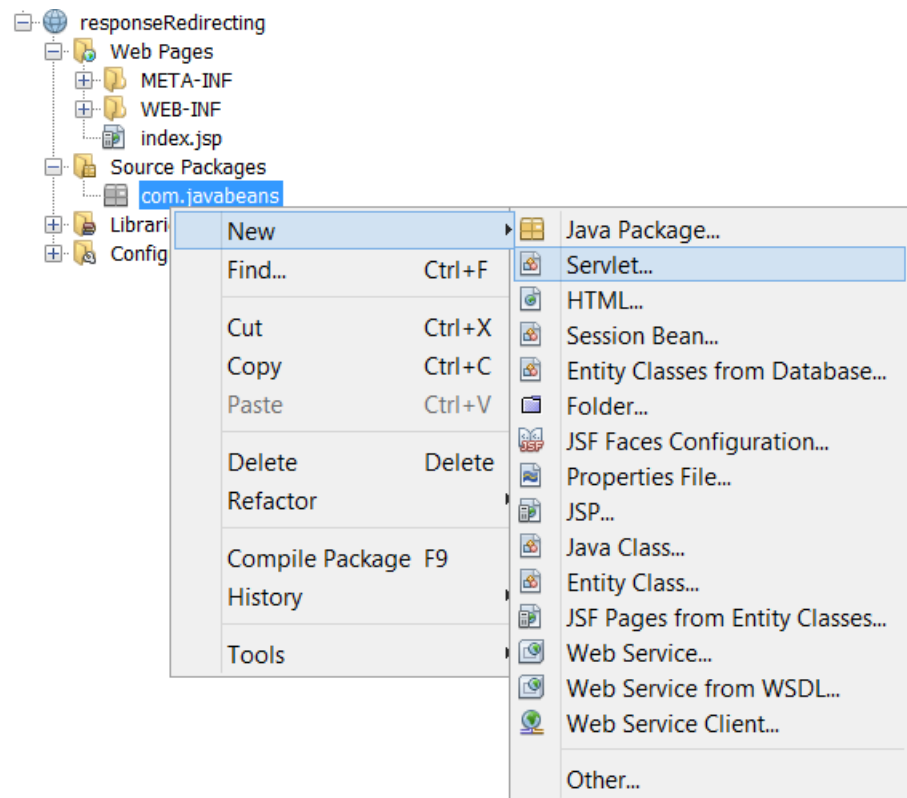
Tambahkan java package dengan klik kanan pada source package
pilih new lalu klik java package



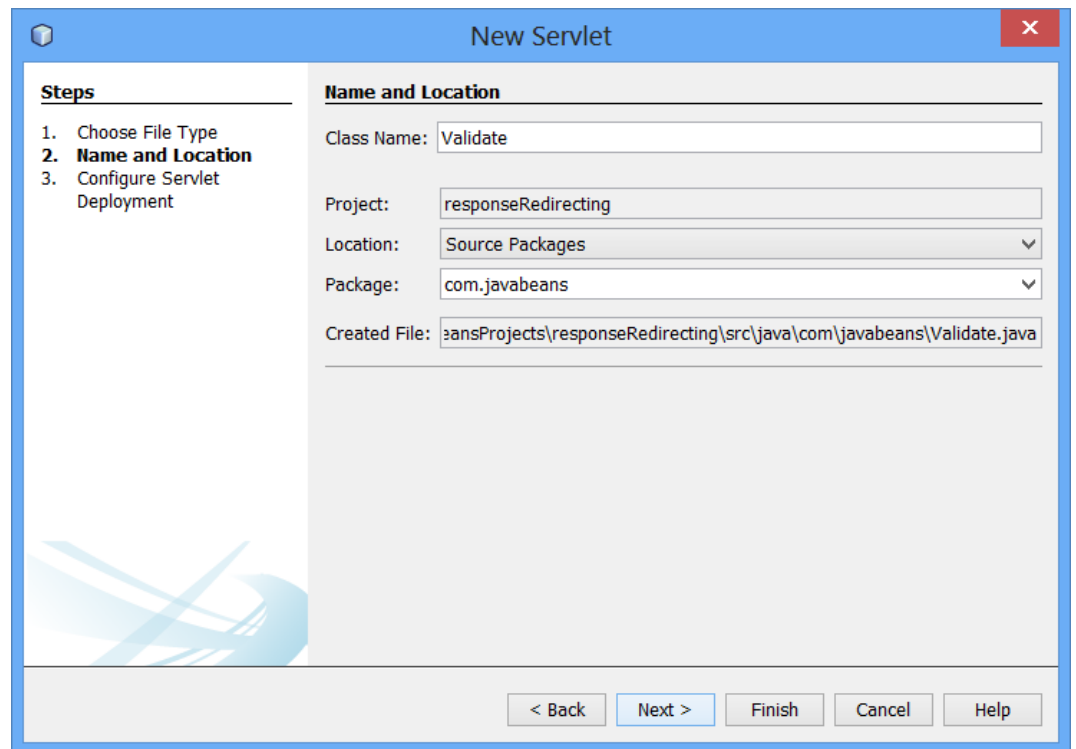
Beri nama pada com.javabeans pada java package



Klik kanan pada com.javabeans lalu pilih new > servlet



Beri nama Validate pada servlet lalu klik next



Pastikan servlet name dan URL pattern(s) sesuai lalu klik **Finish**

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

< Back Next > **Finish** Cancel Help

Ubah isi dari Validate.java menjadi seperti berikut

```

package com.javabeans;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

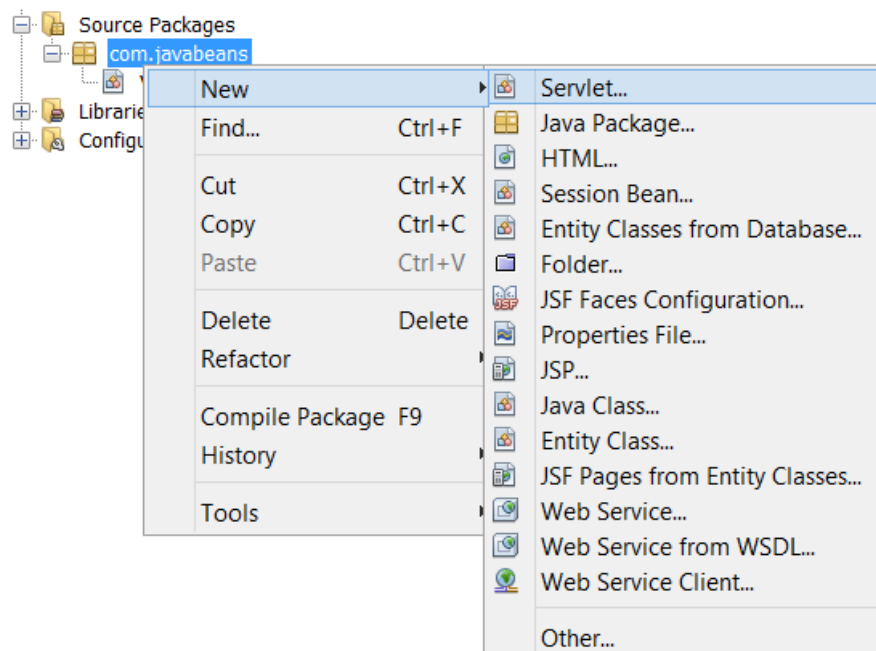
@WebServlet(name = "Validate", urlPatterns = {"/Validate"})
public class Validate extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String name = request.getParameter("user");
            String password = request.getParameter("pass");

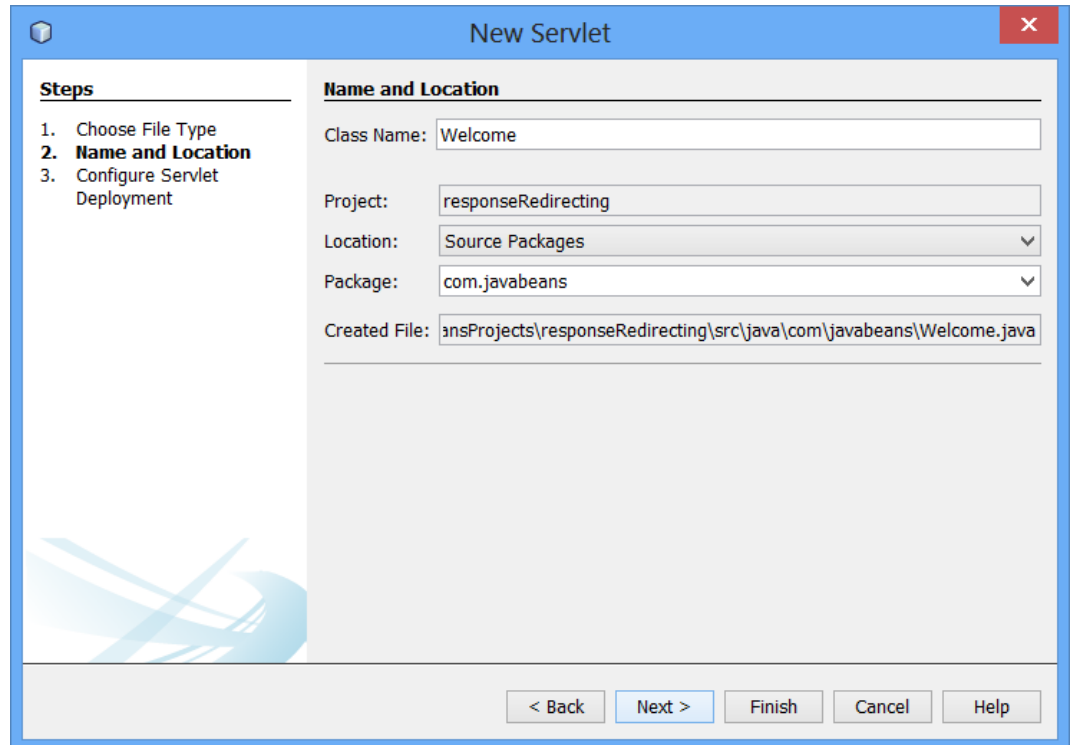
            if(password.equals("budiluhur"))
            {
                RequestDispatcher rd = request.getRequestDispatcher("Welcome");
                rd.forward(request, response);
            }
            else
            {
                out.println("<font color='red'><b>You have entered incorrect password</b></font>");
                RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
                rd.include(request, response);
            }
        } finally {
            out.close();
        }
    }
}

```

Klik kanan pada com.javabeans pilih New > Servlet



Beri nama Welcome pada Servlet klik **Next**



New Servlet

Steps

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

Name and Location

Class Name:

Project:

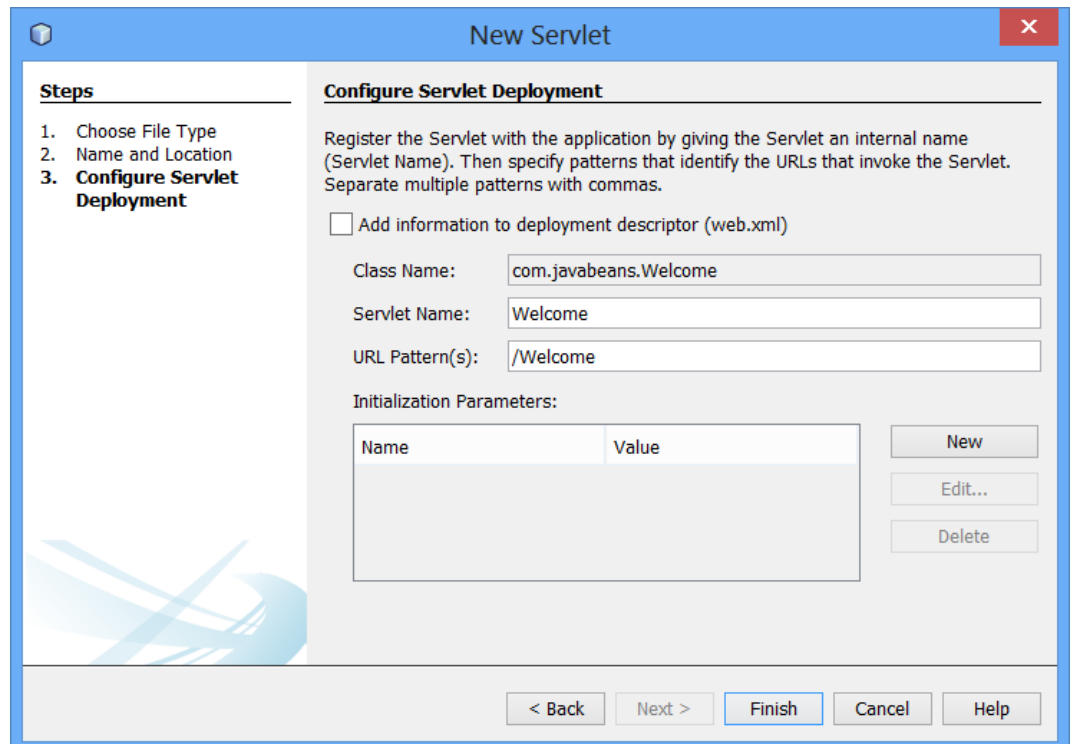
Location:

Package:

Created File:

< Back Next > Finish Cancel Help

Pastikan Servlet Name dan URL Pattern(s) sesuai klik **Finish**



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

< Back Next > Finish Cancel Help

Ubah isi Welcome.java menjadi seperti berikut

```

package com.javabeans;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

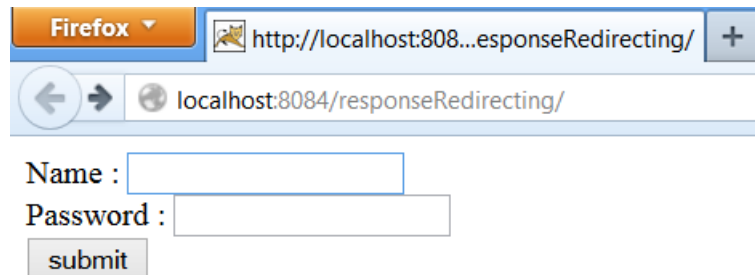
@WebServlet(name = "Welcome", urlPatterns = {"/Welcome"})
public class Welcome extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            out.println("<h2>Welcome user</h2>");
        } finally {
            out.close();
        }
    }
}

```

Setelah di running maka hasilnya adalah sebagai berikut



Firefox | http://localhost:808...esponseRedirecting/ +

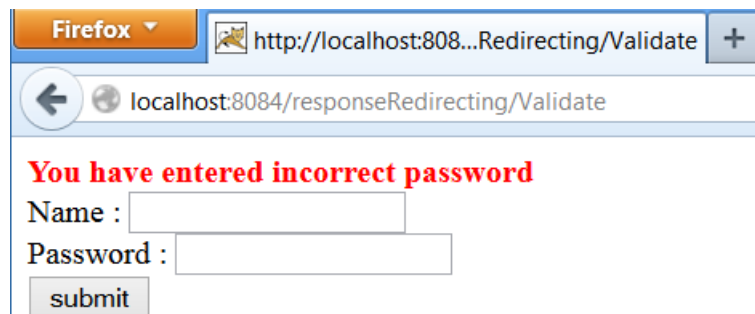
localhost:8084/responseRedirecting/

Name :

Password :

submit

Jika password salah akan muncul



Firefox | http://localhost:808...Redirecting/Validate +

localhost:8084/responseRedirecting/Validate

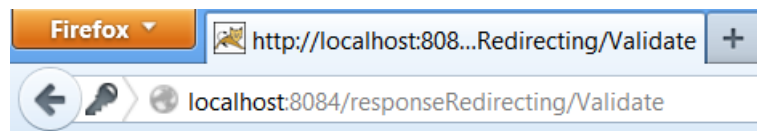
You have entered incorrect password

Name :

Password :

submit

Jika password benar akan muncul



Firefox | http://localhost:808...Redirecting/Validate +

localhost:8084/responseRedirecting/Validate

Welcome user

2. Method Include

Pada `include()` method hasil request dispatcher yang ditambahkan ke hasil keluaran sebelumnya dihasilkan oleh servlet `Validate`. Kita melihat bahwa `RequestDispatcher` objek `'include'` isi `index.jsp` dalam hasil dikirim kembali ke browser. Seperti yang Anda lihat pada Servlet `Validate`

```
package com.javabeans;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

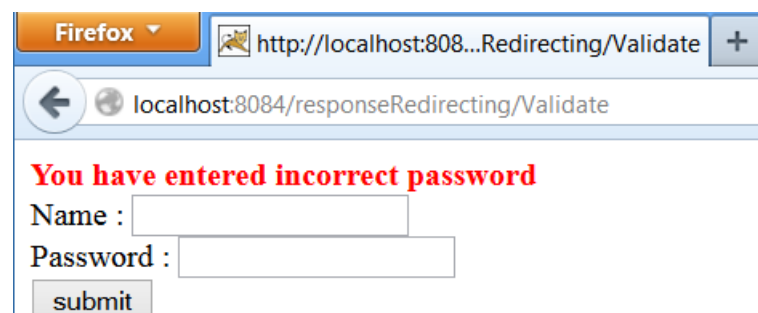
@WebServlet(name = "Validate", urlPatterns = {"/Validate"})
public class Validate extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String name = request.getParameter("user");
            String password = request.getParameter("pass");

            if(password.equals("budiluhur"))
            {
                RequestDispatcher rd = request.getRequestDispatcher("Welcome");
                rd.forward(request, response);
            }
            else
            {
                out.println("<font color='red'><b>You have entered incorrect password</b></font>");
                RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
                rd.include(request, response);
            }
        } finally {
            out.close();
        }
    }
}
```

Kodingan yang berada pada kotak merah akan di eksekusi ketika password tidak sama dengan budiluhur karena `'include'` maka akan mencetak tulisan `"You have entered incorrect password"` dan mencetak form yang ada pada `index.jsp`

Hasilnya :



The screenshot shows a Firefox browser window. The address bar displays `http://localhost:808...Redirecting/Validate`. Below the address bar, the URL `localhost:8084/responseRedirecting/Validate` is shown. The main content area displays the message **You have entered incorrect password** in red text. Below this message, there is a form with two input fields: **Name :** and **Password :**. A **submit** button is located at the bottom of the form.

3. Method Forward

Jika pada forward() objek RequestDispatcher 'forward' permintaan / respon terhadap Welcome.java. Berbeda halnya dengan 'include', method 'forward' akan membuang output sebelumnya yang Servlet Validasi telah menulis untuk respon. Akibatnya, kita hanya melihat output yang dihasilkan oleh Welcome.java.

```
package com.javabeans;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;

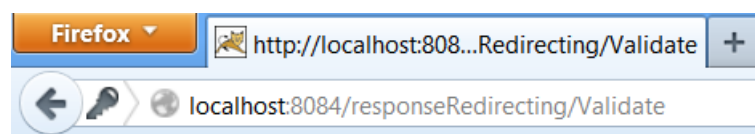
@WebServlet(name = "Validate", urlPatterns = {"/Validate"})
public class Validate extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String name = request.getParameter("user");
            String password = request.getParameter("pass");

            if(password.equals("budiluhur"))
            {
                RequestDispatcher rd = request.getRequestDispatcher("Welcome");
                rd.forward(request, response);
            }
            else
            {
                out.println("<font color='red'><b>You have entered incorrect password</b></font>");
                RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
                rd.include(request, response);
            }
        } finally {
            out.close();
        }
    }
}
```

Kodingan yang berada pada kotak merah akan di eksekusi ketika password sama dengan budiluhur karena 'forward' maka tidak akan memperdulikan Validate.java dan hanya akan mencetak yang ada pada Welcome.java

Hasilnya :



Welcome user

4. Method sendRedirect

The `sendRedirect()` method adalah `HttpServletResponse` interface dapat digunakan untuk mengarahkan respon terhadap sumber daya lain, mungkin servlet, jsp atau file html. Method ini menerima relatif serta URL absolut dan juga bekerja pada sisi client karena menggunakan bar url browser untuk membuat permintaan lain. Jadi, dapat bekerja dalam dan di luar server.

Perbedaan antara `forward()` method dan `sendRedirect()` method

Forward()	sendRedirect()
Ketika kita menggunakan <code>forward</code> permintaan metode adalah transfer ke sumber daya lain dalam server yang sama untuk diproses lebih lanjut.	Dalam kasus permintaan <code>sendRedirect</code> dapat mentransfer ke sumber daya lain untuk domain yang berbeda atau server yang berbeda untuk diproses lebih lanjut.
Dalam <code>forward</code> Web kontainer menangani semua proses internal dan client atau browser tidak terlibat.	Bila Anda menggunakan <code>sendRedirect</code> transfer kontainer permintaan client atau browser, sehingga url yang diberikan dalam method <code>sendRedirect</code> terlihat sebagai permintaan baru client.
Ketika <code>forward</code> dipanggil pada <code>requestDispatcher</code> object maka kita akan melewati permintaan dan object respon sehingga object permintaan lama kita hadir pada sumber daya baru	Dalam kasus <code>sendRedirect</code> permintaan lama dan object respon hilang karena itu diperlakukan sebagai permintaan baru oleh browser.

yang akan memproses permintaan kita.	
--------------------------------------	--

Secara visual kita tidak dapat melihat alamat yang di forwarded, karena bersifat transparent.	Di address bar kita dapat melihat alamat baru yang di redirected karena itu tidak bersifat transparent.
Menggunakan forward () metode lebih cepat dari mengirim redirect.	SendRedirect lebih lambat karena akan meload url tujuan dari awal karena permintaan benar-benar baru dan objek pada permintaan lama hilang.
Ketika kita redirect menggunakan forward dan kita ingin menggunakan data yang sama dalam sumber daya baru kita dapat menggunakan request.setAttribute() maka request object akan tersedia.	Namun dalam sendRedirect jika kita ingin menggunakan kita harus menyimpan data dalam session atau melewatkannya bersama dengan URL.
Syntax untuk forward: Forward(request,response)	Syntax untuk sendRedirect: sendRedirect(String URL)

Pertemuan 4

Cookies & Filtering

5. Send and Receive Cookies

Cookie adalah file teks yang disimpan pada komputer klien dan mereka disimpan untuk melacak informasi berbagai tujuan. Java Servlets secara transparan mendukung cookies HTTP.

Ada tiga langkah yang terlibat dalam mengidentifikasi kembali pengguna:

- Script Server mengirimkan satu set cookies ke browser. Sebagai contoh nama, usia, atau nomor identifikasi dll
- Browser menyimpan informasi ini pada komputer lokal untuk penggunaan masa depan.
- Ketika browser waktu berikutnya mengirim permintaan ke server web kemudian mengirimkan cookie informasi ke server dan server menggunakan informasi tersebut untuk mengidentifikasi pengguna.

Methods pada servlet cookies :

- `public void setDomain(String pattern)`
Metode ini menetapkan domain ke mana cookie berlaku, misalnya `student.budiluhur.ac.id`.
- `public String getDomain()`
Metode ini mendapatkan domain yang cookie berlaku, misalnya `student.budiluhur.ac.id`.
- `public void setMaxAge(int expiry)`
Metode ini menetapkan berapa banyak waktu (dalam hitungan detik) harus dilalui sebelum cookie berakhir. Jika Anda tidak menetapkan ini, cookie akan berlangsung hanya untuk sesi saat ini.

- `public int getMaxAge()`
Metode ini mengembalikan usia maksimal cookie, yang ditentukan dalam hitungan detik, Secara default, -1 menunjukkan cookie akan bertahan sampai browser mati.
- `public String getName()`
Metode ini mengembalikan nama cookie. Nama tidak dapat diubah setelah dibuat.
- `public void setValue(String newValue)`
Metode ini menetapkan nilai yang terkait dengan cookie.
- `public String getValue()`
Metode ini mendapat nilai yang terkait dengan cookie.
- `public void setPath(String uri)`
Metode ini menetapkan path yang berlaku untuk cookie ini. Jika Anda tidak menentukan path, cookie dikembalikan untuk semua URL di direktori yang sama seperti halaman saat ini serta semua subdirektori.
- `public String getPath()`
Metode ini untuk mendapat path yang berlaku pada cookie ini.
- `public void setSecure(boolean flag)`
Metode ini menetapkan nilai boolean yang menunjukkan apakah cookie hanya boleh dikirim melalui koneksi yang terenkripsi (yaitu SSL).
- `public void setComment(String purpose)`
Metode ini menentukan komentar yang menggambarkan tujuan cookie. Komentar sangat berguna jika browser menyajikan cookie kepada pengguna.
- `public String getComment()`
Metode ini mengembalikan komentar menggambarkan tujuan cookie ini, atau null jika cookie tidak memiliki komentar.

6. Filtering

Filter adalah komponen yang terdapat pada spesifikasi servlet version 2.3 yang berfungsi untuk men-intercept request dari client sebelum mereka mengakses resource di server dan untuk memanipulasi response dari server sebelum dikirim kembali ke client. Filter API terdiri dari tiga interface yang terletak di javax.servlet package yaitu :

- Filter Interface

Untuk membuat filter perlu untuk implement interface ini. Filter interface menyediakan tiga method life cycle untuk filter yang dibuat.

- 1) void destroy()

Metode ini dipanggil oleh container web untuk menunjukkan ke filter bahwa ia sedang ditarik dari penggunaannya.

- 2) void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

Metode ini dipanggil oleh container setiap kali sepasang request / response dilewatkan melalui rantai disebabkan oleh permintaan client untuk sumber daya pada akhir dari rantaiannya.

- 3) void init(FilterConfig filterConfig)

Metode ini disebut oleh kontainer web untuk menunjukkan ke filter yang sedang ditempatkan ke dalam layanan.

- FilterChain Interface

Sebuah filterChain adalah obyek yang disediakan oleh kontainer servlet untuk pengembang memberikan pandangan ke dalam rantai request dari request yang disaring untuk sumber daya. Filter menggunakan filterChain untuk memanggil filter berikutnya dalam rantai tersebut, atau jika filter yang dipanggil merupakan filter terakhir dalam rantai tersebut, maka untuk memanggil sumber daya pada akhir rantai.

- **FilterConfig Interface**

Object dari FilterConfig digunakan oleh web container untuk mengirimkan value pada saat inisialisasi filter. FilterConfig interface terdiri dari empat method yaitu :

- 1) **String getFilterName()**

Mengembalikan nama-filter dari filter ini sebagaimana didefinisikan dalam descriptor deployment.

- 2) **String getInitParameter(String name)**

Mengembalikan String yang berisi nilai parameter inisialisasi bernama, atau null jika parameter tidak ada.

- 3) **Enumeration getInitParameterNames()**

Mengembalikan nama-nama parameter inisialisasi servlet sebagai Pencacahan String objek, atau Pencacahan kosong jika servlet tidak memiliki parameter inisialisasi.




- 4) **ServletContext getServletContext()**

Mengembalikan referensi ke ServletContext di mana pemanggil mengeksekusi.

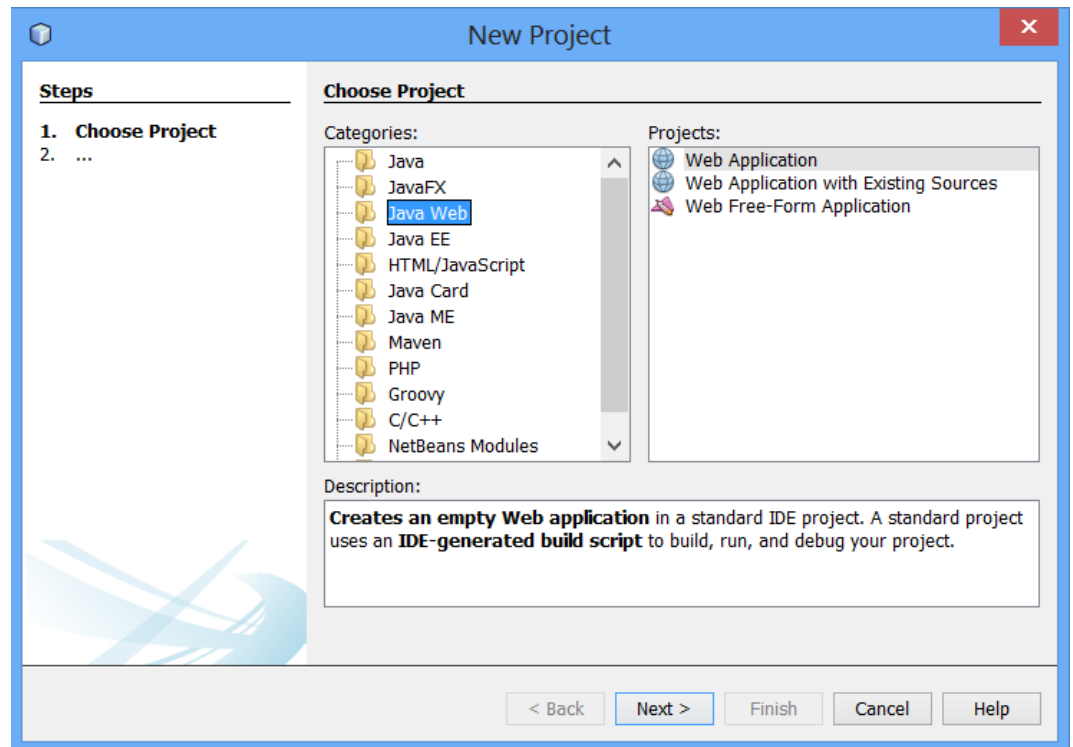
7. Membuat Filter Sederhana

Pertama-tama buat project baru

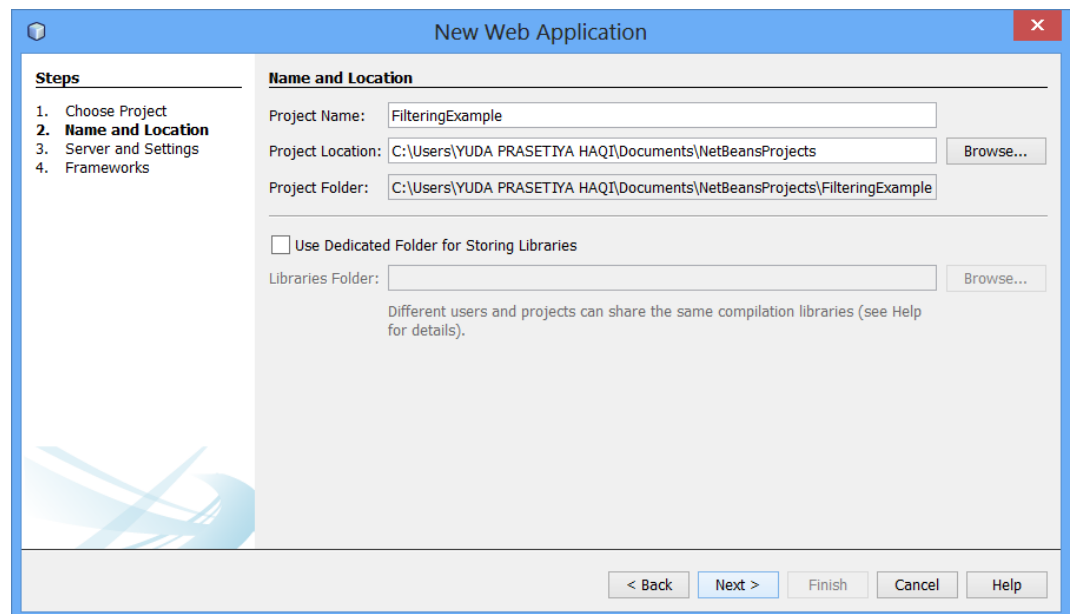
File -> New Project

File	Edit	View	Navigate	Source	Refactor	R
	New Project...	Ctrl+Shift+N				
	New File...	Ctrl+N				
	Open Project...	Ctrl+Shift+O				
	Open Recent Project	▶				
	Close Project					
	Open File...					
	Open Recent File	▶				
	Project Group	▶				
	Project Properties					
	Import Project	▶				
	Export Project	▶				
	Save	Ctrl+S				
	Save As...					
	Save All	Ctrl+Shift+S				
	Page Setup...					
	Print...	Ctrl+Alt+Shift+P				
	Print to HTML...					
	Exit					

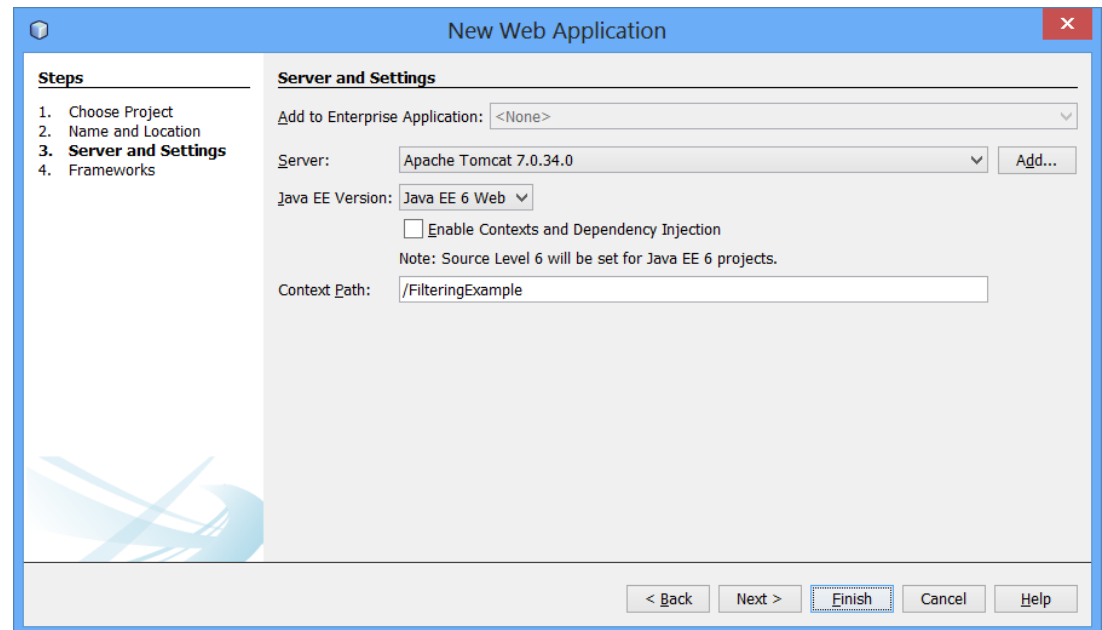
Pilih project : Java Web -> Web Application -> Klik Next



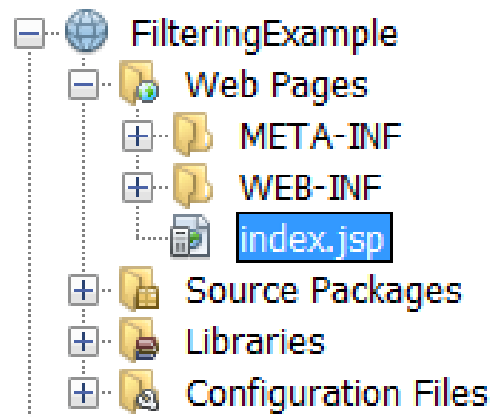
Beri Nama Project FilteringExample -> Klik Next



Klik Finish



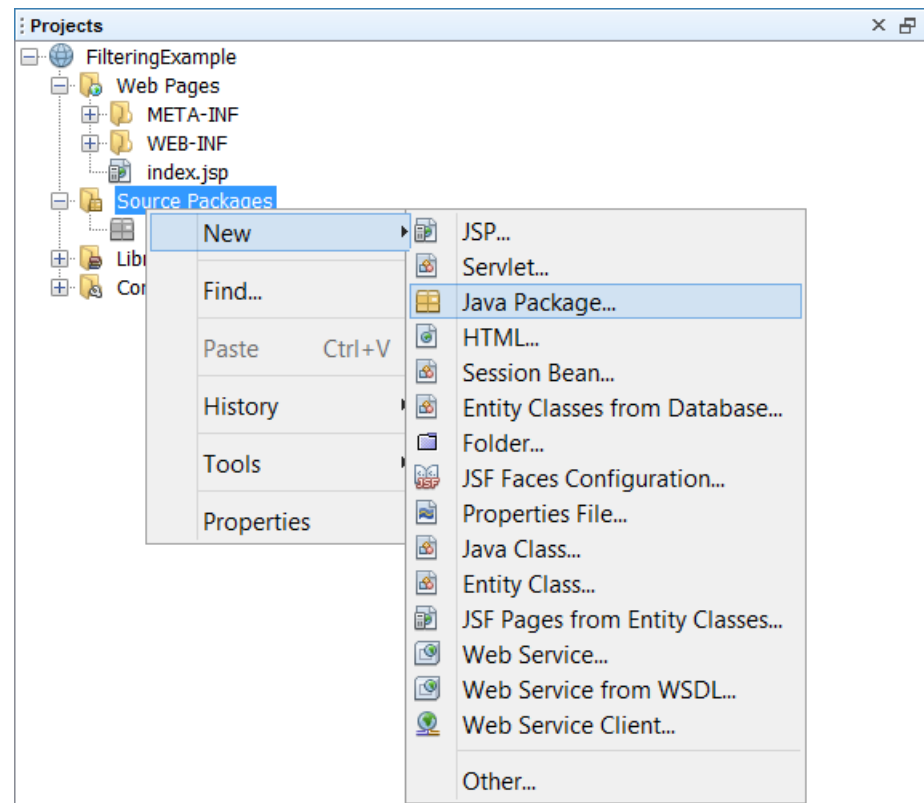
Buka index.jsp



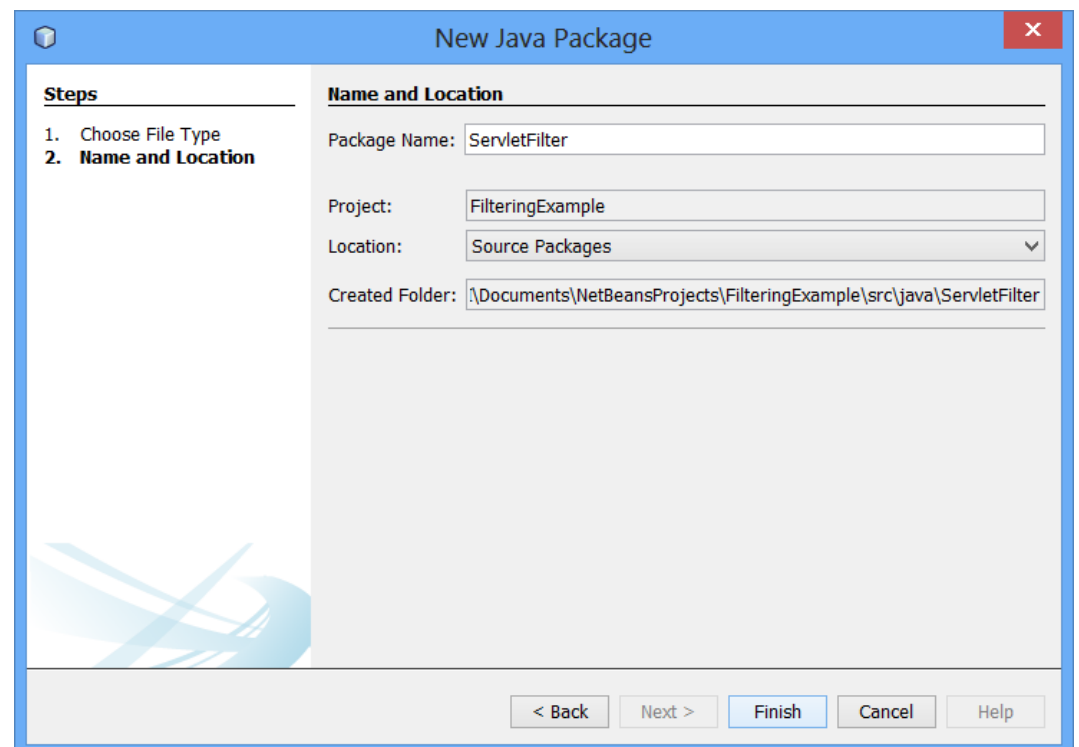
Isi index.jsp sebagai berikut

```
Start Page x index.jsp x
Source History
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <form action="Register" method="GET">
      <table>
        <tr>
          <td>Name</td>
          <td><input type="text" name="name"></td>
        </tr>
        <tr>
          <td>Address</td>
          <td><input type="text" name="address"></td>
        </tr>
        <tr>
          <td>Mobile</td>
          <td><input type="text" name="mobile"></td>
        </tr>
        <tr>
          <td>Password</td>
          <td><input type="password" name="password"></td>
        </tr>
        <tr>
          <td><input type="submit" value="Submit"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

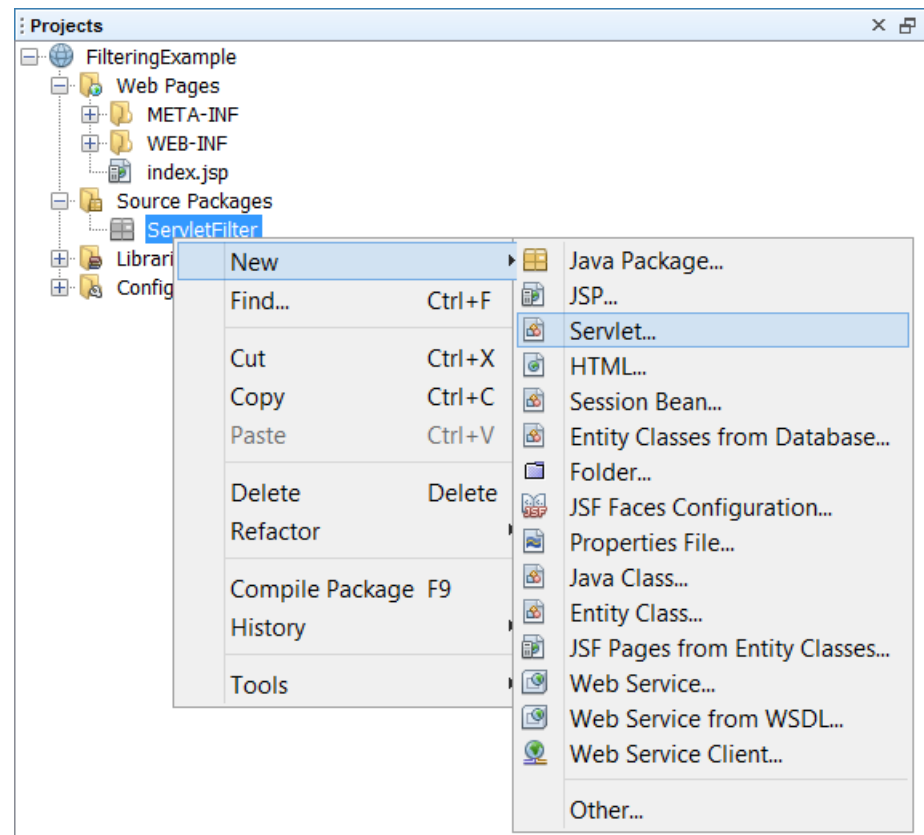
Lalu buat Java Package : Klik kanan Pada Source Packages pilih New
-> Java Package



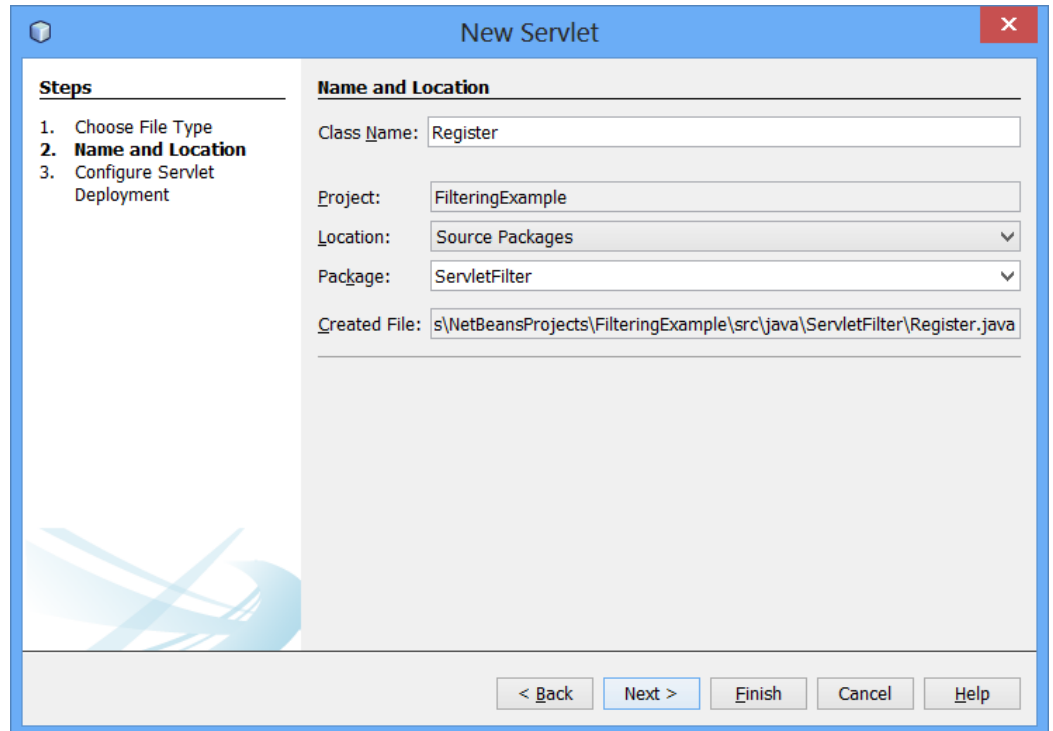
Beri Nama Java Packages ServletFilter Klik Finish



Buat Servlet : Klik kanan Pada ServletFilter Package pilih New -> Servlet



Beri nama Servlet Register lalu Klik Next



New Servlet

Steps

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

Name and Location

Class Name: Register

Project: FilteringExample

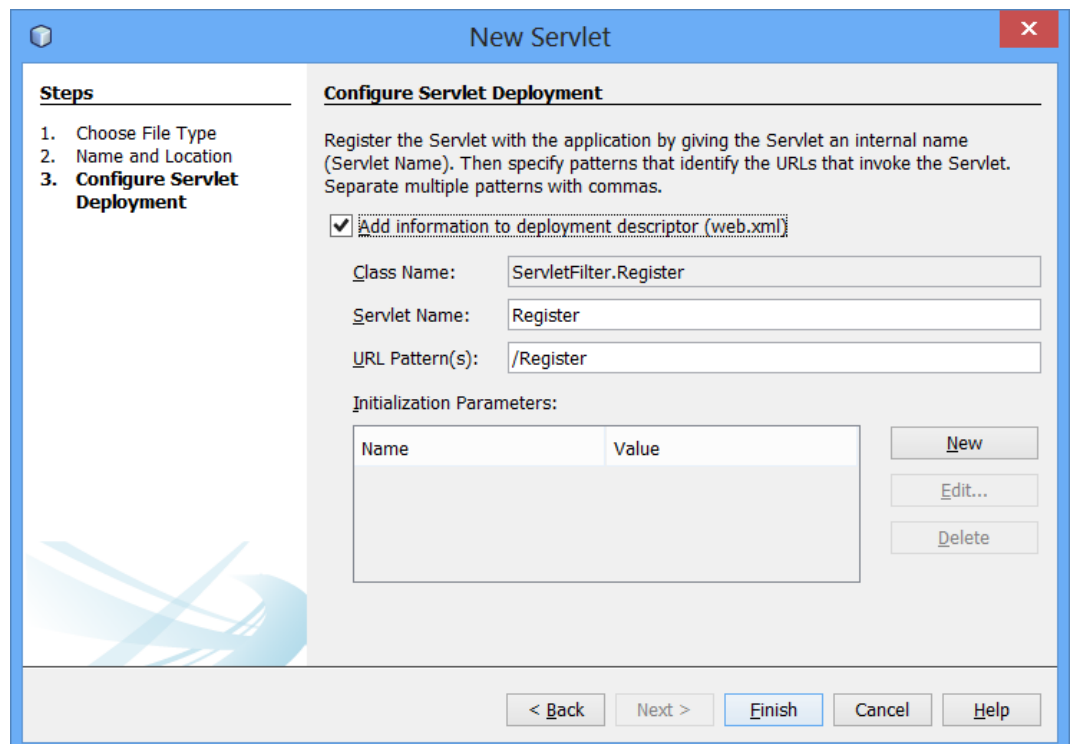
Location: Source Packages

Package: ServletFilter

Created File: s:\NetBeansProjects\FilteringExample\src\java\ServletFilter\Register.java

< Back Next > Finish Cancel Help

Klik Add Information to deployment descriptor lalu Klik Finish



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: ServletFilter.Register

Servlet Name: Register

URL Pattern(s): /Register

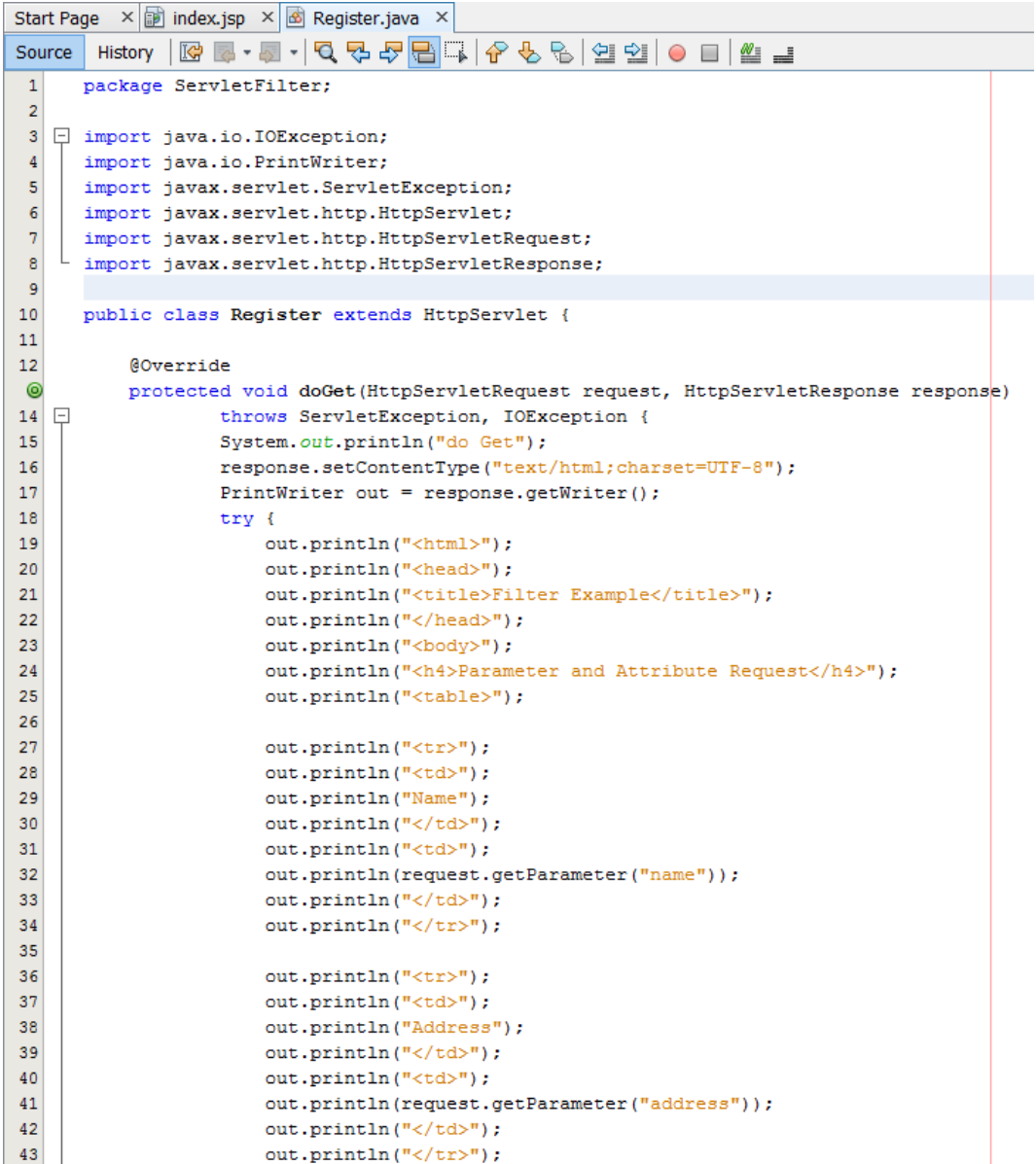
Initialization Parameters:

Name	Value

New Edit... Delete

< Back Next > Finish Cancel Help

Isi Register.java seperti berikut :

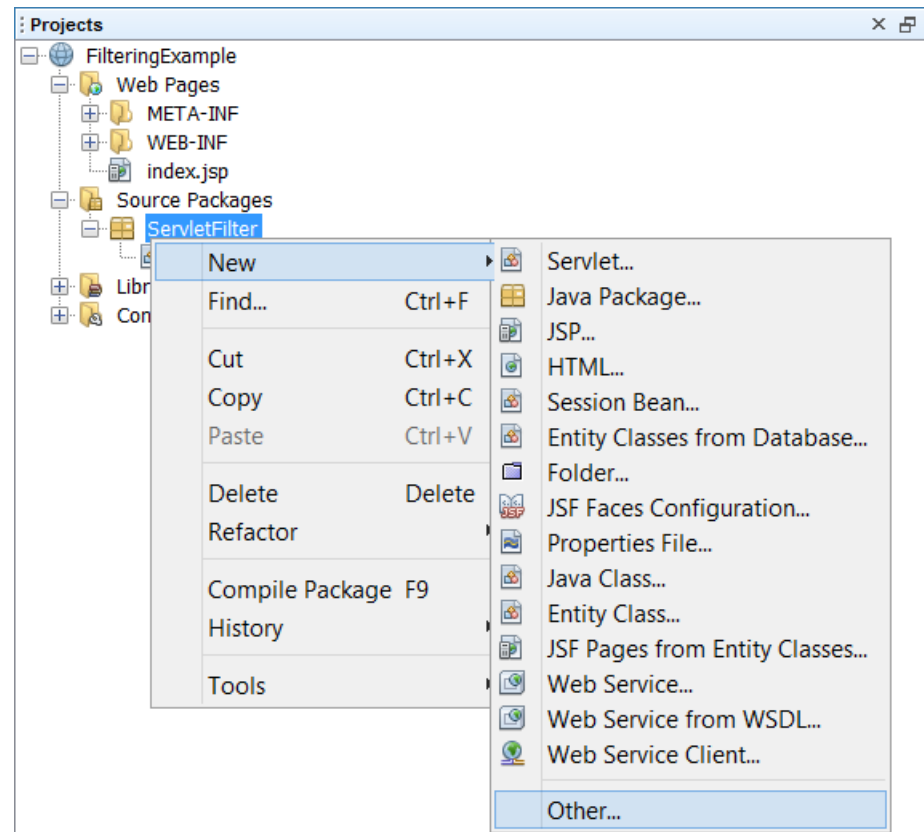


```
1 package ServletFilter;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 public class Register extends HttpServlet {
11
12     @Override
13     protected void doGet(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException {
15         System.out.println("do Get");
16         response.setContentType("text/html;charset=UTF-8");
17         PrintWriter out = response.getWriter();
18         try {
19             out.println("<html>");
20             out.println("<head>");
21             out.println("<title>Filter Example</title>");
22             out.println("</head>");
23             out.println("<body>");
24             out.println("<h4>Parameter and Attribute Request</h4>");
25             out.println("<table>");
26
27             out.println("<tr>");
28             out.println("<td>");
29             out.println("Name");
30             out.println("</td>");
31             out.println("<td>");
32             out.println(request.getParameter("name"));
33             out.println("</td>");
34             out.println("</tr>");
35
36             out.println("<tr>");
37             out.println("<td>");
38             out.println("Address");
39             out.println("</td>");
40             out.println("<td>");
41             out.println(request.getParameter("address"));
42             out.println("</td>");
43             out.println("</tr>");
```

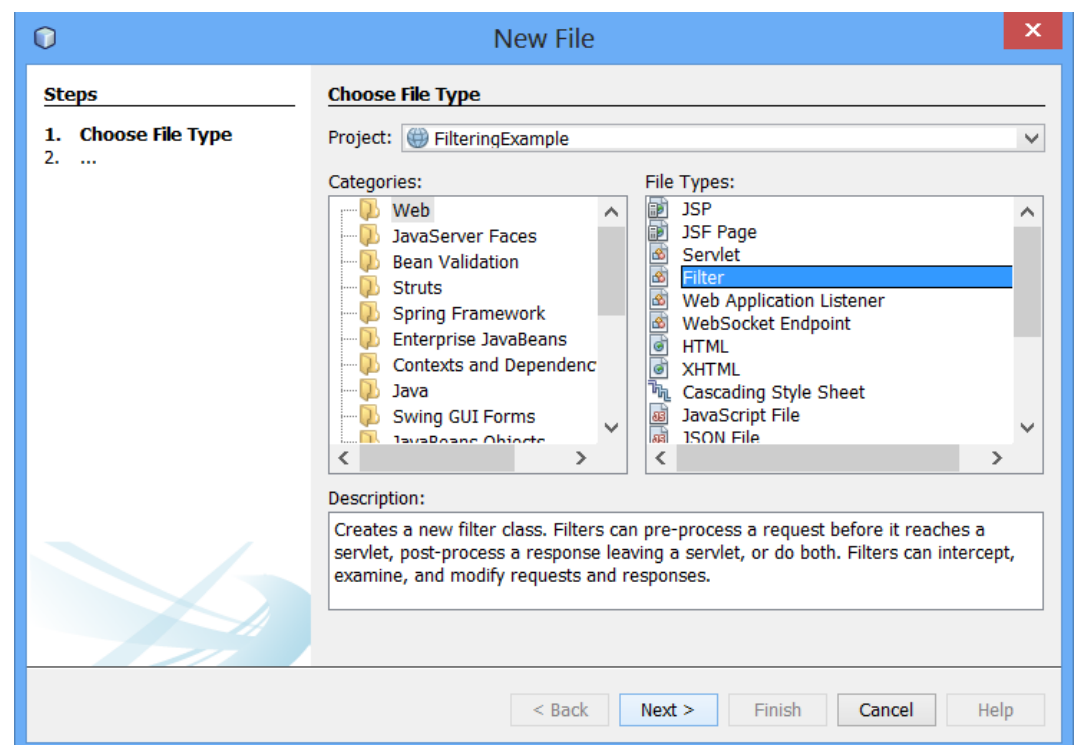
```
Start Page × index.jsp × Register.java ×
Source History
43         out.println("</tr>");
44
45         out.println("<tr>");
46         out.println("<td>");
47         out.println("Mobile");
48         out.println("</td>");
49         out.println("<td>");
50         out.println(request.getParameter("mobile"));
51         out.println("</td>");
52         out.println("</tr>");
53
54         out.println("<tr>");
55         out.println("<td>");
56         out.println("Password");
57         out.println("</td>");
58         out.println("<td>");
59         out.println(request.getParameter("password"));
60         out.println("</td>");
61         out.println("</tr>");
62
63         out.println("<tr>");
64         out.println("<td>");
65         out.println("IP Address");
66         out.println("</td>");
67         out.println("<td>");
68         out.println(request.getAttribute("IPAddress"));
69         out.println("</td>");
70         out.println("</tr>");
71
72         out.println("<tr>");
73         out.println("<td>");
74         out.println("Time Register");
75         out.println("</td>");
76         out.println("<td>");
77         out.println(request.getAttribute("timeReg"));
78         out.println("</td>");
79         out.println("</tr>");
80
81         out.println("</table>");
82         out.println("</body>");
83         out.println("</html>");
84     } finally {
85         out.close();
86     }
87 }
```

```
Start Page x index.jsp x Register.java x
Source History
52 out.println("</td>");
53
54 out.println("<tr>");
55 out.println("<td>");
56 out.println("Password");
57 out.println("</td>");
58 out.println("<td>");
59 out.println(request.getParameter("password"));
60 out.println("</td>");
61 out.println("</tr>");
62
63 out.println("<tr>");
64 out.println("<td>");
65 out.println("IP Address");
66 out.println("</td>");
67 out.println("<td>");
68 out.println(request.getAttribute("IPAddress"));
69 out.println("</td>");
70 out.println("</tr>");
71
72 out.println("<tr>");
73 out.println("<td>");
74 out.println("Time Register");
75 out.println("</td>");
76 out.println("<td>");
77 out.println(request.getAttribute("timeReg"));
78 out.println("</td>");
79 out.println("</tr>");
80
81 out.println("</table>");
82 out.println("</body>");
83 out.println("</html>");
84 } finally {
85     out.close();
86 }
87
88
89 @Override
90 protected void doPost(HttpServletRequest request, HttpServletResponse response)
91     throws ServletException, IOException {
92     System.out.println("do Post");
93 }
94 }
```

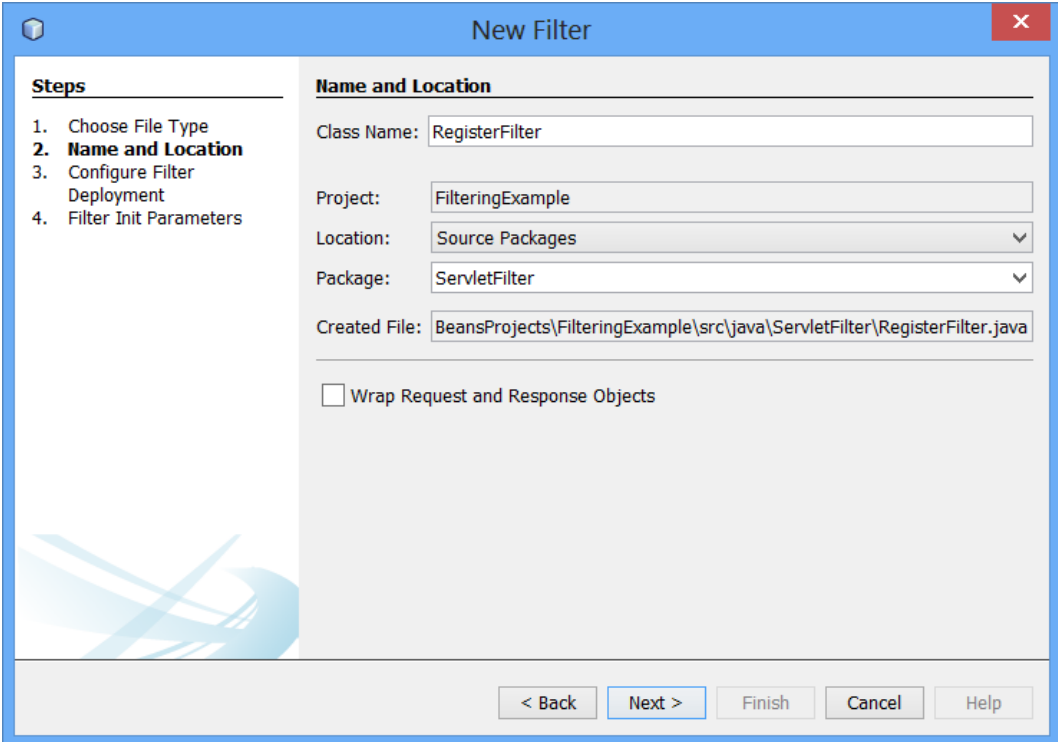
Lalu buat Filter : Klik kanan pada ServletFilter Pilih New -> Other



Pilih Web -> Filter lalu Klik Next



Beri nama RegisterFilter



The 'New Filter' dialog box is shown at the 'Name and Location' step. The 'Steps' list on the left includes: 1. Choose File Type, 2. **Name and Location**, 3. Configure Filter Deployment, and 4. Filter Init Parameters. The 'Name and Location' section contains the following fields: 'Class Name' with the value 'RegisterFilter', 'Project' with 'FilteringExample', 'Location' with 'Source Packages', and 'Package' with 'ServletFilter'. The 'Created File' path is 'BeansProjects\FilteringExample\src\java\ServletFilter\RegisterFilter.java'. There is an unchecked checkbox for 'Wrap Request and Response Objects'. At the bottom are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Steps

1. Choose File Type
2. **Name and Location**
3. Configure Filter Deployment
4. Filter Init Parameters

Name and Location

Class Name: RegisterFilter

Project: FilteringExample

Location: Source Packages

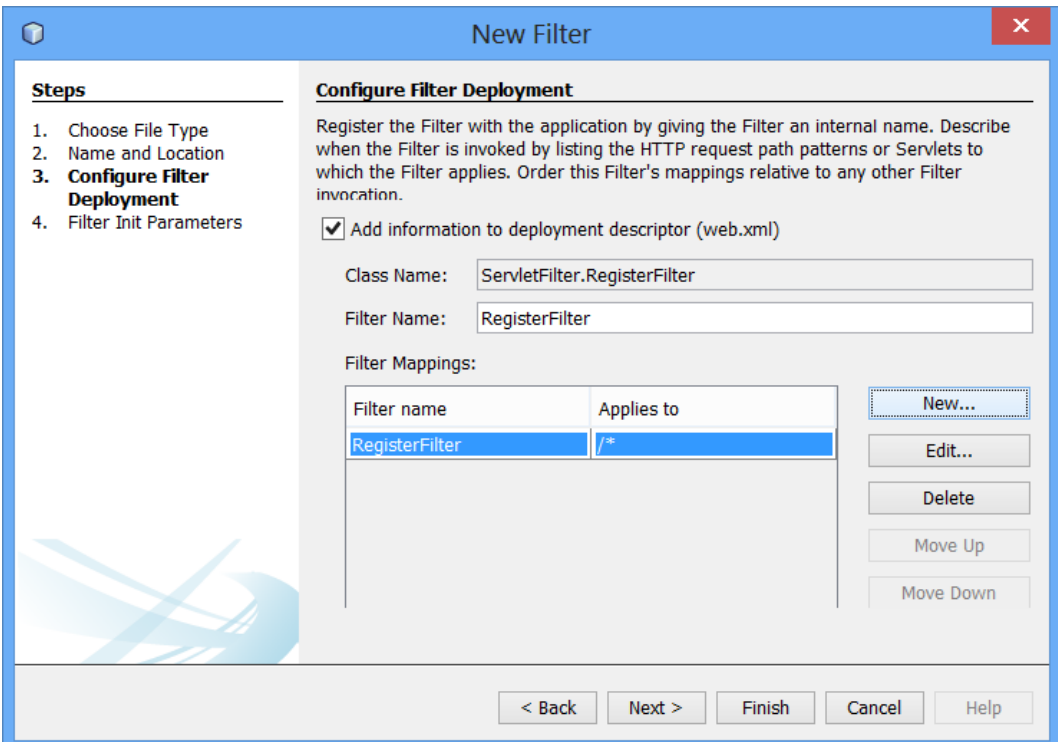
Package: ServletFilter

Created File: BeansProjects\FilteringExample\src\java\ServletFilter\RegisterFilter.java

☐ Wrap Request and Response Objects

< Back Next > Finish Cancel Help

Klik Add information to deployment descriptor, lalu Klik New



The 'New Filter' dialog box is shown at the 'Configure Filter Deployment' step. The 'Steps' list on the left includes: 1. Choose File Type, 2. Name and Location, 3. **Configure Filter Deployment**, and 4. Filter Init Parameters. The 'Configure Filter Deployment' section contains a text area with instructions: 'Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.' Below this is a checked checkbox for 'Add information to deployment descriptor (web.xml)'. The 'Class Name' field contains 'ServletFilter.RegisterFilter' and the 'Filter Name' field contains 'RegisterFilter'. The 'Filter Mappings' section has a table with two columns: 'Filter name' and 'Applies to'. The first row has 'RegisterFilter' and '/*'. To the right of the table are buttons: 'New...', 'Edit...', 'Delete', 'Move Up', and 'Move Down'. At the bottom are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Steps

1. Choose File Type
2. Name and Location
3. **Configure Filter Deployment**
4. Filter Init Parameters

Configure Filter Deployment

Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.

☒ Add information to deployment descriptor (web.xml)

Class Name: ServletFilter.RegisterFilter

Filter Name: RegisterFilter

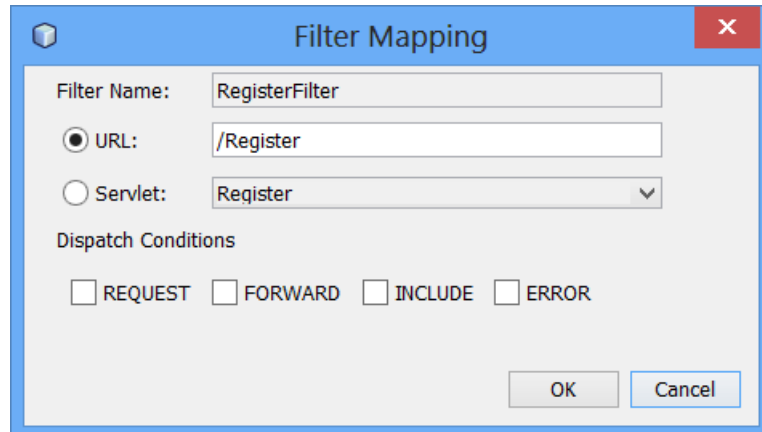
Filter Mappings:

Filter name	Applies to
RegisterFilter	/*

New... Edit... Delete Move Up Move Down

< Back Next > Finish Cancel Help

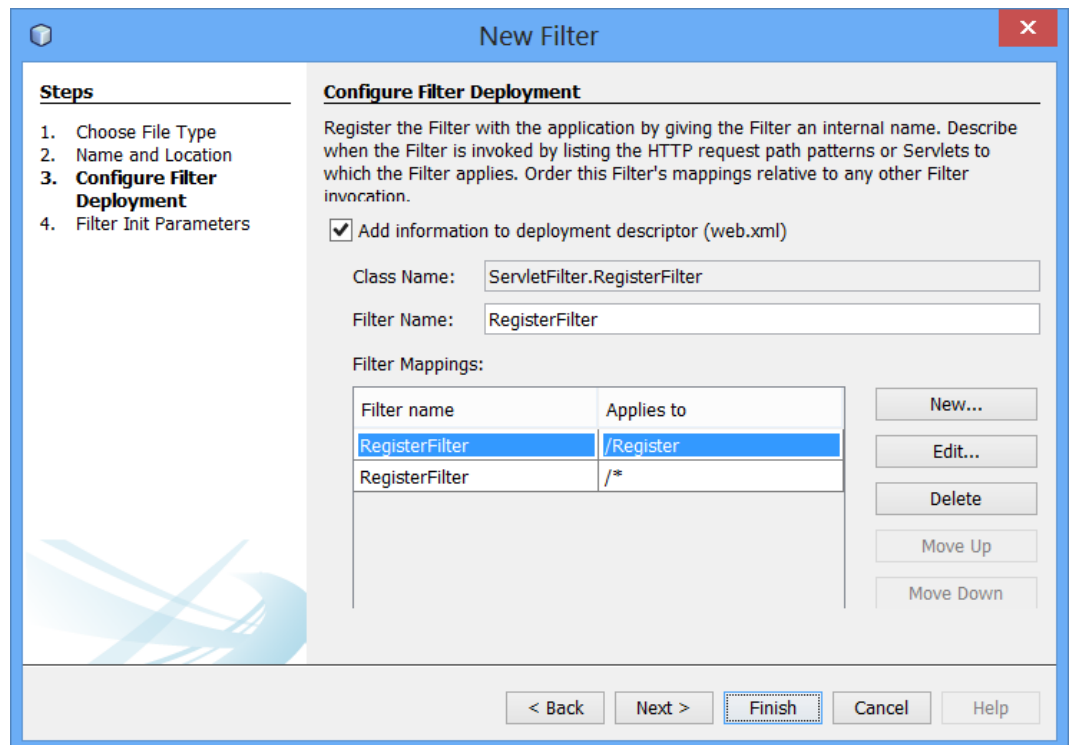
Isi Filter Mapping seperti berikut, Klik OK



The 'Filter Mapping' dialog box is shown with a blue title bar and a red close button. It contains the following fields and options:

- Filter Name:** RegisterFilter
- URL:** /Register (selected with a radio button)
- Servlet:** Register (selected in a dropdown menu)
- Dispatch Conditions:** REQUEST, FORWARD, INCLUDE, ERROR (all unchecked)
- Buttons:** OK, Cancel

Lalu Klik Finish

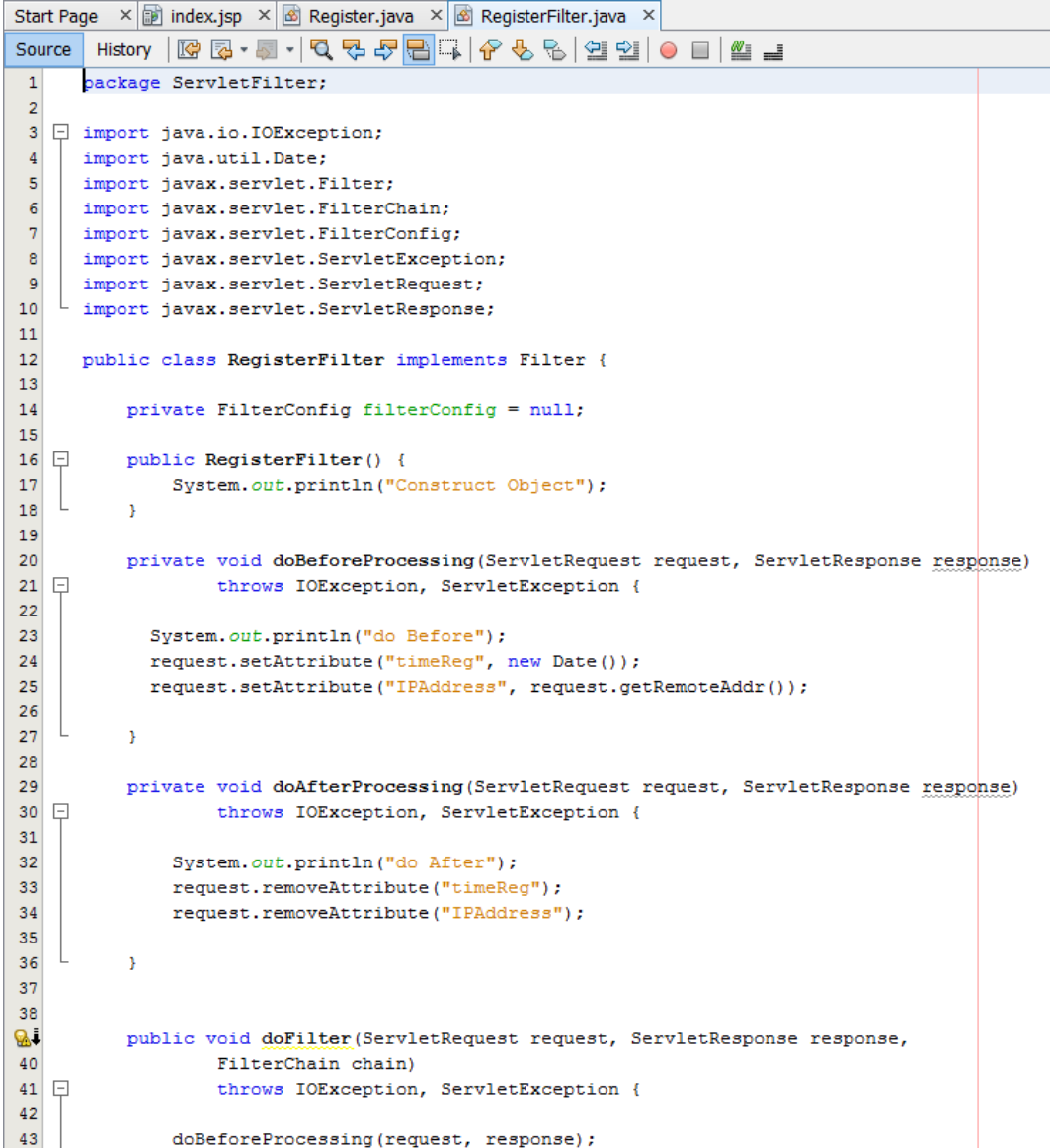


The 'New Filter' dialog box is shown with a blue title bar and a red close button. It contains the following sections and fields:

- Steps:** 1. Choose File Type, 2. Name and Location, 3. **Configure Filter Deployment** (selected), 4. Filter Init Parameters
- Configure Filter Deployment:**
 - Register the Filter with the application by giving the Filter an internal name. Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to any other Filter invocation.
 - ☒ Add information to deployment descriptor (web.xml)
 - Class Name:** ServletFilter.RegisterFilter
 - Filter Name:** RegisterFilter
 - Filter Mappings:**

Filter name	Applies to
RegisterFilter	/Register
RegisterFilter	/*
 - Buttons:** New..., Edit..., Delete, Move Up, Move Down
- Bottom Buttons:** < Back, Next >, **Finish** (highlighted), Cancel, Help

Isi RegisterFilter seperti Gambar dibawah



```
1 package ServletFilter;
2
3 import java.io.IOException;
4 import java.util.Date;
5 import javax.servlet.Filter;
6 import javax.servlet.FilterChain;
7 import javax.servlet.FilterConfig;
8 import javax.servlet.ServletException;
9 import javax.servlet.ServletRequest;
10 import javax.servlet.ServletResponse;
11
12 public class RegisterFilter implements Filter {
13
14     private FilterConfig filterConfig = null;
15
16     public RegisterFilter() {
17         System.out.println("Construct Object");
18     }
19
20     private void doBeforeProcessing(ServletRequest request, ServletResponse response)
21         throws IOException, ServletException {
22
23         System.out.println("do Before");
24         request.setAttribute("timeReg", new Date());
25         request.setAttribute("IPAddress", request.getRemoteAddr());
26     }
27
28
29     private void doAfterProcessing(ServletRequest request, ServletResponse response)
30         throws IOException, ServletException {
31
32         System.out.println("do After");
33         request.removeAttribute("timeReg");
34         request.removeAttribute("IPAddress");
35     }
36
37
38
39     public void doFilter(ServletRequest request, ServletResponse response,
40         FilterChain chain)
41         throws IOException, ServletException {
42
43         doBeforeProcessing(request, response);
```



```

Start Page x index.jsp x Register.java x RegisterFilter.java x
Source History
39
40
41 public void doFilter(ServletRequest request, ServletResponse response,
42                     FilterChain chain)
43     throws IOException, ServletException {
44
45         doBeforeProcessing(request, response);
46
47         System.out.println("Chain RegisterFilter Filter");
48         chain.doFilter(request, response);
49
50         doAfterProcessing(request, response);
51     }
52
53
54 public FilterConfig getFilterConfig() {
55     return (this.filterConfig);
56 }
57
58 public void setFilterConfig(FilterConfig filterConfig) {
59     this.filterConfig = filterConfig;
60 }
61
62 public void destroy() {
63     System.out.println("Filter is destroyed");
64     //add code to release any resource
65 }
66
67
68 public void init(FilterConfig filterConfig) {
69     this.filterConfig = filterConfig;
70     System.out.println("Init.....");
71     //Get init parameter
72 }
73
74 }
75

```

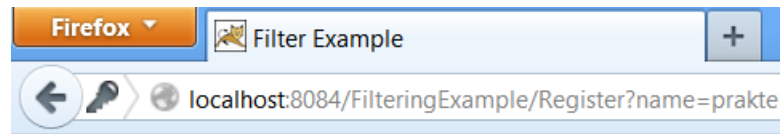
Running program, Bila berhasil maka akan memunculkan web seperti berikut :

The screenshot shows a web browser window with the title "JSP Page". The address bar displays "localhost:8084/FilteringExample/". The page contains a registration form with the following fields and values:

Name	praktek
Address	jl. ciledug reya
Mobile	08192314123
Password	••••••••

Below the form is a "Submit" button.

Hasil setelah parameter dan attribut melewati Filter :



Parameter and Attribute Request

Name	praktek
Address	jl. ciledug reya
Mobile	08192314123
Password	mahasiswa
IP Address	0:0:0:0:0:0:1
Time Register	Sat Mar 01 10:45:55 ICT 2014

Pertemuan 5

Session Tracking

8. Session Tracking

HTTP adalah "stateless" protokol yang berarti setiap kali client mengambil halaman Web, client membuka koneksi terpisah ke server Web dan server secara otomatis tidak menyimpan catatan permintaan client sebelumnya.

Masih ada tiga cara selain HTTPsession untuk menjaga sesi antara client web dan web server:

- **Cookies :**
Sebuah webserver dapat menetapkan ID sesi yang unik sebagai cookie untuk setiap client web dan untuk permintaan berikutnya dari client mereka dapat dikenali dengan menggunakan cookie yang diterima.
Ini mungkin bukan cara yang efektif karena banyak browser tidak mendukung cookie, jadi saya tidak akan merekomendasikan untuk menggunakan prosedur ini untuk menjaga sesi.

- **Hidden Form Fields**
Catatan ini berarti bahwa, ketika form dikirimkan, nama tertentu dan nilai secara otomatis dimasukkan dalam GET atau POST data. Setiap saat browser web mengirimkan permintaan kembali, maka nilai session_id dapat digunakan untuk menyimpan lagu dari web browser yang berbeda.
Ini bisa menjadi cara yang efektif untuk melacak sesi, namun mengklik (<A HREF...>) link hypertext biasa tidak mengakibatkan pengiriman form, jadi kolom form tersembunyi juga tidak dapat mendukung pelacakan sesi umum.

- URL Rewriting:

Anda dapat menambahkan beberapa data tambahan pada akhir setiap URL yang mengidentifikasi sesi, dan server dapat mengaitkan bahwa pengidentifikasi sesi dengan data yang telah disimpan tentang sesi tersebut.

Misalnya, dengan `http://contoh.com/page1.htm;sessionid=12345`, pengidentifikasi sesi terpasang sebagai `sessionid = 12345` yang dapat diakses di web server untuk mengidentifikasi client.

URL Rewriting adalah cara yang baik untuk mempertahankan sesi dan bekerja untuk browser ketika mereka tidak mendukung cookies tapi di sini kelemahan adalah bahwa Anda akan menghasilkan setiap URL dinamis untuk menetapkan ID sesi meskipun halaman adalah halaman HTML sederhana yang statis.

Selain tiga cara yang disebutkan di atas, servlet menyediakan `HttpSession` Interface yang menyediakan cara untuk mengidentifikasi pengguna di lebih dari satu permintaan halaman atau kunjungan ke situs Web dan untuk menyimpan informasi tentang pengguna tersebut.

Servlet kontainer menggunakan interface ini untuk membuat sesi antara client HTTP dan server HTTP. Sesi ini berlangsung selama jangka waktu tertentu, di lebih dari satu koneksi atau halaman permintaan dari pengguna.

Anda perlu memanggil `request.getSession ()` sebelum Anda mengirim konten dokumen ke client. Berikut adalah ringkasan dari metode penting tersedia melalui objek `HttpSession`:

- `public Object getAttribute(String name)`

Metode ini mengembalikan objek terikat dengan nama tertentu dalam sesi ini, atau null jika tidak ada objek yang terikat dengan nama.

- `public Enumeration getAttributeNames()`
Metode ini mengembalikan sebuah Pencacahan String objek yang berisi nama-nama dari semua benda terikat untuk sesi ini.
- `public long getCreationTime()`
Metode ini mengembalikan waktu ketika sesi ini diciptakan, diukur dalam milidetik sejak tengah malam 1 Januari 1970 GMT.
- `public String getId()`
Metode ini mengembalikan sebuah string yang berisi identifier unik yang diberikan untuk sesi ini.
- `public long getLastAccessedTime()`
Metode ini mengembalikan waktu terakhir klien mengirimkan permintaan terkait dengan sesi ini, karena jumlah milidetik sejak tengah malam 1 Januari 1970 GMT.
- `public int getMaxInactiveInterval()`
Metode ini mengembalikan interval waktu maksimum, dalam hitungan detik, bahwa kontainer servlet akan membuat sesi ini terbuka antara klien mengakses.
- `public void invalidate()`
Metode ini membatalkan sesi ini dan melepaskan setiap objek yang terikat dengan itu.
- `public boolean isNew()`
Metode ini mengembalikan nilai true jika klien belum pernah mengakses sesi atau jika klien memilih untuk tidak bergabung dengan sesi.
- `public void removeAttribute(String name)`

Metode ini akan menghapus objek yang terikat dengan nama tertentu dari sesi ini.

- `public void setAttribute(String name, Object value)`

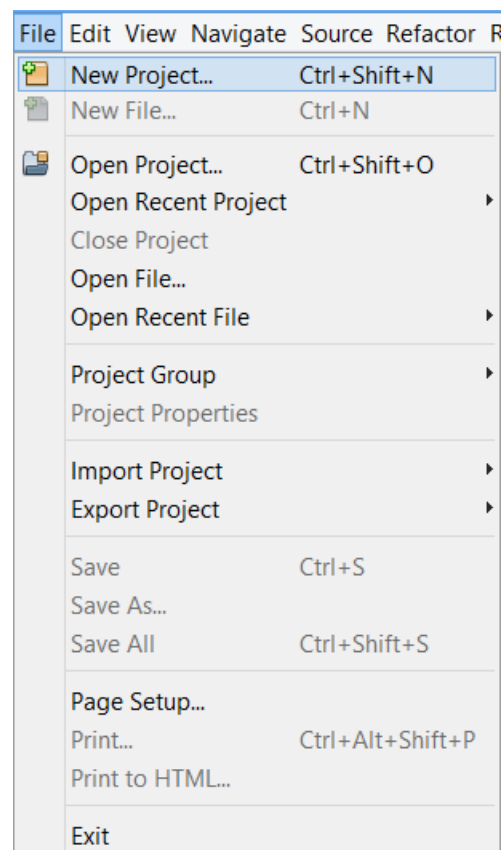
Metode ini mengikat objek untuk sesi ini, dengan menggunakan nama tertentu.

- `public void setMaxInactiveInterval(int interval)`

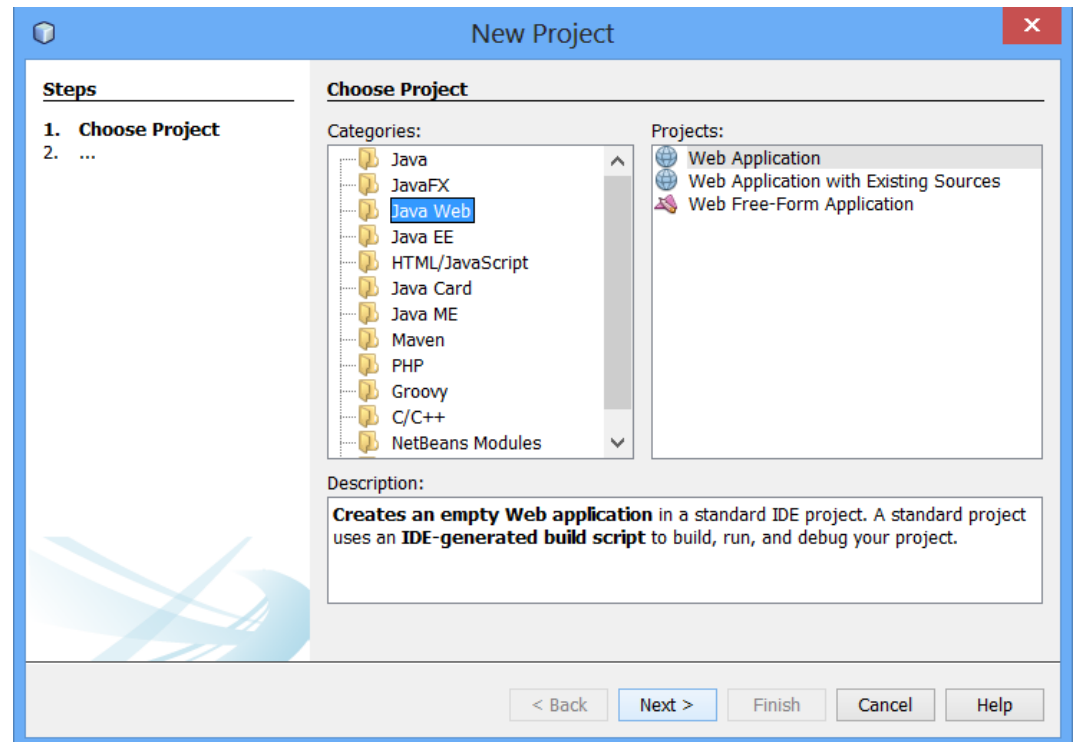
Metode ini menentukan waktu, dalam hitungan detik, antara permintaan klien sebelum kontainer servlet akan membatalkan sesi ini.

9. Aplikasi Shopping Cart

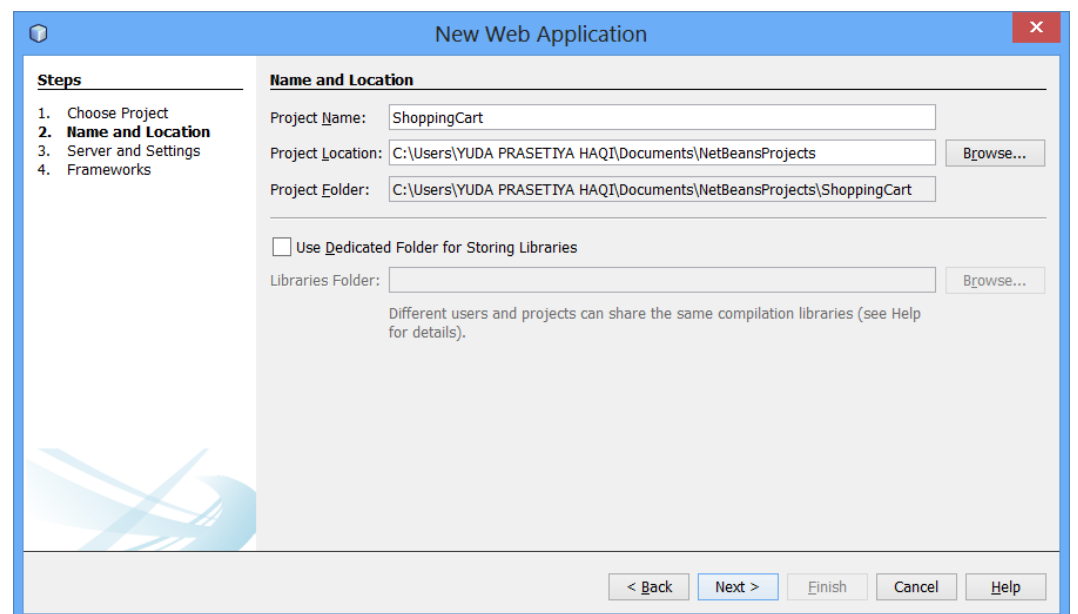
Pertama-tama buat Project baru, Pilih File -> New Project



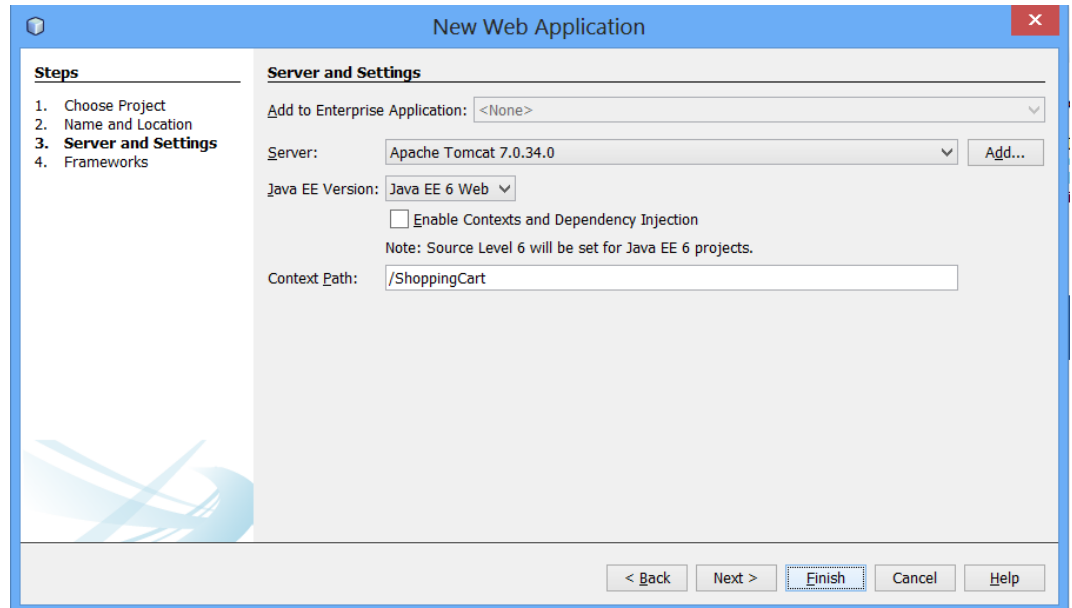
Pilih Java Web -> Web Application



Beri Nama Project : ShoppingCart



Pilih Apache Tomcat sebagai Server



The screenshot shows the 'New Web Application' wizard in an IDE. The window has a blue title bar with the text 'New Web Application' and a close button. On the left, a 'Steps' pane lists four steps: 1. Choose Project, 2. Name and Location, 3. **Server and Settings** (highlighted), and 4. Frameworks. The main area is titled 'Server and Settings' and contains the following options:

- 'Add to Enterprise Application:' with a dropdown menu showing '<None>'.
- 'Server:' with a dropdown menu showing 'Apache Tomcat 7.0.34.0' and an 'Add...' button.
- 'Java EE Version:' with a dropdown menu showing 'Java EE 6 Web'.
- An unchecked checkbox labeled 'Enable Contexts and Dependency Injection'.
- A note: 'Note: Source Level 6 will be set for Java EE 6 projects.'
- 'Context Path:' with a text input field containing '/ShoppingCart'.

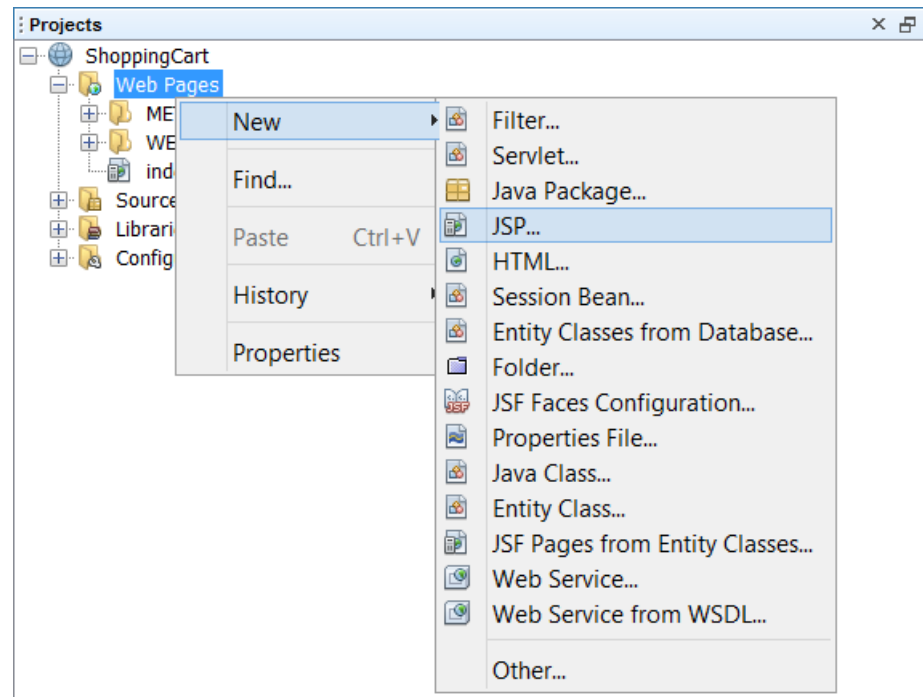
At the bottom of the wizard, there are five buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a dashed border), 'Cancel', and 'Help'.

Isi index.jsp dengan kodingan berikut :

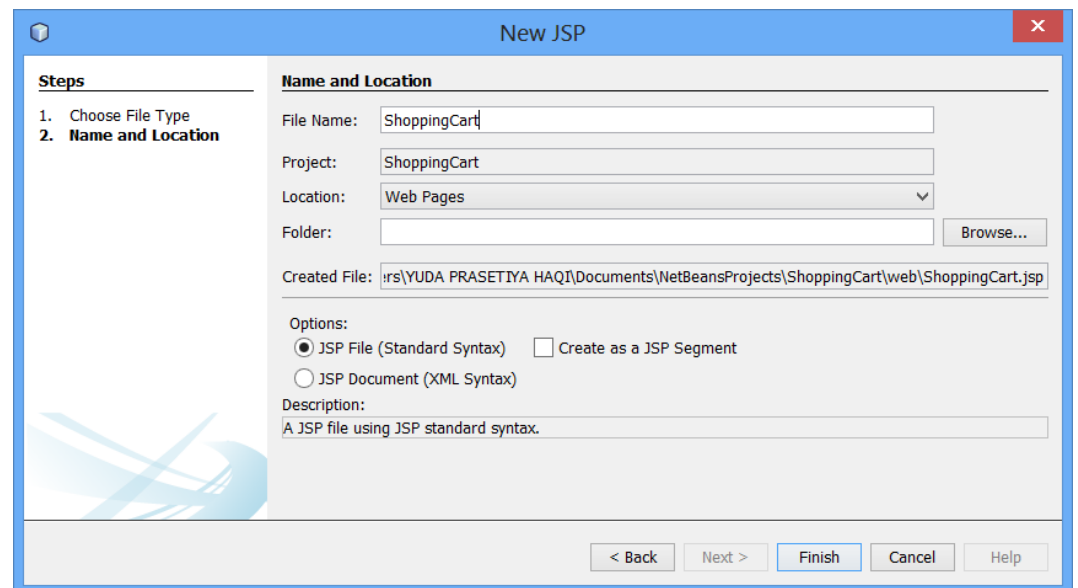
```
Start Page x index.jsp x
Source History
<jsp:useBean id="cart" scope="session" class="beans.ShoppingCart" />
<html>
<head>
<title>DVD Catalog</title>
</head>
<body>
<%
String id = request.getParameter("id");
if ( id != null ) {
String desc = request.getParameter("desc");
Float price = new Float(request.getParameter("price"));

cart.addItem(id, desc, price.floatValue(), 1);
}
%>
<a href="ShoppingCart.jsp">Shopping Cart Quantity:</a>
<%=cart.getNumOfItems() %>
<hr>
<center><h3>DVD Catalog</h3></center>
<table border="1" width="300" cellspacing="0"
cellpadding="2" align="center">
<tr><th>Description</th><th>Price</th></tr>
<tr>
<td>
<form action="index.jsp" method="post">
<td>Book</td>
<td>$19.95</td>
<td><input type="submit" name="Submit" value="Add"></td>
<input type="hidden" name="id" value="1">
<input type="hidden" name="desc" value="Book">
<input type="hidden" name="price" value="19.95">
</form>
</td>
<td>
<form action="index.jsp" method="post">
<td>Toy</td>
<td>$19.95</td>
<td><input type="submit" name="Submit" value="Add"></td>
<input type="hidden" name="id" value="2">
<input type="hidden" name="desc" value="Toy">
<input type="hidden" name="price" value="19.95">
</form>
</td>
</tr>
</table>
</body>
</html>
```

Buat file JSP baru : Klik kanan di Web Pages -> JSP



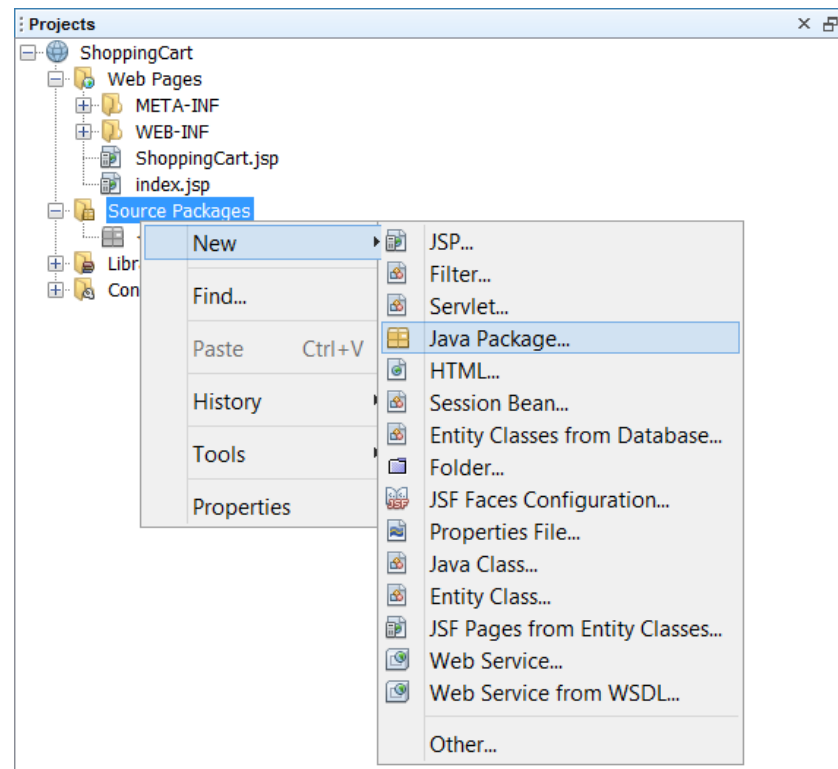
Beri nama file JSP tersebut Shopping Cart Klik Finish



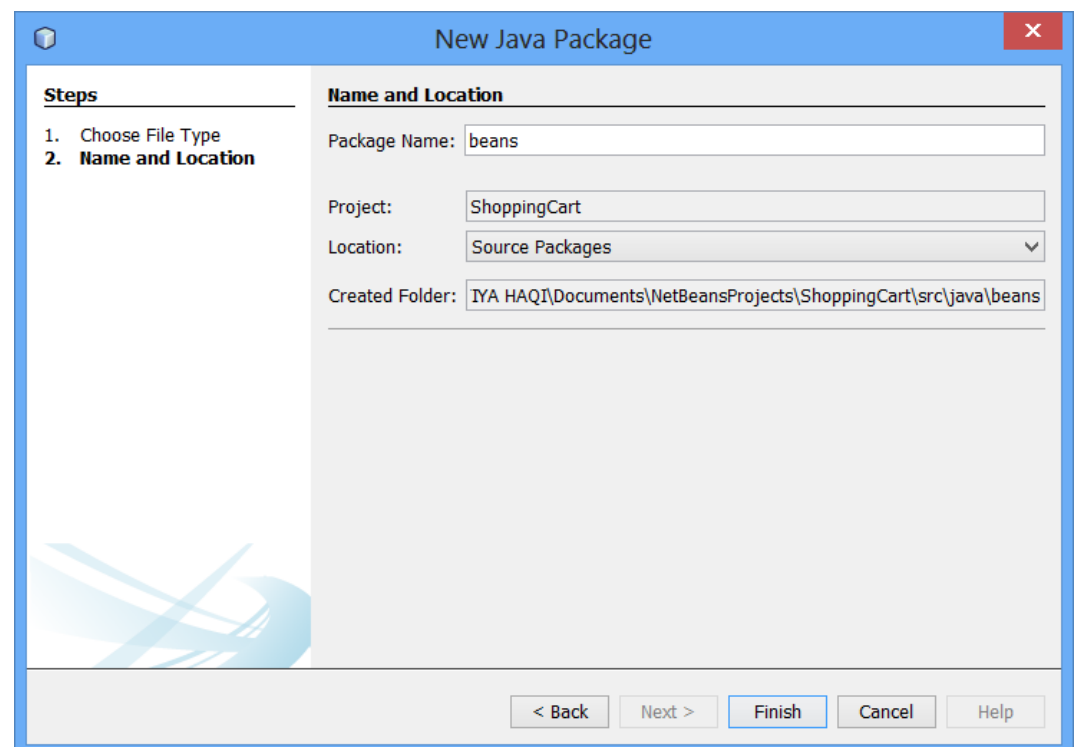
Isi ShoppingCart.jsp dengan kodingan berikut

```
Start Page x index.jsp x ShoppingCart.jsp x
Source History
<%@ page import="java.util.*" %>
2
3 <jsp:useBean id="cart" scope="session" class="beans.ShoppingCart" />
4 <html>
5 <head>
6 <title>Shopping Cart Contents</title>
7 </head>
8 <body>
9 <center>
10 <table width="300" border="1" cellspacing="0"
11 cellpadding="2" border="0">
12 <caption><b>Shopping Cart Contents</b></caption>
13 <tr>
14 <th>Description</th>
15 <th>Price</th>
16 <th>Quantity</th>
17 </tr>
18 <%
19 Enumeration e = cart.getEnumeration();
20 String[] tmpItem;
21 // Iterate over the cart
22 while (e.hasMoreElements()) {
23 tmpItem = (String[])e.nextElement();
24 %>
25 <tr>
26 <td><%=tmpItem[1] %></td>
27 <td align="center">${<%=tmpItem[2] %>}</td>
28 <td align="center"><%=tmpItem[3] %></td>
29 </tr>
30 <%
31 }
32 %>
33 </table>
34 </center>
35 <a href="index.jsp">Back to Catalog</a>
36 </body>
37 </html>
```

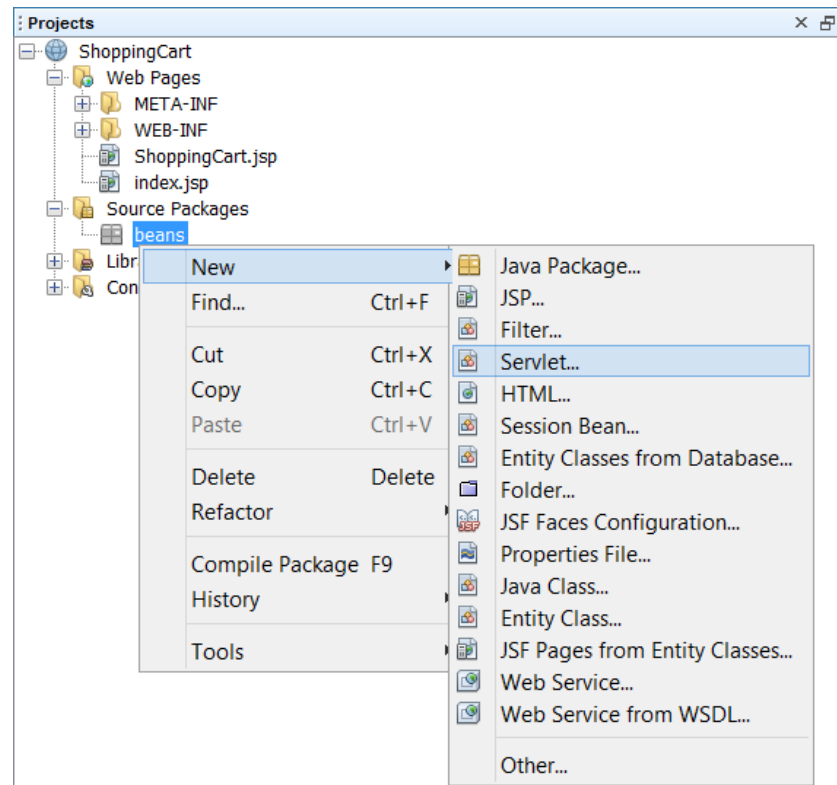
Buat Java Package : Klik kanan pada Source Packages pilih New->Java Package



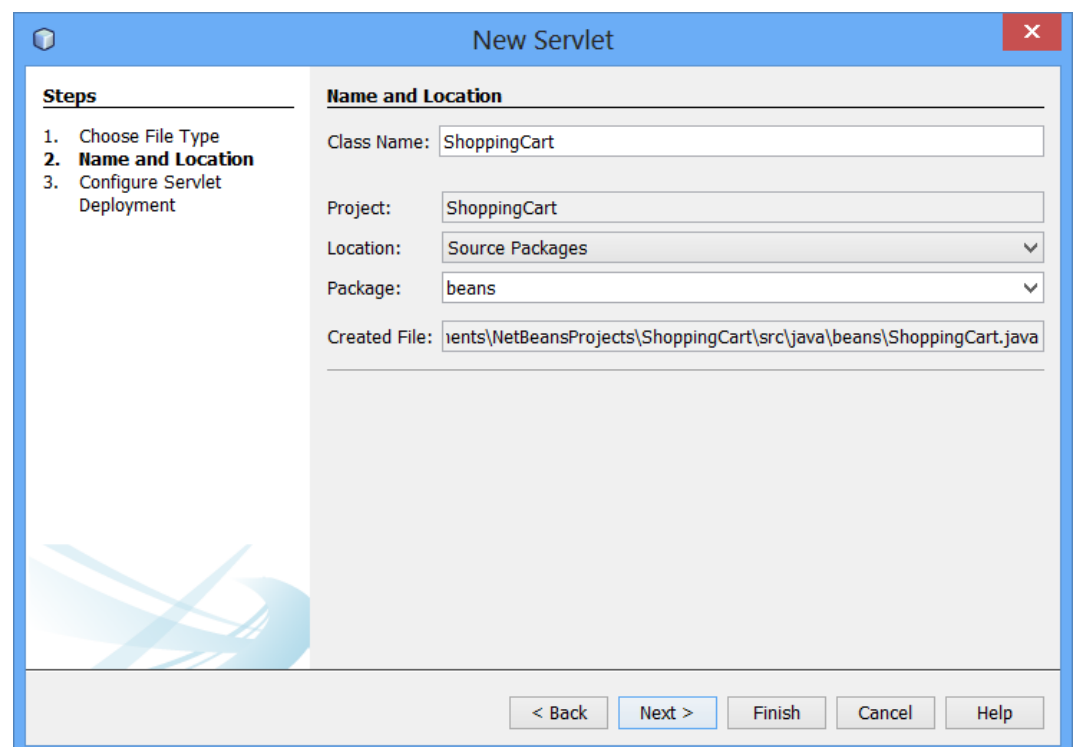
Beri nama beans pada Java Package Klik Finish



Klik kanan pada Java Package Pilih New->Servlet



Berinama ShoppingCart pada Servlet



Centang Add information to deployment descriptor (web.xml); Klik Finish

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):

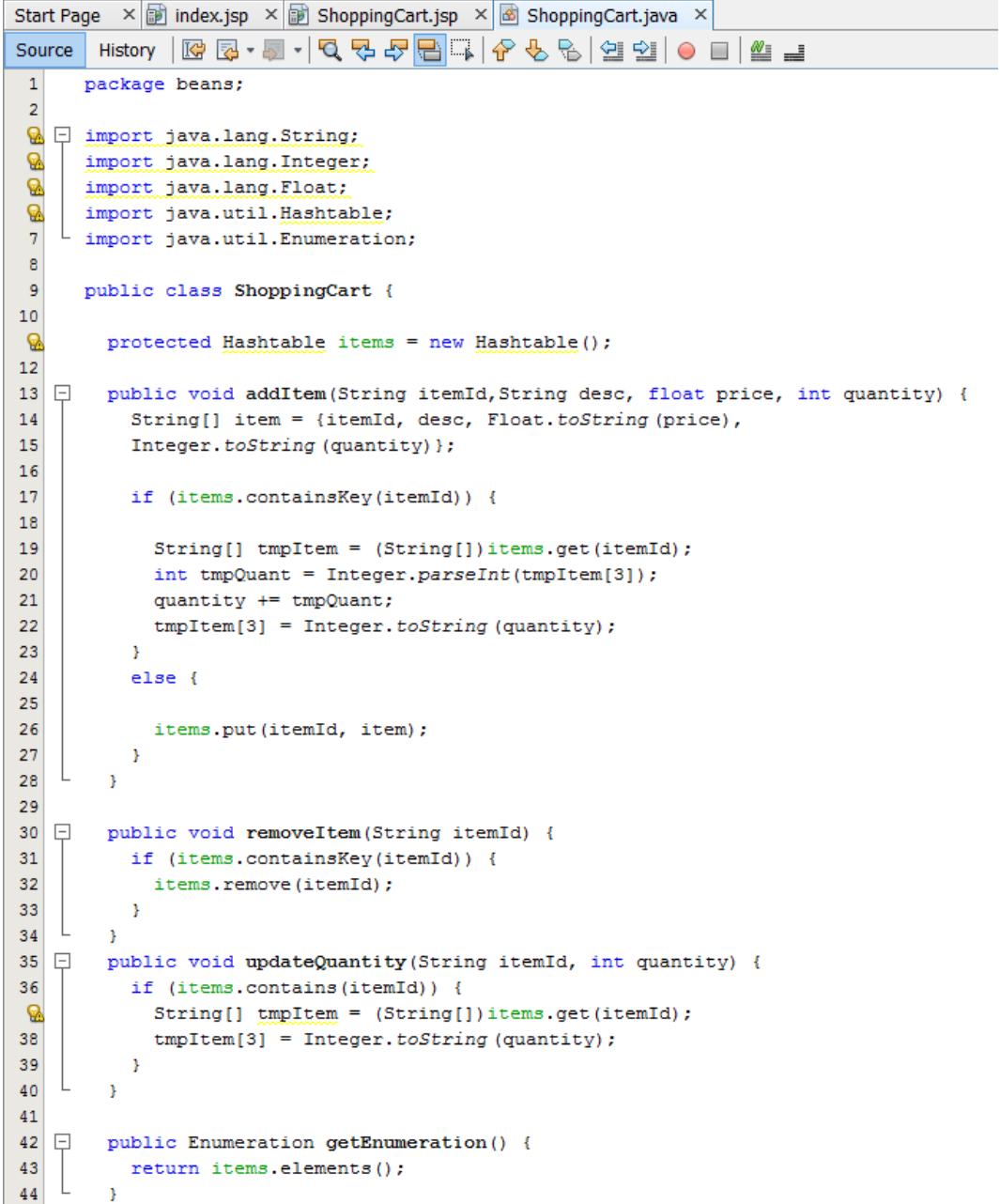
Initialization Parameters:

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

New
Edit...
Delete

< Back Next > **Finish** Cancel Help

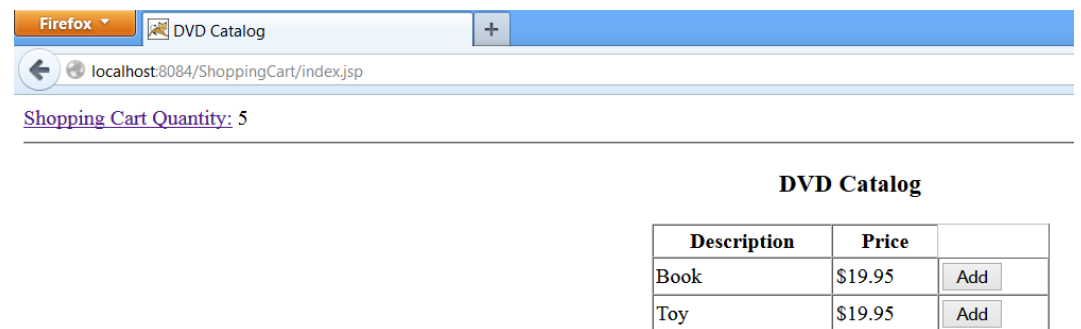
Isi ShoppingCart.java dengan kodingan berikut :



```
1 package beans;
2
3 import java.lang.String;
4 import java.lang.Integer;
5 import java.lang.Float;
6 import java.util.Hashtable;
7 import java.util.Enumeration;
8
9 public class ShoppingCart {
10
11     protected Hashtable items = new Hashtable();
12
13     public void addItem(String itemId, String desc, float price, int quantity) {
14         String[] item = {itemId, desc, Float.toString(price),
15             Integer.toString(quantity)};
16
17         if (items.containsKey(itemId)) {
18
19             String[] tmpItem = (String[])items.get(itemId);
20             int tmpQuant = Integer.parseInt(tmpItem[3]);
21             quantity += tmpQuant;
22             tmpItem[3] = Integer.toString(quantity);
23         }
24         else {
25
26             items.put(itemId, item);
27         }
28     }
29
30     public void removeItem(String itemId) {
31         if (items.containsKey(itemId)) {
32             items.remove(itemId);
33         }
34     }
35
36     public void updateQuantity(String itemId, int quantity) {
37         if (items.containsKey(itemId)) {
38             String[] tmpItem = (String[])items.get(itemId);
39             tmpItem[3] = Integer.toString(quantity);
40         }
41     }
42
43     public Enumeration getEnumeration() {
44         return items.elements();
45     }
46 }
```

```
Start Page x index.jsp x ShoppingCart.jsp x ShoppingCart.java x
Source History
31     if (items.containsKey(itemId)) {
32         items.remove(itemId);
33     }
34 }
35 public void updateQuantity(String itemId, int quantity) {
36     if (items.containsKey(itemId)) {
37         String[] tmpItem = (String[])items.get(itemId);
38         tmpItem[3] = Integer.toString(quantity);
39     }
40 }
41
42 public Enumeration getEnumeration() {
43     return items.elements();
44 }
45
46 public float getCost() {
47     Enumeration e = items.elements();
48     String[] tmpItem;
49     float totalCost = 0.00f;
50
51     while (e.hasMoreElements()) {
52         tmpItem = (String[])e.nextElement();
53         totalCost += (Integer.parseInt(tmpItem[3]) *
54             Float.parseFloat(tmpItem[2]));
55     }
56     return totalCost;
57 }
58
59 public int getNumOfItems() {
60     Enumeration e = items.elements();
61     String[] tmpItem;
62     int numOfItems = 0;
63
64     while (e.hasMoreElements()) {
65         tmpItem = (String[])e.nextElement();
66         numOfItems += Integer.parseInt(tmpItem[3]);
67     }
68
69     return numOfItems;
70 }
71
72 }
73
74 }
75 }
```


Hasil Running Project : Bila add di Klik maka akan masuk 1 item ke dalam Shopping Cart



Firefox DVD Catalog +

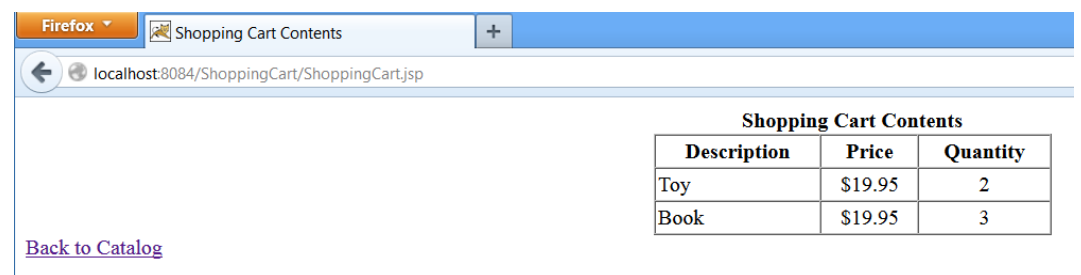
localhost:8084/ShoppingCart/index.jsp

[Shopping Cart Quantity: 5](#)

DVD Catalog

Description	Price	
Book	\$19.95	Add
Toy	\$19.95	Add

Setelah Klik Shopping Cart Quantity : Tampilan Sesuai Banyaknya Tombol Add di Klik



Firefox Shopping Cart Contents +

localhost:8084/ShoppingCart/ShoppingCart.jsp

Shopping Cart Contents

Description	Price	Quantity
Toy	\$19.95	2
Book	\$19.95	3

[Back to Catalog](#)

Pertemuan 6

Dasar-dasar JSP (Java Server Pages)

I. Gambaran Umum JSP

a. Pengertian JSP

JSP adalah teknologi pengembangan web site yang mensupport konten secara dinamis dengan menyisipkan sintak java didalam HTML.

b. Kelebihan JSP

Dengan menggunakan JSP maka web yang kita bangun akan memiliki beberapa kelebihan sebagai berikut:

1. Teknologi JSP dibangun dengan teknologi Java sehingga bisa running di multiplatform operating System seperti Windows, Linux, Mac OS dan Operating System yang lainnya.
2. Performance JSP lebih powerful dibanding dengan pemrograman CGI/Perl. JSP selalu di compile terlebih dahulu oleh server sebelum page di request. Tidak seperti CGI/Perl yang mengharuskan server untuk melakukan load interpreter dan target script setiap page tersebut di request.
3. JSP dapat digunakan dengan mengkombinasikan dengan Servlet sehingga dapat menangani permasalahan pada lapisan logika dalam pengolahan bisnis proses tersebut.
4. Dengan pemrograman JSP maka aplikasi yang dibangun berdasarkan pondasi Object Oriented Programming sehingga mudah di maintenance dan dapat menangani permasalahan yang kompleks.

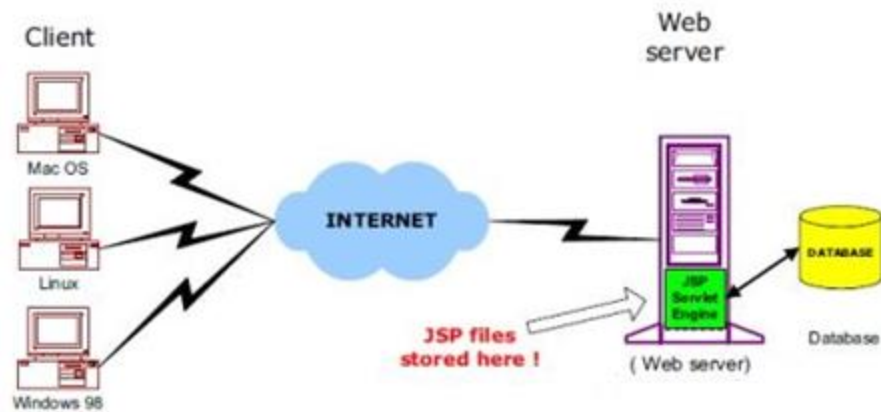
II. Kebutuhan Software

Untuk software yang diperlukan dalam proses pembelajaran JSP ini adalah :

1. jdk-7-windows-i586
2. XAMPP
3. MySQL Front
4. Netbeans 7.4.1

Untuk urutan proses instalasi software sesuai urutan dari nomor tersebut. Jadi yang diinstal pertama kali adalah JDK, lalu XAMPP, MySQL Front dan yang terakhir adalah Netbeans 7.4.1. Ketika menginstal netbeans ditahap pertama instalasi mohon klik tombol "Customize" dan check list untuk mengaktifkan paket apache tomcat untuk aktivasi web server berbasis apache tomcat.

III. Arsitektur JSP



Gb 3.1. Arsitektur JSP

(http://www.tutorialspoint.com/jsp/jsp_architecture.htm, 24 Feb 2014)

Web yang dibangun dengan JSP terdiri dari beberapa komponen sebagai berikut:

1. Web Server

Web Server merupakan sebuah software yang digunakan untuk menjalankan aplikasi web di komputer client. Web Server yang digunakan dalam instalasi JSP ini adalah Apache tomcat yang terinstal satu paket dalam Netbeans.

2. Client

Client adalah komputer yang digunakan oleh user untuk menampilkan aplikasi web yang dibangun dengan JSP. JSP bisa ditampilkan pada computer client dengan operating system yang multiplatform seperti Windows, Linux, MaC OS dan Operating System yang lain.

3. Internet

Internet adalah sebuah media yang dapat menampilkan aplikasi web yang kita bangun sehingga dapat dibuka di komputer client dari berbagai benua,negara atau kota.

4. Database

Database digunakan untuk menyimpan data yang diinput dari Form JSP. Data yang tersimpan didatabase dapat dilakukan proses pencarian,perubahan dan penghapusan data. Data didatabase dapat digunakan untuk kebutuhan pelaporan dari proses bisnis yang ada dalam web tersebut.

IV. Dasar Sintak JSP

a. Deklarasi Variabel JSP

Variabel adalah tempat untuk menampung data yang akan tersimpan dalam memori yang bisa berubah nilainya. Pada JSP bentuk

```
<%! Tipe Data nama variable= value; %>
```

Sebagai contoh:

```
<%! int angka = 10; %>
```

```
<div id="tampil">Nilai dari angka= <%=angka%> </div>
```

penulisan variabel adalah sebagai berikut:

b. Ekspresi di JSP

Penulisan ekspresi di JSP dituliskan sbb:

```
<%= expression %>
```

Contoh:

```
<p>
    Tgl Sekarang:
    <%= (new java.util.Date()).toLocaleString() %>
</p>
```

c. Komentar di JSP

Komentar di JSP memiliki dua kegunaan. Kegunaan yang pertama digunakan untuk membuat suatu keterangan dari suatu coding yang dibuat. Kegunaan kedua untuk menonaktifkan sintak JSP sehingga jika dicompile maka baris yang diberikan komentar tersebut tidak dibaca sebagai perintah program.

Bentuk Penulisan dari komentar JSP sebagai berikut:

```
<%--
Ini merupakan contoh dari komentar
--%>
```

d. JSP Directives

JSP Directives berpengaruh terhadap struktur dari class Servlet.

Tiga tipe dari Directives adalah:

Directive	Deskripsi
<% @page...%>	Mendefinisikan atribut page seperti scripting language, error page dan buffering requirements
<% @ include...%>	Memanggil file lain selama fase translasi
<% @ taglib...%>	Mendeklarasikan tag library, berisi proses action yang digunakan dalam page

e. JSP Actions

JSP Action digunakan untuk memproses setiap event atau aksi yang ada di JSP. JSP Action terdiri dari 3 komponen yakni:

1. <jsp:include> Action

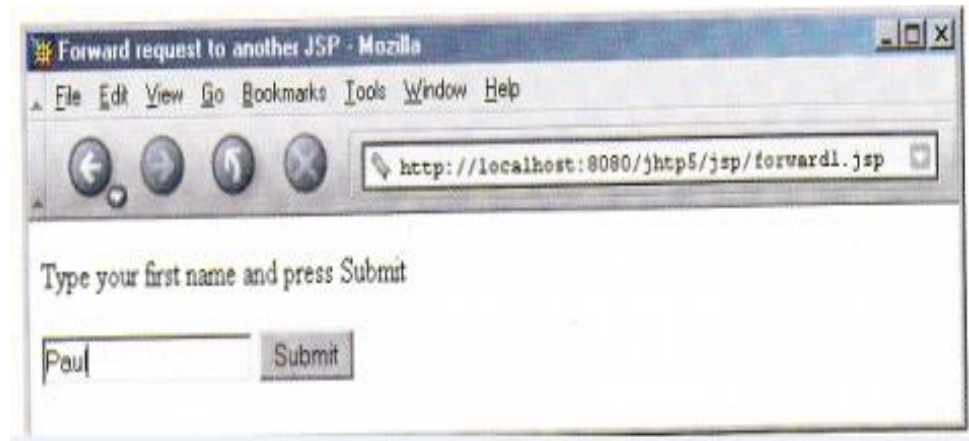
<jsp:include> digunakan untuk memanggil file jsp kedalam file jsp lain. Sebagai contoh: digunakan dalam pembuatan template yang mengintegrasikan beberapa Form kedalam template utama jika dipanggil dari menu di template utama. untuk lebih jelasnya lihat penggalan program dibawah ini:

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 25.10: include.jsp -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8
9   <head>
10     <title>Using jsp:include</title>
11
12     <style type = "text/css">
13       body {
14         font-family: tahoma, helvetica, arial, sans-serif;
15       }
16
17       table, tr, td {
18         font-size: .9em;
19         border: 3px groove;
20         padding: 5px;
21         background-color: #dddddd;
22       }
23     </style>
24   </head>
25
26   <body>
27     <table>
28       <tr>
29         <td style = "width: 160px; text-align: center">
30           <img src = "images/logotiny.png"
31             width = "140" height = "93"
32             alt = "Deitel & Associates, Inc. Logo" />
33         </td>
34
35         <td>
36           <!-- include banner.html in this JSP -->
37           <jsp:include page = "banner.html"
38             flush = "true" />
39         </td>
40       </tr>
41
42       <tr>
43         <td style = "width: 160px">
```

2. <jsp:forward> Action

<jsp:forward> digunakan untuk memforward suatu request dari satu page ke page yang lain. sebagai contoh: jika ada request yang di arahkan ke form A.jsp. Tetapi, pada Form A.jsp di forward ke B.jsp maka request dari Form A akan ditampilkan di Form B.jsp. Untuk lebih jelasnya lihat contoh dibawah ini:

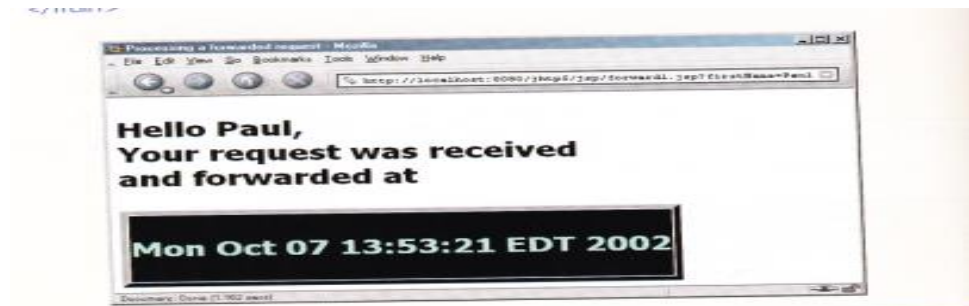
forward1.jsp



Sintak forward1.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%
      String name = request.getParameter("firstName");
      if(name!=null){
    %>
    <jsp:forward page="forward2.jsp">
      <jsp:param name="date" value="<%= new java.util.Date() %>"/>
    </jsp:forward>
    <%
      }else{
    %>
    <h1>Contoh Forward</h1>
    <form action="forward1.jsp">
      Input Nama Awal: <input type="text" name="firstName"/><br/>
      <input type="submit" value="Simpan"/>
    </form>
    <%
      }
    %>
  </body>
</html>
```

Forward2.jsp



Detail Sintak forward2.jsp

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
    <style type="text/css">
      .big{
        font-family:Tahoma,sans-serif,arial;
        font-weight:bold;
        font-size:2em;
      }
    </style>
  </head>
  <body>
    <h1>Hasil Forward</h1>
    <p class="big">
      <h2>Selamat Datang <%=request.getParameter("firstName") %></h2>
      <h2> Request sudah diterima dan diforward dengan melampirkan</h2>
    </p>
    <table style="border:6px outset;">
      <tr>
        <td style="background-color:green">
          <p class="big" style="color:cyan">
            <%=request.getParameter("date") %>
          </p>
        </td>
      </tr>
    </table>
  </body>
</html>
```

3. <jsp:useBean> Action

Untuk memanipulasi objek di java. Pada semester ini belum diajarkan.

f. JSP Implicit Object

Implicit Object adalah konsep bagaimana menyediakan akses ke servlet dari jsp. Objek Implicit (Implicit Object) memiliki 4 komponen yaitu:

1. Application

menyediakan kontainer dalam jsp

2. Page

Untuk pengolahan, pencetakan, penanganan error dalam page dengan sintak : out, response, config, exception.

3. request

Untuk menangani request dari inputan client.

4. Session

Objek yang menyediakan informasi dari client jika session tersebut dibuat. Biasanya digunakan untuk proses Login sebelum masuk ke web site.

Contoh dari Objek Implicit dapat dilihat pada penggalan program dibawah ini:

Contoh (oimpl.jsp):

```
<%@page contentType="text/html"%>
<html>
<head><title>Objek Implisit</title></head>
<body>
<BODY>
<H2>Mencoba Objek Implisit</H2>
<UL>
<LI>Waktu saat ini: <%= new java.util.Date() %>
<LI>Server: <%= application.getServerInfo() %>
<LI>Session ID: <%= session.getId() %>
<LI>Nilai parameter <CODE>Nama</CODE> :
<%= request.getParameter("Nama") %>
</UL>
</BODY></HTML>
```



Mencoba Objek Implisit

- Waktu saat ini: Thu Feb 19 21:40:44 GMT+07:00 2004
- Server: Apache Tomcat/4.0.6
- Session ID: B5F8E479CB02244369F56E7C8A1FB028
- Nilai parameter Nama : Budi Susanto

g. Struktur Kondisi JSP

Struktur kondisi digunakan untuk menyelesaikan permasalahan dengan logika manusia yang diubah dalam bahasa pemrograman JSP. Bentuk struktur kondisi yang pertama menggunakan IF...Else (Jika True....Jika False....). Contoh IF..Else sbb:

```
<%! int hari = 3; %>
<html>
<head><title>Contoh IF...ELSE</title></head>
<body>
<% if (hari == 1 | hari == 7) { %>
    <p> Merupakan Hari Weekend</p>
<% } else { %>
    <p> Hari ini bukan weekend</p>
<% } %>
</body>
</html>
```

Bentuk IF..Else IF..Else..

Bentuk ini digunakan untuk menyelesaikan suatu permasalahan, jika terdapat lebih dari satu kondisi. contoh dari penggunaannya sbb:

```
<%! int nilai = 80; %>
<html>
<head><title>Contoh IF..ELSE IF..ELSE ..</title></head>
<body>
<% if ( nilai > 84 ) { %>
    <p> Grade Anda A</p>
<% } else if(nilai > 74 ) { %>
    <p> Grade Anda B </p>
<% } else{
    <p> Grade Anda Kurang Sekali </p>
} %>
</body>
```

h. Struktur Perulangan di JSP

Struktur Perulangan memiliki beberapa kegunaan, salah satunya digunakan untuk mencetak bilangan dalam jumlah banyak sekaligus ataupun untuk menyelesaikan permasalahan yang berulang secara rutin. Bentuk penulisan dari struktur perulangan sbb:

```
<%! int nilai = 80; %>
<html>
<head><title>Contoh Perulangan ..</title></head>
<body>
<% if ( nilai > 84 ) { %>
    <p> Grade Anda A</p>
<% } else if(nilai > 74 ) { %>
    <p> Grade Anda B </p>
i. <% } else{
j.    <p> Grade Anda Kurang Sekali </p>
    } %>
</body>
```

Pertemuan 7

Pra UTS

Pada pertemuan ini digunakan untuk mereview dari pembahasan materi sebelumnya. Review dengan garis besar lebih ditekankan pada pembahasan konsep dasar Servlet dan JSP. Untuk mereview mahasiswa diberikan tugas tambahan untuk melihat sampai sejauh mana pemahaman mahasiswa tentang konsep Servlet dan JSP. Untuk tugasnya adalah mahasiswa dapat membuat program seperti pada gambar dibawah ini:

The image shows a web application interface with a green header bar containing the text "Web Perubahan". Below the header is a form titled "Master Dosen". The form contains the following fields and controls:

- NIP : Cari
- Nama :
- Pendidikan : S1 ▼
- Alamat :

At the bottom of the form, there are three buttons: "simpan" (circled in blue), "Ubah", and "Hapus".

1. Tugas JSP

Mahasiswa membuat Form seperti pada gambar diatas dan outputnya. setelah diklik tombol simpan maka tampilkan NIP, Nama, Pendidikan dan Alamat sesuai data yang diinput dengan konsep JSP.

2. Tugas Servlet

Mahasiswa membuat Form seperti pada gambar diatas dan outputnya. setelah diklik tombol simpan maka tampilkan NIP, Nama, Pendidikan dan Alamat sesuai data yang diinput dengan konsep Servlet dengan menggunakan konsep Response redirection dan Session Tracking.

Pertemuan 8

Ujian Tengah Semester

Pada pembahasan ini mahasiswa dibagi dalam kelompok dengan jumlah anggota minimal dan maksimal 3 orang. Tiap kelompok diharapkan dapat membuat program dengan konsep dasar JSP dan servlet dengan studi kasus:

1. Web Kuliner
2. Web Pemesanan Tiket (Web Kereta Api)
3. Web Penjualan (Web Jual Beli Bunga, Buku, dll)

Target dari studi kasus tersebut adalah mahasiswa dapat membuat Satu Form dengan data yang diinput dapat ditampilkan dengan logika dan pengembangan serta kreasi dari sesuai studi kasus ditiap kelompok.

Pertemuan 9

PembuatanTemplate

Pada pembahasan setelah uts ini, mahasiswa diarahkan untuk dapat mengembangkan dari setiap konsep dan materi dari JSP dan servlet yang sudah didapatkan pada pertemuan 1 s.d. pertemuan ke 6. Pada pertemuan ini mahasiswa lebih ditekankan untuk memahami dan membuat template dengan dapat mengintegrasikan semua Form baik Master maupun Transaksi kedalam Template yang sama. Untuk lebih jelasnya dapat mengikuti langkah-langkah dibawah ini:

1. Hasil Template

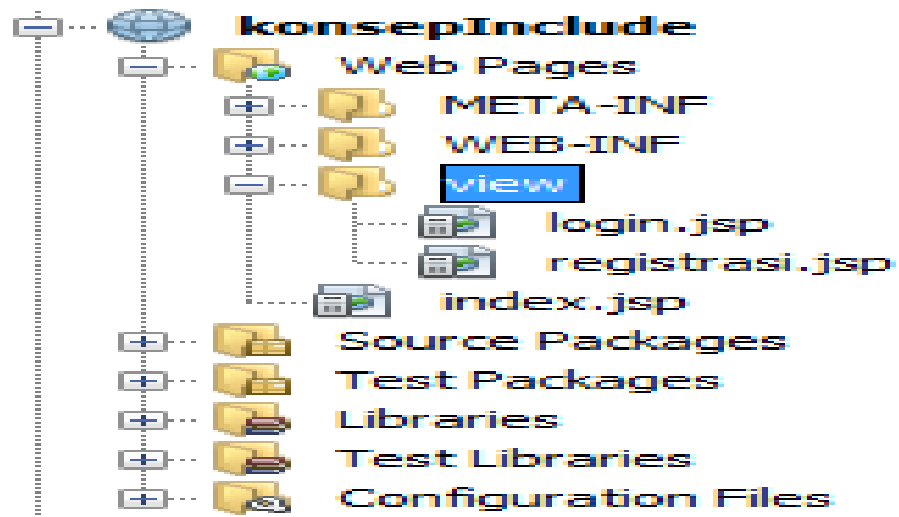
Hasil template yang akan dibuat dapat dilihat seperti pada gambar dibawah ini:

The image shows a web application template. At the top is a green header with the text "Web Perubahan". Below the header is a sidebar on the left with a green background, containing two links: "Menu Login" and "Menu Dosen". The main content area is titled "Master Dosen" and contains a form with the following fields and controls:

- NIP : Cari
- Nama :
- Pendidikan : S1 ▾
- Alamat :

At the bottom of the form are three buttons: "simpan", "Ubah", and "Hapus".

2. Kerangka Hasil Akhir Struktur File



3. Script Template Kerangka Utama

Script ini dibuat pada file index.jsp

```

<div id="konten">
  <div id="header">
    <h1>Web Perubahan</h1>
  </div>
  <div id="left">
    <h3><a href="index.jsp?id=login">Menu Login</a></h3><br/>
    <h3><a href="index.jsp?id=form">Menu Dosen</a></h3>
  </div>

  <div id="right"></div>
</div>

```

4. Script Desain Template CSS

Script ini ditempatkan diantara tag <head></head> pada file index.jsp. Untuk lebih jelasnya lihat gambar dibawah ini:

```

<style type="text/css" rel="stylesheet">
    #left{
        float:left;
        width: 20%;
        border-style:groove;
        background-color: #b8e186;
    }
    #right{
        margin-left: 300px;
        width: 40%;
        border-style:groove;
    }
    #header{
        width: 66%;
        height: 150px;
        border-style: groove;
        font-family: Tahoma;
        font-size: 32px;|
        font-weight: bold;
        background-color: #7fbc41; }
</style>

```

5. Script Integrasi template (Dengan <jsp:include>)

Script ini diterapkan diantara <div id="right"> pada file index.jsp.

Untuk lebih jelasnya lihat pada gambar dibawah ini:

```

<div id="right">
    <%
        //menangkap parameter dengan nama id
        if(request.getParameter("id")==null){
            out.print("data kosong");
        }else{
            String kode=request.getParameter("id");
            // cek jika menu yang kita pilih sama dengan form atau login
            if(kode.equals("form")){
                %>
                <jsp:include page="view/registrasi.jsp" flush="true" />
                <%
            }else if(kode.equals("login")){
                %>
                <jsp:include page="view/login.jsp" flush="true"></jsp:include>
                <%
            }
        }
    %>
</div>

```

pada pembahasan <jsp:include> ini digunakan untuk memanggil Form Login dan Form registrasi sehingga tertampil dalam template utama yaitu index.jsp. Dengan sintak tersebut maka setiap Form yang dipanggil akan ditampilkan dalam Menu Utama. Jadi Satu template berisi beberapa Form yang akan dipanggil sesuai kebutuhan. Mahasiswa diharapkan dapat membuat Form Login dan Form Mahasiswa seperti pada tahapan berikutnya dibawah ini.

6. Sintak Form Registrasi .jsp

```

<div id="dosen" align="center">
  <h1>Master Dosen</h1><br/>
  <form action="" method="post">
    <table>
      <tr>
        <td>NIP</td>
        <td>:</td>
        <td><input type="text" name="txtnip"><input type="submit" value="Cari"></td>
      </tr>
      <tr>
        <td>Nama</td>
        <td>:</td>
        <td><input type="text" name="txtnama"></td>
      </tr>
      <tr>
        <td>Pendidikan</td>
        <td>:</td>
        <td><select name="pddk">
          <option value="S1">S1</option>
          <option value="S2">S2</option>
          <option value="S3">S3</option>
        </select>
      </td>
    </tr>
    <tr>
      <td>Alamat</td>
      <td>:</td>
      <td><textarea rows="6" cols="10"></textarea>
    </td>
  </tr>
  <tr>
    <td colspan="3">
      <input type="submit" value="simpan">
      <input type="submit" value="Ubah">
      <input type="submit" value="Hapus">
    </td>
  </tr>
</table>
</form></div>

```

7. Sintak Form Login.jsp

```
<h1>Anda di Form Login</h1>
```

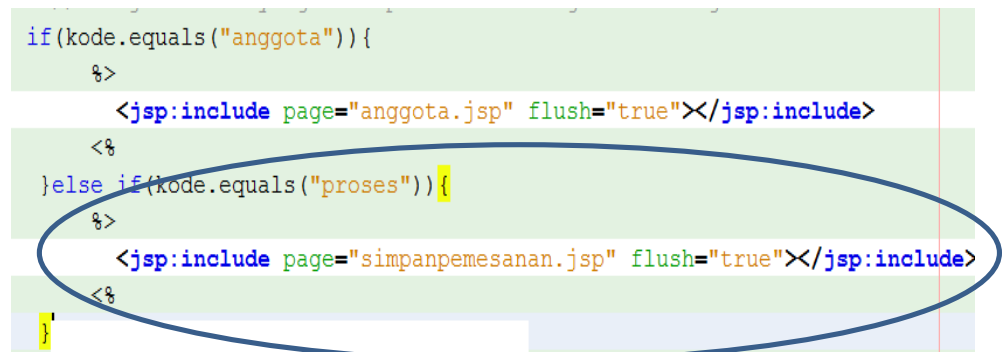
Tambahkan secara mandiri pembuatan form login dengan inputan berupa username dan password.

Pertemuan 10

Pembuatan Template Tingkat Lanjut

Pada pertemuan 10 ini, masih direview terkait pembahasan pembuatan template. Mahasiswa secara keseluruhan ditargetkan untuk bisa mendesain dan membangun sebuah template dengan kreasi seni dan inovasi masing-masing. Setelah mahasiswa dapat membuat template dengan integrasi Form yang dipanggil, maka mahasiswa diarahkan untuk mengolah setiap Form yang dipanggil dengan menampilkan inputan dari Form Registrasi.jsp dan kembali ke menu utama.

Untuk mengolah inputan supaya kembali ketemplate utama ada 2 tahap. Tahap pertama adalah Modifikasi dari Form Registrasi.jsp adalah pada tag `<form action="..." >` diubah menjadi `<Form action="index.jsp?id=proses">`. perubahan sintak ini digunakan untuk setiap inputan dari form registrasi akan ditampilkan dalam template utama. Tahap kedua adalah modifikasi pada file index.jsp dengan menambahkan sintak diantara `<div id="right">` sehingga hasilnya seperti pada gambar dibawah ini:



```
if(kode.equals("anggota")){  
    %>  
    <jsp:include page="anggota.jsp" flush="true"></jsp:include>  
    <%  
}else if(kode.equals("proses")){  
    %>  
    <jsp:include page="simpanpemesanan.jsp" flush="true"></jsp:include>  
    <%  
}
```

pada coding diatas adalah untuk pengolahan dari inputan form registrasi dengan parameter yang bernama "proses". Untuk menampilkan inputan dari Form Registrasi.jsp akan ditampilkan dalam file pengolahan dengan nama `simpanpemesanan.jsp`. Didalam file `simpanpemesanan.jsp` akan menampung dari setiap inputan dengan penggalan sintak sebagai berikut:

```
<% String id=request.getParameter("txtid1");  
String nama=request.getParameter("txtnama1");  
String alamat=request.getParameter("txtalamat1");  
String kodepos=request.getParameter("txtkodepos1");  
String telp=request.getParameter("txttelp1");
```

Tahap berikutnya tinggal pengembangan dari tiap mahasiswa untuk mengembangkan konsep template ini dalam studi kasus masing-masing.

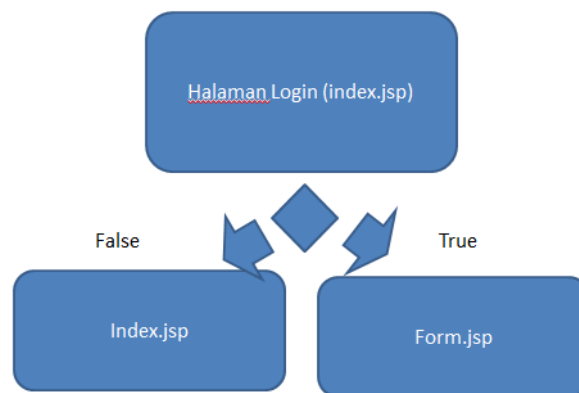
Pertemuan 11

Konsep Pembuatan Login di JSP

Pada pertemuan 11 ini mahasiswa diharapkan dapat membuat login dengan integrasi template di JSP. Untuk lebih jelasnya mohon ikuti tahapan dari langkah-langkah berikut:

1. Alur Web Login

Alur Web Session



Form Login akan dibuat pada file index.jsp, jika login true maka akan diarahkan ke Form.jsp. tetapi, jika Login gagal maka akan diarahkan kembali ke halaman login yaitu index.jsp.

2. Buat Project dan Form Login
 - a. Buat project dengan nama "SessionJavaWeb"
 - b. Modifikasi index.jsp sehingga menjadi sebagai berikut:

Username :	<input type="text"/>
Password :	<input type="password"/>
<input type="button" value="Login"/>	

3. Script Index.jsp

```

] <html>
]   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
-   </head>
]   <body>
]     <%
      session.removeAttribute("user");//hapus session
-     %>
]     <Form method="POST" action="OlahLogin">
      Username : <input type="text" name="txtnama"><br/>
      Password : <input type="password" name="txtpass"><br/>
      <input type="submit" name="cmdlogin" value="Login">
-     </Form>
-   </body>
- </html>

```

4. Pengolahan Session

pengolahan login dengan konsep Session pada class Servlet dengan nama file "OlahLogin.java". modifikasi sintak pada processRequest() sehingga hasil akhir sebagai berikut:

```

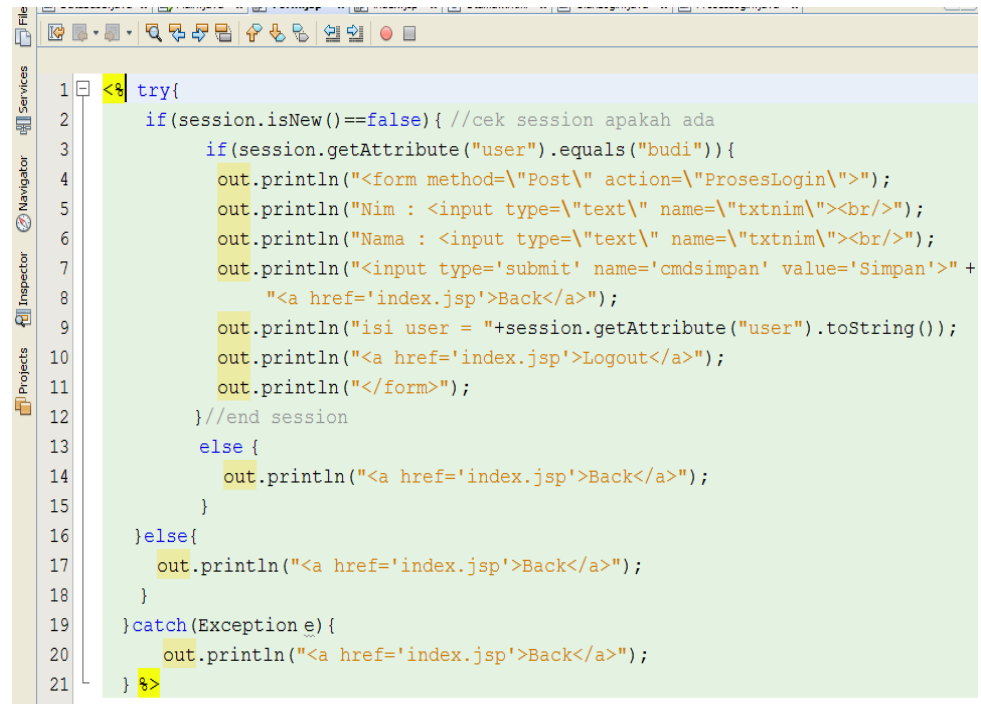
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();

    //deklarasi Session
    HttpSession session = request.getSession(true);
    String user = request.getParameter("txtnama");
    String pass = request.getParameter("txtpass");

    try {
        // pembuatan session
        if (user.equals("budi") && pass.equals("pass")) {
            session.setAttribute("user", user);
            if (session.getAttribute("user").equals("budi")) {
                response.sendRedirect("Form.jsp");
            } else {
                response.sendRedirect("index.jsp");
            }
        } else {
            response.sendRedirect("index.jsp");
        }
    }
}

```

5. Jika Login Valid akan diarahkan ke Form.jsp dengan sintak sbb:



```
1 <% try{
2     if(session.isNew()==false){ //cek session apakah ada
3         if(session.getAttribute("user").equals("budi")){
4             out.println("<form method=\"Post\" action=\"ProsesLogin\">");
5             out.println("Nim : <input type=\"text\" name=\"txtnim\"><br/>");
6             out.println("Nama : <input type=\"text\" name=\"txtnim\"><br/>");
7             out.println("<input type='submit' name='cmdsimpan' value='Simpan'>" +
8                 "<a href='index.jsp'>Back</a>");
9             out.println("isi user = "+session.getAttribute("user").toString());
10            out.println("<a href='index.jsp'>Logout</a>");
11            out.println("</form>");
12        } //end session
13        else {
14            out.println("<a href='index.jsp'>Back</a>");
15        }
16    } else {
17        out.println("<a href='index.jsp'>Back</a>");
18    }
19 } catch (Exception e) {
20     out.println("<a href='index.jsp'>Back</a>");
21 }
```

Pertemuan 12

Aplikasi Untuk Menyimpan, Merubah dan Menghapus Data ke Database MySQL

Pada pertemuan 12 ini mahasiswa diharapkan dapat membuat Form JSP yang terkoneksi ke database dengan Servlet sebagai class control. konsep ini mengacu pada pola model, view dan controller. Dengan model adalah model disisi database, view adalah JSP sebagai form inputan dan controller adalah class servlet. Untuk lebih ringkasnya ikuti tahapan dibawah ini:

1. Buat project di java netbeans dengan nama “KoneksiServletDB”
2. Modifikasi file index.jsp dengan script dibawah ini:

```
10
11 <html>
12 <head>
13   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14   <title>JSP Page</title>
15 </head>
16 <body>
17   <h1>Form Biodata Pegawai</h1>
18   <pre>
19   <form method="post" action="prosesKoneksi">
20     NIP : <input type="text" name="txtnip">
21     <br/>
22     Nama : <input type="text" name="txtnama">
23     <br/>
24     Gol : <select name="gol"><option value="IB" >IB</option><option value="IIB" checked="">IIB</option><option value="IIIB">IIIB</option>
25   </select><br/>
26     WILAYAH KERJA : <input type="checkbox" name="semarang" value="1">Semarang <input type="checkbox" name="jakarta" value="1">Jakarta<br/>
27
28     <input type="submit" name="cmdsimpan" value="Simpan"> <input type="submit" value="Tampil" name="cmdtampil"> <input type="reset" value="Reset">
29   </form>
30   </pre>
31 </body>
32 </html>
33
```

hasil Form index.jsp sbb:

Form Biodata Pegawai

NIP :

Nama :

Gol :

WILAYAH KERJA : ☐ Semarang ☐ Jakarta

3. Buat class Koneksi.java pada package jdbc dengan sintak sbb:

```
package jdbc;
import java.sql.*;

/**
 *
 * @author indra
 */
public class Koneksi {
    public Connection bukaKoneksi() throws SQLException{
        Connection connect;
        try{
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            connect=DriverManager.getConnection("jdbc:mysql://localhost:3306/biodata","root","");
            return connect;
        }
        catch(Exception exc){

        }
        return null;
    }
}
```

pada tahapan ini pastikan libray mysql JDBC Driver sudah diadd pada folder libraries diproject netbeans anda. Dengan klik kanan “Libraries” pilih add Library lalu pilih MySQL JDBC Driver.

4. Buat Class ProsesKoneksi.java . Class ini digunakan untuks mengolah proses simpan, cari, ubah dan hapus dalam class servlet. Untuk lebih jelasnya lihat pada script dibawah ini:

Modifikasi method processRequest pada class Servlet

- a. Proses Simpan

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try{
        String nip = request.getParameter("txtnip");
        String nama = request.getParameter("txtnama");
        String gol = request.getParameter("gol");
        int semarang = Integer.parseInt(request.getParameter("semarang"));
        int jakarta = Integer.parseInt(request.getParameter("jakarta"));
        //memanggil class Koneksi
        Koneksi obj = new Koneksi();
        Connection conn=obj.bukaKoneksi();
        Statement stm=conn.createStatement();
        ///////////////////////////////////////////////////
        if(request.getParameter("cmdsimpan")!=null){
            String query = "insert into pegawai values ('"+nip+"','"+nama+"','"+gol+"','"+semarang+"','"+jakarta+"')";
            out.println("isi query "+query);
            /// menjalankan query
            stm.executeUpdate(query);
            out.println("data sudah disimpan");
            out.println(" <a href='index.jsp'>BACK</a> ");
        }
    }
}

```

b. Proses Cari/Tampil Data

```

else if(request.getParameter("cmdtampil")!=null){
    ////// proses menampilkan dari database ///////////////
    String query2 = "select * from pegawai";
    ResultSet rs = st.executeQuery(query2);
    out.println("<table border='1'><tr><td>Nip</td><td>Nama</td> " +
        "<td>Gol</td><td>Wilayah Kerja</td></tr>");
    String wil="";
    while(rs.next()){
        out.println("<tr><td>"+rs.getString(1)+"</td>");
        out.println("<td>"+rs.getString(2)+"</td>");
        out.println("<td> "+rs.getString(3)+"</td>");
        out.println("<td> "+wil+"</td>");
        out.println("<td><a href='prosesKoneksi?lihat="+rs.getString(1)+"' > " +
            "lihat</a> </td>");
        out.println("</tr>");
    }
    out.println("</table>");
}

```



```

else if(request.getParameter("lihat") != null ){
    String nim = request.getParameter("lihat");
    ///// proses menampilkan dari database /////
    String query3 = "select * from pegawai where nip='"+nip+"'";
    ResultSet rs = stm.executeQuery(query3);
    if(rs.next()){
        out.println("<form method='post' action='prosesKoneksi'>");
        out.println("<table border='0'><tr><td>Nip</td> +
            "<td><input type='text' name='txtnip' value='"+rs.getString(1)+"'></td> +
            "</tr>");
        out.println("<tr><td>Nama</td> +
            "<td><input type='text' name='txtnama' value='"+rs.getString(2)+"' +
            "</td></tr>");
        out.println("<tr><td>Gol</td><td><select name='gol'>");
        if(rs.getString(3).equals("IB")){
            out.println("<option value='IB' selected>IB</option> ");
            out.println("<option value='IIB' >IIB</option> ");
            out.println("<option value='IIIB' >IIIB</option> ");
        }else if(rs.getString(3).equals("IIB")){
            out.println("<option value='IB' >IB</option> ");
            out.println("<option value='IIB' selected>IIB</option> ");
            out.println("<option value='IIIB' >IIIB</option> ");
        }else if(rs.getString(3).equals("IIIB")){
            out.println("<option value='IB' >IB</option> ");
            out.println("<option value='IIB' >IIB</option> ");
            out.println("<option value='IIIB' selected >IIIB</option> ");
        }
        out.println("</select></td></tr>");
        out.println("<tr><td>Wilayah Kerja </td><td> ");

        if(rs.getInt(4)==1){
            out.println("<input type='checkbox' name='semarang' value='1' checked>Semarang ");
            out.println("<input type='checkbox' name='jakarta' value='1' >Jakarta ");
        }else if(rs.getInt(5)==1){
            out.println("<input type='checkbox' name='semarang' value='1' >Semarang ");
            out.println("<input type='checkbox' name='jakarta' value='1' checked>Jakarta ");
        }out.println("</td></tr>");
        out.println("<tr> +
            "<td> <input type='submit' value='ubah' name='cmdubah'></td> +
            "<td><input type='submit' value='Hapus' name='cmdhapus'></td>");
        out.println("</tr>");

        out.println("</table></form>");
    }
}

```

c. Proses Ubah dan Hapus ke database

```
}else if(request.getParameter("cmdubah")!=null){

    String query = "update pegawai set nama='"+nama+"'," +
        "gol='"+gol+"',semarang='"+semarang+"'," +
        "jakarta='"+jakarta+"' where nip='"+nip+"'";
    stm.executeUpdate(query);
    out.println("data sudah diubah");
    out.println(" <a href='index.jsp'>BACK</a> ");
}else if(request.getParameter("cmdhapus")!=null){
    String query = "delete from pegawai where nip='"+nip+"'";
    stm.executeUpdate(query);
    out.println("data sudah dihapus");
    out.println(" <a href='index.jsp'>BACK</a> ");
}

// menutup koneksi
conn.close();
stm.close();
}catch(Exception e){
    out.println("error : "+e);
}
```

Pertemuan 13

Aplikasi Untuk ke Database Tingkat Lanjut

Pada pertemuan 13 ini mahasiswa diharapkan untuk dapat menguasai dengan tuntas konsep dan pembuatan aplikasi koneksi Java Web ke database MySQL dengan konsep JSP dan Java Servlet. Pada pertemuan mahasiswa akan dicek secara keseluruhan untuk memastikan bahwa setiap mahasiswa dapat membuat aplikasi koneksi java web ke database dengan tuntas dan betul.

Pertemuan 14

Hosting dan Domain Java Web

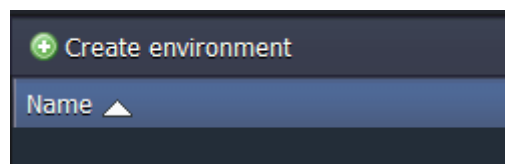
Pada pertemuan ini, mahasiswa diharapkan dapat melakukan instalasi dan upload file java web yang telah dibuat kedalam web hosting. file yang diupload adalah file dengan extension .war yang berada pada folder dist. Hosting yang digunakan harus berbasis apache Tomcat, bukan apache seperti biasa. Karena apache untuk web berbasis PHP.

Hosting berbasis Apache Tomcat salah satunya ada di url <http://jelastic.com>. Untuk mempelajari dan melakukan hosting pada jelastic tersebut dapat dilihat dengan mengikuti langkah-langkah dibawah ini:

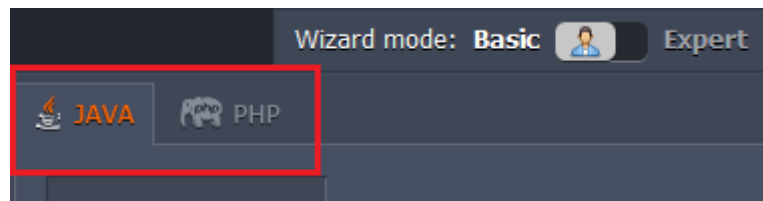
How to Create your Jelastic Environment ***(<http://docs.jelastic.com/setting-up-environment>, 3 Feb 2014)***

To create your Jelastic environment, follow these steps:

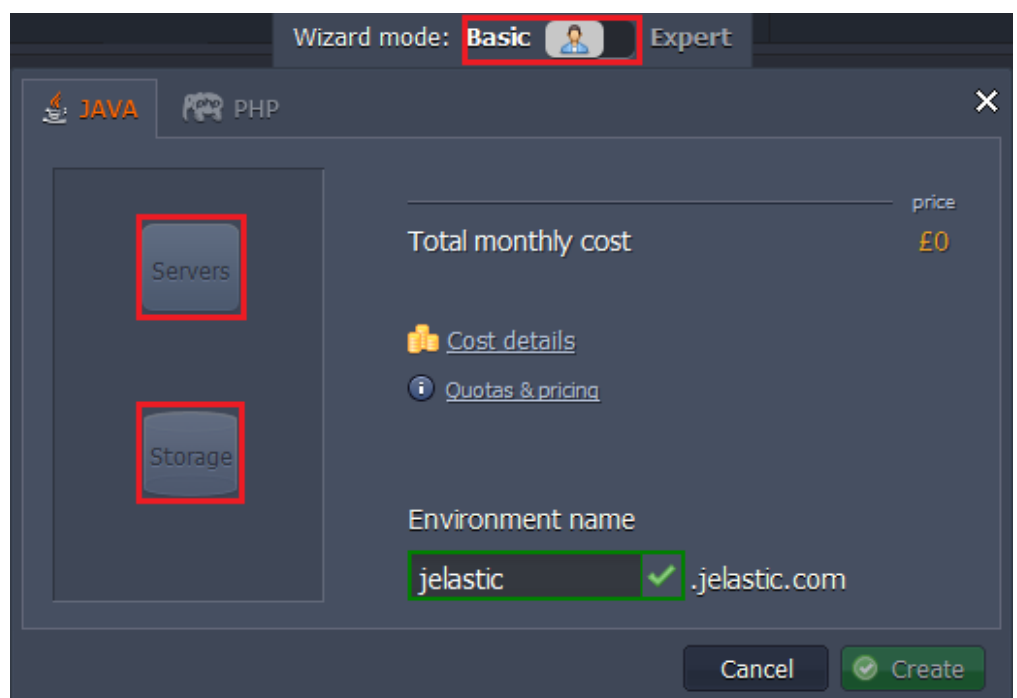
1. Log into the Jelastic Manager
2. Click **Create environment** in the upper left corner of the dashboard.



3. The **Environment topology** dialog box will be opened, where you can customize your environment settings. For the beginning choose your programming language, by clicking on the appropriate tab in the upper part of dialog box:

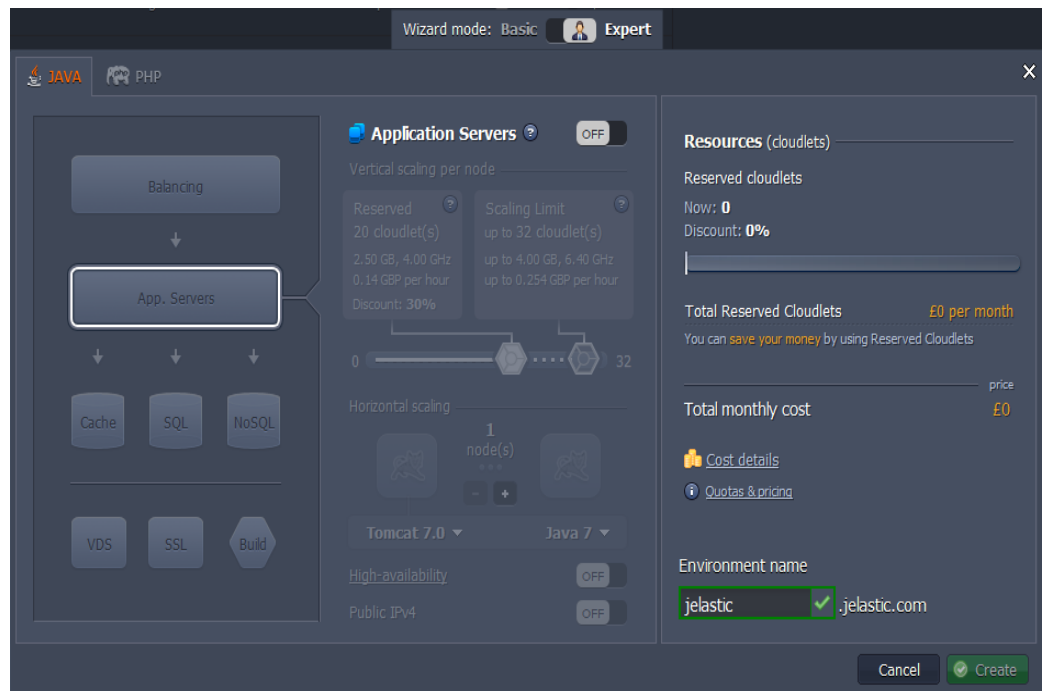


4. You have two modes available to you within the Wizard - **Basic** and **Expert**.
- Choose **Basic** mode by clicking on the button as shown below:



Using this mode you can create a simple environment quickly. All you have to do is select your application server and database, and type your desired environment name and click **Create**.

- If you want to be able to fully customize your environment, use **Expert**. The Wizard will look like this:

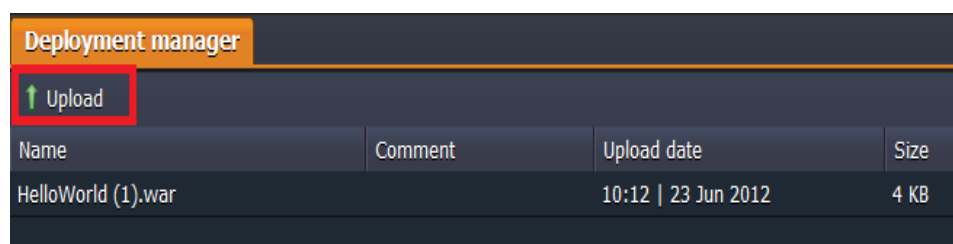


Untuk setting file .war ikuti langkah dibawah ini:

The **Jelastix Deployment Manager** gives you a convenient way to control your deployed web applications. You can easily add and deploy any of them to a specified environment or delete them if needed.

In order to add new Java application package:

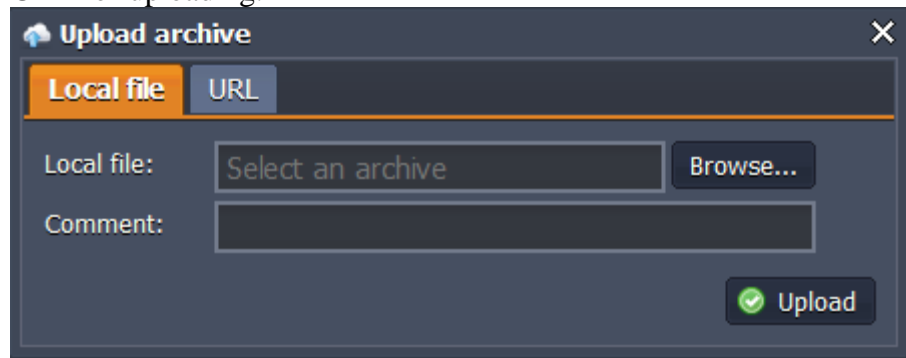
1. Upload your application package by clicking **Upload** in **Jelastix Deployment Manager**.



2. Browse to your local file. The **Jelastix Deployment Manager** supports **.WAR**, **.ZIP** or **.EAR** formats.

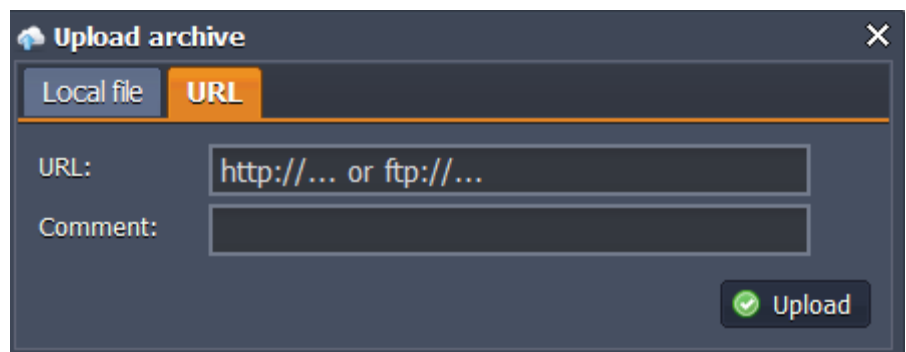
Note that the maximum available size of the uploaded local file is

150 MB. If your application size exceeds this limit, please use the URL for uploading.



The screenshot shows a dark-themed dialog box titled "Upload archive" with a close button (X) in the top right corner. It has two tabs: "Local file" (selected and highlighted in orange) and "URL". Under the "Local file" tab, there is a "Local file:" label, a text input field containing the placeholder "Select an archive", and a "Browse..." button to its right. Below this is a "Comment:" label and an empty text input field. At the bottom right, there is a green "Upload" button with a checkmark icon.

3. To add new application via URL (**http://**, **https://** or **ftp://**) navigate to the appropriate tab and enter there the required link.



The screenshot shows the same "Upload archive" dialog box, but with the "URL" tab selected and highlighted in orange. The "Local file" tab is now greyed out. Under the "URL" tab, there is a "URL:" label, a text input field containing the placeholder "http://... or ftp://...", and a "Comment:" label with an empty text input field below it. The green "Upload" button with a checkmark icon remains at the bottom right.

NOTE:

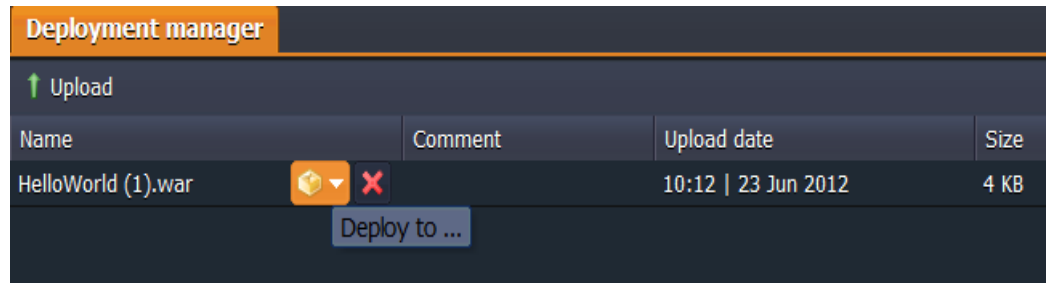
- Upload manager just links to the file
- If you modify your linked file but do not change the link to it there is no need to enter URL again
- If the *ftp* server needs authorization you'll have to put your URL in the following format:

ftp://{login}:{password}@{server_name}/{path_to_file}/{MyPackage.war}

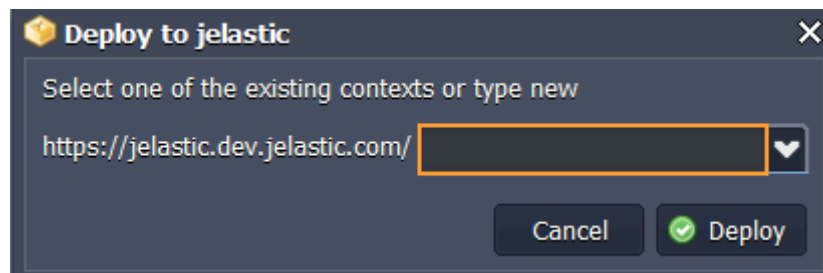
- If you happen to select a file with an unsupported extension or you enter an incorrect URL, you will get an error message informing you of it.

4. You can track the progress of the package uploading in the Tasks pane. Once the file is uploaded, it gets listed along with all your previous packages. Once uploaded, you have access to the **Deploy** and **Delete** options for that file.

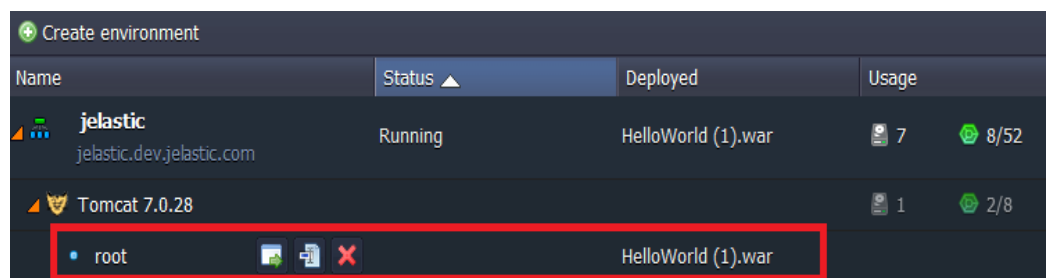
5. Select your package in the list and click **Deploy to** drop-down menu.



6. To get your java web application hosted choose the environment where you want to deploy it. In the opened window, specify the application's target context. Click **Deploy**.



7. Your application will be deployed to the chosen environment in just a minute.



Now you can open and use your Java web application

Anda dapat menggunakan hosting jelastic ini secara Gratis pada 1 minggu pertama, untuk selanjutnya anda dapat melakukan hosting berbayar berbasis cloud dengan mengikuti langkah-langkah di web <http://jelastic.com> tersebut.

Pertemuan 15

Pra UAS

Pada pertemuan ini mahasiswa dibagi dalam beberapa kelompok untuk mendapatkan topik terkait uas. kelompok minimal 1 dan maksimal harus 3 mahasiswa. Topik yang bisa dipilih oleh mahasiswa adalah sebagai berikut:

1. Web SIM Rumah Sakit
2. Web Pemesanan Tiket Pesawat, Kereta Api
3. Zakat Online
4. Qurban Online
5. dll

Mahasiswa harus memilih salah satu dari topik tersebut dan dibuat programnya dengan menggunakan konsep JSP sebagai view dan Java Servlet sebagai Controller serta database dengan MySQL. Rule atau aturan dari pembuatan project tersebut adalah:

1. Web harus menggunakan template
2. Ada login untuk admin dan user biasa
3. Menu Form terdiri dari minimal 3 Form Master dan 1 Form Transaksional sesuai topik terpilih
4. Kembangkan dalam validasi di tiap Form
5. Di setiap form proses Simpan, Tampil, Ubah dan Hapus harus berjalan dengan benar
6. Jika sampai hosting jelastik maka bonus tiap kelompok 5 point.

Pertemuan 16
Ujian Akhir Semester

Mahasiswa diharapkan mendemonstrasikan program setiap topik yang dipilih dari tiap kelompok di depan kelas. Tolak Ukur penilaian adalah:

1. Desain template
2. Validasi
3. Proses Master
4. Proses Transaksi
5. Proses Login
6. Hosting

DAFTAR PUSTAKA

Patzer, Andrew. 2002. JSP Examples and Best Practices. Apress: USA

Bryan, et al. 2008. Head First Servlet and JSP Second Edition. O'Reilly Media: USA.

Deitel, Paul., Deitel, Harvey., 2010. Java How to Program 8 th Edition. Prentice Hall.

Hall, Marty., Brown, Larry. "Core Servlet and Java Server Pages 2 nd Edition". Prentice Hall

Hall, Marty. "More Servlets and Java Server Pages",Prentice Hall

Tutorial JSP. 2013. <http://www.tutorialspoint.com/jsp/>. 3rd Apr 2013

Application Configuration. 2012.
<http://docs.jelastic.com/application-configuration>. 2nd
March 2012