

# Membuat Aplikasi Web dengan Eclipse Web Tools Platform

**Yanu Widodo**

[yanuwid@gmail.com](mailto:yanuwid@gmail.com)

## **Lisensi Dokumen:**

Copyright © 2003-2007 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Tulisan ini berisi langkah-langkah dasar pembuatan aplikasi web menggunakan Eclipse Web Tools Platform (Eclipse WTP). Eclipse WTP merupakan sebuah *platform* yang dikembangkan Eclipse Foundation begitu rupa sehingga memudahkan programmer yang ingin melakukan *deploying*, *running*, dan *testing* terhadap sebuah aplikasi web.

## **1. Pendahuluan**

Untuk membuat aplikasi web berbasis java, servlet khususnya, paling tidak ada enam langkah. Pertama, meng-*compile* servlet. Kedua, membuat folder aplikasi web (*document root*). Ketiga, membuat folder WEB-INF di folder aplikasi tadi. Keempat, membuat web.xml dan folder class di dalam WEB-INF. Kelima, meng-*copy* class yang sudah dikompilasi ke dalam folder class sebelumnya. Dan yang terakhir, menyesuaikan file web.xml sesuai dengan servlet yang telah dibuat [1]

Dapat dibayangkan bagaimana jika seorang pemrogram ingin melakukan perubahan terhadap aplikasi tersebut. Tentu akan sangat merepotkan bukan? Nah, untuk memudahkan langkah-langkah itu, dibuatlah Eclipse WTP.

## **2. Persiapan**

Karena kode-kode JSP yang digunakan pada artikel ini sangat dasar, pembaca diharapkan telah cukup familiar dengan HTML, java, servlet, dan JSP. Selain itu, juga diperlukan beberapa software pendukung, yaitu: Java Development Kit (JDK), Tomcat, dan Eclipse IDE for Java EE Developers.



## 2.1. Java Development Kit

JDK versi terlengkap dan terbaru telah disediakan Sun Microsystem di <http://java.sun.com/javase> baik untuk sistem operasi Solaris, Windows, atau Linux (yang digunakan pada tulisan ini merupakan versi Windows). Sedangkan proses instalasinya, dapat dilihat di <http://java.sun.com/javase/6/webnotes/install>



## 2.2. Eclipse IDE for Java EE Developers

Berbeda dengan *Eclipse Clasic*, pada IDE tipe ini sudah dilengkapi fitur *Web Tools Platform* yang dapat digunakan untuk pengembangan aplikasi Web dan J2EE. Fitur ini dilengkapi sejumlah tool dan API untuk memudahkan proses *deploying*, *running*, dan *testing* aplikasi. [3]. Installer-nya dapat diperoleh di halaman: <http://www.eclipse.org/downloads/>. Jika sebelumnya sudah terinstall eclipse classic, cara lain mendapatkan WTP adalah melalui *update manager*. Informasinya ada di <http://download.eclipse.org/webtools/updates/>



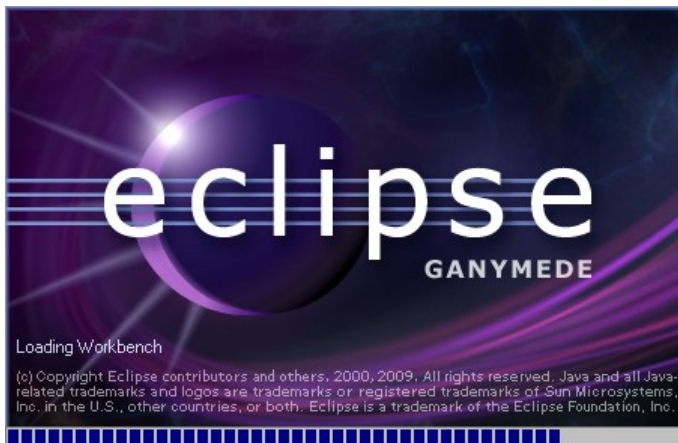
## 2.3. Tomcat

Tomcat adalah *servlet container* yang dikembangkan oleh Apache Software Foundation yang merupakan implementasi spesifikasi Java Servlet dan Java Server Pages yang dikeluarkan Sun Microsystem [2]. Installer tomcat dapat diperoleh di <http://tomcat.apache.org/download-60.cgi>. Sedang cara instalasinya dapat dilihat di <http://tomcat.apache.org/tomcat-6.0-doc/setup.html>



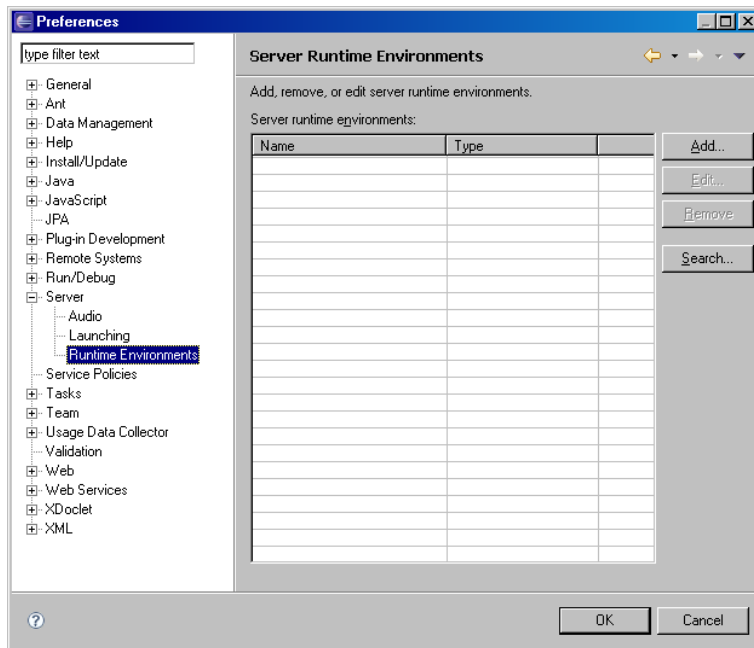
## 3. Membuat Aplikasi Web

Setelah proses instalasi selesai, akan dijelaskan konfigurasi eclipse, pembuatan JSP, pembuatan servlet, dan cara *deployment*.

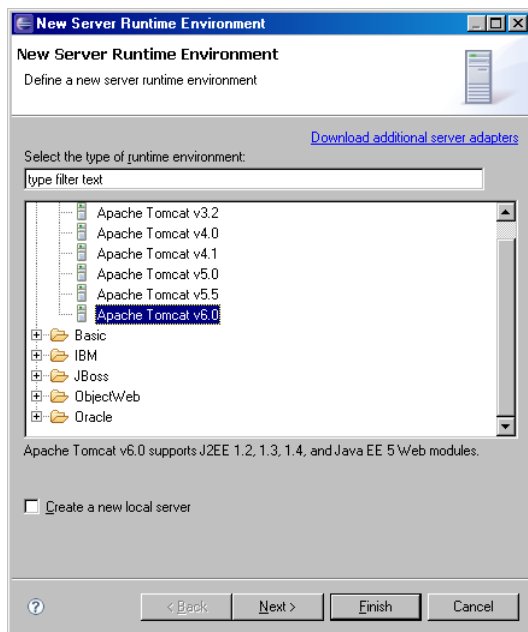


### 3.1. Konfigurasi

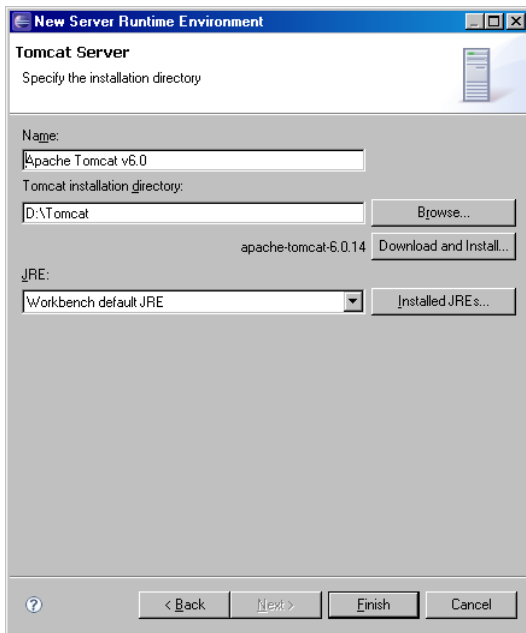
Untuk membuat sebuah aplikasi web, yang pertama sekali dilakukan adalah menambahkan Servlet Container, ke dalam Server Runtime Environments. Caranya, pada menu bar, klik: Window > Preferences > Server > Runtime Environments.



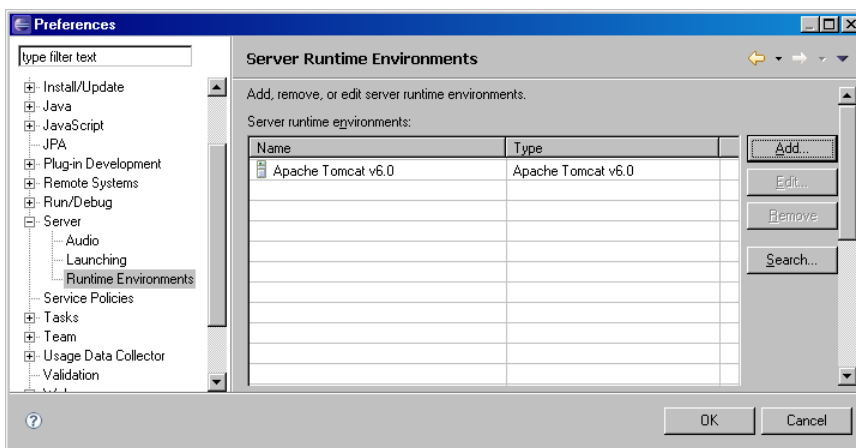
Klik Add. Akan muncul window seperti dibawah:



Pilih tipe tomcat yang sesuai dengan yang telah ter-install (di sini yang dipilih Tomcat v6.0). Kemudian klik Next.

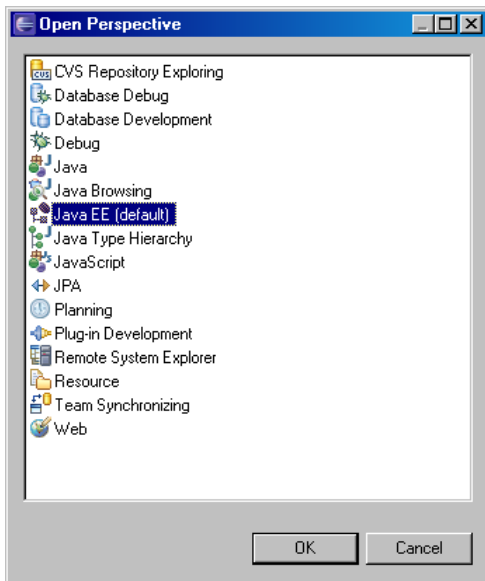


Setelah tombol Finish diklik, muncul window seperti ini.



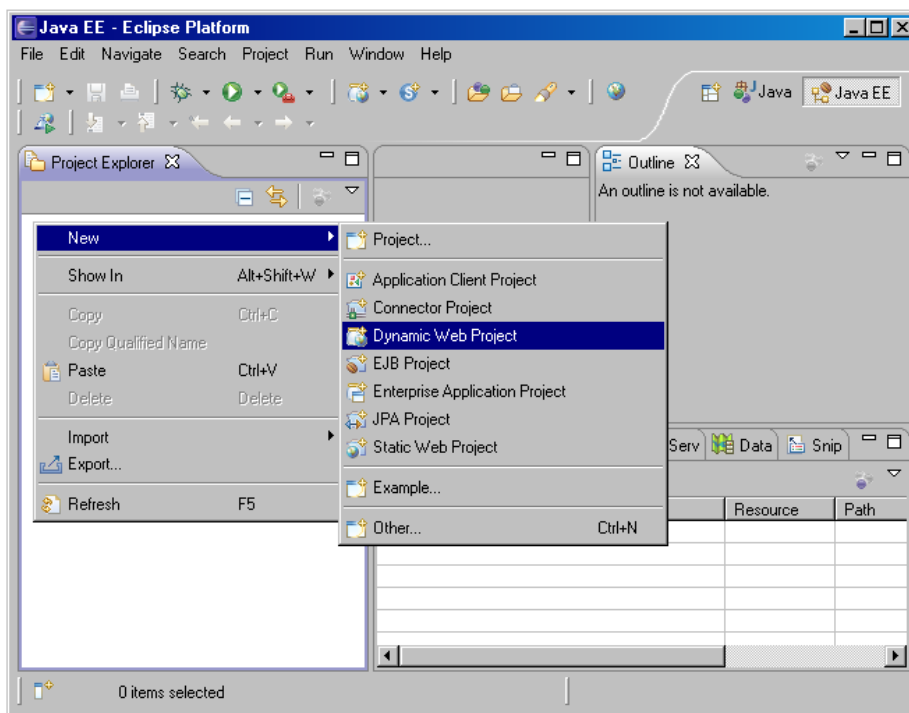
Klik OK.

Kemudian, langkah selanjutnya adalah membuat Dynamic Web Project. Sebelumnya, *show perspective*-nya sebaiknya diset terlebih dahulu ke Java EE. Dengan perspektif ini, kode-kode html atau xml akan lebih mudah dibaca. Caranya, pada menu bar, klik: Windows > Open Perspective > Other. Akan muncul window seperti ini:

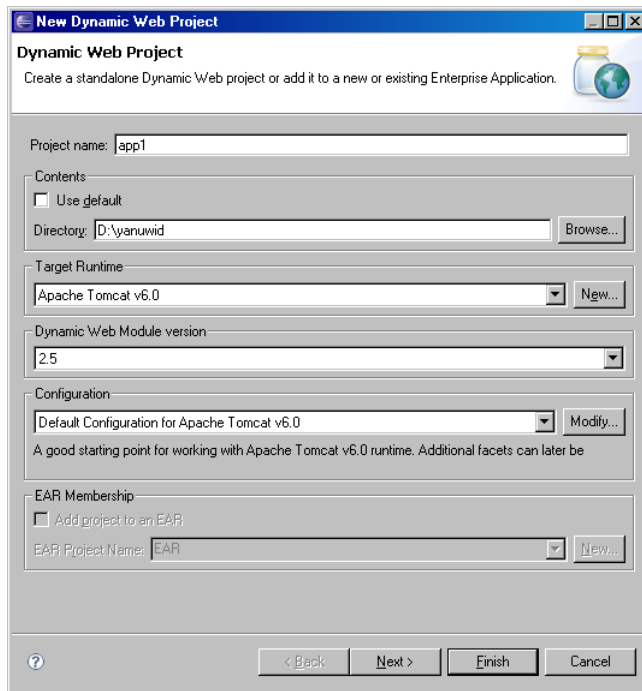


Klik OK, maka perspektif akan berubah ke Java EE.

Setelah itu, arahkan pointer di atas Project Explorer. Klik kanan > New > Dynamic Web Project.

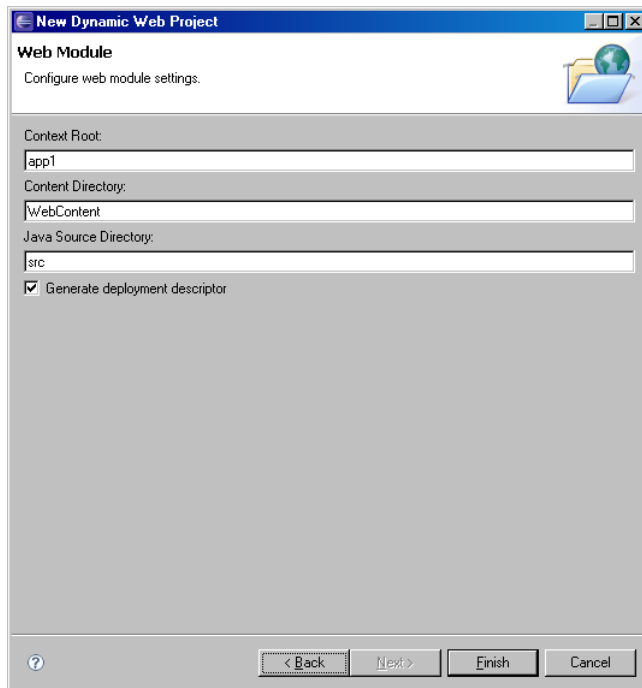


Akan muncul window seperti di bawah.

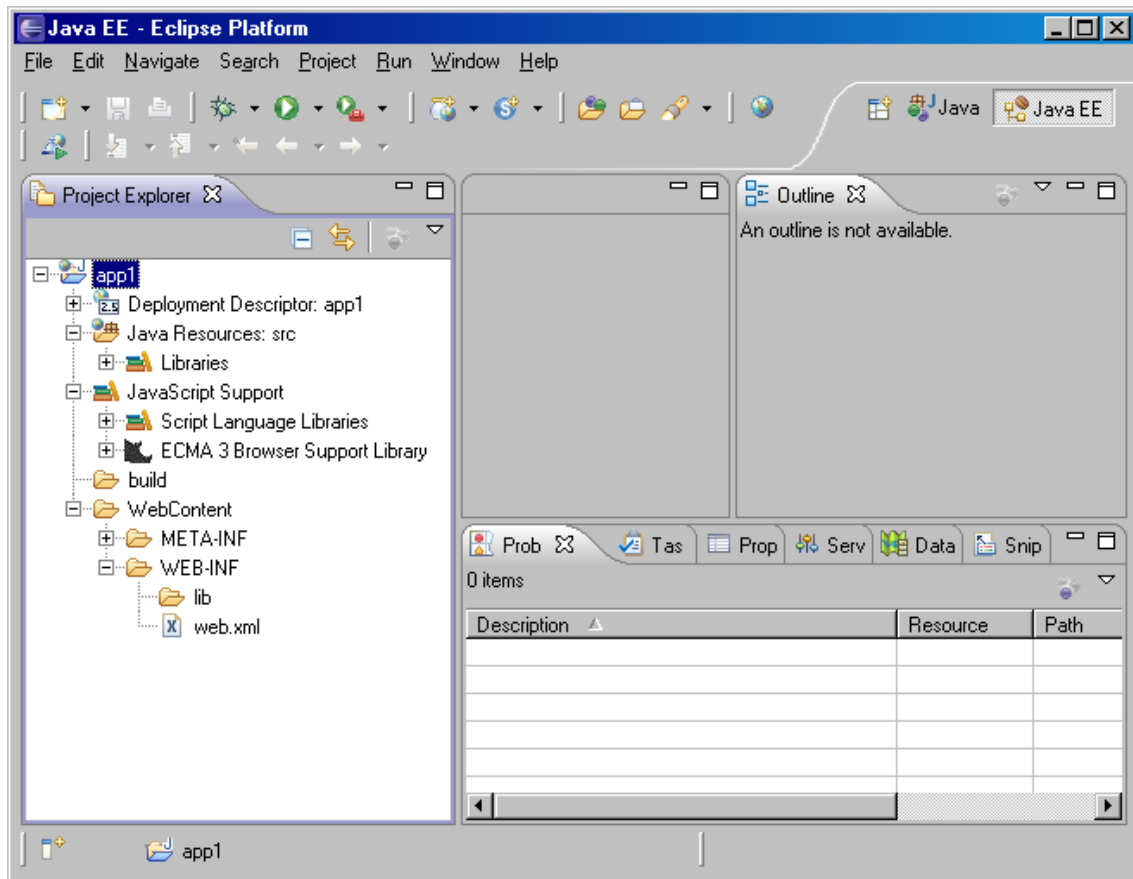


Misalnya, project yang akan dibuat diberi nama “app1”.

Kemudian klik next.



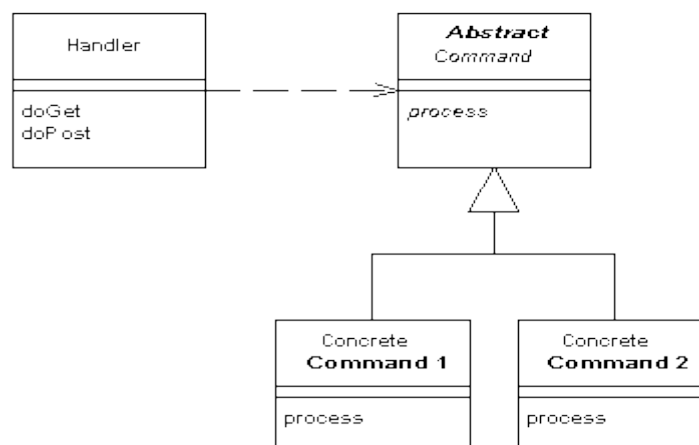
Inisialisasi Dynamic Web Project akan selesai setelah tombol finish ditekan. Setelah itu, pada *project explorer* akan nampak struktur modul aplikasi web yang akan dibuat.



Kode-kode JSP dan HTML berada di dalam folder **WebContent**. Kode-kode servlet dan java berada dalam folder **Java Resources: src**. File konfigurasi, berada dalam folder **WEB-INF**.

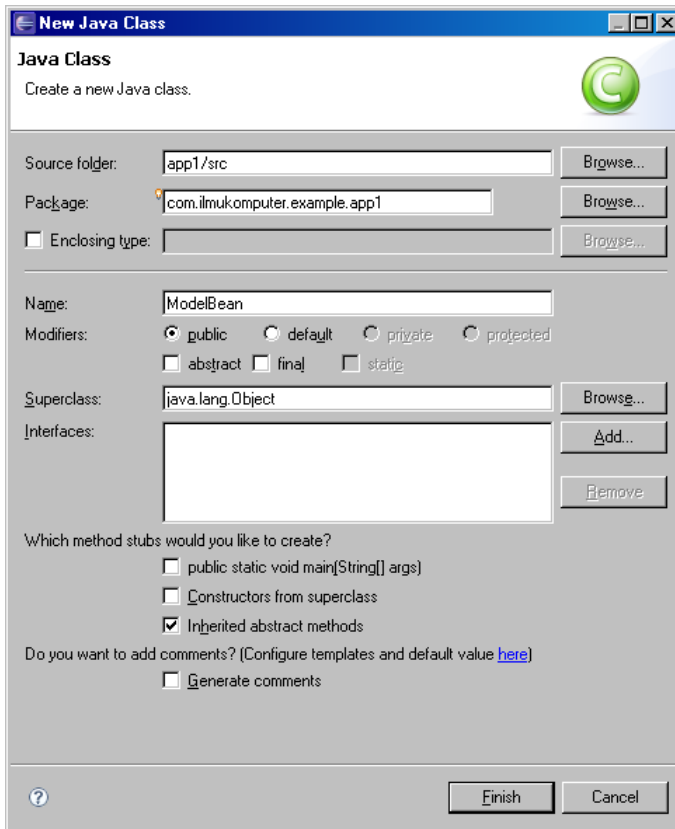
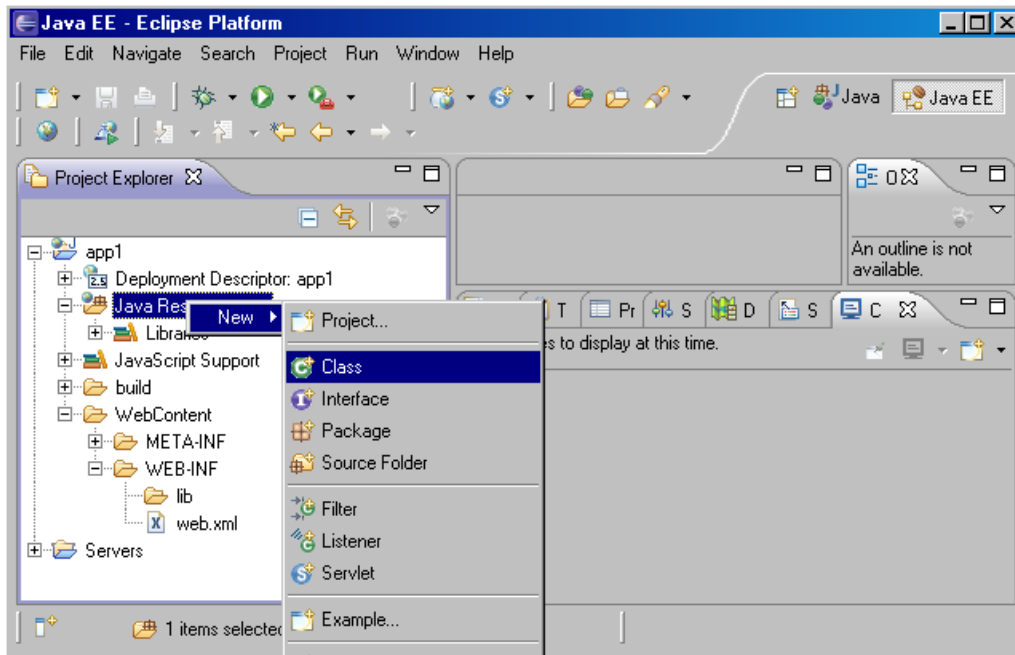
### 3.2. Membuat Bean, Servlet, dan JSP

Kode-kode dibawah ini merupakan sebuah contoh aplikasi sederhana Front Controller[5] yang merujuk contoh yang ditulis oleh Ben Souther [4]. Front Controller dipakai untuk menggabungkan penanganan sejumlah request dengan meneruskannya melalui sebuah *handler object* tunggal [6].



#### 3.2.1 Membuat Bean

Bean yang dimaksud di sini adalah sebuah class yang mempunyai properti private, getters, dan setters [7]. Pada project explorer, klik kanan Java Resource > New > Class



Klik finish dan tuliskan kode ModelBean.java berikut:

```
package com.ilmukomputer.example.app1;

public class ModelBean {
    private String firstName;
    private String lastName;
    private String email;
    private String phone;

    public String getFirstName() {
        return fixNull(this.firstName);
    }
}
```

```
public void setFirstName(String firstName){
    this.firstName = firstName;
}

public String getLastName(){
    return fixNull(this.lastName);
}

public void setLastName(String lastName){
    this.lastName = lastName;
}

public String getEmail(){
    return fixNull(this.email);
}

public void setEmail(String email){
    this.email = email;
}

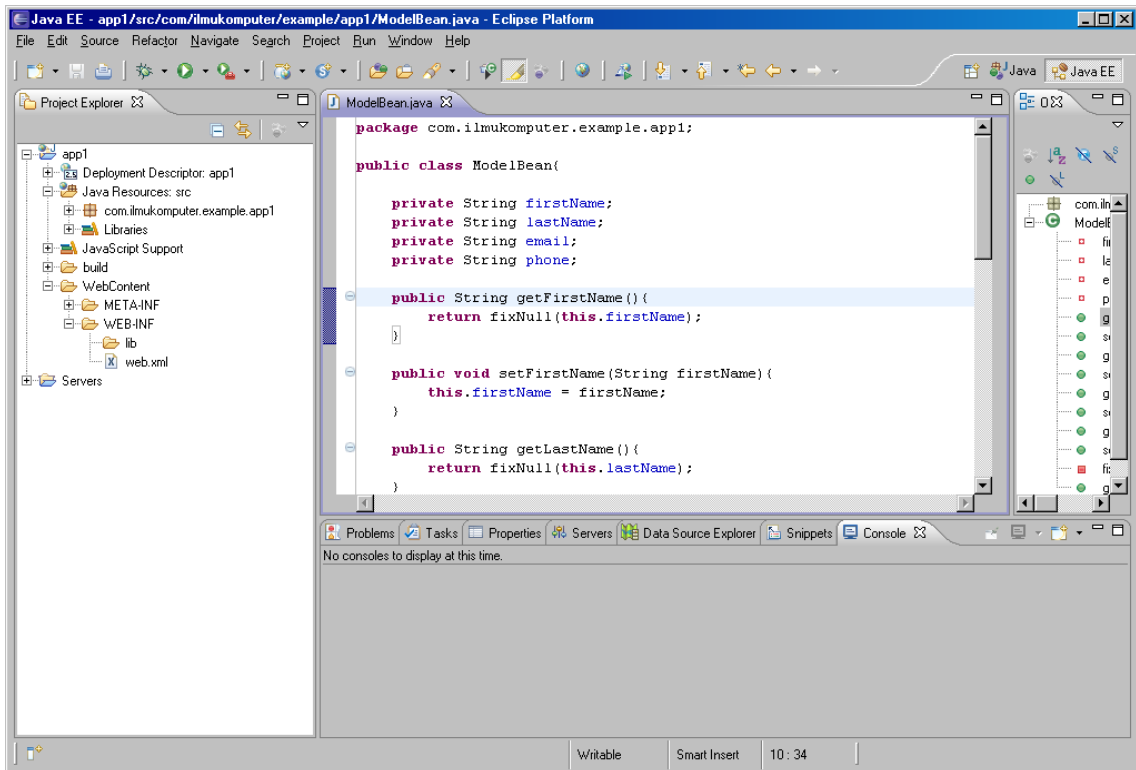
public String getPhone(){
    return fixNull(this.phone);
}

public void setPhone(String phone){
    this.phone = phone;
}

private String fixNull(String in){
    return (in == null) ? "" : in;
}

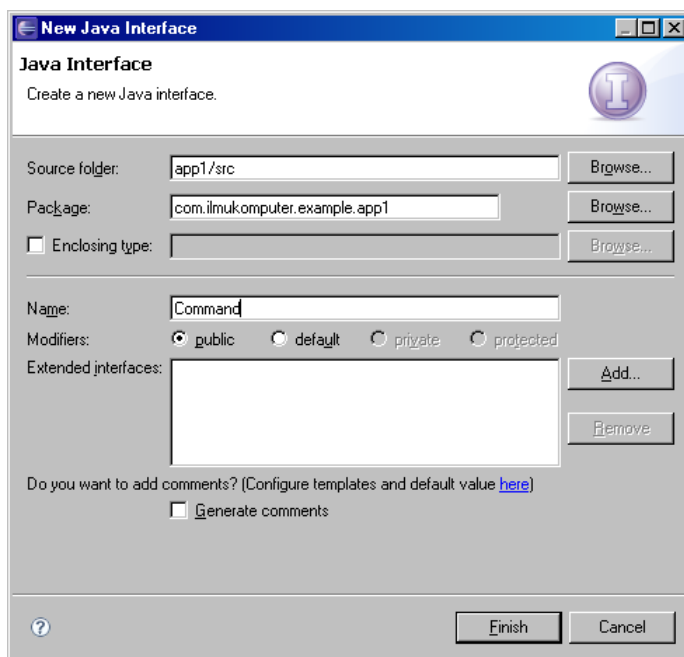
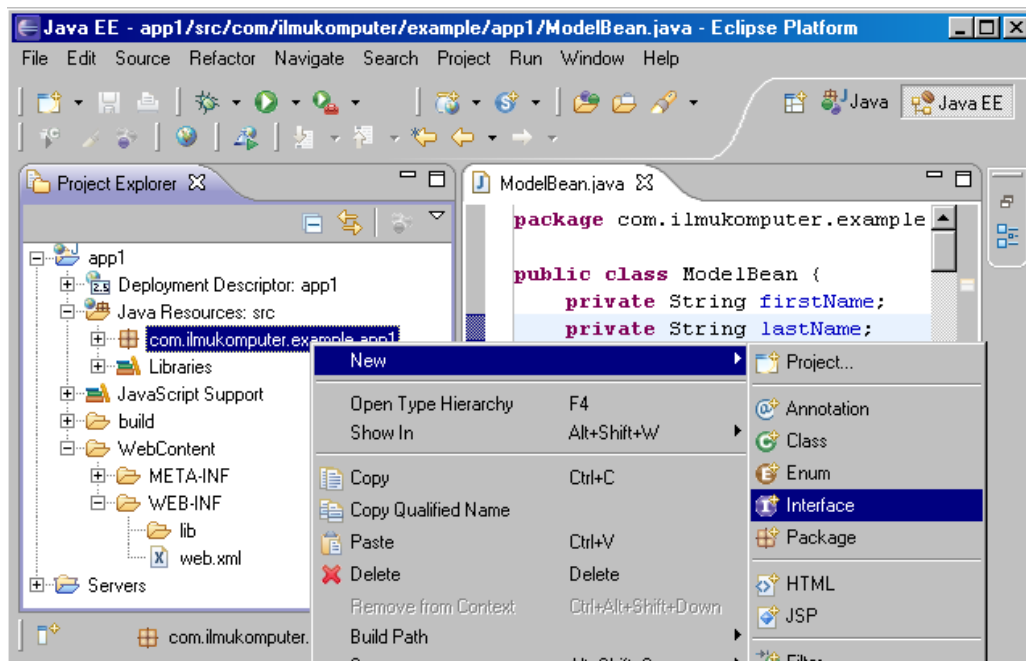
public String getMessage(){

    return "\nFirst Name: " + getFirstName() + "\n"
        + "Last Name: " + getLastName() + "\n"
        + "Email: " + getEmail() + "\n"
        + "Phone: " + getPhone() + "\n";
}
}
```



### 3.2.2 Membuat servlet

Sesuai dengan UML FrontController, servlet yang akan dibuat membutuhkan beberapa class: satu interface dan dua sub-class.



Klik finish dan tuliskan kode Command.java berikut:

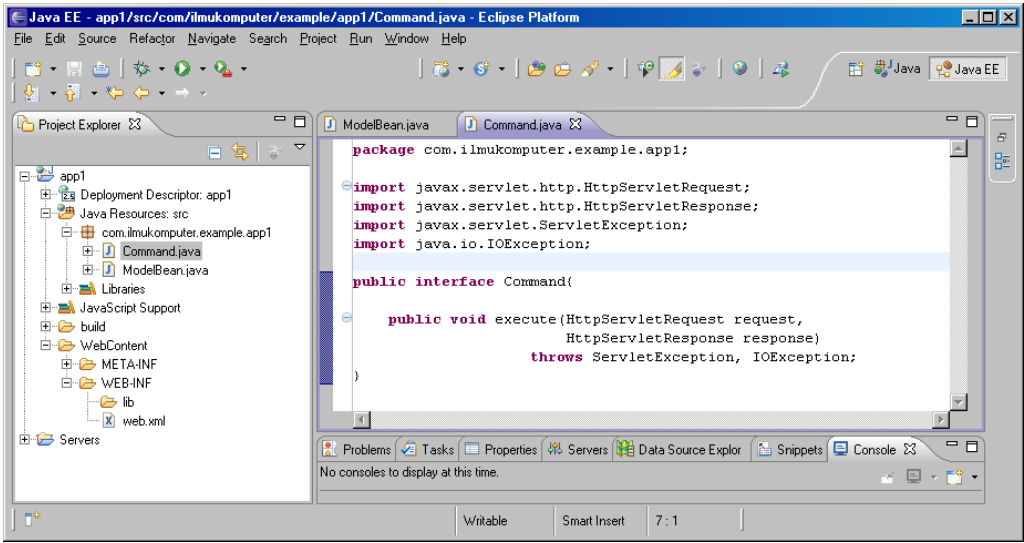
```
package com.ilmukomputer.example.app1;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;

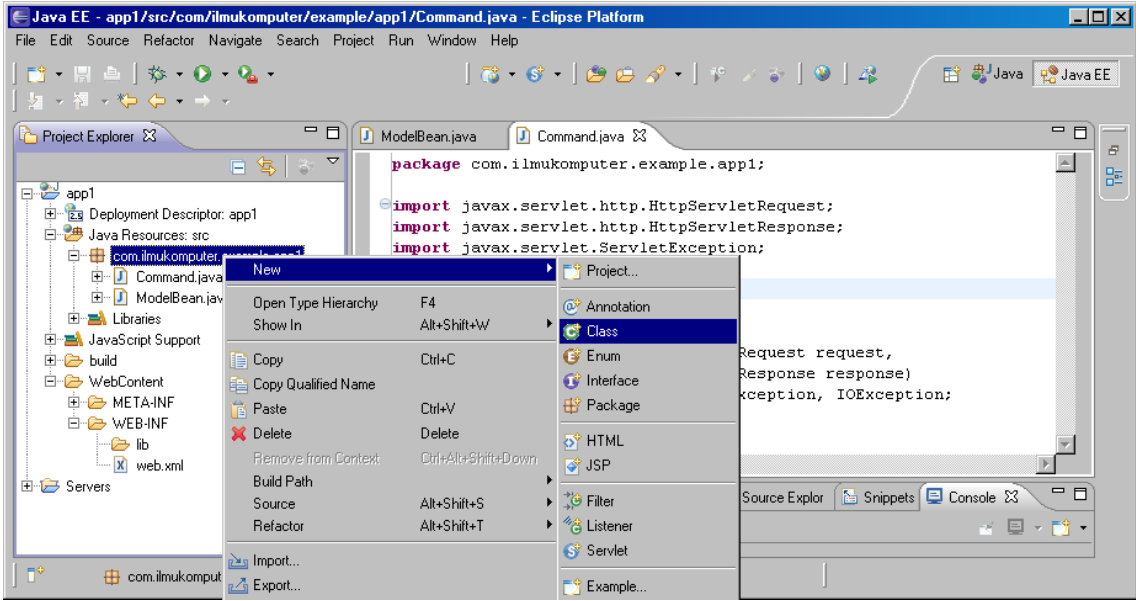
public interface Command{

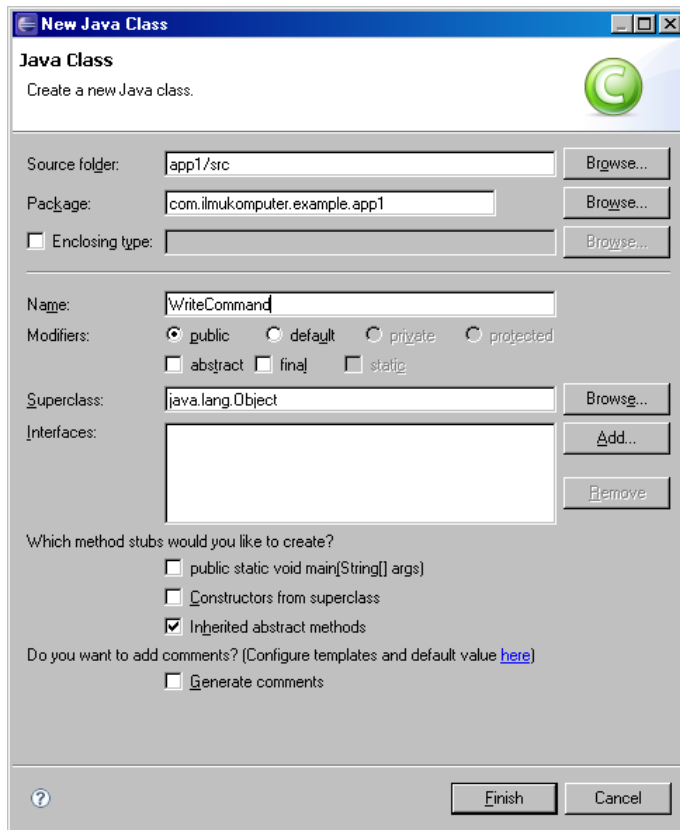
    public void execute (HttpServletRequest request,
```

```
HttpServletResponse response)  
throws ServletException, IOException;  
}
```



Kemudian buat sub-classnya Action





Klik finish dan tuliskan kode WriteCommand.java berikut:

```
package com.ilmukomputer.example.app1;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import java.io.IOException;

public class WriteCommand implements Command{

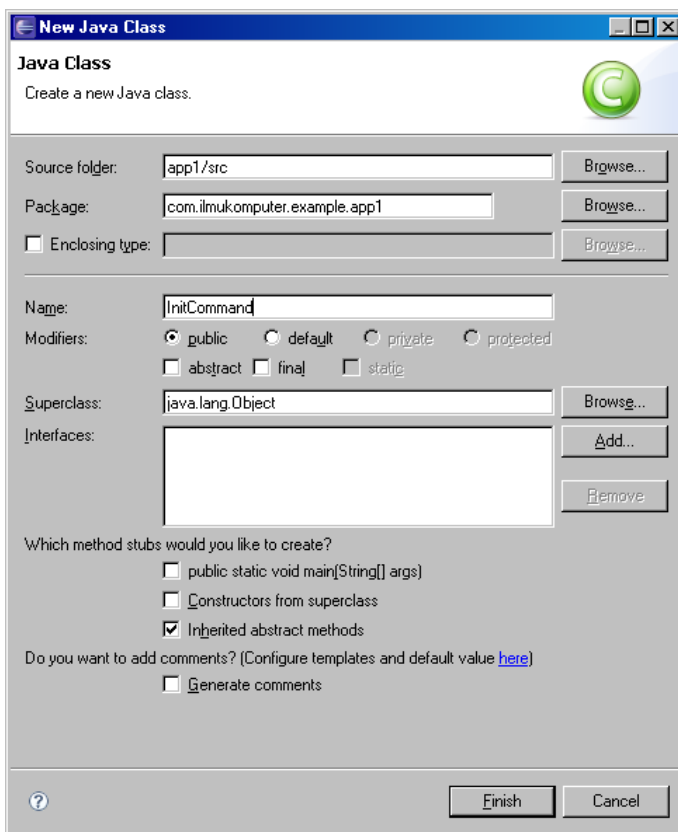
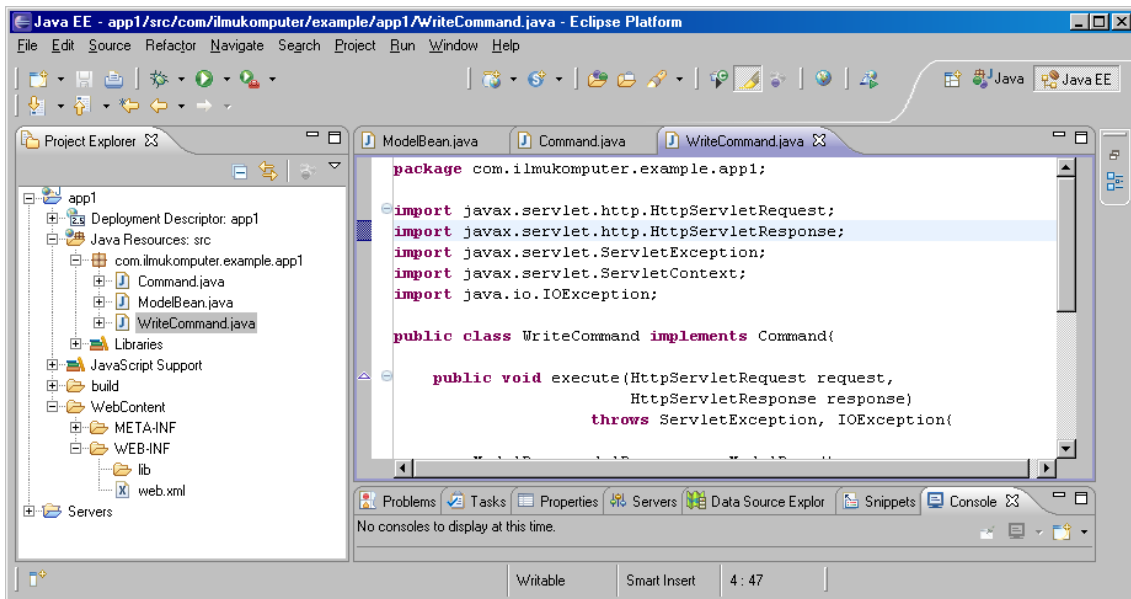
    public void execute(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException{

        ModelBean modelBean = new ModelBean();

        modelBean.setFirstName(request.getParameter("first_name"));
        modelBean.setLastName( request.getParameter("last_name"));
        modelBean.setEmail(    request.getParameter("email"));
        modelBean.setPhone(    request.getParameter("phone"));

        request.setAttribute("modelBean", modelBean);

        ServletContext context = request.getSession().getServletContext();
        context.getRequestDispatcher("/view.jsp").forward(request, response);
    }
}
```



Klik finish dan tuliskan kode InitCommand.java berikut:

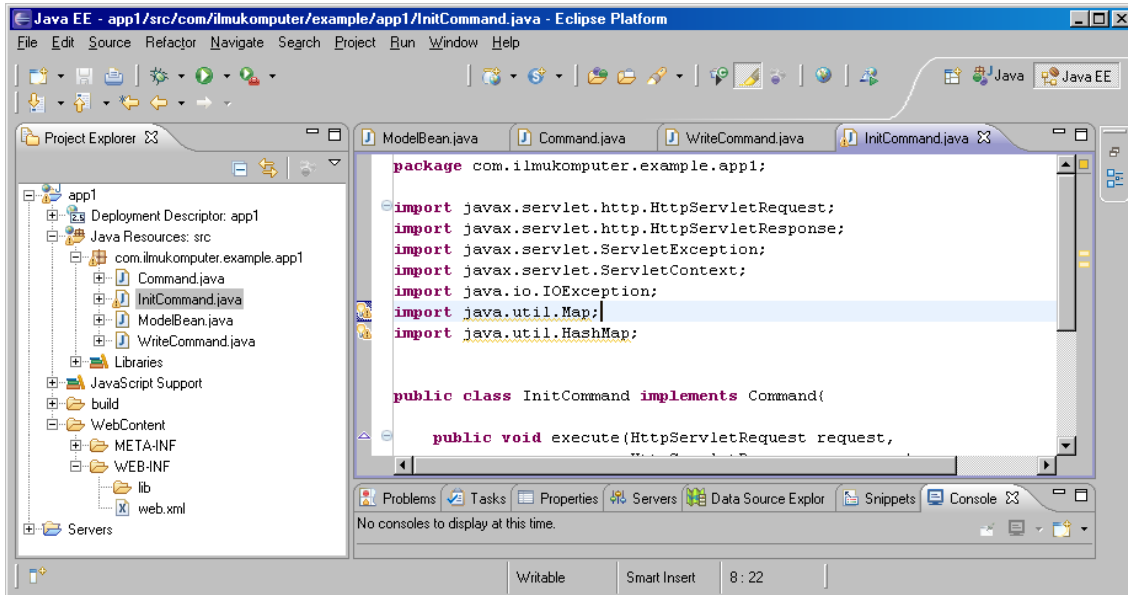
```
package com.ilmukomputer.example.app1;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import java.io.IOException;

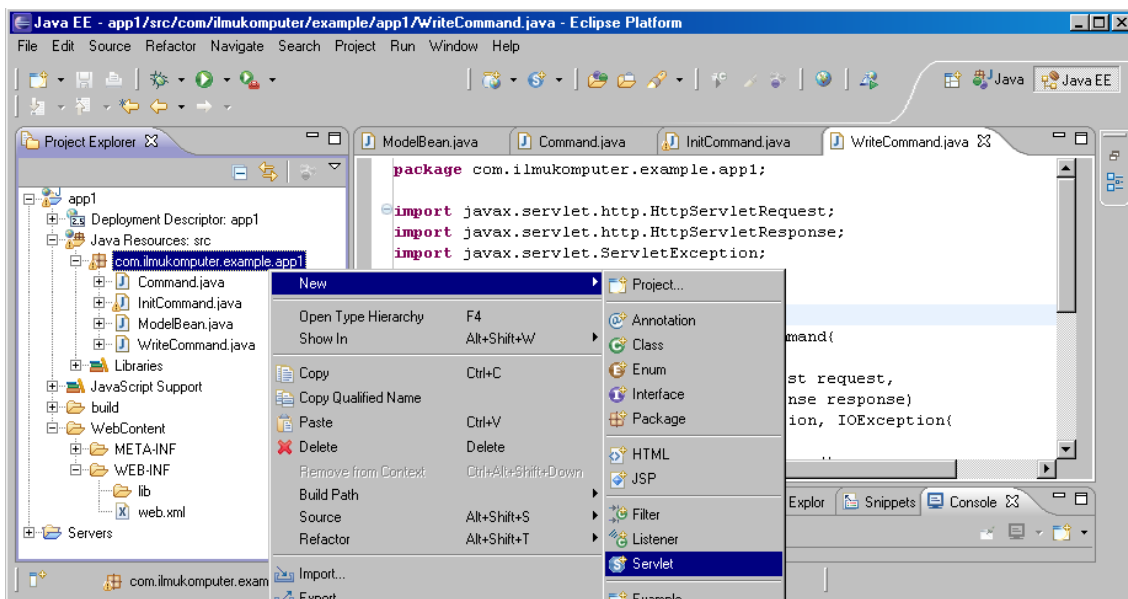
public class InitCommand implements Command{

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

```
request.setAttribute("modelBean", new ModelBean());  
  
ServletContext context = request.getSession().getServletContext();  
context.getRequestDispatcher("/view.jsp").forward(request, response);  
}  
}
```



Sekarang membuat servlet. Pada project explorer, klik kanan nama paket > New > Servlet



Web project: app1

Source folder: /app1/src

Java package: com.ilmuKomputer.example.app1

Class name: ControllerServlet

Superclass: javax.servlet.http.HttpServlet

Use an existing Servlet class or JSP

Class name: ControllerServlet

< Back Next > Finish Cancel

Klik Next

Name: ControllerServlet

Description:

Initialization Parameters:

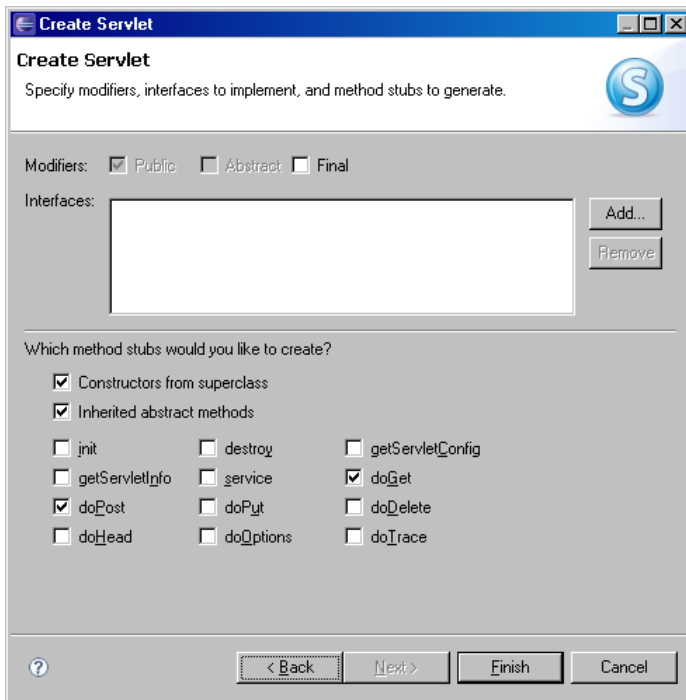
Name	Value	Description
------	-------	-------------

URL Mappings:

/ControllerServlet
--------------------

< Back Next > Finish Cancel

Nilai-nilai yang dimasukkan dalam field-field seperti diatas, akan secara otomatis ditulis pada file deployment (web.xml)



Klik finish. Lalu tuliskan kode-kode berikut:

```
package com.ilmukomputer.example.app1;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.Map;
import java.util.HashMap;

public class ControllerServlet extends HttpServlet{

    private Map commands = new HashMap();

    public void init(){
        this.commands.put("init", new InitCommand());
        this.commands.put("write", new WriteCommand());
    }

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

        String formAction = request.getParameter("form_action");

        if(null == formAction){
            formAction = "init";
        }

        Command command = (Command) commands.get(formAction);

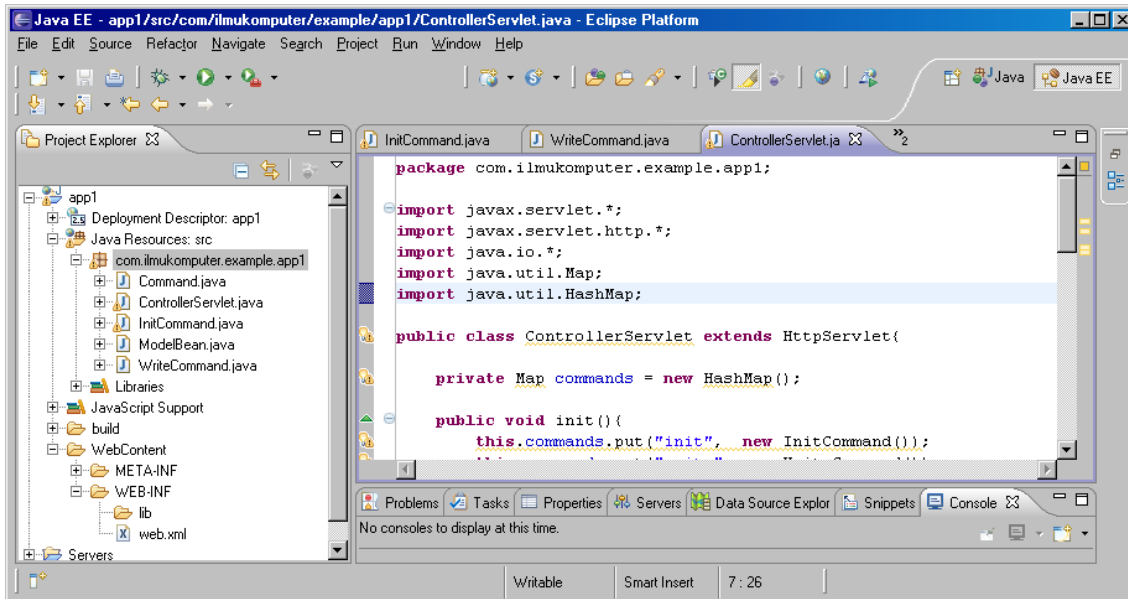
        if(null == command){
            throw new IllegalArgumentException(
                "No command for form action: " + formAction);
        }
        command.execute(request, response);
    }

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

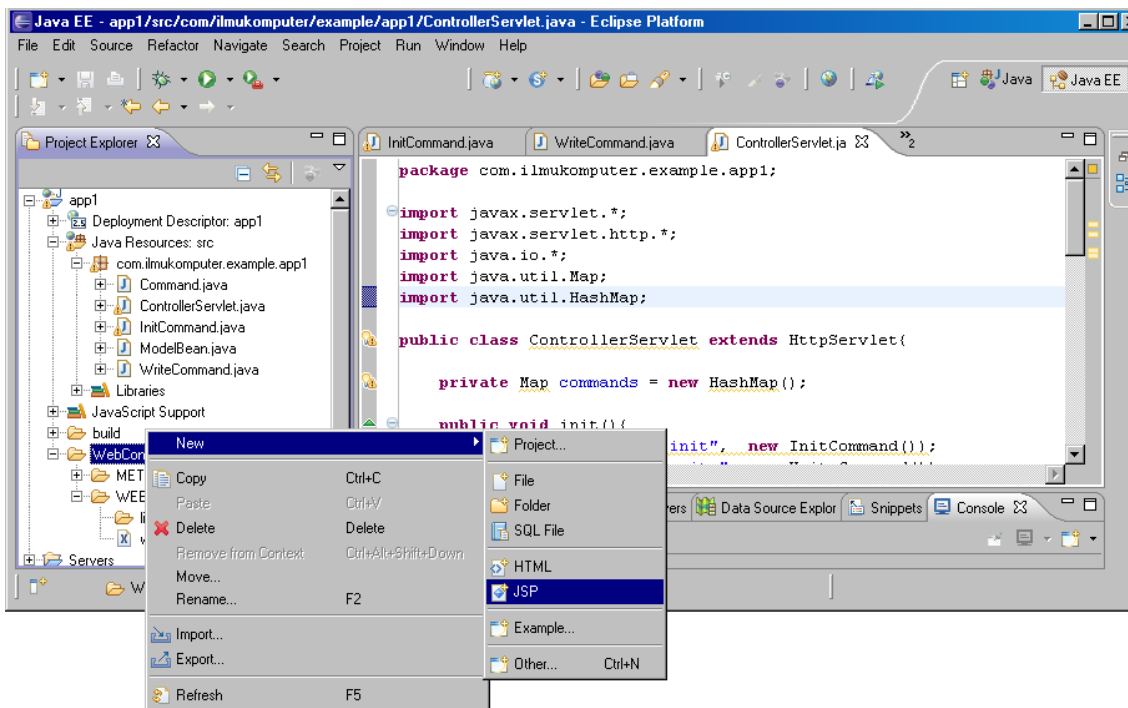
        processCommand(request, response);
    }

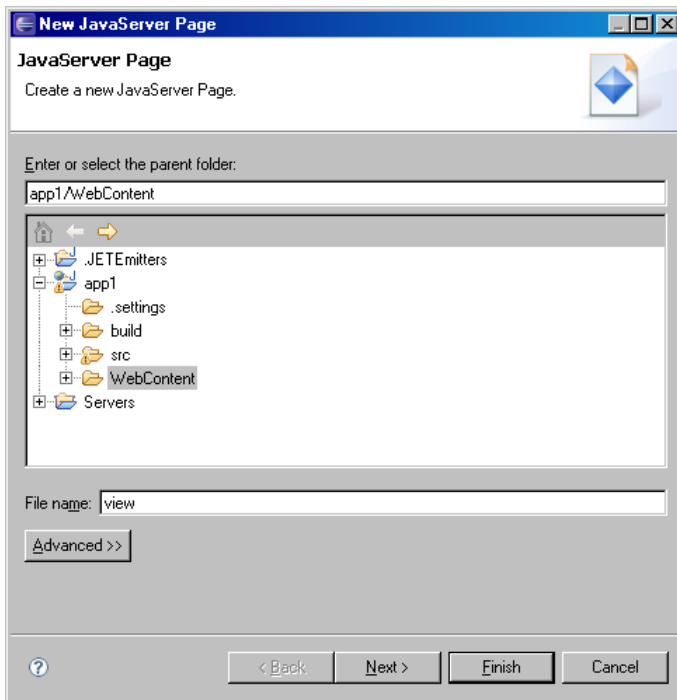
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException{
```

```
        processCommand(request, response);  
    }  
}
```

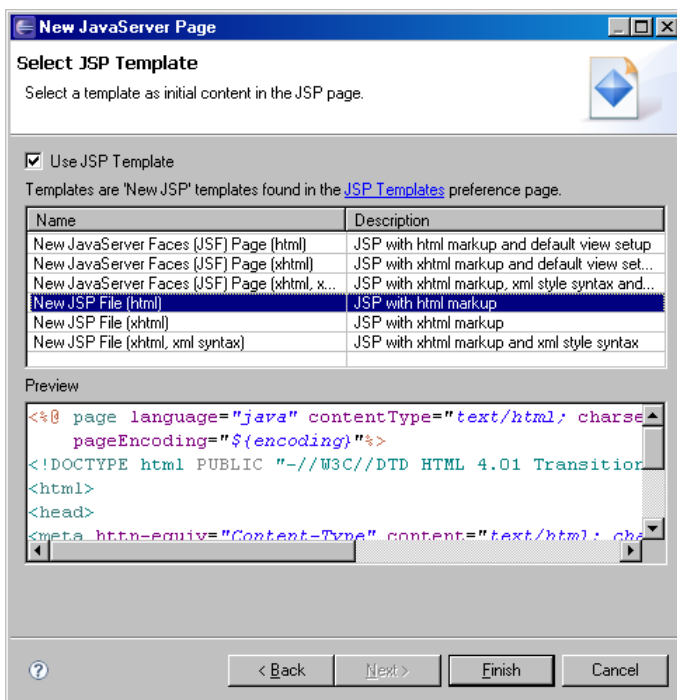


### 3.2.3 Membuat JSP





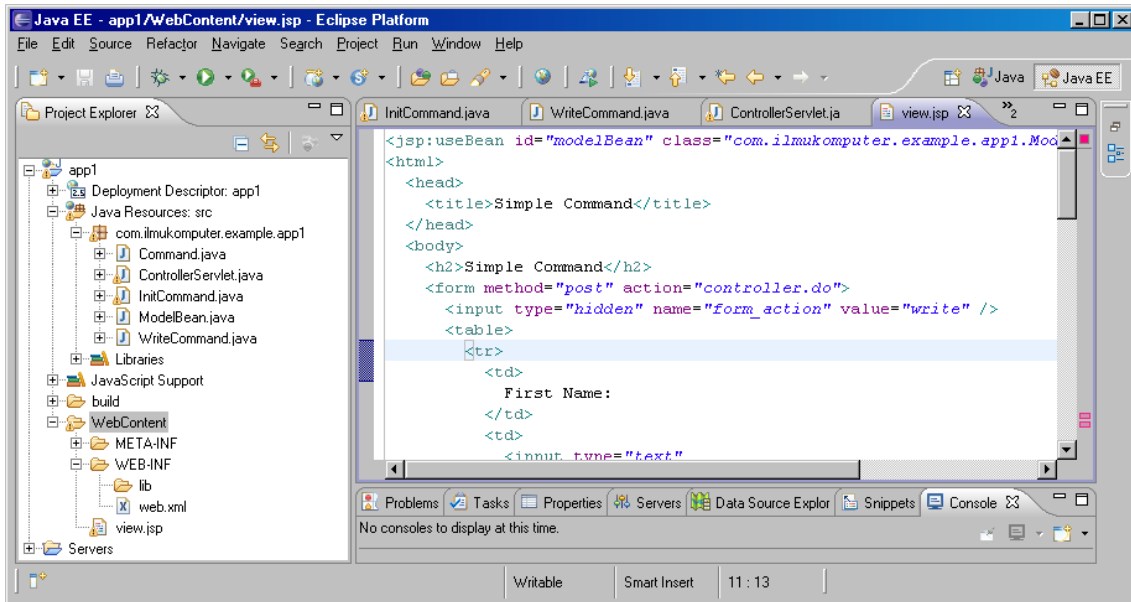
Klik next



Klik finish dan tuliskan kode view.jsp berikut:

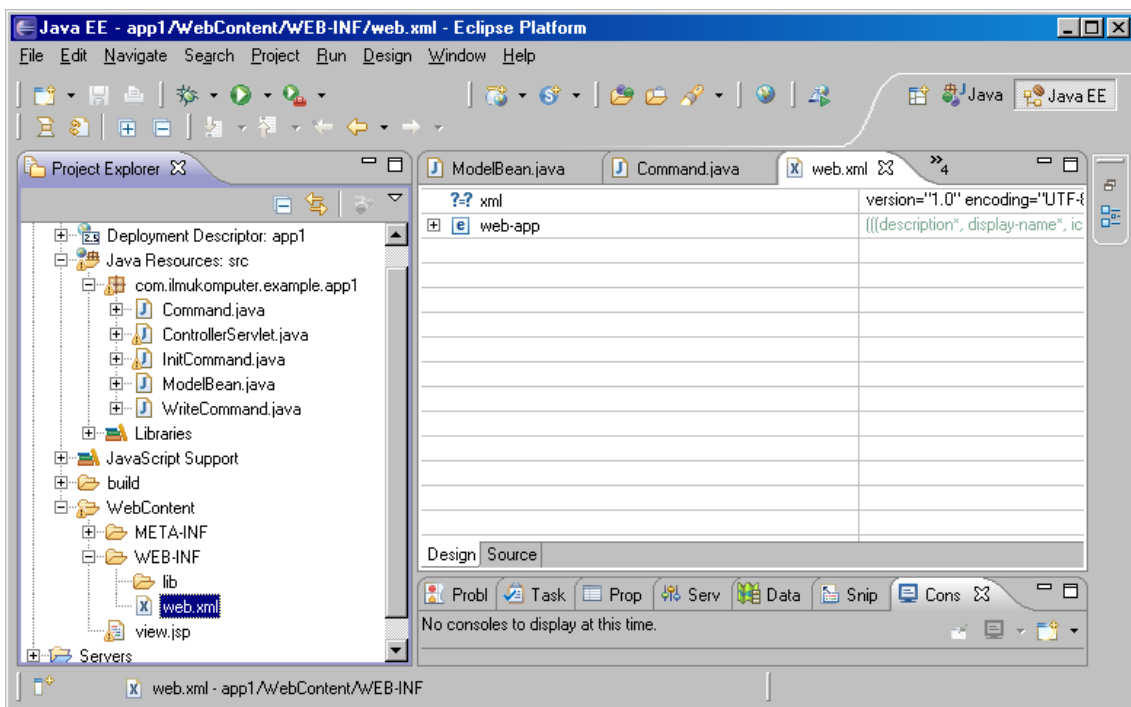
```
<jsp:useBean id="modelBean" class="com.ilmukomputer.example.app1.ModelBean"
scope="request" />
<html>
  <head>
    <title>Simple Command</title>
  </head>
  <body>
    <h2>Simple Command</h2>
```

```
<form method="post" action="controller.do">
  <input type="hidden" name="form_action" value="write" />
  <table>
    <tr>
      <td>
        First Name:
      </td>
      <td>
        <input type="text"
          name="first_name"
          value="<jsp:getProperty name="modelName" property="firstName" />" />
      </td>
    </tr>
    <tr>
      <td>
        Last Name:
      </td>
      <td>
        <input type="text"
          name="last_name"
          value="<jsp:getProperty name="modelName" property="lastName" />" />
      </td>
    </tr>
    <tr>
      <td>
        Email:
      </td>
      <td>
        <input type="text"
          name="email"
          value="<jsp:getProperty name="modelName" property="email" />" />
      </td>
    </tr>
    <tr>
      <td>
        Phone:
      </td>
      <td>
        <input type="text"
          name="phone"
          value="<jsp:getProperty name="modelName" property="phone" />" />
      </td>
    </tr>
    <tr>
      <td>
        <input type="submit" name="enter_button" value="Enter" />
      </td>
      <td>
      </td>
    </tr>
  </table>
</form>
<pre>
  <jsp:getProperty name="modelName" property="message" />
</pre>
</body>
</html>
```

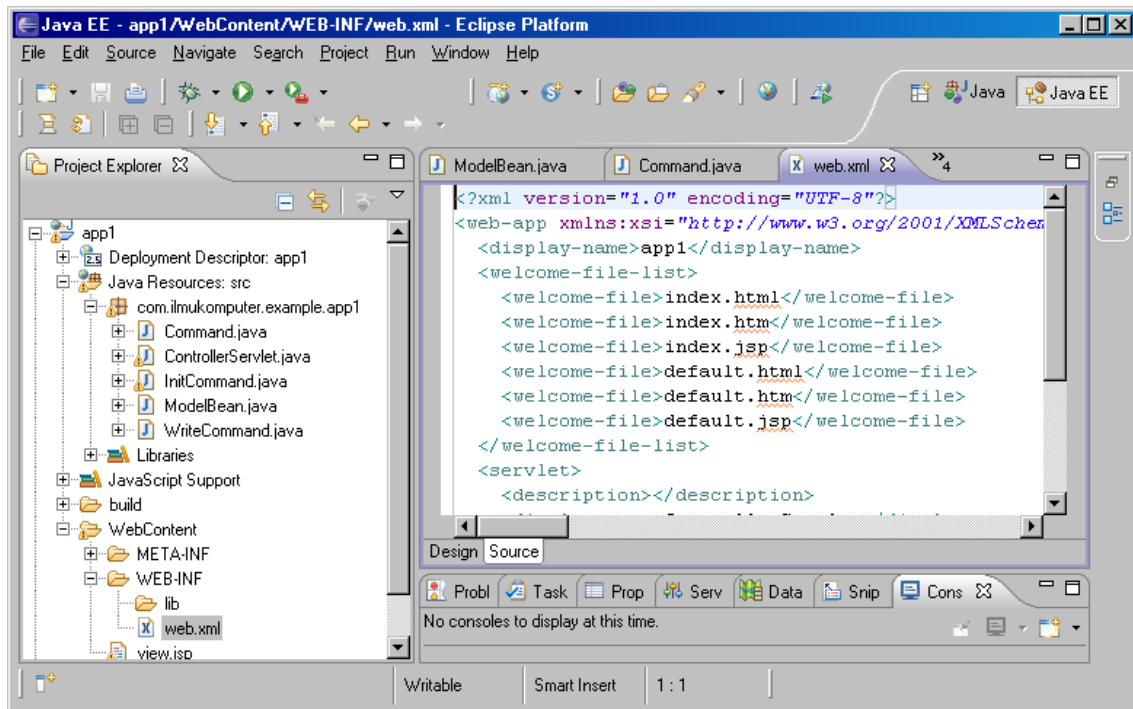


### 3.3. Deployment

Langkah selanjutnya adalah Deployment. Caranya: edit web.xml untuk menyesuaikan dengan kode-kode yang telah dibuat sebelumnya.



Pada jendela utama, klik tab Source



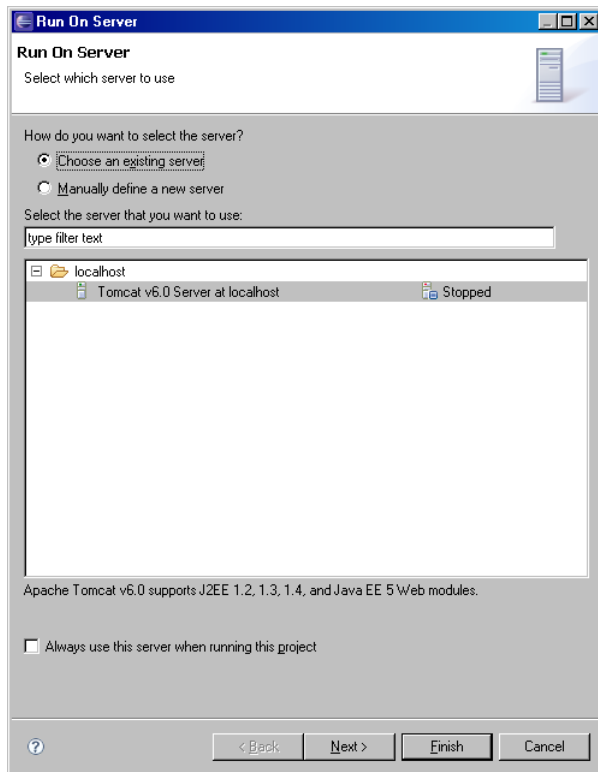
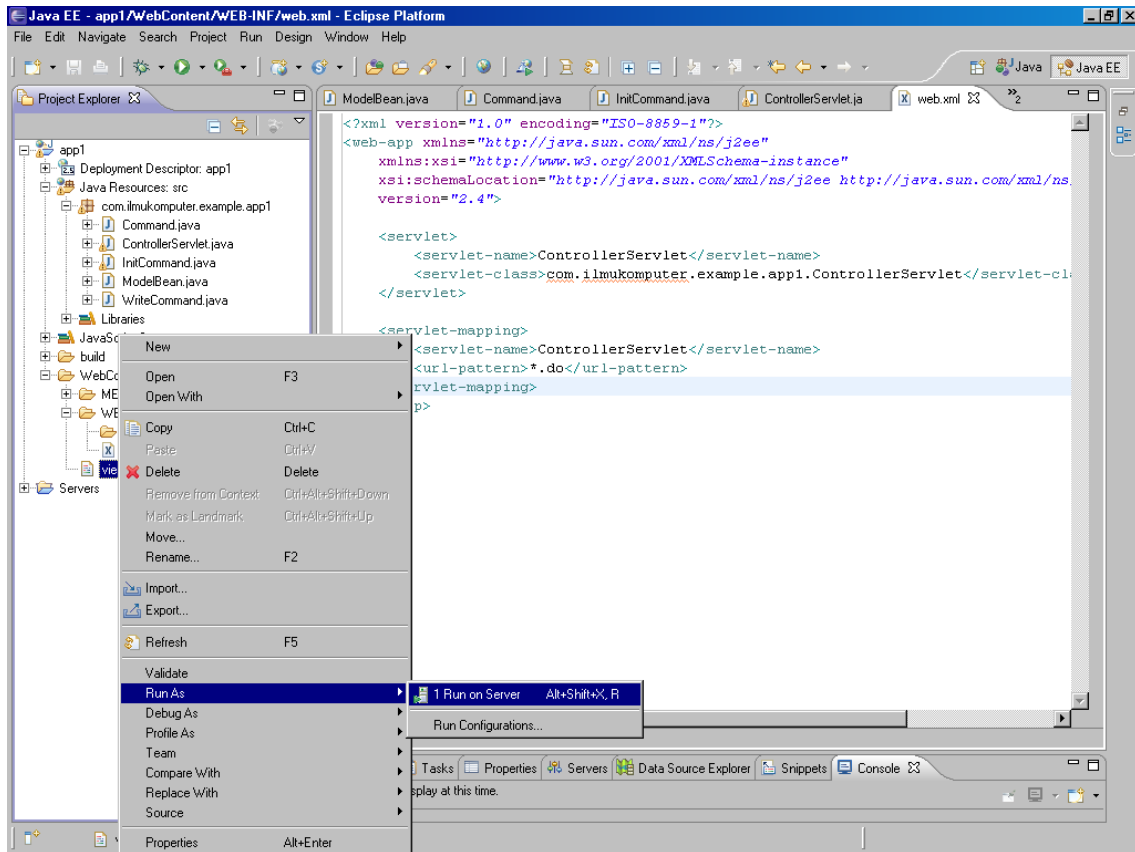
Ganti web.xml yang ada dengan web.xml berikut:

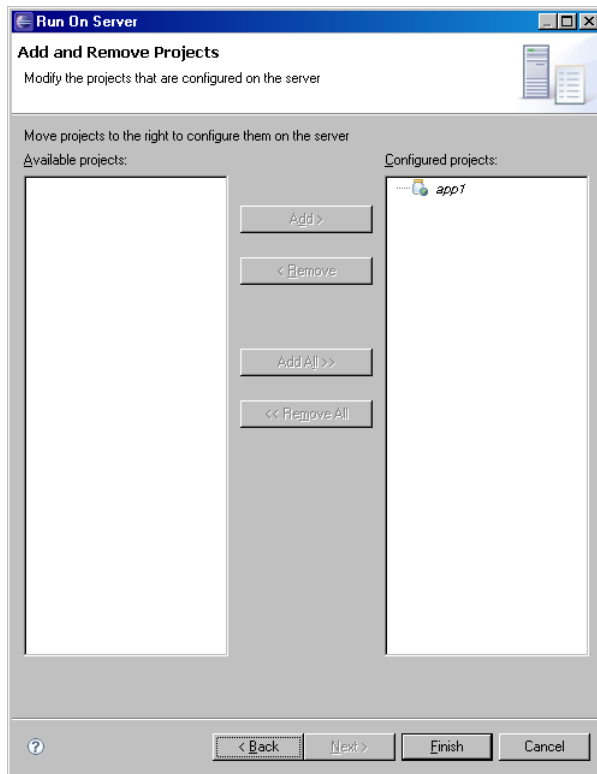
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/
web-app_2_4.xsd"
  version="2.4">

  <servlet>
    <servlet-name>ControllerServlet</servlet-name>
    <servlet-class>com.ilmukomputer.example.app1.ControllerServlet</servlet-class>
  </servlet>

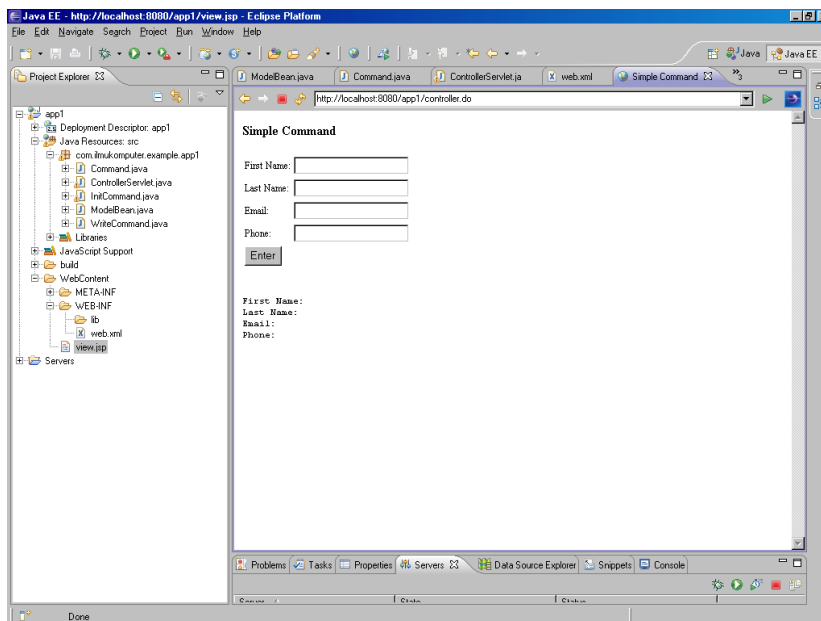
  <servlet-mapping>
    <servlet-name>ControllerServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
```

Setelah itu, aplikasi yang telah dibuat siap untuk dijalankan.





Klik finish. Kalau browser defaultnya menggunakan internal web browser, maka akan tampilan akhirnya adalah seperti ini.



#### 4. Penutup

Jika dibandingkan dengan membuat aplikasi web dengan cara manual, WTP tentu sangat memudahkan pengembang. Baik itu untuk editing maupun deploying. Serta sangat menghemat waktu. Tidak mengherankan, eclipse yang sudah terintegrasi dengan WTP sangat populer dikalangan pengembang Java.

## 5. Referensi

- [1] <http://www.roseindia.net/servlets/introductiontoconfigurationservlet.shtml>
- [2] [http://en.wikipedia.org/wiki/Apache\\_Tomcat](http://en.wikipedia.org/wiki/Apache_Tomcat)
- [3] <http://www.eclipse.org/webtools/>
- [4] <http://faq.javaranch.com/java/BioBenSouther>
- [5] <http://faq.javaranch.com/java/CodeBarnSimpleCommand>
- [6] <http://martinfowler.com/eaCatalog/frontController.html>
- [7] Kathy Sierra & Beart Bates, SCJP Sun® Certified Programmer for Java™ 6 Study Guide Exam (310-065), Mc Graw Hill, 2008, hal. 8

## Biografi Penulis



Yanu Widodo, lahir di Tulungagung pada tahun 1981. Menekuni ilmu permesinan saat masih sekolah di STM Negeri Tulungagung. Setelah lulus pada tahun 1999, lalu melanjutkan ke Teknik Perancangan dan Konstruksi di Politeknik Perkapalan Negeri Surabaya, Institut Teknologi Sepuluh Nopember (PPNS-ITS). Pemasaran dengan dunia elektronika dan komputer, lalu pada tahun 2001 hingga 2005 belajar Teknik Komputer Kontrol di Jurusan Teknik Elektro ITS. Selepas bekerja di Optical Disc Drive Department, PT. Panasonic Shikoku Electronics Batam (PSECB), pada pertengahan tahun 2006 lalu mendalami pengetahuan bidang Teknologi Informasi di Politeknik Elektronika Negeri Surabaya (PENS-ITS). Setelah mendapatkan gelar Sarjana Sains Terapan (S.ST) pada bulan september 2008, lalu bekerja di BaliCamp.