

Pemrograman Jaringan

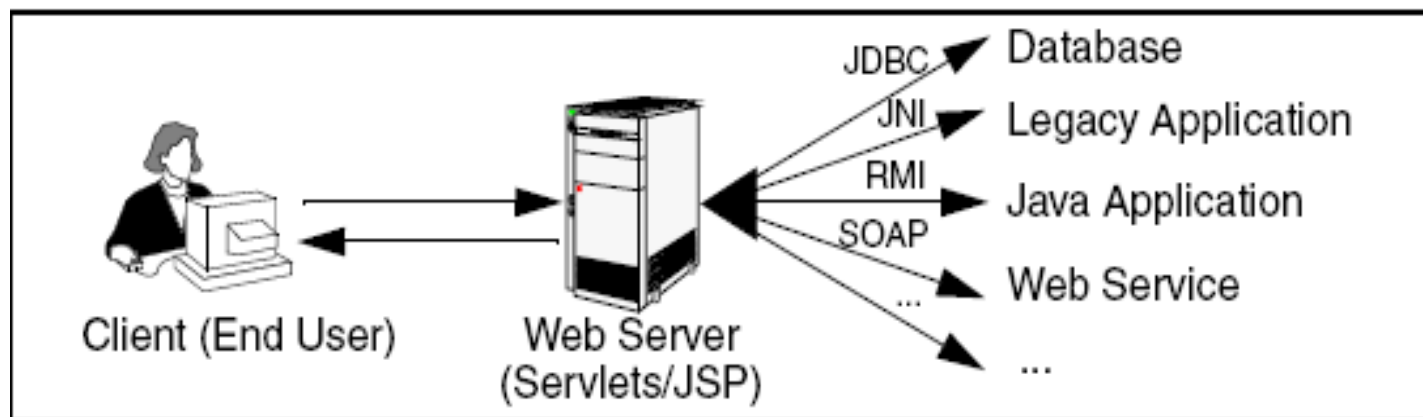
Servlet

Static VS Dynamic Web

- HTML -> statis
- Servlet/JSP -> dinamis
- Kenapa dinamis?
 - The web page is based on data sent by the **client**.
 - The web page is derived from data that changes **frequently**.
 - The web page uses information from corporate **databases** or other server-side sources.
- Servlet -> kode java yang dicompile & dieksekusi di server pada sebuah web server
- Applet -> kode java yang dieksekusi di client
- Servlet termasuk dalam **J2EE**

Servlet

- Servlets are Java programs that run on **Web** or **application servers**, acting as a **middle** layer between requests coming from Web browsers or other HTTP clients and databases or applications on the HTTP server.



Servlet

- **It is regular Java code.** There are new APIs, but no new syntax.
- **It has unfamiliar import statements.**
 - Because servlet is not part of J2SE, but J2EE
- **It extends a standard class (HttpServlet).**
 - Servlets provide a rich infrastructure for dealing with HTTP.
- **It overrides the doGet/doPost method.**
 - Servlets have different methods to respond to different types of HTTP commands.

Kemampuan Servlet

- **Read the explicit** data sent by the client.
 - Misal: lewat form
- **Read the implicit** HTTP request data sent by the browser.
 - The HTTP information includes cookies, information about media types and compression schemes the browser understands
- **Generate the results.**
 - may require talking to a database, executing an RMI or EJB call, invoking a Web service, or computing the response directly.

Kemampuan Servlet

- **Send the explicit** data (i.e., the document) to the client.
 - This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), or even a compressed format
 - HTML is by far the most common format, so an important servlet task is to wrap the results inside of HTML.
- **Send the implicit** HTTP response data.
 - Sending HTTP response data involves telling the browser or other client what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet vs traditional CGI

- Servlet is more **efficient**, **easier** to use, more **powerful**, more **portable**, **safer**, and **cheaper**
- Efficient
 - With traditional CGI, a new process is started for each HTTP request
 - With servlets, the Java virtual machine **stays running** and **handles** each request with a **lightweight** Java **thread**, not a **heavyweight** operating system **process**
- Convenient
 - Servlets have an **extensive infrastructure** for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions
 - With CGI, you must do it yourself
- Multi Vendor support

Servlet vs traditional CGI

- Powerful
 - Servlets can **talk directly** to the Web server, whereas regular CGI programs cannot, at least not without using a server-specific API.
- Portable
 - Because Servlets are written in the **Java** programming language and follow a standard API.
- Inexpensive
- Secure
 - vulnerabilities in traditional CGI stems from the fact that the programs are often executed by **general-purpose operating system shells**.
 - some CGI programs are processed by languages that do **not automatically check array or string bounds**.

Advantages of Servlets

- Efficiency
 - More efficient – uses **lightweight** java threads as opposed to individual processes
- Persistency
 - Servlets **remain** in memory
 - Servlets can **maintain state** between requests
- Portability
 - Since servlets are written in **Java**, they are platform independent
- Robustness
 - **Error handling, Garbage collector** to prevent problems with memory leaks
 - **Large class library** – network, file, database, distributed object components, security, etc.

Advantages of Servlets

- Extensibility
 - Creating new subclasses that suite your needs
 - Inheritance, polymorphism, etc. **All OOP.**
- Security
 - Security provided by the server as well as the **Java Security Manager**
 - **Eliminates** problems associated with executing cgi scripts using operating system “shells”
- Powerful
 - Servlets can **directly talk** to web server
 - Facilitates **database connection** pooling, session tracking etc.
- Convenient
 - **Parsing and decoding** HTML form data, reading and setting HTTP headers, handling cookies, etc.

Java Servlet Architecture

- Two packages make up the servlet architecture
 - *javax.servlet*
 - Contains generic interfaces and classes that are implemented and extended by all servlets
 - *javax.servlet.http*
 - Contains classes that are extended when creating HTTP-specific servlets
- The heart of servlet architecture is the interface class *javax.servlet.Servlet*
- It provides the framework for **all servlets**
- Defines five basic methods – init, service, destroy, getServletConfig and getServletInfo

Life Cycle of a Servlet

- Applet life cycle methods: *init()*, *start()*, *paint()*, *stop()*, and *destroy()* – appropriate methods called based on user action
- Similarly, servlets operate in the context of a **request** and **response** model managed by a **servlet engine**
- The engine does the following
 - **Loads** the servlet when it is first requested
 - Calls the servlet's *init()* method
 - Handles any number of requests by calling the servlet's *service()* method
 - When shutting down, calls each servlet's *destroy()* method

Life Cycle – *init()* method

- **Request** for a servlet received by the servlet engine
- **Checks** to see if the servlet is already loaded
 - If not, uses a class loader to get the required servlet class and instantiates it by calling the constructor method
- After the servlet is loaded, but before it services any requests, the *init()* method is called
- Inside *init()*, the resources used by the servlet are initialized. E.g: establishing database connection
- This method is called only once just before the servlet is placed into service
- The *init()* method takes a ***ServletConfig*** object as a parameter
- Most common way of doing this is to have it call the *super.init()* passing it the *ServletConfig* object

Life Cycle – *service()* method

- The *service()* method handles **all requests** sent by a client
- It cannot start servicing requests **until the *init()*** method has been executed
- The *service()* method is used only when extending ***GenericServlet*** class
- Since servlets are designed to operate in the HTTP environment, the ***HttpServlet*** class is extended
- The *service(HttpServletRequest, HttpServletResponse)* method examines the request and calls the appropriate ***doGet()*** or ***doPost()*** method.
 - A typical Http servlet includes overrides to one or more of these subsidiary methods rather than an override to ***service()***

Life Cycle – *destroy()* method

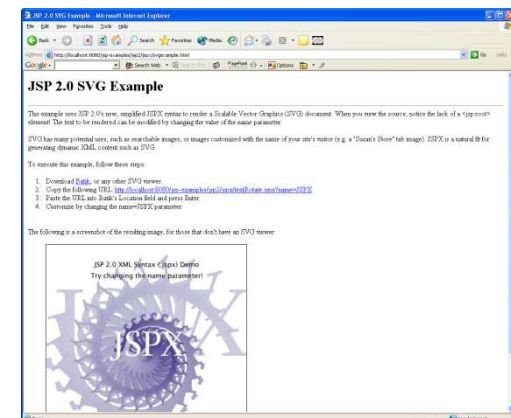
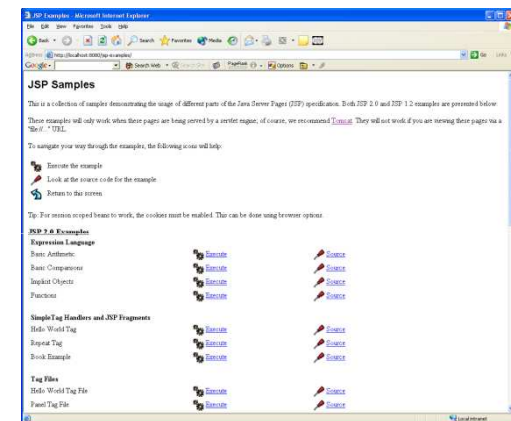
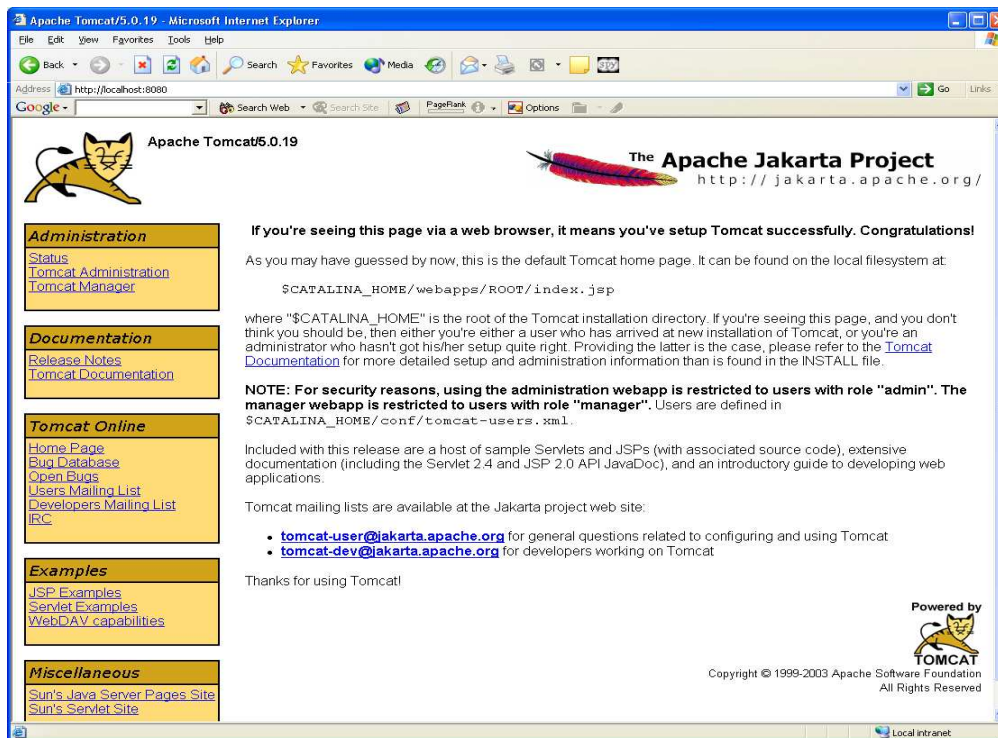
- This method signifies the **end** of a servlet's life
- The resources allocated during `init()` are **released**
- **Save** persistent information that will be used the next time the servlet is loaded
- The servlet engine **unloads** the servlet
- Calling **`destroy()`** yourself **will not** actually unload the servlet.
 - Only the servlet engine can do this

Free Servlet and JSP Engines

- Apache Tomcat
 - <http://tomcat.apache.org/>
- Allaire/Macromedia JRun
 - <http://www.macromedia.com/software/jrun/>
- GlassFish
 - <http://www.glassfish.org>
- Caucho's Resin
 - <http://www.caucho.com/>

Installing Tomcat

- After tomcat was successfully installed,
- Open your browser and point to <http://localhost:8080>.
 - You should see the Tomcat welcome page.
- The .sh files are for running Tomcat on Linux/Unix

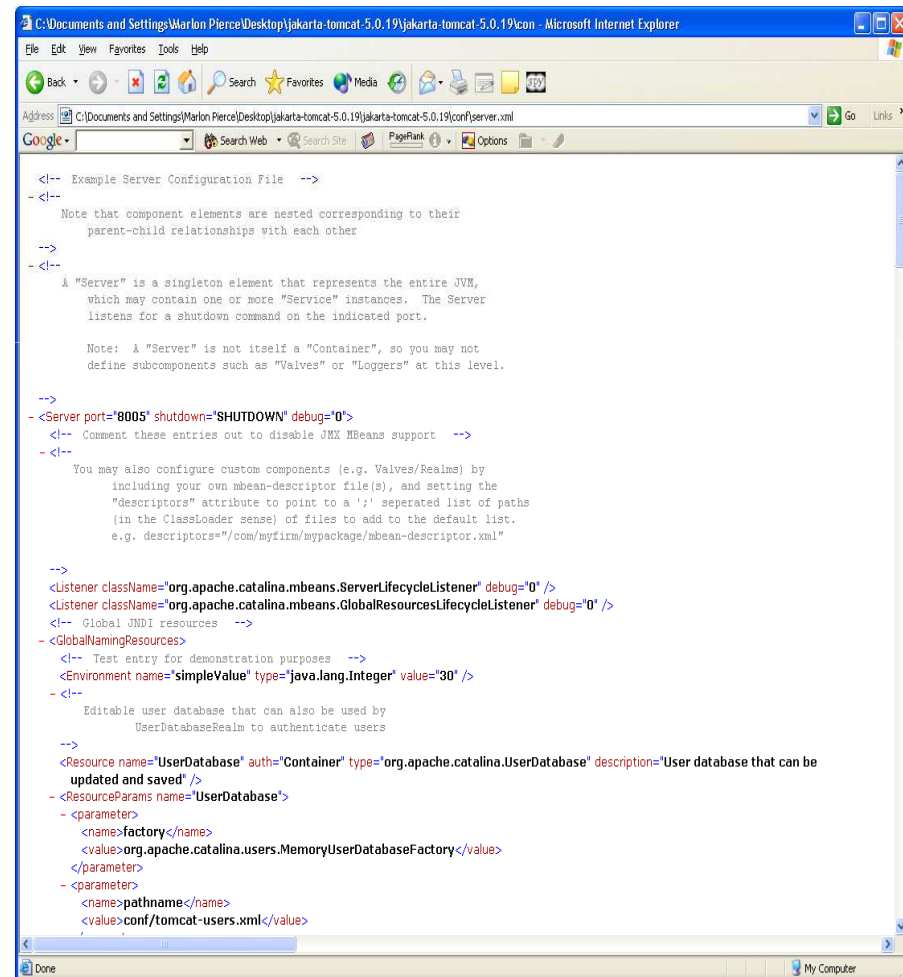


Tomcat

- Tomcat failures to start correctly if
 - you either do not have **the Java SDK** installed on, or
 - your **JAVA_HOME** environment variable is set incorrectly.
 - Bisa ditambahkan juga CATALINA_HOME
- File konfigurasi server: **server.xml**
 - Difold : conf\
- File konfigurasi Servlet mapping: **web.xml**
 - Di dalam folder WEB-INF
- File-file servlet diletakkan di webapps\
 - Bisa dibuat dalam folder tersendiri

Tomcat Ports

- Tomcat's default settings listen to three ports: 8080, 8005, 8009.
 - 8080 is the **http** port number.
 - 8005 is the **shutdown** port.
 - You can contact this to shutdown Tomcat from another process.
 - 8009 is the port for running Tomcat **behind** an Apache server.
 - Not needed here, but port opened
- Tomcat can use other ports
 - 8443 for **SSL connections**
 - Commented out by default.
 - Requires some additional configuration
 - 8082 is for **proxy** connections
 - Redirecting HTTP to other servers.
 - Commented out by default.
 - You don't have to edit these.



```
<!-- Example Server Configuration File -->
- <!--
  Note that component elements are nested corresponding to their
  parent-child relationships with each other
-->
- <!--
  A "Server" is a singleton element that represents the entire JVM,
  which may contain one or more "Service" instances. The Server
  listens for a shutdown command on the indicated port.

  Note: A "Server" is not itself a "Container", so you may not
  define subcomponents such as "Valves" or "Loggers" at this level.

-->
- <Server port="8005" shutdown="SHUTDOWN" debug="0">
  <!-- Comment these entries out to disable JMX MBeans support -->
  - <!--

  You may also configure custom components (e.g. Valves/Realms) by
  including your own mbean-descriptor file(s), and setting the
  "descriptors" attribute to point to a ';' separated list of paths
  (in the ClassLoader sense) of files to add to the default list.
  e.g. descriptors="/com/myfirm/mypackage/mbean-descriptor.xml"

  -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" debug="0" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" debug="0" />
  <!-- Global JNDI resources -->
  - <GlobalNamingResources>
    <!-- Test entry for demonstration purposes -->
    <Environment name="simpleValue" type="java.lang.Integer" value="30" />
  - <!--

  Editable user database that can also be used by
  UserDatabaseRealm to authenticate users

  -->
  <Resource name="UserDatabase" auth="Container" type="org.apache.catalina.UserDatabase" description="User database that can be
  updated and saved" />
  - <ResourceParams name="UserDatabase">
    - <parameter>
      <name>factory</name>
      <value>org.apache.catalina.users.MemoryUserDatabaseFactory</value>
    </parameter>
    - <parameter>
      <name>pathname</name>
      <value>conf/tomcat-users.xml</value>
    </parameter>
  </ResourceParams>
  </GlobalNamingResources>
</Server>
```

Compiling and Invoking Servlets

- Set your CLASSPATH
 - To Servlet JAR file (`\TOMCAT_HOME\lib\servlet-api.jar`).
- Put your servlet classes in proper location
 - Locations vary from server to server. E.g.,
 - `tomcat_install_dir\webapps\<folder>\WEB-INF\classes`
 - Invoke your servlets
 - **`http://localhost/<folder>/ServletName`**
 - Custom URL-to-servlet mapping (via **`web.xml`**)

Jika menggunakan editor Jcreator / Editplus

- Pastikan Anda menginclude-kan berkas:
- C:\Program Files\Apache Software Foundation\Tomcat 6.0\lib
- Yaitu:
 - servlet-api.jar
 - jsp-api.jar (untuk tag-tag JSP)
- Pada profile JDK anda sehingga bisa compile

web.xml

```
<servlet>
  <servlet-name>HelloWorldExample</servlet-name>
  <servlet-class>HelloWorldExample</servlet-class>
</servlet>
<servlet>
  <servlet-name>HelloServlet</servlet-name>
  <servlet-class>HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HelloWorldExample</servlet-name>
  <url-pattern>/HelloWorldExample</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>HelloServlet</servlet-name>
  <url-pattern>/HelloServlet</url-pattern>
</servlet-mapping>
```

<http://localhost:8080/<folder>/HelloServlet>

<http://localhost:8080/<folder>/HelloWorldExample>

Anda juga bisa menambah **<param-name>** dan **<param-value>** di dlm **<servlet>**

Context.xml

- Buat folder **META-INF**
- Buat file **context.xml** didalamnya
- `<context reloadable="true">`
 - Agar servlet selalu direload tanpa harus restart ulang Tomcat

Lib

- Di dalam folder **WEB-INF** bisa dibuat folder **Lib** selain classes
- Isi folder lib adalah file-file **JAR**
 - Misalnya JDBC
 - Anda juga bisa meletakkan JDBC di lib
TOMCAT_HOME

Simple Servlet Template

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletTemplate extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        // Use "request" to read incoming HTTP headers
        // (e.g. cookies) and HTML form data (query data)

        // Use "response" to specify the HTTP response status
        // code and headers (e.g. the content type, cookies).

        PrintWriter out = response.getWriter();
        // Use "out" to send content to browser
    }
}
```

Important Steps

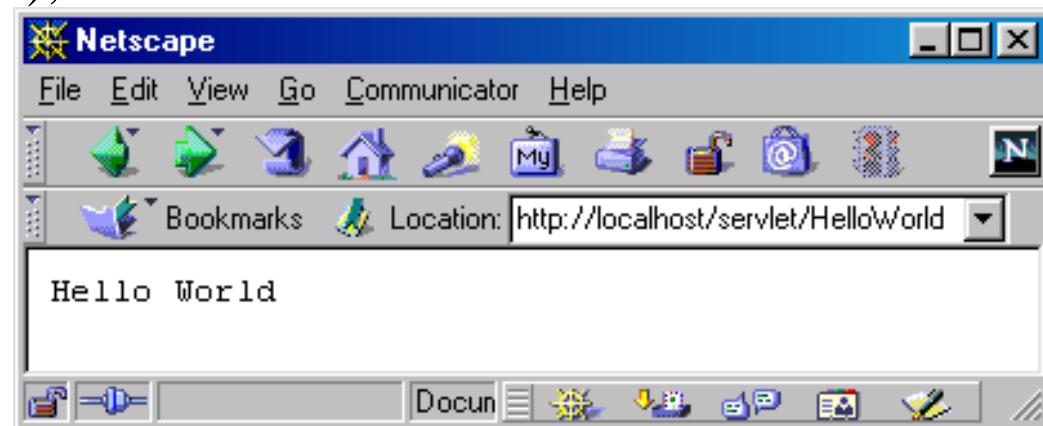
- Import the Servlet API:

```
import javax.servlet.*;  
import javax.servlet.http.*;
```
- Extend the `HTTPServlet` class
- You need to override at least one of the **request handlers!**
 - `doGet` / `doPost`
- Get an output stream to send the response back to the client
 - All output is channeled to the browser.

A Simple Servlet That Generates Plain Text

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```



Generating HTML

- Set the **Content-Type** header
 - Use `response.setContentType("text/html")`
- Output HTML
- Use an HTML validation service
 - <http://validator.w3.org/>
 - <http://www.htmlhelp.com/tools/validator/>

Servlet generating HTML

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");

        java.util.Date tgl = new java.util.Date();

        out.println("<head>");
        out.println("<title>Hallo</title>");
        out.println("</head>");
        out.println("<body>"
            + "Hallo, saya Antonius RC<hr>"
            + "Tanggal sekarang : " + tgl
            );
        out.println("</body>");
        out.println("</html>");
    }
}
```

Hallo, saya Antonius RC

Tanggal sekarang : Fri Aug 20 21:19:14 ICT 2010

Penggunaan CSS

- CSS diletakkan pada folder **utama** servlet
- Selevel dengan folder WEB-INF dan META-INF
- Misal diberi nama: css
 - Di dalamnya ada file my.css
- Cara pemanggilan:
- `out.println("<link rel='stylesheet' type='text/css' href='\" + request.getContextPath() + \"/css/my.css' />");`

Contoh CSS

Antonius

```
public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");

        String title = "Antonius";

        out.println("<title>" + title + "</title>");
        out.println("<link rel='stylesheet' type='text/css' href='" + request.getContextPath() + "/css/my.css' />");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>" + title + "</h1>");
        out.println("<p>This is a paragraph.</p>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

doGet and doPost

- The handler methods each take two parameters:
 - `HttpServletRequest`: encapsulates all information regarding the browser request.
 - Form data, client host name, HTTP request headers.
 - `HttpServletResponse`: encapsulate all information regarding the servlet response.
 - HTTP Return status, outgoing cookies, HTML response.
- If you want the same servlet to handle both GET and POST, you can have `doGet` call `doPost` or vice versa.

```
Public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException  
{doPost(req,res);}
```

BrowserInfo.java

```
public void doGet(HttpServletRequest request, HttpServletResponse r
    PrintWriter out = response.getWriter();
    response.setContentType("text/html");

    out.println("<html>");
    out.println("<head><title>Browser Information</title></head>");
    out.println("<body>");
    out.println("Your host name is " + request.getLocalName());
    String userAgent = request.getHeader("user-agent");
    String browser = "unknown";
    out.println("<br/>and your browser is ");
    if (userAgent != null) {
        if (userAgent.indexOf("MSIE") > -1) {
            browser = "MS Internet Explorer";
        }
        else if (userAgent.indexOf("Firefox") > -1) {
            browser = "Mozilla Firefox";
        }
    }
    out.println(browser);
    out.println("</body>");
    out.println("</html>");
}
```

Your host name is 0.0.0.0
and your browser is Mozilla Firefox

getParameter()

- Use *getParameter()* to retrieve parameters from a form by name.

Named Field values HTML FORM

```
<INPUT TYPE="TEXT" NAME="nilai1">
```

In a Servlet

```
String nilai1 = request.getParameter("nilai1");
```



Menjumlahkan 2 bilangan

Nilai 1: + Nilai 2:

```
public void doGet(HttpServletRequest request, HttpServletResponse
    PrintWriter out = response.getWriter();
    response.setContentType("text/html");

    int nilai1 = Integer.parseInt(request.getParameter("nilai1"));
    int nilai2 = Integer.parseInt(request.getParameter("nilai2"));
    int hasil = nilai1+nilai2;

    out.println("<html>");
    out.println("<head><title>Hasil Penjumlahan</title></head>");
    out.println("<body>");
    out.println("Hasil "
        + nilai1
        + " + " + nilai2 + " = "
        + hasil);
    out.println("</body>");
    out.println("</html>");
}
```

Hasil $4 + 5 = 9$

getParameter() *cont'd*

- getParameter() can return three things:
 - **String**: corresponds to the parameter.
 - **Empty String**: parameter exists, but no value provided.
 - **null**: Parameter does not exist.

getParameterValues()

- Used to retrieve **multiple** form parameters with the same name.
- For example, a series of **checkboxes** all have the same name, and you want to determine which ones have been selected.
- Returns an **array of Strings**.

```
request.getParameterValues("nilai1");  
for(i=0;i<hasilarray.length;i++){  
    out.println(hasilarray[i]);  
}
```

getParameterNames()

- Returns an **Enumeration** object.
- By cycling through the enumeration object, you can obtain the **names** of all parameters submitted to the servlet.
- Note that the Servlet API does not specify the **order** in which parameter names appear.


```
Enumeration hasIenum = request.getParameterNames();  
while(hasIenum.hasMoreElements()){  
    out.println(hasIenum.nextElement());  
}
```

Circle Servlet


```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class circle extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(" <BODY><H1 ALIGN=CENTER> Circle Info </H1>\n");
        try{
            String sdiam = request.getParameter("diameter");
            double diam = Double.parseDouble(sdiam);
            out.println("<BR><H3>Diam:</H3>" + diam +
                "<BR><H3>Area:</H3>" + diam/2.0 * diam/2.0 * 3.14159 +
                "<BR><H3>Perimeter:</H3>" + 2.0 * diam/2.0 * 3.14159);
        } catch ( NumberFormatException e ){
            out.println("Please enter a valid number");
        }
        out.println("</BODY></HTML>");
    }
}
```

Subclass
HttpServlet.



Specify HTML
output.



Attach a PrintWriter
to Response Object



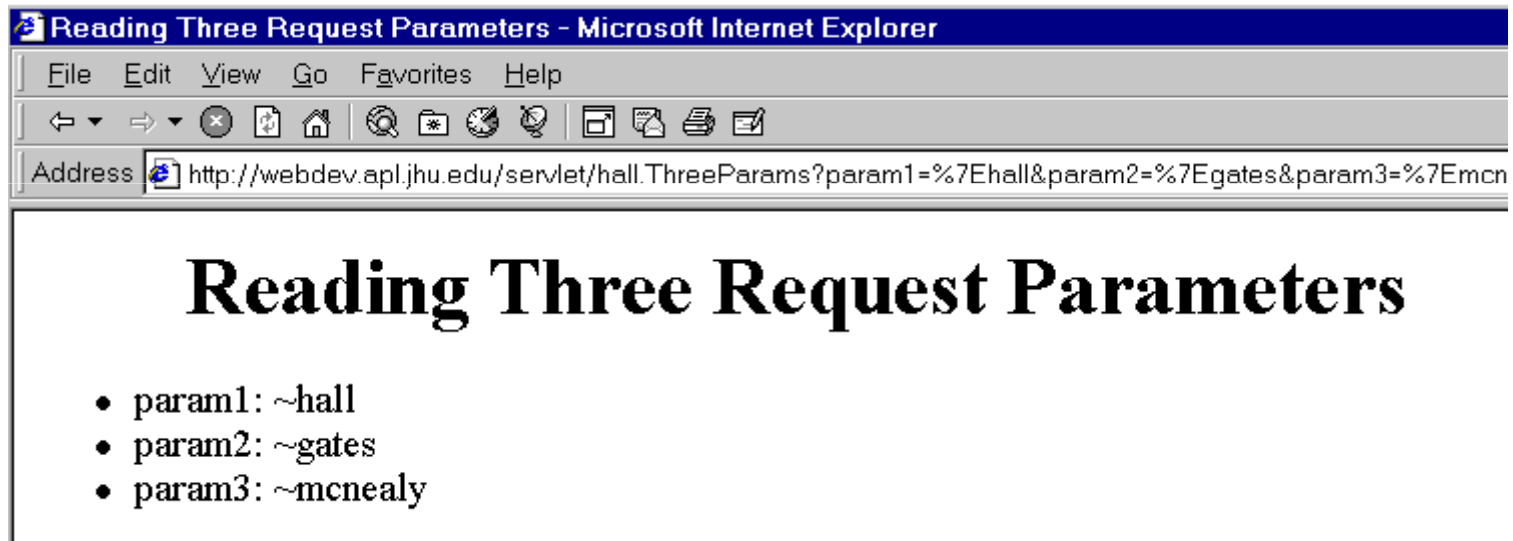
Handling parameters

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Reading Three Request Parameters";
        out.println(ServletUtilities.headWithTitle(title) +
                   "<BODY>\n" +
                   "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                   "<UL>\n" +
                   "  <LI>param1: "
                   + request.getParameter("param1") + "\n" +
                   "  <LI>param2: "
                   + request.getParameter("param2") + "\n" +
                   "  <LI>param3: "
                   + request.getParameter("param3") + "\n" +
                   "</UL>\n" +
                   "</BODY></HTML>");
    }

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

Three parameters



Instance Persistence Example

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SimpleCounter extends HttpServlet {
    int count = 0;

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();

        int local_count;
        synchronized(this) {
            local_count = ++count;
        }
        out.println("Since loading, this servlet has been accessed " +
            local_count + " times.");
    }
}
```

Retrieving Information

- Server Info
 - `getServerName()`
 - `getServerPort()`
 - `getServerInfo()`
 - `getAttribute(name)`
- Client Machine
 - `getRemoteAddr()`
 - `getRemoteHost()`
- Request
 - `getRemoteUser()`
 - `getAuthType()`

Retrieving Information (continued)

- Request
 - `getParameter(name)`
 - `getParameterNames()`
 - `getQueryString()`
 - `getRequestURL(request)`
 - `getRequestURI()`
 - `getServletPath()`
 - `getScheme()`
 - `getProtocol()`
 - `getMethod()`
 - `getHeader(name)`
 - `getDateHeader(name)`
 - `getIntHeader(name)`
 - `getHeaderNames()`

HTTP Headers

- A servlet sets HTTP headers to provide extra information about its response
- Use `setHeader(name, value)` to set headers
 - Cache-control – HTTP 1.1 cache control
 - Pragma – HTTP 1.0 cache control
 - Connection – for persistence
 - Retry-after – when server can again handle requests
 - Expires – when a document becomes invalid
 - Location – new location of a document
 - www-authenticate – authorization scheme
 - Content-encoding – scheme used to encode response

Cookies and Servlets

- The `HttpServletRequest` class includes the “`getCookies()`” function.
 - This returns an array of cookies, or null if there aren't any.
- Cookies can then be accessed using three methods.
 - `String getName()`
 - `String getValue()`
 - `String getVersion()`
 - `String getComment()`

Cookies & Servlets cont'd

- Cookies can be created using **HttpServletResponse.addCookie()** and the constructor **new Cookie(String name, String value);**
 - Expiration can be set using **setMaxAge(int seconds)**

Contoh Cookies

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FCookie extends HttpServlet {
    public void doGet ( HttpServletRequest request, HttpServletResponse response )
        throws ServletException, IOException {
        PrintWriter out;

        // Set content type and other response header fields first
        response.setContentType("text/html");

        // Write the data of the response
        out = response.getWriter();

        // Create a cookie
        Cookie c1 = new Cookie("CName1", "Cookie Value1");
        Cookie c2 = new Cookie("CName2", "Cookie Value2");
        response.addCookie(c1);
        response.addCookie(c2);
        out.println("<HTML><HEAD><TITLE>");
        out.println(" This output is generated from a Servlet");
        out.println("</TITLE></HEAD><BODY>");
        out.println(" This has set 2 Cookies");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

Sessions & Servlets

- Servlets also support simple transparent sessions
 - Implements Interface HttpSession
 - Get one by using HttpServletRequest.getSession()
- You can store & retrieve values in the session
 - putValue(String name, String value)
 - String getValue(String name)
 - String[] getNames()

Sessions & Servlets cont'd

- Various other information is stored
 - long `getCreationTime()`
 - String `getId()`
 - long `getLastAccessedTime()`
- Also can set timeout before session destruction
 - int `getMaxInactiveInterval()`
 - `setMaxInactiveInterval(int seconds)`

Praktikum

- Cookies untuk Login
- Session untuk Login
- Penggunaan REGEX
- Koneksi Database (JDBC)

Cookies

Username
Password

Keterangan : Cookies akan disimpan selama 1 menit (60 detik).

Selamat Datang User 'anton'

IP Komputer Anda : '0:0:0:0:0:0:1'

Anda pernah login sebanyak : 3 kali

Last Login Anda : **Fri Aug 20 22:49:59 ICT 2010**

```
<servlet>
  <servlet-name>login</servlet-name>
  <servlet-class>login</servlet-class>
  <init-param>
    <param-name>inisial</param-name>
    <param-value>anton</param-value>
  </init-param>
</servlet>
```

```
<servlet-mapping>
  <servlet-name>login</servlet-name>
  <url-pattern>/login</url-pattern>
</servlet-mapping>
```

```

public class login extends HttpServlet {

    String def_pass;    //untuk mengambil default password di web.xml

    public void init() throws ServletException{ //ambil parameter default password di web.xml
        def_pass = getInitParameter("inisial");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
    {
        response.setContentType("text/html; charset=iso-8859-1");
        PrintWriter out = response.getWriter();

        out.println("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\r\n<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 :
        out.println("\r\n<html xmlns=\"http://www.w3.org/1999/xhtml\">\r\n<head>\r\n<title>Tugas II Web Database :: Ha

        String username = request.getParameter("username");
        String password = request.getParameter("password");

        String user_rev = "";    //untuk tampungan password
        String last_login = "";    //untuk mencatat last_login pada cookie
        int jml_login = 0;    //untuk mencatat jml_login pada cookie
        Date tgl_skrng = new Date(); //untuk mencatat tgl hari ini
    }
}

```

```

if(username != null && password != null && username.length()>0 && password.length()>0){
|   for(int i=username.length()-1;i>=0;i--){    //untuk mencocokkan password
        user_rev = user_rev + username.charAt(i);
    }

    if((password.equals(def_pass)) || (password.equals(user_rev))){

        out.println("<h1>Selamat Datang User '<b><font color='blue'>" + username + "</font></b>'</h1><hr>" );
        out.println("IP Komputer Anda  : '<b><font color='red'>" + request.getRemoteAddr() + "</font></b>'<br>");

        Cookie first = new Cookie("tanda","0"); //isi cookie awal dulu
        first.setMaxAge(60);
        response.addCookie(first);

        Cookie[] cek_cookie = request.getCookies(); //ambil isi cookie

        if(cek_cookie == null || cek_cookie.length<=1){
            out.println("<font color ='FF00FF'><i>Anda Baru Login Pertama kalinya</i></font><hr>");

            Cookie j_1 = new Cookie("jml_login","1");    //simpan cookie
            Cookie l_1 = new Cookie("last_login",tgl_skrng.toString());
            j_1.setMaxAge(60);
            l_1.setMaxAge(60);
            response.addCookie(j_1);
            response.addCookie(l_1);
        }
    }
}

```

```

    }
    else
    {
        for(int i=0;i<cek_cookie.length;i++){ //cek isi cookie
            if(cek_cookie[i].getName() != null && cek_cookie[i].getName().equals("jml_login")){
                jml_login = Integer.parseInt(cek_cookie[i].getValue()); //ambil jml_login dari cookie
            }

            if(cek_cookie[i].getName() != null && cek_cookie[i].getName().equals("last_login")){
                last_login = cek_cookie[i].getValue(); //ambil last_login dari cookie
            }
        }

        out.println("Anda pernah login sebanyak : <b><font color='brown'>" + jml_login + "</font></b> kali.");
        out.println("Last Login Anda : <b><font color='green'>" + last_login + "</font></b><hr>");

        jml_login++; //tambah jumlah login dengan 1
        Integer jlogin = new Integer(jml_login); //tampung ke integer agar bisa dijadikan String
        Cookie j_1 = new Cookie("jml_login", jlogin.toString()); //simpan cookie
        Cookie l_1 = new Cookie("last_login", tgl_skrng.toString());
        j_1.setMaxAge(60);
        l_1.setMaxAge(60);
        response.addCookie(j_1);
        response.addCookie(l_1);
    }
    } else out.println("<h2><font color='red'>Username atau Password Anda salah!</font></h2>");
} else out.println("<h1><font color='red'>Anda Belum Mengisi Username & Password, silahkan ulangi!</font></h1>");

    out.println("\r\n</body>\r\n</html>\r\n");
}

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

```

Username
Password

Keterangan : username & password harus sama dan session dibuat selama 30 detik..

Session

Selamat Datang User 'anton'

Masukkan teks : (Setiap alamat email harus diakhiri enter!)

```
a@a.com  
b@b.com  
a@.com  
b@  
anton
```

REGULAR EXPRESSION :

Teks yang diinputkan : 'a@.com b@b.com anton'

Tabel semua substring :

Substring	Alamat Email ?
a@.com	Tidak OK
b@b.com	OK
anton	Tidak OK

[Isi Lagi ?](#)

File-file Pendukung

- File: session.java
- File: regeks.java
- File: regex.html

Koneksi DB

- `import java.sql.*;`
- `Class.forName("org.gjt.mm.mysql.Driver");`
- `Connection conn
=DriverManager.getConnection("jdbc:mysql://localhost:3306/
test?user=root&password=");`
- `try{
 //baris program
} catch (ClassNotFoundException e) {
} catch (SQLException esql){
}`

Yang harus dilakukan

- Kopi file JDBC dalam bentuk JAR ke dalam TOMCAT_HOME\lib anda!

```

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class table extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
    {
        response.setContentType("text/html; charset=iso-8859-1");
        PrintWriter out = response.getWriter();

        out.println("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\r\n<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\" \"http://\r\n<html xmlns=\"http://www.w3.org/1999/xhtml\">\r\n<head>\r\n<title>Browse Data Mahasiswa</title>\r\n<meta http-equiv=\"Cor

String nim = request.getParameter("nim");

if (nim != null && nim.length()>0)
{
    try{

        //Untuk MySQL
        Class.forName("org.gjt.mm.mysql.Driver");
        Connection conn =DriverManager.getConnection("jdbc:mysql://localhost:3306/test?user=root&password=");
        //Untuk PostgreSQL
        //Class.forName("org.postgresql.Driver");
        //Connection conn =DriverManager.getConnection("jdbc:postgresql://192.168.3.8:5432/test?user=root&password=");

        Statement stmt=conn.createStatement();

        out.println("<div align='center'><h1><font color='red' face='Verdana, Arial, Helvetica, sans-serif'>HASIL QUERY DARI DATABASE</for
        out.println("NIM : <font color='blue'>" + nim + "</font><br>");

        ResultSet myResultSet = stmt.executeQuery("select * from mahasiswa where nim="+ nim);
    }
}
}

```

```

if(myResultSet!=null)
{
    while(myResultSet.next())
    {
        String nama = myResultSet.getString("nama");
        String ipk = myResultSet.getString("ipk");
        String jk = myResultSet.getString("jk");
        out.println("<font size='2' face='Arial, Helvetica, sans-serif'>" + nama + "</font><br>");
        out.println("<font size='2' face='Arial, Helvetica, sans-serif'>" + ipk + "</font><br>");
        out.println("<font size='2' face='Arial, Helvetica, sans-serif'>" + jk + "</font><br>");
    }
}

myResultSet.last();
out.println("<B>" + myResultSet.getRow() + "</B> baris terpilih.");
stmt.close();
conn.close();

} catch (ClassNotFoundException e){
    out.println("<H2>Exception: <font color='red'>" + e.getMessage() + "</font></H2>");
} catch (SQLException esql) {
    out.println("<H2>Exception: <font color='red'>" + esql.getMessage() + "</font></H2>");
}
} else {
    try{

//Untuk MySQL
Class.forName("org.gjt.mm.mysql.Driver");
Connection conn =DriverManager.getConnection("jdbc:mysql://localhost:3306/test?user=root&password=");

Statement stmt=conn.createStatement();

out.println("<table border='1'>");
out.println("<tr><td>NIM</td><td>NAMA</td><td>IPK</td><td>JENIS KELAMIN</td></tr>");

ResultSet myResultSet = stmt.executeQuery("select * from mahasiswa order by nim asc");

```

```

if(myResultSet!=null)
{
    while(myResultSet.next())
    {
        nim = myResultSet.getString("nim");
        String nama = myResultSet.getString("nama");
        String ipk = myResultSet.getString("ipk");
        String jk = myResultSet.getString("jk");
        out.println("<tr><td><font size='2' face='Arial, Helvetica, sans-serif'>" + nim + "</font></td>");
        out.println("<td><font size='2' face='Arial, Helvetica, sans-serif'>" + nama + "</font></td>");
        out.println("<td><font size='2' face='Arial, Helvetica, sans-serif'>" + ipk + "</font></td>");
        out.println("<td><font size='2' face='Arial, Helvetica, sans-serif'>" + jk + "</font></td>");
    }
}

myResultSet.last();
out.println("<B>" + myResultSet.getRow() + "</B> baris terpilih.");
stmt.close();
conn.close();

} catch (ClassNotFoundException e){
    out.println("<H2>Exception: <font color='red'>" + e.getMessage() + "</font></H2>");
} catch (SQLException esql) {
    out.println("<H2>Exception: <font color='red'>" + esql.getMessage() + "</font></H2>");
}

}

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

```

Hasil

4 baris terpilih.

NIM	NAMA	IPK	JENIS KELAMIN
22001234	Lusia	3.52	P
22002515	Putri	3.28	P
22002521	Mahas	3.50	L
22002529	anton	3.68	L



Browser address bar: <http://localhost:8080/servletku/table?nim=22002529>

Browser tabs: Browse Data Mahasiswa, localhost / localhost / test / mahasi...

HASIL QUERY DARI DATABASE

NIM : 22002529
anton
3.68
L
1 baris terpilih.

NEXT

- Multimedia Server