

Bengkel Rekayasa Piranti Lunak

Aplikasi Web Dengan Java

By ekobs@developerforce.net

Draft 0.1.2

Last update 4 November 2003

Untuk release terbaru, question, troubleshoot dan feedback
kami undang Anda bergabung dengan
Masyarakat J2EE/Linux Indonesia
Bisa dilakukan dengan mengirim email ke

jlinux-subscribe@yahogroups.com

PENGANTAR

Bagaimana mengembangkan aplikasi Web dengan Java ...

Table of Contents

1Jakarta Tomcat	4
CS-071-011.....	5
CS-071-012.....	10
2Aplikasi Web.....	13
CS-071-021.....	14
CS-071-022.....	17
3HTML.....	20
CS-071-031.....	21
CS-071-033.....	23
CS-071-034.....	26
CS-071-035.....	29
CS-071-036.....	32
CS-071-037.....	35
4Cara Kerja Servlet	38
5Context	39
6Java Servlet	40
7Deployment Servlet di Jakarta Tomcat	41
8Initialization	42
9HttpServletRequest.....	43
10HttpServletResponse	44
11Akses Database Melalui Servlet.....	45
CS-071-111.....	46
CS-071-112.....	49
CS-071-113.....	53
CS-071-114.....	56
CS-071-115.....	59
CS-071-141.....	65
CS-071-142.....	66
CS-071-191.....	74
CS-071-191.....	77

1 Jakarta Tomcat

Jakarta Tomcat adalah web application server, yang mempunyai kemampuan sebagai Servlet container dan JSP container di mana Anda bisa men-deploy Servlet dan JSP. Di atas Jakarta Tomcat, Servlet dan JSP akan bekerja melayani request dari client, yang lumrahnya adalah berupa browser.

Untuk bisa menjalankan Jakarta Tomcat, Anda membutuhkan Java Development Kit (JDK). Untuk instalasi Jakarta Tomcat, Anda bisa men-download binary dari <http://jakarta.apache.org> , dalam format .zip, .tar.gz. Yang Anda perlu lakukan hanyalah men-decompress file tersebut.

Dalam bekerja dengan Jakarta Tomcat, Anda mempunyai sebuah directory yang dikenal sebagai TOMCAT_HOME. TOMCAT_HOME adalah directory di mana Jakarta Tomcat di-install.

Selanjutnya di bawah TOMCAT_HOME Anda akan menemukan beberapa sub-directory, di antaranya bin/, conf/, logs/ dan webapp/. Di dalam sub-directory bin/ terdapat file-file executable terutama untuk menjalankan dan menghentikan Jakarta Tomcat. Di dalam sub-directory conf/ terdapat file-file untuk configuration. Di dalam sub-directory logs/ terdapat file-file log. Dan sub-directort webapp/ adalah di mana Anda bisa meletakkan aplikasi Web yang Anda bangun dengan Servlet dan JSP.

Di bawah sub-directory webapp/ Anda bisa meng-create sub-directory. Sub directory ini akan dijadikan sebagai Context oleh Jakarta Tomcat.

Anda menjalankan Jakarta Tomcat dengan meng-execute **startup.sh** di subdirectory bin/. Sedangkan untuk menghentikan Tomcat Anda meng-execute **shutdown.sh** di sub directory bin/ juga.

Secara default Jakarta Tomcat siap melayani request dari client melalui port 8080. Melalui Web browser, Anda bisa menghubungi **<http://localhost:8080>**

.

CS-071-011

Sebuah praktikum dengan Servlet yang menggenerate satu halaman Web yang menampilkan sebuah pesan.

Persiapan

Buat sebuah directory untuk bekerja
misalnya /home/lab/myjava
lalu buat sub directory WEB-INF
dan sub directory WEB-INF/classes

Langkah

Langkah ke-1

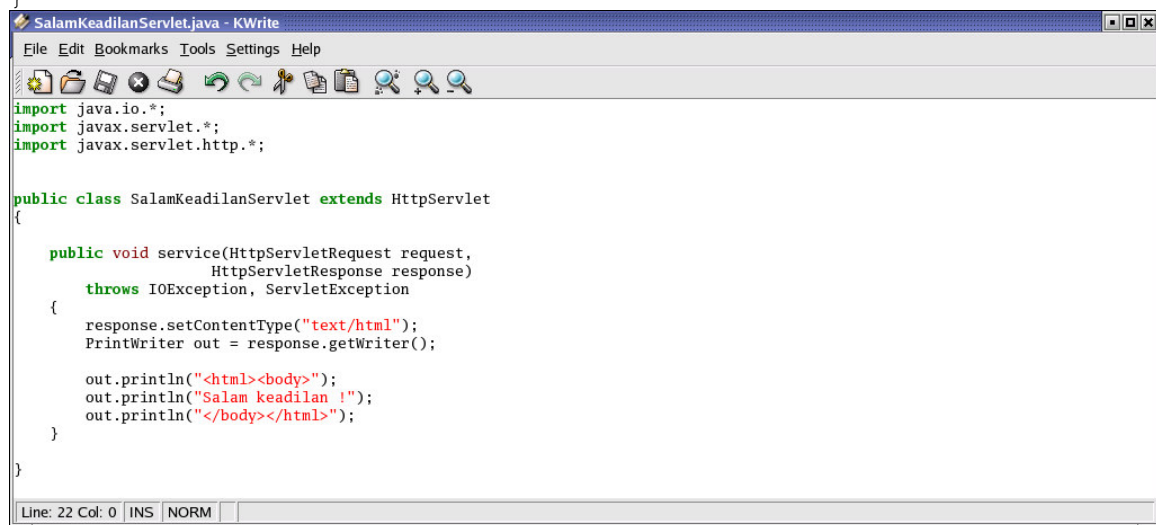
Tulis SalamKeadilanServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes

SalamKeadilanServlet.java

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

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

        out.println("<html><body>");
        out.println("Salam keadilan !");
        out.println("</body></html>");
    }
}
```



Langkah ke-2

Compile ...

```
$ export CLASSPATH=/home/lab/jakarta-tomcat-4.1.18/common/lib/servlet.jar
$ javac WEB-INF/classes/SalamKeadilanServlet.java
```

Langkah ke-3

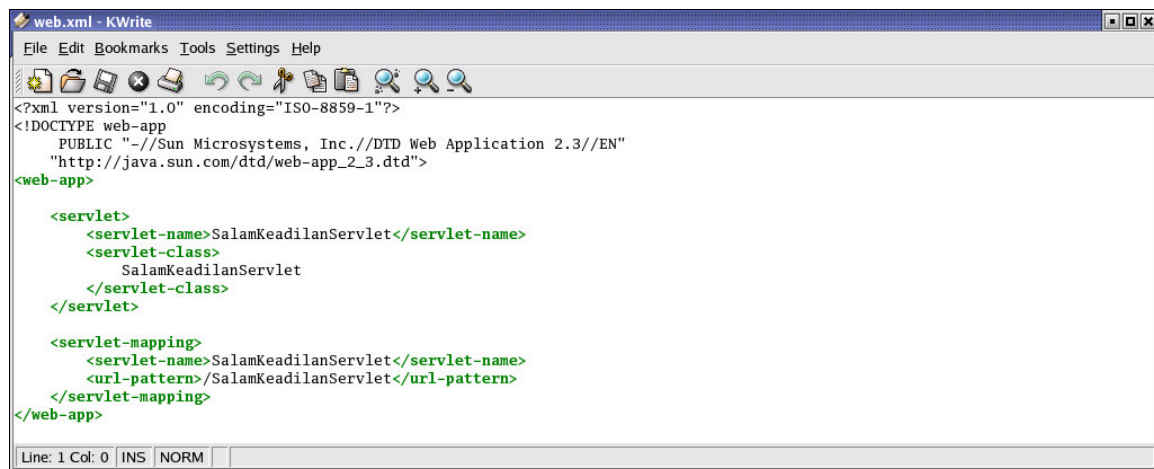
Tulis web.xml, simpan ke sub directory WEB-INF

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <servlet-class>
            SalamKeadilanServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <url-pattern>/SalamKeadilanServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

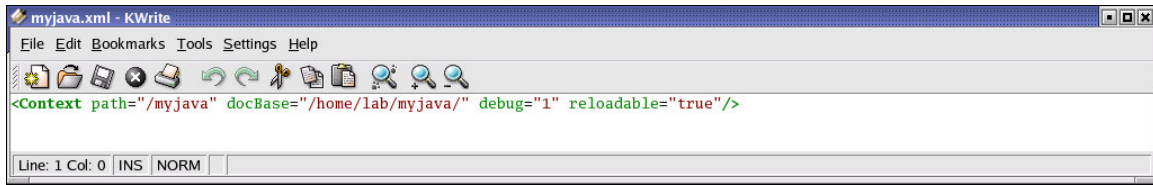


Langkah ke-4

Tulis myjava.xml yang mendefinisikan Web application Context myjava

myjava.xml

```
<Context path="/myjava" docBase="/home/lab/myjava/" debug="1"
    reloadable="true"/>
```



Langkah ke-5

Pindahkan myjava.xml ke TOMCAT_HOME/webapps

```
$ mv myjava.xml /home/lab/jakarta-tomcat-4.1.18/webapps
```

Langkah ke-6

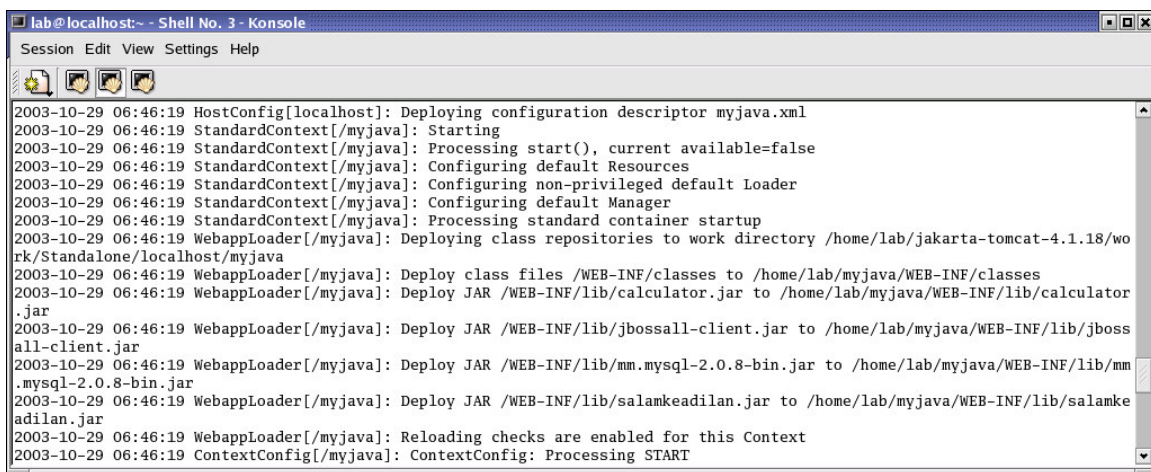
Terpantau di log dari Tomcat :

```
2003-10-29 06:46:19 HostConfig[localhost]: Deploying configuration descriptor myjava.xml
2003-10-29 06:46:19 StandardContext[/myjava]: Starting
2003-10-29 06:46:19 StandardContext[/myjava]: Processing start(), current available=false
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring default Resources
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring non-privileged default Loader
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring default Manager
2003-10-29 06:46:19 StandardContext[/myjava]: Processing standard container startup
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploying class repositories to work directory /home/lab/jakarta-tomcat-4.1.18/work/Standalone/localhost/myjava
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy class files /WEB-INF/classes to /home/lab/myjava/WEB-INF/classes
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/calculator.jar to /home/lab/myjava/WEB-INF/lib/calculator.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/jbossall-client.jar to /home/lab/myjava/WEB-INF/lib/jbossall-client.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/mm.mysql-2.0.8-bin.jar to /home/lab/myjava/WEB-INF/lib/mm.mysql-2.0.8-bin.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/salamkeadilan.jar to /home/lab/myjava/WEB-INF/lib/salamkeadilan.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Reloading checks are enabled for this Context
2003-10-29 06:46:19 ContextConfig[/myjava]: ContextConfig: Processing START
2003-10-29 06:46:19 StandardContext[/myjava]: Setting deployment descriptor public ID to '-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN'
2003-10-29 06:46:19 StandardContext[/myjava]: Setting deployment descriptor public ID to '-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN'
2003-10-29 06:46:19 ContextConfig[/myjava]: Accumulating TLD resource paths
2003-10-29 06:46:19 ContextConfig[/myjava]: Scanning JAR at resource path '/WEB-INF/lib/mm.mysql-2.0.8-bin.jar'
2003-10-29 06:46:19 ContextConfig[/myjava]: Scanning JAR at resource path '/WEB-INF/lib/jbossall-client.jar'
2003-10-29 06:46:19 ContextConfig[/myjava]: Scanning JAR at resource path '/WEB-INF/lib/calculator.jar'
```

```

2003-10-29 06:46:19 ContextConfig[/myjava]: Scanning JAR at resource
path '/WEB-INF/lib/salamkeadilan.jar'
2003-10-29 06:46:19 ContextConfig[/myjava]: Pipeline Configuration:
2003-10-29 06:46:19 ContextConfig[/myjava]:
org.apache.catalina.core.StandardContextValve/1.0
2003-10-29 06:46:19 ContextConfig[/myjava]: =====
2003-10-29 06:46:19 NamingContextListener[/Standalone/localhost/myjava]:
Creating JNDI naming context
2003-10-29 06:46:19 StandardManager[/myjava]: Seeding random number
generator class java.security.SecureRandom
2003-10-29 06:46:19 StandardManager[/myjava]: Seeding of random number
generator has been completed
2003-10-29 06:46:19 StandardContext[/myjava]: Posting standard context
attributes
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring application
event listeners
2003-10-29 06:46:19 StandardContext[/myjava]: Sending application start
events
2003-10-29 06:46:19 StandardContext[/myjava]: Starting filters
2003-10-29 06:46:19 StandardWrapper[/myjava:default]: Loading container
servlet default
2003-10-29 06:46:19 StandardWrapper[/myjava:invoker]: Loading container
servlet invoker
2003-10-29 06:46:19 StandardContext[/myjava]: Starting completed

```



```

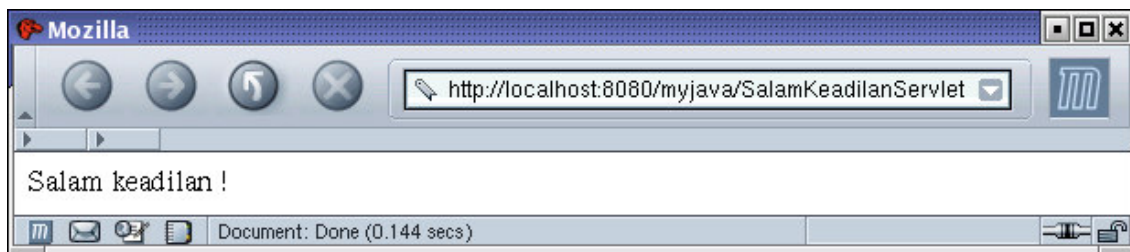
2003-10-29 06:46:19 HostConfig[localhost]: Deploying configuration descriptor myjava.xml
2003-10-29 06:46:19 StandardContext[/myjava]: Starting
2003-10-29 06:46:19 StandardContext[/myjava]: Processing start(), current available=false
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring default Resources
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring non-privileged default Loader
2003-10-29 06:46:19 StandardContext[/myjava]: Configuring default Manager
2003-10-29 06:46:19 StandardContext[/myjava]: Processing standard container startup
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploying class repositories to work directory /home/lab/jakarta-tomcat-4.1.18/work/Standalone/localhost/myjava
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy class files /WEB-INF/classes to /home/lab/myjava/WEB-INF/classes
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/calculator.jar to /home/lab/myjava/WEB-INF/lib/calculator.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/jbossall-client.jar to /home/lab/myjava/WEB-INF/lib/jbossall-client.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/mm.mysql-2.0.8-bin.jar to /home/lab/myjava/WEB-INF/lib/mm.mysql-2.0.8-bin.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Deploy JAR /WEB-INF/lib/salamkeadilan.jar to /home/lab/myjava/WEB-INF/lib/salamkeadilan.jar
2003-10-29 06:46:19 WebappLoader[/myjava]: Reloading checks are enabled for this Context
2003-10-29 06:46:19 ContextConfig[/myjava]: ContextConfig: Processing START

```

Langkah ke-8

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/SalamKeadilanServlet>



Jika Anda melihat source dari halaman Web yang ditampilkan adalah :

```
<html><body>  
Salam keadilan !  
</body></html>
```

CS-071-012

Praktikum lain ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis DateServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes

DateServlet.java

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

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

        Date date = new Date();

        SimpleDateFormat format
            = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");
        out.println("<html><body>");
        out.println(format.format(date));
        out.println("</body></html>");
    }
}
```

Langkah ke-2

Compile ...

```
$ javac WEB-INF/classes/DateServlet.java
```

Langkah ke-3

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>DateServlet</servlet-name>
        <servlet-class>
```

```

        DateServlet
    </servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>DateServlet</servlet-name>
    <url-pattern>/DateServlet</url-pattern>
</servlet-mapping>
</web-app>

```

Langkah ke-4

Terpantau di log dari Tomcat :

```

2003-10-17 07:00:07 StandardContext[/myjava]: Stopping
2003-10-17 07:00:07 StandardContext[/myjava]: Stopping filters
2003-10-17 07:00:07 StandardContext[/myjava]: Processing standard
container shutdown
2003-10-17 07:00:07 ContextConfig[/myjava]: ContextConfig: Processing
STOP
2003-10-17 07:00:07 StandardContext[/myjava]: Sending application stop
events
2003-10-17 07:00:07 StandardContext[/myjava]: Stopping complete
2003-10-17 07:00:07 StandardContext[/myjava]: Starting
2003-10-17 07:00:07 StandardContext[/myjava]: Processing start(),
current available=false
2003-10-17 07:00:07 StandardContext[/myjava]: Processing standard
container startup
2003-10-17 07:00:07 WebappLoader[/myjava]: Deploying class repositories
to work directory /home/lab/jakarta-tomcat-
4.1.18/work/Standalone/localhost/myjava
2003-10-17 07:00:07 WebappLoader[/myjava]: Deploy class files /WEB-
INF/classes to /home/lab/myjava/WEB-INF/classes
2003-10-17 07:00:07 WebappLoader[/myjava]: Reloading checks are enabled
for this Context
2003-10-17 07:00:07 ContextConfig[/myjava]: ContextConfig: Processing
START
2003-10-17 07:00:07 StandardContext[/myjava]: Setting deployment
descriptor public ID to '-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN'
2003-10-17 07:00:07 StandardContext[/myjava]: Setting deployment
descriptor public ID to '-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN'
2003-10-17 07:00:07 ContextConfig[/myjava]: Accumulating TLD resource
paths
2003-10-17 07:00:07 ContextConfig[/myjava]: Pipeline Configuration:
2003-10-17 07:00:07 ContextConfig[/myjava]:
org.apache.catalina.core.StandardContextValve/1.0
2003-10-17 07:00:07 ContextConfig[/myjava]: =====
2003-10-17 07:00:07 NamingContextListener[/Standalone/localhost/myjava]:
Creating JNDI naming context
2003-10-17 07:00:07 StandardManager[/myjava]: Seeding random number
generator class java.security.SecureRandom
2003-10-17 07:00:07 StandardManager[/myjava]: Seeding of random number
generator has been completed
2003-10-17 07:00:07 StandardContext[/myjava]: Posting standard context
attributes
2003-10-17 07:00:07 StandardContext[/myjava]: Configuring application
event listeners
2003-10-17 07:00:07 StandardContext[/myjava]: Sending application start
events
2003-10-17 07:00:07 StandardContext[/myjava]: Starting filters
2003-10-17 07:00:07 StandardWrapper[/myjava:default]: Loading container
servlet default
2003-10-17 07:00:07 StandardWrapper[/myjava:invoker]: Loading container

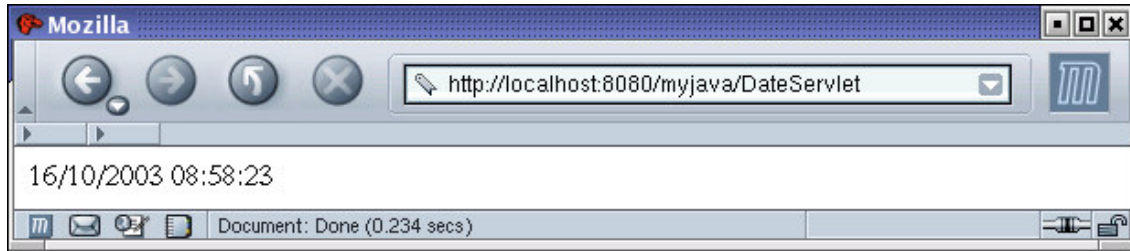
```

```
servlet invoker  
2003-10-17 07:00:07 StandardContext[/myjava]: Starting completed
```

Langkah ke-5

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

`http://localhost:8080/myjava/DateServlet`



Jika Anda melihat source dari halaman Web yang ditampilkan adalah :

```
<html><body>  
16/10/2003 08:58:23  
</body></html>
```

2Aplikasi Web

Teknologi inti untuk mengembangkan aplikasi Web dengan Java adalah Servlet. Servlet adalah sebuah class yang digunakan untuk menerima request dan memberikan response, terutama melalui protokol HTTP. Anda menulis source code dari Servlet, lalu meng-compile dan men-deploy di java web server. Selanjutnya client dapat berinteraksi dengan Servlet melalui browser.

Servlet bisa dipandang sebagai class yang bisa digunakan untuk menulis response dalam format HTML. Ia ditulis sebagaimana lumrahnya sebuah class di dalam bahasa pemrograman Java. Servlet disimpan sebagai file .java. Untuk mengirimkan response dalam format HTML, Anda bisa menulisnya melalui obyek `PrintWriter`, yang didapatkan dari `HttpServletResponse`.

Dalam perjalanannya, dikembangkan teknologi Java Server Page (JSP) di mana Anda bisa menulis script untuk aplikasi Web dengan bahasa Java.

Berbeda dengan Servlet, JSP bisa dipandang sebagai HTML yang di dalamnya bisa mempunyai kode-kode Java. JSP disimpan sebagai file .jsp. Menulis JSP adalah seperti menulis file HTML, kecuali di dalamnya dapat disisipkan kode-kode Java sebagai presentation logic. Kode-kode Java ini disisipkan melalui directive, sebagai scriplet, atau sebagai expression.

Servlet dan JSP mempunyai kemampuan yang kembar. Keduanya bisa membaca input yang dikirimkan melalui form di Web, mengakses database melalui JDBC, mengolah data dan menulis response ke browser. Response lumrahnya dalam format HTML.

Perbedaan Servlet dan JSP lebih kepada proses pengembangannya. Sedangkan dalam operasinya, keduanya adalah sama. Oleh web application server, JSP akan di-rewrite menjadi Servlet, di-compile dan selanjutnya akan diperlakukan sebagaimana Servlet.

CS-071-021

Praktikum membaca input dari Servlet ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=SalamKeadilanServlet method=post>
    Nama Anda : <input type=text name=name>
    <input type=submit value='Click me!'/>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis SalamKeadilanServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

SalamKeadilanServlet.java

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

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

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

        out.println("<html><body>");
        out.println("Hi " + name + ", salam keadilan !");
        out.println("</body></html>");
    }
}
```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/SalamKeadilanServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <servlet-class>
            SalamKeadilanServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <url-pattern>/SalamKeadilanServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

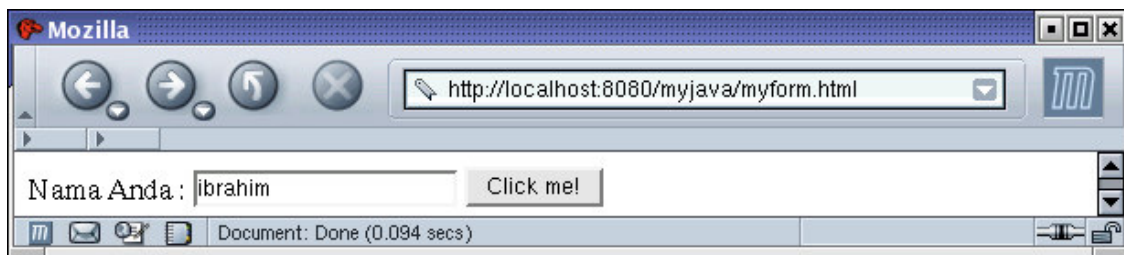
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

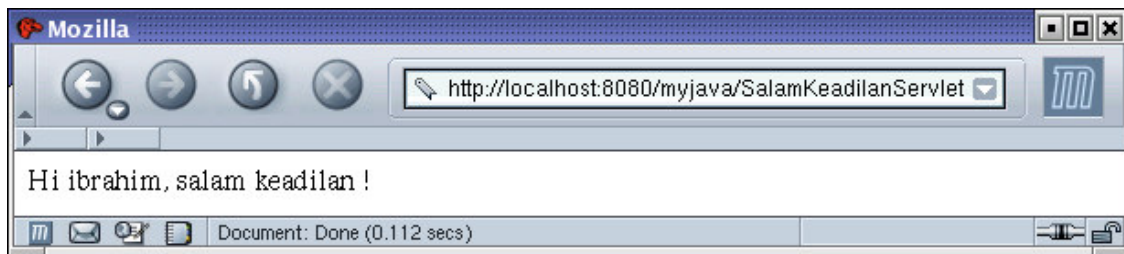
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, SalamKeadilanServlet akan di-invoke dari browser :



CS-071-022

Praktikum membaca input dari Servlet ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=DateServlet method=post>
    Date Format :

    <select name=format>
      <option value='d/M/y'>d/M/y</option>
      <option value='d/M/yyyy'>d/M/yyyy</option>
      <option value='d/MMM/y'>d/MMM/y</option>
      <option value='d/MMM/yyyy'>d/MMM/yyyy</option>
      <option value='d/MMMM/yyyy'>d/MMMM/yyyy</option>
      <option value='d/M/y h:m:s'>d/M/y h:m:s</option>
      <option value='d/MMM/y h:m:s'>d/MMM/y h:m:s</option>
      <option value='d/MMMM/yyyy hh:mm:ss'>d/MMMM/yyyy
hh:mm:ss</option>
    </select>
    <input type=submit value='Show time !'>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis DateServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

DateServlet.java

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

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

        Date date = new Date();
```



```

        String formatStr = request.getParameter("format");
        SimpleDateFormat format = new SimpleDateFormat(formatStr);
        out.println("<html><body>");

        out.println(format.format(date));

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

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/DateServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>DateServlet</servlet-name>
        <servlet-class>
            DateServlet
        </servlet-class>
    </servlet>

        <servlet-mapping>
            <servlet-name>DateServlet</servlet-name>
            <url-pattern>/DateServlet</url-pattern>
        </servlet-mapping>
</web-app>

```

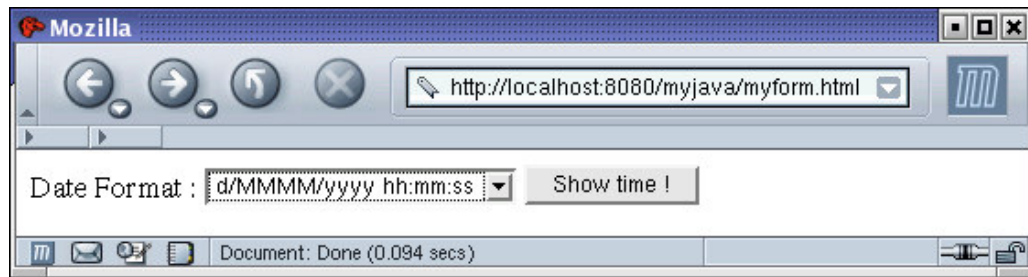
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

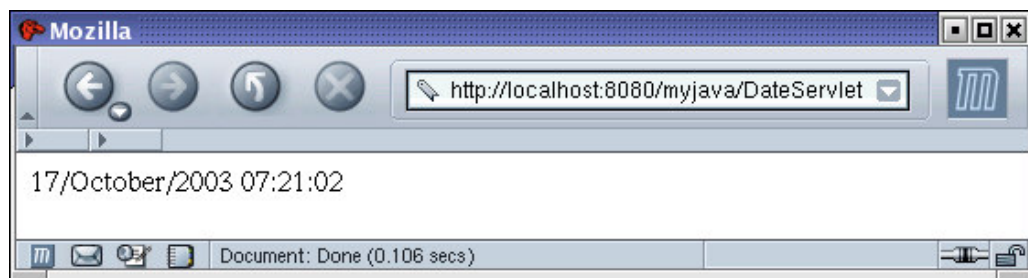
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, DateServlet akan di-invoke dari browser :



3HTML

Hypertext Markup Language (HTML) merupakan salah satu teknologi inti untuk World Wide Web (WWW). Dokumen-dokumen ditulis dalam format HTML untuk bisa ditampilkan di atas Web browser. Aplikasi web juga mengembalikan response ke Web browser dalam format HTML. Dalam aplikasi Java, Servlet atau JSP mempunyai kemampuan untuk menulis HTML ini.

HTML menyediakan tag-tag yang menunjukkan bagaimana informasi ditampilkan oleh Web browser. Di antaranya bagaimana font colour, bagaimana meletakkan image, dan bagaimana membuat table.

HTML juga menyediakan tag untuk hyperlink. Hyperlink adalah sebuah label yang bisa di-click oleh user, lalu Web browser akan membuka URL yang dideklarasikan dalam hyperlink tsb. Tag yang digunakan adalah `<a>...`

Untuk aplikasi Web, HTML menyediakan tag untuk membuat form, di mana user bisa melakukan data entry. Form ini selanjutnya bisa di-submit ke web application server untuk diproses. Data yang di-entry user dikirimkan sebagai parameter. Servlet atau JSP bisa membaca parameter ini.

Untuk membuat form, tag yang digunakan adalah `<form>...</form>`. HTML form mempunyai attribute bernama action, dan method. Action menyatakan Servlet atau JSP yang akan di-invoke saat form di-submit. Sedangkan method menyatakan HTTP transfer method yang akan digunakan. Method yang digunakan bisa POST atau GET.

Komponen-komponen input, yang digunakan user untuk melakukan data entry, ditulis di antara `<form>` dan `</form>`. Terdapat beberapa komponen input yang bisa ditulis, di antaranya text, password, radio button, checkbox, select serta button.

Button bisa digunakan untuk men-submit form ke web application server, untuk selanjutnya Servlet atau JSP yang ditentukan akan di-invoke.

CS-071-031

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis my.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

my.html

```
<html>
<body>
<a href='InputReaderServlet?myInput=DeveloperForce.Net'>
click me ! </a>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

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

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

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>InputReaderServlet</servlet-name>
        <url-pattern>/InputReaderServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

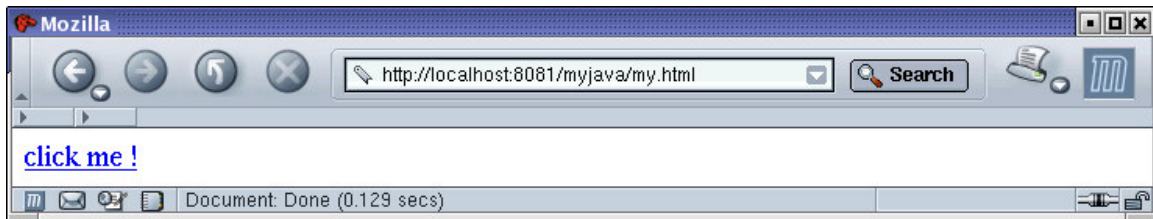
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

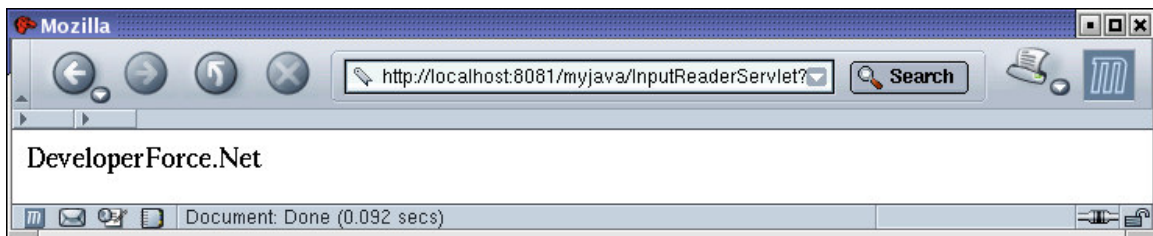
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/my.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



CS-071-033

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=InputReaderServlet method=post>
    <textarea name=myInput rows=3 cols=40></textarea>
    <br>
    <input type=submit value='Click me! '>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

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

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

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

        <servlet-mapping>
            <servlet-name>InputReaderServlet</servlet-name>
            <url-pattern>/InputReaderServlet</url-pattern>
        </servlet-mapping>
</web-app>
```

Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

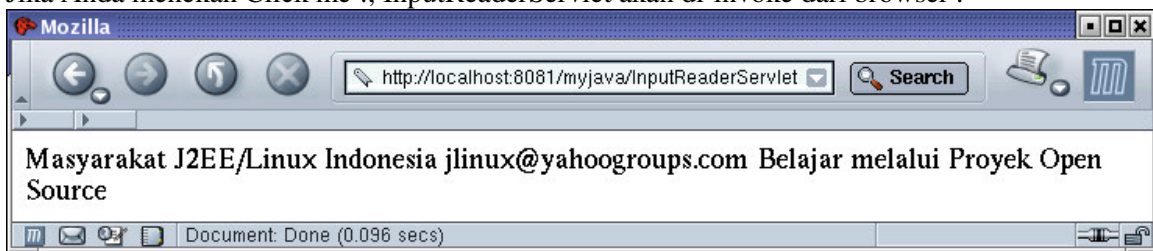
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



CS-071-034

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=InputReaderServlet method=post>
    <input type=checkbox name=bananaCheckBox> Banana
    <br>
    <input type=checkbox name=mangoCheckBox> Mango
    <br>
    <input type=checkbox name=orangeCheckBox> Orange
    <br>
    <input type=submit value='Click me!'/>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

        String bananaCheckBox = request.getParameter("bananaCheckBox");
        String mangoCheckBox = request.getParameter("mangoCheckBox");
        String orangeCheckBox = request.getParameter("orangeCheckBox");

        out.println("<html><body>");
        out.println("bananaCheckBox = " + bananaCheckBox);
        out.println("<br>");
        out.println("mangoCheckBox = " + mangoCheckBox);
        out.println("<br>");
    }
}
```

```

        out.println("orangeCheckBox = " + orangeCheckBox);
        out.println("<br>");
        out.println("</body></html>");
    }
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

        <servlet-mapping>
            <servlet-name>InputReaderServlet</servlet-name>
            <url-pattern>/InputReaderServlet</url-pattern>
        </servlet-mapping>
</web-app>

```

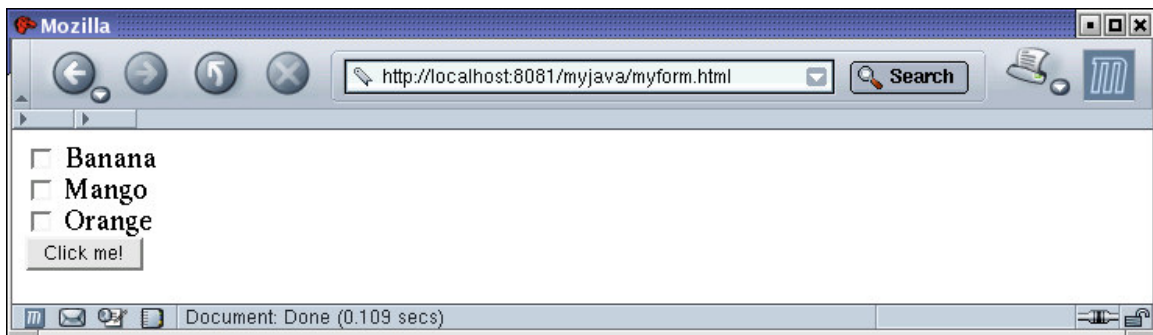
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

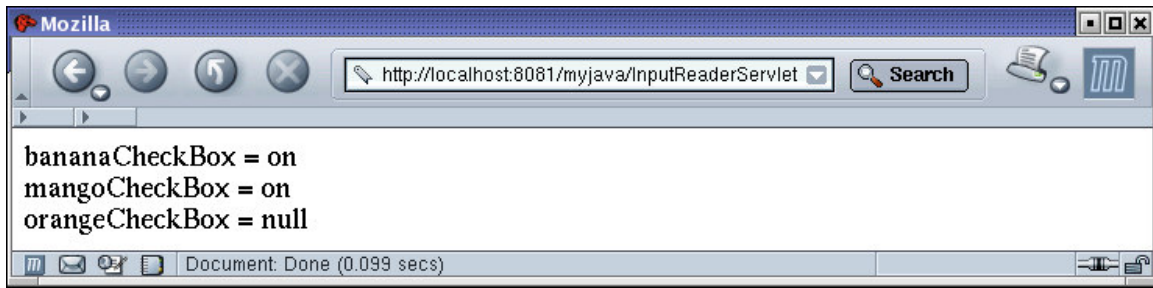
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



CS-071-035

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=InputReaderServlet method=post>
    <input type=radio name=fruitRadioButton value=Banana> Banana
    <br>
    <input type=radio name=fruitRadioButton value=Mango> Mango
    <br>
    <input type=radio name=fruitRadioButton value=Orange> Orange
    <br>
    <input type=submit value='Click me!'/>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

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

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

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>InputReaderServlet</servlet-name>
        <url-pattern>/InputReaderServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

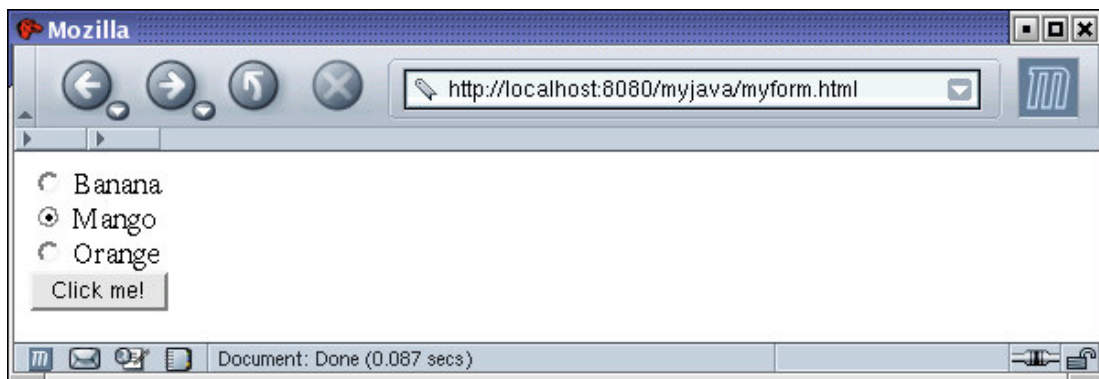
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

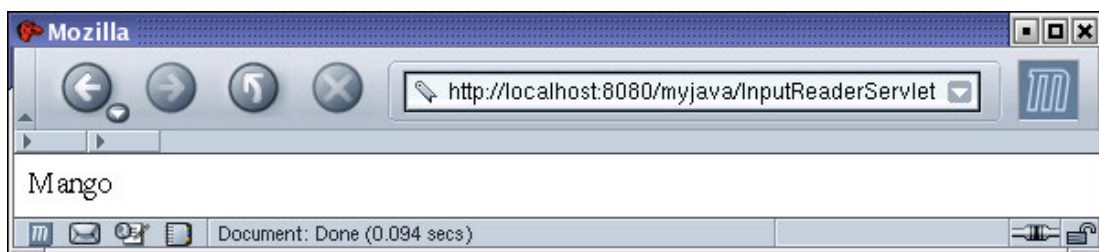
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



CS-071-036

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=InputReaderServlet method=post>
    <select name=fruitSelect>
      <option value=Banana>Banana</option>
      <option value=Mango>Mango</option>
      <option value=Orange>Orange</option>
    </select>
    <br>
    <input type=submit value='Click me! '>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

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

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

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

        <servlet-mapping>
            <servlet-name>InputReaderServlet</servlet-name>
            <url-pattern>/InputReaderServlet</url-pattern>
        </servlet-mapping>
</web-app>
```

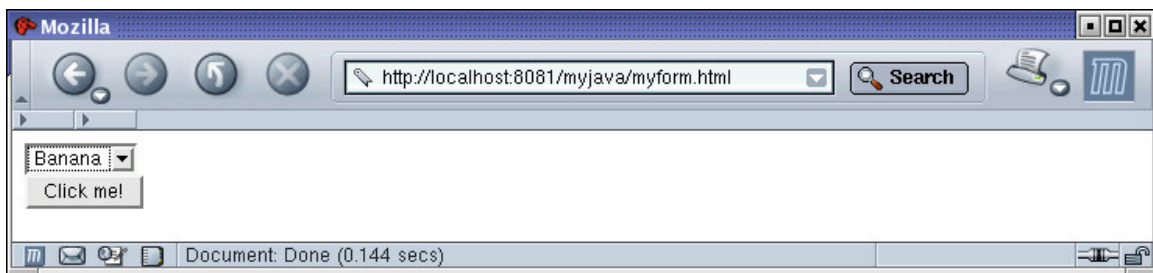
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

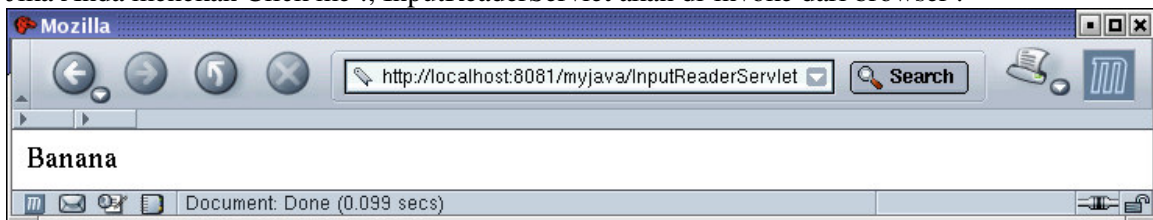
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



CS-071-037

Praktikum tentang HTML form ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis myform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

myform.html

```
<html>
<body>
  <form name=myForm action=InputReaderServlet method=post>
    <select name=fruitSelect size=5 multiple>
      <option value=Banana>Banana</option>
      <option value=Mango>Mango</option>
      <option value=Orange>Orange</option>
    </select>
    <br>
    <input type=submit value='Click me! '>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InputReaderServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InputReaderServlet.java

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

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

        String[] values = request.getParameterValues("fruitSelect");

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

        String fruitSelect = null;
        for(int i=0;i<values.length;i++)
        {
            if(i==0)
                fruitSelect = values[0];
        }
    }
}
```

```

        else
            fruitSelect += ", " + values[i];
    }

    out.println(fruitSelect);
    out.println("</body></html>");
}
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InputReaderServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InputReaderServlet</servlet-name>
        <servlet-class>
            InputReaderServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>InputReaderServlet</servlet-name>
        <url-pattern>/InputReaderServlet</url-pattern>
    </servlet-mapping>
</web-app>

```

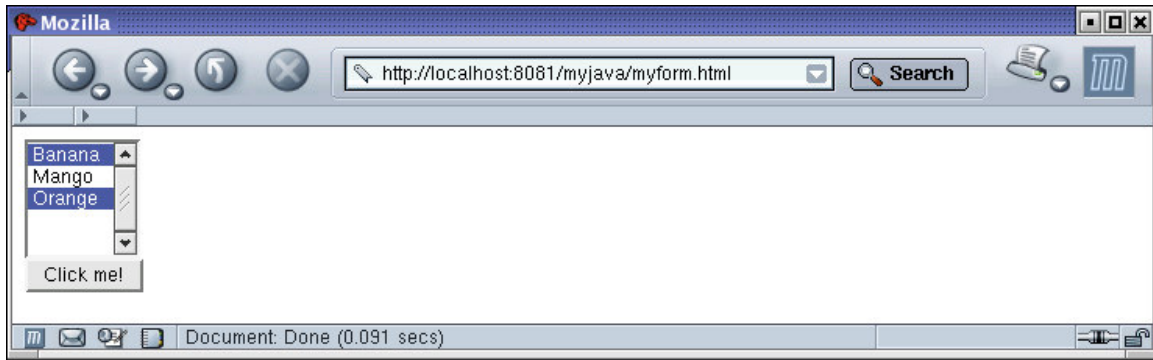
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/myform.html>



Jika Anda menekan Click me !, InputReaderServlet akan di-invoke dari browser :



4Cara Kerja Servlet

Servlet bekerja melayani request dari client, yang lumrahnya adalah Web browser. Untuk bisa melayani client, Servlet terlebih dahulu harus di-deploy di web application server, yang menyediakan kemampuan sebagai Servlet container.

Client memanggil Servlet dengan mengirimkan HTTP request ke web application server. HTTP request ini bisa di-transfer dengan method GET, POST atau lainnya. Method GET selalu terjadi jika user membuka sebuah URL. Method POST bisa digunakan saat user men-submit sebuah form.

Saat web application server menerima HTTP request dari client, ia akan menyerahkan request ini ke Servlet container. Servlet container akan meng-create dua buah obyek yaitu obyek `HttpServletRequest` dan obyek `HttpServletResponse`. Obyek `HttpServletRequest` meng-encapsulate HTTP request dari client, sedangkan obyek `HttpServletResponse` dipersiapkan untuk meng-encapsulate HTTP response ke client.

Selanjutnya Servlet container akan meng-invoke method dari Servlet dengan melewati dua obyek ini. Servlet yang di-invoke oleh Servlet container ditentukan oleh URI yang dikirimkan oleh Web browser, dan pemetaan yang dibuat melalui configuration. Dalam configuration dapat ditentukan bahwa URI dengan pola tertentu akan dilayani oleh Servlet tertentu.

Servlet bisa membaca data yang dikirimkan oleh client melalui obyek `HttpServletRequest`. Melalui obyek ini Servlet membaca parameter, cookie, dan juga informasi tentang client.

Selanjutnya untuk mengembalikan response ke client, Servlet bisa melakukannya melalui obyek `HttpServletResponse`. Lumrahnya Servlet menuliskan response dalam format HTML.

Sebelum menuliskan response, Servlet terlebih dahulu bisa mengolah data yang dikirimkan oleh client, mengakses dengan database dan melakukan proses-proses lain.

5Context

Sebuah **Context** adalah sebuah aplikasi Web yang terpisah, berdiri sendiri, independen. Sebuah Context mempunyai configuration masing-masing. Library dari sebuah Context juga tidak bisa dibaca oleh Context lain. Obyek di sebuah Context tidak bisa meng-akses obyek di Context lain.

Di atas sebuah web application server seperti Jakarta Tomcat bisa di-deploy lebih dari satu Context.

Anda bisa membuat sebuah Context dengan meng-create sebuah sub-directory di bawah **TOMCAT_HOME/webapp/**.

Sebuah Context yang lengkap mempunyai sub-directory WEB-INF/ di mana terdapat **web.xml** yang merupakan configuration file dari Context ini. Di dalam WEB-INF/ bisa terdapat sub-directory **classes/** dan **lib/**.

Sub-directory **classes/** adalah di mana file-file **.class** diletakkan, sedangkan **lib/** adalah di mana file-file **.jar**, yang merupakan kumpulan file-file **.class**, diletakkan.

6Java Servlet

Java Servlet ditulis sebagai lumrahnya Java class lain. Ia disimpan dalam file berekstension **.java**. Sebuah Java Servlet harus merupakan sub-class dari **HttpServlet**

Untuk melayani request dari client, Anda perlu meng-override method **service()**. Parameter yang dilewatkan ke dalam method **service()** ini berupa obyek **HttpServletRequest** dan obyek **HttpServletResponse**.

Untuk lebih spesifik terhadap HTTP method, Anda juga bisa meng-override method **doGet()** dan atau **doPost()**. Kedua method ini mempunyai parameter yang sama dengan **service()**.

Method **doGet()** akan dijalankan jika client mengirimkan HTTP request dengan method **GET**. Contoh dari method **GET**, adalah jika user meng-click sebuah link di halaman Web. Dalam kasus ini, Web browser akan mengirimkan HTTP request dengan method **GET** ke server.

Method **doPost()** akan dijalankan jika client mengirimkan HTTP response dengan method **POST**. Ini terjadi misalnya, saat user mengisi HTML form dengan method **POST**, dan men-submit request tersebut ke server.

Di dalam method **service()**, **doGet()**, atau **doPost()** ini Anda bisa membaca parameter yang dikirimkan client, mengolah data, mengakses database dan menulis response ke client.

7Deployment Servlet di Jakarta Tomcat

Untuk melakukan deployment, Anda perlu mempunyai **Context**. Ini bisa dibuat dengan membuat sub-directory di bawah **TOMCAT_HOME/webapp/**

Untuk men-deploy Servlet, pertama Anda meng-compile Servlet. Lalu Anda meletakkan file-file **.class** ke sub-directory **WEB-INF/classes** di bawah directory yang dibuat untuk Context Anda.

Jika dibutuhkan Anda bisa meletakkan file-file **.jar** di sub-directory **WEB-INF/lib**.

Selanjutnya Anda bisa memanggil dari browser, sesuai Context dan nama Servlet Anda.

Anda bisa juga meng-edit **web.xml**, di subdirectory **WEB-INF/** di bawah directory dari Context Anda. Melalui **web.xml** Anda, di antaranya, bisa membuat mapping yang mengatur bahwa pola URI tertentu akan dilayani oleh Servlet tertentu.

8Initialization

Anda bisa melakukan initialization terhadap Servlet sebelum Servlet melayani client.

```
init()  
init(ServletConfig)  
ServletConfig  
getInitParameter()  
getInitParameterNames()
```


9HttpServletRequest

Class `HttpServletRequest` digunakan untuk meng-encapsulate HTTP request dari client.

Untuk membaca parameter-parameter yang dikirimkan client, tersedia method-method `getParameter()`, `getParameterNames()` dan `getParameterValues()`.

Untuk mendapatkan header dari HTTP request tersedia method `getHeader()`, dan membaca cookie tersedia method `getCookie()`.

Untuk mendapatkan informasi tentang server di mana Servlet bekerja, tersedia method `getServerName()` dan `getServerPort()`.

Untuk mendapatkan informasi tentang client yang memanggil Servlet, tersedia method `getRemoteAddr()`, `getRemoteHost()` dan `getRemoteUser()`.

10HttpServletResponse

Class HttpServletResponse digunakan untuk meng-encapsulate HTTP response yang dikirimkan Servlet ke client.

Method `getWriter()` bisa digunakan untuk mendapatkan obyek `PrintWriter`. Method `getOutputStream()` bisa digunakan untuk mendapatkan obyek `ServletOutputStream`.

Melalui obyek `PrintWriter` atau `ServletOutputStream`, Anda bisa menuliskan response ke client. Obyek `PrintWriter` cocok jika response Anda adalah character, misal dalam format HTML, sedangkan obyek `ServletOutputStream` cocok jika response Anda adalah binary, misalnya berupa graphics.

Method `setHeader()` bisa digunakan untuk menuliskan header, dan method `setCookie()` bisa digunakan untuk menuliskan cookie.

Method `setStatus()` bisa digunakan untuk mengirimkan status code ke client.

11Akses Database Melalui Servlet

Servlet bisa berinteraksi dengan database menggunakan teknologi JDBC. JDBC bisa ditulis di class Servlet, atau di dalam sebuah Data Access Object yang selanjutnya di-invoke oleh Servlet.

CS-071-111

Praktikum untuk membuat JDBC Connection ke database dari Servlet ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis ConnectionServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

ConnectionServlet.java

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

public class ConnectionServlet extends HttpServlet
{
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");

        Connection conn = null;
        try
        {
            String jdbcDriver
                = "org.gjt.mm.mysql.Driver";
            Class.forName(jdbcDriver);

            String url = "jdbc:mysql://localhost:3306/MYAPP_DB";
            String user = "ekobs";
            String pwd = "j2ee";

            out.println
                ("Mencoba membangun connection ke '" + url
                 + "' dengan user '" + user
                 + "' dan password '" + pwd + "' ...");
            out.println("<br>");
            conn = DriverManager.getConnection(
                url, user, pwd);
            out.println("Success.");
            out.println("<br>");

        }
        catch(Exception e)
        {
            out.println(e.toString());
            e.printStackTrace();
        }
        finally
        {

```

```

        try
        {
            if(conn != null)
            {
                conn.close();
                out.println("Menutup connection.");
            }
        }
        catch(SQLException sqle)
        {
            sqle.printStackTrace();
        }
    }
    out.println("</body></html>");
}
}

```

Langkah ke-2

Compile ...

```
$ javac WEB-INF/classes/ConnectionServlet.java
```

Langkah ke-3

Jika belum ada, buat directory lib di bawah directory WEB-INF/

Copy JDBC driver untuk mySQL, misalnya mm.mysql-2.0.8-bin.jar ke sub-directory WEB-INF/lib

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>ConnectionServlet</servlet-name>
        <servlet-class>
            ConnectionServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ConnectionServlet</servlet-name>
        <url-pattern>/ConnectionServlet</url-pattern>
    </servlet-mapping>
</web-app>

```

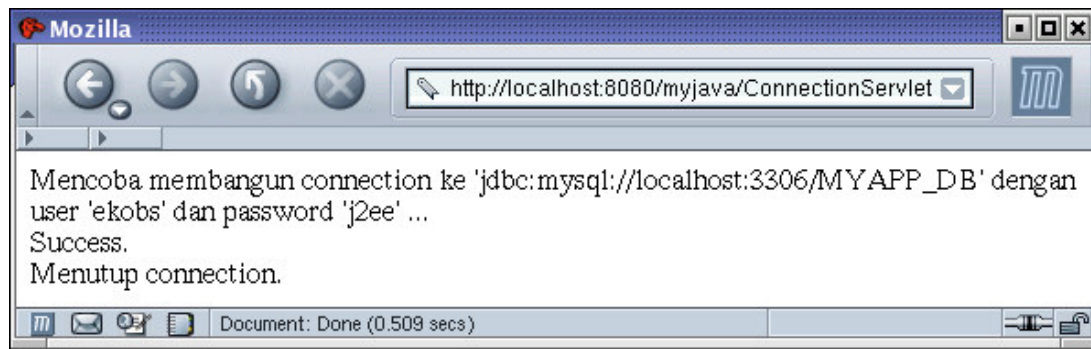
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap di-invoke dari browser melalui URL

<http://localhost:8080/myjava/ConnectionServlet>



CS-071-112

Praktikum melakukan insert ke sebuah table di database ...

Prasyarat

Anda harus melakukan praktikum CS-071-111 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis insertform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

insertform.html

```
<html>
<body>
  <form name=insetForm action=InsertServlet method=post>
    Id : <input type=text name=id>
    <br>
    Name : <input type=text name=name>
    <br>
    Department : <input type=text name=department>
    <br>
    Job Title : <input type=text name=jobTitle>
    <br>
    Hire Date (DDMMYYYY) : <input type=text name=hireDate>
    <br>
    Permanent Employee : <input type=checkbox name=permanentEmployee
value=true>
    <br>
    Salary : <input type=text name=salary>
    <br>
    <input type=submit value='Insert !'>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis InsertServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

InsertServlet.java

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

public class InsertServlet extends HttpServlet
{
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
    }
}
```

```

PrintWriter out = response.getWriter();
out.println("<html><body>");

Connection conn = null;
try
{
    String jdbcDriver
        = "org.gjt.mm.mysql.Driver";
    Class.forName(jdbcDriver);

    String url = "jdbc:mysql://localhost:3306/MYAPP_DB";
    String user = "ekobs";
    String pwd = "j2ee";
    conn = DriverManager.getConnection(
        url, user, pwd);

    SimpleDateFormat format = new SimpleDateFormat("ddMMyyyy");

    String id = request.getParameter("id");
    String name = request.getParameter("name");
    String department = request.getParameter("department");
    String jobTitle = request.getParameter("jobTitle");
    java.util.Date hireDate
        = format.parse(request.getParameter("hireDate"));
    boolean permanentEmployee
        = Boolean.valueOf(
            request.getParameter("permanentEmployee"))
            .booleanValue();
    double salary
        = Double.parseDouble(request.getParameter("salary"));

    String sqlInsert
        = "INSERT INTO EMPLOYEE_TBL VALUES (?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement stmt
        = conn.prepareStatement(sqlInsert);

    stmt.setString(1, id);
    stmt.setString(2, name);
    stmt.setString(3, department);
    stmt.setString(4, jobTitle);
    stmt.setDate(5, new java.sql.Date(hireDate.getTime()));
    stmt.setBoolean(6, permanentEmployee);
    stmt.setDouble(7, salary);

    int count = stmt.executeUpdate();
    stmt.close();
    out.println("Query OK, " + count + " row affected");
}
catch(Exception e)
{
    out.println(e.toString());
    e.printStackTrace();
}
finally
{
    try
    {
        if(conn != null)
        {
            conn.close();
        }
    }
    catch(SQLException sqle)
    {

```



```

        sqle.printStackTrace();
    }
}
out.println("</body></html>");
}
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/InsertServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>InsertServlet</servlet-name>
        <servlet-class>
            InsertServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>InsertServlet</servlet-name>
        <url-pattern>/InsertServlet</url-pattern>
    </servlet-mapping>
</web-app>

```

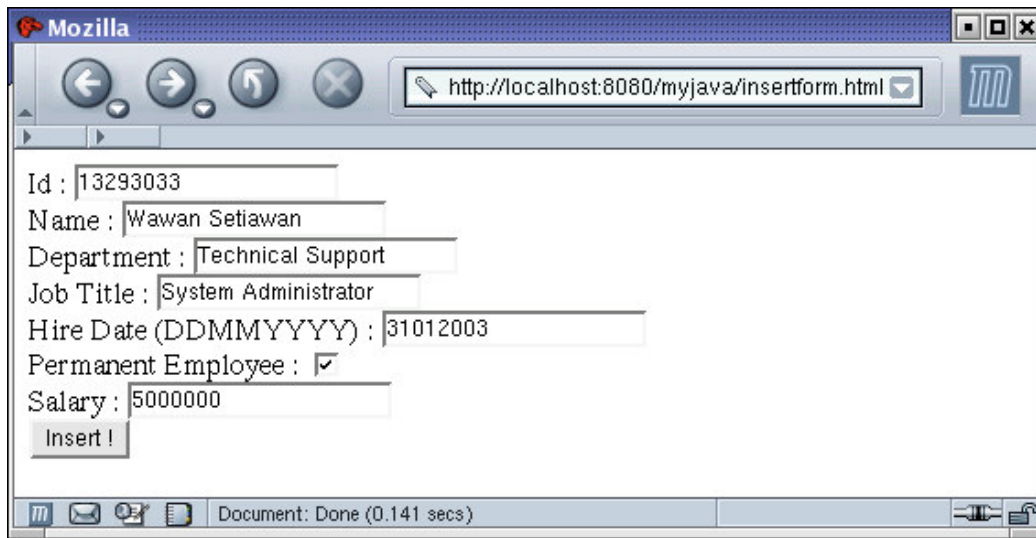
Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

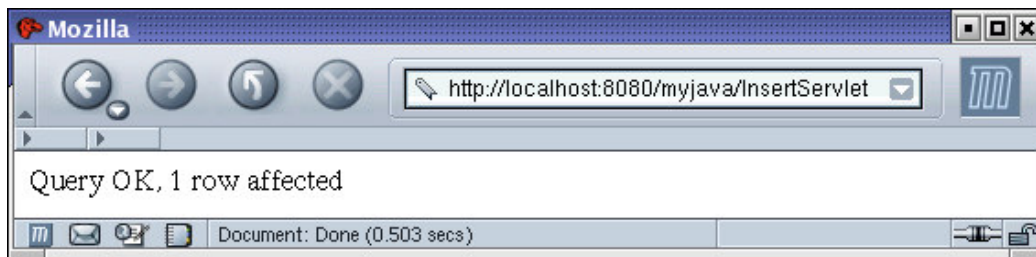
Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/insertform.html>



Jika Anda menekan Insert !, InsertServlet akan di-invoke dari browser :



CS-071-113

Praktikum melakukan select dari sebuah table di database ...

Prasyarat

Anda harus melakukan praktikum CS-071-111 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis selectform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

selectform.html

```
<html>
<body>
    <form name=selectForm action=SelectServlet method=post>
        Id : <input type=text name=id>
        <br>
        <input type=submit value='Select !'>
    </form>
</body>
</html>
```

Langkah ke-2

Tulis SelectServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

SelectServlet.java

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

public class SelectServlet extends HttpServlet
{
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");

        Connection conn = null;
        try
        {
            String jdbcDriver
                = "org.gjt.mm.mysql.Driver";
            Class.forName(jdbcDriver);

            String url = "jdbc:mysql://localhost:3306/MYAPP_DB";
            String user = "ekobs";
            String pwd = "j2ee";
```

```

conn = DriverManager.getConnection(
    url, user, pwd);

String id = request.getParameter("Id");

String sqlSelect =
    "SELECT * FROM EMPLOYEE_TBL ";

if( id != null && !id.trim().equals(""))
{
    sqlSelect += " WHERE ID = ?";
}

PreparedStatement stmt = conn.prepareStatement(sqlSelect);

if( id != null && !id.trim().equals(""))
{
    stmt.setString(1, id);
}

System.out.println(sqlSelect);
System.out.println("id = " + id);

ResultSet rs = stmt.executeQuery();

SimpleDateFormat format
    = new SimpleDateFormat("dd/MMMM/yyyy");

while(rs.next())
{
    id = rs.getString("ID");
    String name = rs.getString("NAME");
    String department = rs.getString("DEPARTMENT");
    Date hireDate = rs.getDate("HIRE_DATE");
    boolean permanentEmployee
        = rs.getBoolean("PERMANENT_EMPLOYEE");
    double salary = rs.getDouble("SALARY");

    out.println("E M P L O Y E E");
    out.println("<br>");
    out.println("Id : " + id);
    out.println("<br>");
    out.println("Name : " + name);
    out.println("<br>");
    out.println("Department : " + department);
    out.println("<br>");
    out.println("Hire Date : " + format.format(hireDate));
    out.println("<br>");
    out.println("Permanent Emp : " + permanentEmployee);
    out.println("<br>");
    out.println("Salary : " + salary);
    out.println("<br>");
    out.println("<br>");
    out.println("<br>");

}
rs.close();
stmt.close();
}
catch(Exception e)
{
    out.println(e.toString());
    e.printStackTrace();
}

```

```

        finally
        {
            try
            {
                if(conn != null)
                {
                    conn.close();
                }
            }
            catch(SQLException sqle)
            {
                sqle.printStackTrace();
            }
        }
        out.println("</body></html>");
    }
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/SelectServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>SelectServlet</servlet-name>
        <servlet-class>
            SelectServlet
        </servlet-class>
    </servlet>

        <servlet-mapping>
            <servlet-name>SelectServlet</servlet-name>
            <url-pattern>/SelectServlet</url-pattern>
        </servlet-mapping>
</web-app>

```

Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

```
http://localhost:8080/myjava/selectform.html
```

Jika Anda menekan Select !, SelectServlet akan di-invoke dari browser :

CS-071-114

Praktikum melakukan update ke sebuah table di database ...

Prasyarat

Anda harus melakukan praktikum CS-071-111 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis updateform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

updateform.html

```
<html>
<body>
  <form name=updateForm action=UpdateServlet method=post>
    Id : <input type=text name=id>
    <br>
    Name : <input type=text name=name>
    <br>
    Department : <input type=text name=department>
    <br>
    Job Title : <input type=text name=jobTitle>
    <br>
    Hire Date (DDMMYYYY) : <input type=text name=hireDate>
    <br>
    Permanent Employee : <input type=checkbox name=permanentEmployee
value=true>
    <br>
    Salary : <input type=text name=salary>
    <br>
    <input type=submit value='Insert !'>
  </form>
</body>
</html>
```

Langkah ke-2

Tulis UpdateServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

UpdateServlet.java

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

public class UpdateServlet extends HttpServlet
{
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
```

```

response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><body>");

Connection conn = null;
try
{
    String jdbcDriver
        = "org.gjt.mm.mysql.Driver";
    Class.forName(jdbcDriver);

    String url = "jdbc:mysql://localhost:3306/MYAPP_DB";
    String user = "ekobs";
    String pwd = "j2ee";
    conn = DriverManager.getConnection(
        url, user, pwd);

    SimpleDateFormat format = new SimpleDateFormat("ddMMyyyy");

    String id = request.getParameter("id");
    String name = request.getParameter("name");
    String department = request.getParameter("department");
    String jobTitle = request.getParameter("jobTitle");
    java.util.Date hireDate
        = format.parse(request.getParameter("hireDate"));
    boolean permanentEmployee
        = Boolean.valueOf(
            request.getParameter("permanentEmployee"))
            .booleanValue();

    double salary =
Double.parseDouble(request.getParameter("salary"));

    String sqlUpdate =
        "UPDATE EMPLOYEE_TBL "
        + "SET "
        + "NAME = ?, "
        + "DEPARTMENT = ?, "
        + "JOB_TITLE = ?, "
        + "HIRE_DATE = ?, "
        + "PERMANENT_EMPLOYEE = ?, "
        + "SALARY = ? "
        + "WHERE "
        + "ID = ?";
    PreparedStatement stmt = conn.prepareStatement(sqlUpdate);

    stmt.setString(7, id);
    stmt.setString(1, name);
    stmt.setString(2, department);
    stmt.setString(3, jobTitle);
    stmt.setDate(4, new java.sql.Date(hireDate.getTime()));
    stmt.setBoolean(5, permanentEmployee);
    stmt.setDouble(6, salary);

    int count = stmt.executeUpdate();
    stmt.close();
    out.println("Query OK, " + count + " row affected");
}
catch(Exception e)
{
    out.println(e.toString());
    e.printStackTrace();
}
finally
{

```

```

        try
        {
            if(conn != null)
            {
                conn.close();
            }
        }
        catch(SQLException sqle)
        {
            sqle.printStackTrace();
        }
    }
    out.println("</body></html>");
}
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/UpdateServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>UpdateServlet</servlet-name>
        <servlet-class>
            UpdateServlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>UpdateServlet</servlet-name>
        <url-pattern>/UpdateServlet</url-pattern>
    </servlet-mapping>
</web-app>

```

Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

```
http://localhost:8080/myjava/updateform.html
```

Jika Anda menekan Update !, UpdateServlet akan di-invoke dari browser :

CS-071-115

Praktikum melakukan delete di sebuah table di database ...

Prasyarat

Anda harus melakukan praktikum CS-071-111 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis deleteform.html, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

deleteform.html

```
<html>
<body>
    <form name=deleteForm action=DeleteServlet method=post>
        Id : <input type=text name=id>
        <br>
        <input type=submit value='Delete !'>
    </form>
</body>
</html>
```

Langkah ke-2

Tulis DeleteServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

DeleteServlet.java

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

public class DeleteServlet extends HttpServlet
{
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");

        Connection conn = null;
        try
        {
            String jdbcDriver
                = "org.gjt.mm.mysql.Driver";
            Class.forName(jdbcDriver);

            String url  = "jdbc:mysql://localhost:3306/MYAPP_DB";
            String user = "ekobs";
```

```

        String pwd = "j2ee";
        conn = DriverManager.getConnection(
            url, user, pwd);

        SimpleDateFormat format = new SimpleDateFormat("ddMMyyyy");

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

        String sqlDelete =
            "DELETE FROM EMPLOYEE_TBL WHERE ID = ?";
        PreparedStatement stmt = conn.prepareStatement(sqlDelete);

        stmt.setString(1, id);
        int count = stmt.executeUpdate();
        stmt.close();
        out.println("Query OK, " + count + " row affected");
    }
    catch(Exception e)
    {
        out.println(e.toString());
        e.printStackTrace();
    }
    finally
    {
        try
        {
            if(conn != null)
            {
                conn.close();
            }
        }
        catch(SQLException sqle)
        {
            sqle.printStackTrace();
        }
    }
    out.println("</body></html>");
}
}

```

Langkah ke-3

Compile ...

```
$ javac WEB-INF/classes/DeleteServlet.java
```

Langkah ke-4

Tulis web.xml yang baru ...

WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>DeleteServlet</servlet-name>
        <servlet-class>
            DeleteServlet
        </servlet-class>
    </servlet>

```

```
</servlet>

<servlet-mapping>
    <servlet-name>DeleteServlet</servlet-name>
    <url-pattern>/DeleteServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Langkah ke-5

Di log Tomcat akan terpantau proses di mana Context myjava mengalami auto re-deployment.

Langkah ke-6

Servlet telah di-deploy di atas Tomcat web application server, siap melayani request dari browser. melalui URL :

`http://localhost:8080/myjava/insertform.html`

Jika Anda menekan Insert !, InsertServlet akan di-invoke dari browser :

12 – HttpSession

Obyek HttpSession bisa digunakan untuk memegang conversational state dengan sebuah client tertentu. Di dalam obyek HttpSession bisa disimpan user information, shopping cart dll.

Untuk mendapatkan obyek HttpSession, Anda memanggil method getSession() dari obyek HttpServletRequest.

Untuk menyimpan obyek di HttpSession Anda bisa meng-invoke method setAttribute().

Untuk membaca obyek di HttpSession Anda bisa meng-invoke method getAttribute()

Untuk me-remove obyek dari HttpSession Anda bisa meng-invoke method removeAttribute().

Dan Anda bisa meng-invalidate obyek HttpSession dengan method invalidate().

13 – ServletContext

Obyek ServletContext hanya ada satu untuk setiap web application atau Context.

14 – Java Server Pages

Menulis JSP berbeda dengan menulis source code Java. Pada dasarnya menulis JSP adalah menulis HTML dengan menambahkan kode-kode dalam bahasa pemrograman Java. Kode Java di HTML digunakan untuk presentation logic. Kode-kode ini ditambahkan melalui directive, declaration, scriptlet dan expression.

Source code JSP disimpan sebagai file berekstension .jsp. Oleh web application server, JSP akan di-rewrite menjadi Servlet, di-compile dan selanjutnya akan diperlakukan sebagaimana Servlet.

Saat di-rewrite menjadi Servlet, akan digunakan PrintWriter untuk menuliskan semua HTML dari JSP ini, sebagai bagian dari HTTP response ke client. Sedangkan kode Java, akan menjadi bagian dari Servlet tersebut.

CS-071-141

Praktikum dengan JSP sederhana ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis `salamkeadilan.jsp`, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah `/home/myjava`

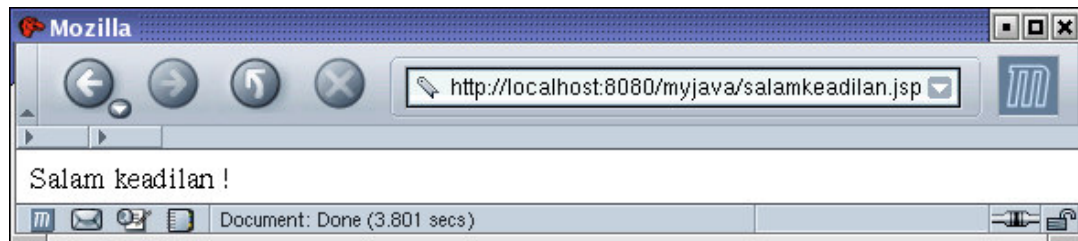
`salamkeadilan.jsp`

```
<html><body>  
Salam keadilan !  
</body></html>
```

Langkah ke-2

JSP siap melayani request dari browser. melalui URL :

`http://localhost:8080/myjava/salamkeadilan.jsp`



CS-071-142

Praktikum dengan JSP sederhana ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis date.jsp, simpan di-directory yang sudah dipersiapkan, dalam contoh adalah /home/myjava

date.jsp

```
<%@ page import=' java.text.SimpleDateFormat' %>
<%@ page import=' java.util.Date' %>

<%
    Date date = new Date();

    SimpleDateFormat format
    = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");
%>
<html><body>

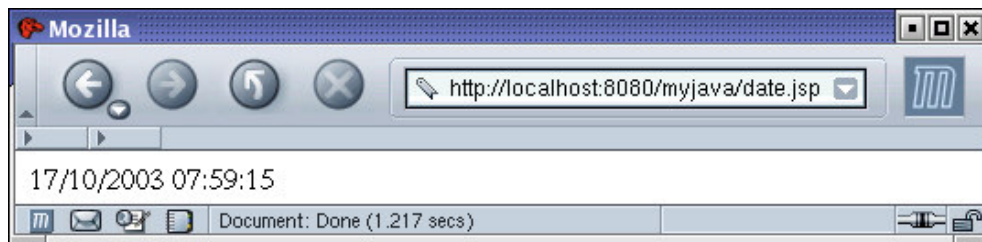
<%= format.format(date) %>

</body></html>
```

Langkah ke-2

JSP siap melayani request dari browser. melalui URL :

<http://localhost:8080/myjava/date.jsp>



15 – Directive

Directive bisa digunakan untuk mendeklarasikan import package, include dokumen, dan error page.

Melalui directive, Anda bisa mendeklarasikan package-package yang dibutuhkan di dalam JSP Anda.

Melalui directive, Anda bisa juga meng-include dokumen lain, berupa JSP, HTML atau text.

Melalui directive, Anda juga bisa mendeklarasikan sebuah JSP yang ditetapkan sebagai error page, yaitu halaman yang digunakan untuk menampilkan error.

16 – Declaration, Scriptlet dan Expression

Declaration ditulis di antara `<%! ... %>` . Di dalam declaration, dapat di-deklarasikan variable, method dan method. Variable dan method ini akan menjadi bagian dari Java class yang di-generate dari JSP.

Scriptlet ditulis di antara `<% ... %>` . Di dalam scriptlet, dapat di-deklarasikan, statement-statement. Statement-stament ini akan menjadi bagian dari method service() dari Servlet.

Expression ditulis di antara `<%= ... %>` . Di dalam expression bisa dituliskan variable, yang akan diubah menjadi String dan menjadi bagian dari response ke client.

17 – Implicit Variable

Terdapat beberapa variable, yang dapat digunakan tanpa perlu mendeklasikan terlebih dahulu. Ini dikenal sebagai implicit variable, yaitu request, response, out, session , application, pageContext, config, page dan exception. Implicit variable ini bisa digunakan dari dalam scriptlet dan expression.

Dalam teknologi Java, saat JSP di-rewrite menjadi Servlet, variable-variable ini akan di-deklarasikan dalam method service().

Variable request adalah sebuah obyek HttpServletRequest. Melalui request, Anda bisa membaca parameter-parameter.

Variable response adalah sebuah obyek HttpServletResponse. Melalui response, Anda bisa menuliskan response ke client.

Variable out adalah sebuah obyek PrintWriter.

Variable session adalah sebuah obyek HttpSession.

Variable application adalah sebuah obyek ServletContext.

...

18 – include dan forward

Saat sebuah Servlet melayani sebuah request dari client, dalam response yang dikembalikan ke client, Servlet tsb bisa meng-include response dari Servlet lain. Untuk ini digunakan obyek `RequestDispatcher`, dengan memanggil method `include()`.

Servlet juga bisa mem-forward request dari client ke Servlet lain, dan menyerahkan kepada Servlet lain tsb untuk mengembalikan response ke client. Untuk ini juga digunakan obyek `RequestDispatcher`, dengan memanggil method `forward()`.

Hal yang sama bisa dilakukan dengan JSP. Sebuah JSP dapat meng-include JSP lain. Untuk ini digunakan action tag `include`.

Dan JSP juga bisa men-forward ke JSP lain. Untuk ini digunakan action tag `forward`.

19– JavaServlet vs JSP

JavaServlet dan JSP mempunyai kemampuan yang kembar. Apa yang bisa dilakukan dengan JSP bisa dilakukan dengan JavaServlet. Anda bisa membaca form yang di-submit user menggunakan JSP atau Servlet. Anda bisa mengakses database melalui JSP atau Servlet. Anda bisa menulis HTML dengan JSP atau Servlet.

Tetapi JavaServlet dan JSP mempunyai kelebihan masing-masing. Lebih mudah menulis HTML di dalam JSP dibanding di dalam Servlet. Servlet bisa mengembalikan response dalam format binary, seperti graphics, sedangkan JSP tidak.

Best practise yang direkomendasikan adalah memadukan Servlet dan JSP. Servlet digunakan untuk menangani workflow, membaca parameter, mengakses database dan mengolah data, sedangkan JSP bertanggung jawab untuk menuliskan HTML.

20 – Model View Controller

Dalam arsitektur Model-View-Controller, sebuah aplikasi dipecah secara fungsional ke dalam 3 komponen utama, yaitu Model, View dan Controller. Model merepresentasikan business obyek yang ditangani, diolah atau ditampilkan di dalam aplikasi. View bertanggung jawab untuk menampilkan model, dan secara umum sebagai User Interface. Controller bertanggung jawab untuk mengendalikan alur kerja dari aplikasi.

Dalam teknologi Java, Model diimplementasikan dengan Java Bean, serta Helper classes yang bertanggung jawab untuk melakukan pengolahan data, maupun interaksi dengan database. View lumrahnya diimplementasikan sebagai Java Server Pages. Sedangkan Controller lumrahnya diimplementasikan dengan Java Servlet.

21 – Security

Sebuah web application dapat dilindungi dalam satu security realm. Di mana Servlet atau JSP hanya bisa diakses oleh user dengan role tertentu.

CS-071-191

Praktikum mengamankan Servlet dengan menerapkan security constraints ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis SalamKeadilanServlet.java, simpan di-directory yang sudah dipersiapkan WEB-INF/classes di bawah /home/myjava

SalamKeadilanServlet.java

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

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

        out.println("<html><body>");
        out.println("Salam keadilan !");
        out.println("</body></html>");
    }
}
```

Langkah ke-2

Compile ...

```
$ javac WEB-INF/classes/SalamKeadilanServlet.java
```

Langkah ke-3

Tulis web.xml yang baru ...

WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <servlet-class>
            SalamKeadilanServlet
        </servlet-class>
    </servlet>
```



```

<servlet-mapping>
  <servlet-name>SalamKeadilanServlet</servlet-name>
  <url-pattern>/SalamKeadilanServlet</url-pattern>
</servlet-mapping>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Salam Keadilan</web-resource-name>
    <url-pattern>/SalamKeadilanServlet</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>consultant</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Salam Keadilan</realm-name>
</login-config>
</web-app>

```

Langkah ke-4

Un-comment baris tentang MemoryRealm di TOMCAT_HOME/conf/server.xml. Caranya, ubah dari :

```

<!--
  <Realm className="org.apache.catalina.realm.MemoryRealm" />
-->

```

menjadi :

```

  <Realm className="org.apache.catalina.realm.MemoryRealm" />

```

Langkah ke-5

Tulis tomcat-users.xml, simpan di TOMCAT_HOME/conf/

tomcat-users.xml

```

<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="executive"/>
  <role rolename="engineer"/>
  <role rolename="consultant"/>
  <user username="eko" password="j2ee" roles="executive"/>
  <user username="fikri" password="oracle9i" roles="engineer"/>
  <user username="millati" password="j2me" roles="consultant"/>
</tomcat-users>

```

Langkah ke-6

Stop dan kemudian start kembali Tomcat ...

Langkah ke-7

Servlet telah di-deploy di atas Tomcat web application server dalam pengamanan ...

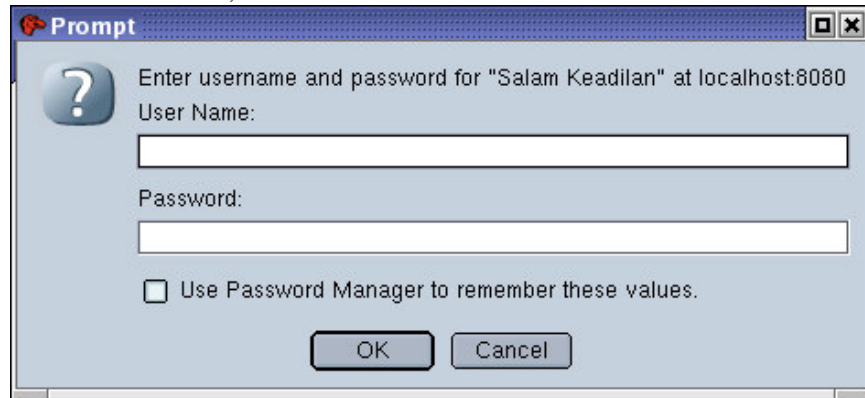
Langkah ke-8

Buka dari browser URL :

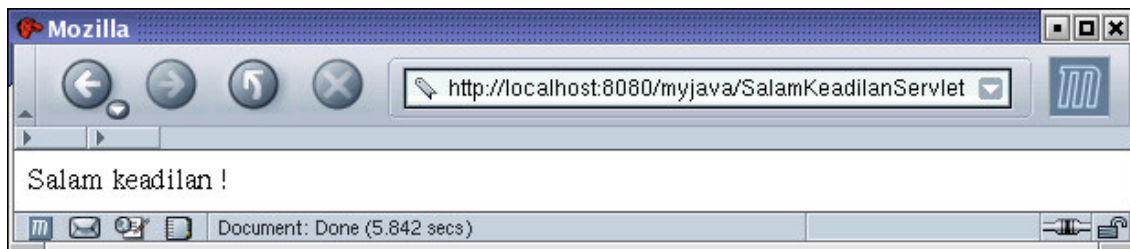
<http://localhost:8080/myjava/SalamKeadilanServlet>

Anda akan mendapatkan form untuk memasukkan User dan Password :

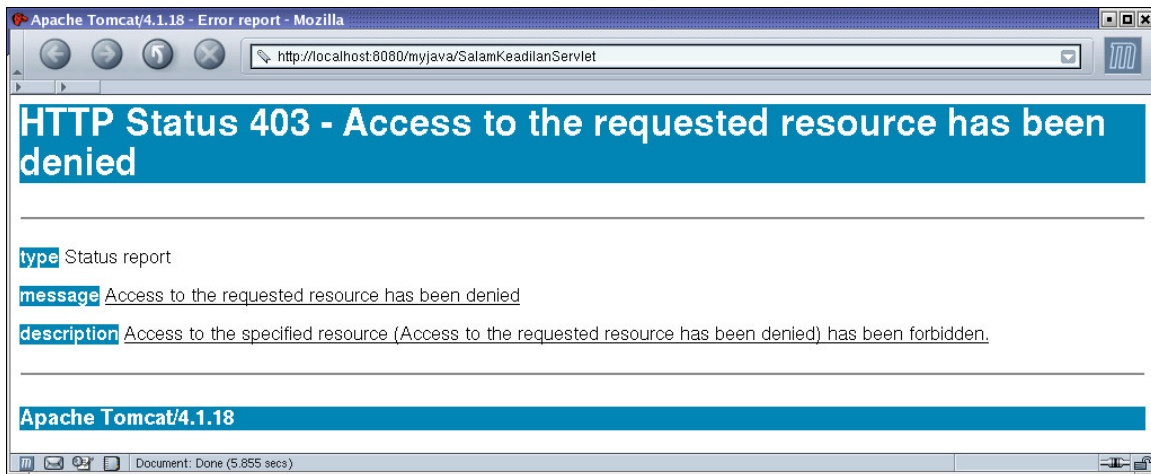
Jika Anda menekan Click me !, SalamKeadilanServlet akan di-invoke dari browser :



Jika Anda memasukkan millati/j2me maka akan mendapatkan :



Jika memasukkan user yang tidak mempunyai hak :



CS-071-191

Praktikum mengamankan Servlet dengan menerapkan security constraints ...

Prasyarat

Anda harus melakukan praktikum CS-071-011 terlebih dahulu ...

Langkah

Langkah ke-1

Tulis `SalamKeadilanServlet.java`, simpan di-directory yang sudah dipersiapkan `WEB-INF/classes` di bawah `/home/myjava`

`SalamKeadilanServlet.java`

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

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

        out.println("<html><body>");
        out.println("Salam keadilan !");
        out.println("</body></html>");
    }
}
```

Langkah ke-2

Compile ...

```
$ javac WEB-INF/classes/SalamKeadilanServlet.java
```

Langkah ke-3

Tulis `web.xml` yang baru ...

`WEB-INF/web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>SalamKeadilanServlet</servlet-name>
        <servlet-class>
            SalamKeadilanServlet
        </servlet-class>
    </servlet>
</web-app>
```

```

        <servlet-name>SalamKeadilanServlet</servlet-name>
        <url-pattern>/SalamKeadilanServlet</url-pattern>
    </servlet-mapping>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Salam Keadilan</web-resource-name>
            <url-pattern>/SalamKeadilanServlet</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>consultant</role-name>
        </auth-constraint>
    </security-constraint>

    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Salam Keadilan</realm-name>
    </login-config>
</web-app>

```

Langkah ke-4

Un-comment baris tentang MemoryRealm di TOMCAT_HOME/conf/server.xml. Caranya, ubah dari :

```

<!--
    <Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
        driverName="org.gjt.mm.mysql.Driver"
        connectionURL="jdbc:mysql://localhost/authority"
        connectionName="test" connectionPassword="test"
        userTable="users" userNameCol="user_name"
userCredCol="user_pass"
        userRoleTable="user_roles" roleNameCol="role_name" />
-->

```

menjadi :

```

    <Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
        driverName="org.gjt.mm.mysql.Driver"
        connectionURL="jdbc:mysql://localhost:3306/MYAPP_DB"
        connectionName="ekobs" connectionPassword="j2ee"
        userTable="USER_TBL" userNameCol="USER_NAME"
userCredCol="PASSWORD"
        userRoleTable="ROLE_TBL" roleNameCol="ROLE_NAME" />

```

Langkah ke-5

Tulis myAuth.sql untuk meng-create dan meng-insert data :

myAuth.sql

```

create table USER_TBL(USER_NAME varchar(20), PASSWORD varchar(20));
create table ROLE_TBL(USER_NAME varchar(20), ROLE_NAME varchar(20),

insert into USER_TBL values('sigitwk', 'php');
insert into ROLE_TBL values('sigitwk', 'consultant');

insert into USER_TBL values('arisp', 'cisco');
insert into ROLE_TBL values('arisp', 'entrepreneur');
commit;

```

Langkah ke-6

Buka sebuah terminal. Lalu melalui terminal tersebut jalankan mysql client untuk meng-execute myDomain.sql

```
$ mysql -uekobs -pj2ee MYAPP_DB < myAuth.sql
```

Langkah ke-7

Stop dan kemudian start kembali Tomcat ...

Langkah ke-8

Servlet telah di-deploy di atas Tomcat web application server dalam pengamanan ...

Langkah ke-9

Buka dari browser URL :

```
http://localhost:8080/myjava/SalamKeadilanServlet
```

Anda akan mendapatkan form untuk memasukkan User dan Password.