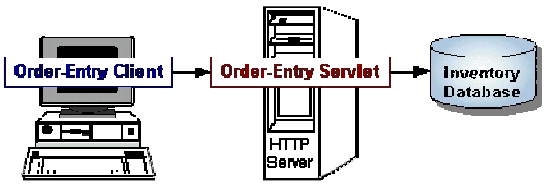
Servlets

Servlets

Servlets are modules that extend Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.



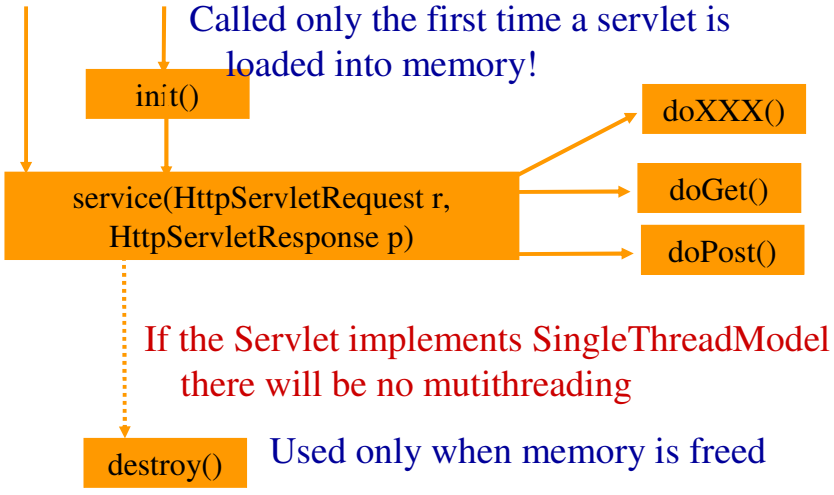
Servlets are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface.

For a full tutorial, see
http://java.sun.com/docs/books/tutorial/servlets/overview/index.html

Applets vs. Servlets

	Applet	Servlet
Runs on:	Client	Server
Has a main:	NO	NO
Extends:	java.applet.Applet	javax.servlet.http. HttpServlet
Graphics	SI	NO
Hearth:	handleEvent()	service()

Servlet Lifecycle



Get vs Post

What are "Get" and "Post"?

Get and Post are methods used to send data to the server:
With the **Get** method, the browser appends the data onto the URL.
With the **Post** method, the data is sent as "standard input."

Why Do I Care?

It's important for you to know which method you are using. The Get method is the default, so if you do not specify a method, the Get method will be used automatically.

The Get method has several disadvantages:

- There is a limit on the number of characters which can be sent to the server, generally around 100 - 150 characters.
- Your user will see the "messy codes" when the data is sent.

service()

This code is part of the class HttpServlet

```
protected void service (HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
    {
     String method = req.getMethod ();
     if (method.equals ("GET")) {
        long       ifModifiedSince;  long       lastModified;    long      now;
        ifModifiedSince = req.getDateHeader ("If-Modified-Since");
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
        else {
            now = System.currentTimeMillis ();
            if (now < ifModifiedSince || ifModifiedSince < lastModified)
                doGet (req, resp);
           else
                resp.sendError (HttpServletResponse.SC_NOT_MODIFIED);
        }
```

service()

This code is part of the class HttpServlet

```
protected void service (HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
    {
     String method = req.getMethod ();
     if (method.equals ("GET")) {
        long       ifModifiedSince;  long       lastModified;   long     now;
        ifModifiedSince = req.getDateHeader ("If-Modified-Since");
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        if (ifModifiedSince == -1 || lastModified == -1) doGet (req, resp);
        else {
            now = System.currentTimeMillis ();
            if (now < ifModifiedSince || ifModifiedSince < lastModified)
                doGet (req, resp);
           else
               resp.sendError (HttpServletResponse.SC_NOT_MODIFIED);
        }
```

service()

```
    } else if (method.equals ("HEAD")) {
        long       lastModified;
        lastModified = getLastModified (req);
        maybeSetLastModified (resp, lastModified);
        doHead (req, resp);
    } else if (method.equals ("POST")) {
        doPost (req, resp);
    } else if (method.equals ("PUT")) {
        doPut(req, resp);
    } else if (method.equals ("DELETE")) {
        doDelete(req, resp);
    } else if (method.equals ("OPTIONS")) {
        doOptions(req,resp);
    } else if (method.equals ("TRACE")) {
        doTrace(req,resp);
    } else {
        resp.sendError (HttpServletResponse.SC_NOT_IMPLEMENTED,
            "Method '" + method + "' is not defined in RFC 2068");
    }
}
```

A taste of servlet programming-1

```
public class SimpleServlet extends HttpServlet {
/** Handle the HTTP GET method by building a simple web page.
   */
   public void doGet (HttpServletRequest request,
                      HttpServletResponse response)        throws
                            ServletException, IOException {

      PrintWriter out;
      String title = "Simple Servlet Output";
```

A taste of servlet programming-2

```
      // set content type and other response header fields first

      response.setContentType("text/html");
      // then write the data of the response
      out = response.getWriter();
      out.println("<HTML><HEAD><TITLE>");
      out.println(title);
      out.println("</TITLE></HEAD><BODY>");
      out.println("<H1>" + title + "</H1>");
      out.println("<P>This is output from
          SimpleServlet.");
      out.println("</BODY></HTML>");
      out.close();
   }
}
```

Forms (a recall)

See also: http://www.cs.tut.fi/~jkorpela/forms/

Accessibility

See http://jimthatcher.com/webcourse8.htm for accessibility when using forms

http://jimthatcher.com/webcourse1.htm for accessibility in general.

Forms

Give to the user the possibility to di send information to the Web server

The FORM tag defines a form and has the following attributes:

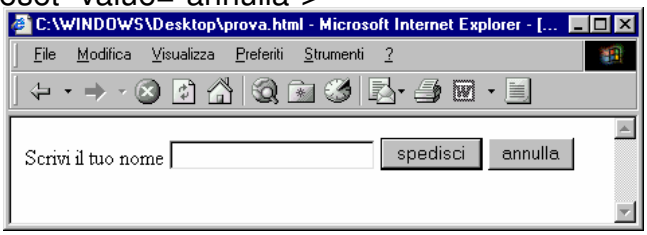
- •ACTION identifies the processing engine
- •ENCTYPE specificies the MIME type used to pass data to the server (Es. Text/html)

FORM contains the sub-tag:

- •several tags for collecting data
- •An INPUT tag must be of type SUBMIT for sending the data
- •An INPUT can be of tye RESET to cancel all the gathered data

Form - input

```
<FORM method="POST" action="/cgi-bin/elabora">
  Scrivi il tuo nome
  <Input type="text" size“=25" maxlength="15“ name=“a”>
  <Input type="submit" value="spedisci">
  <Input type="reset" value="annulla">
</FORM>
```



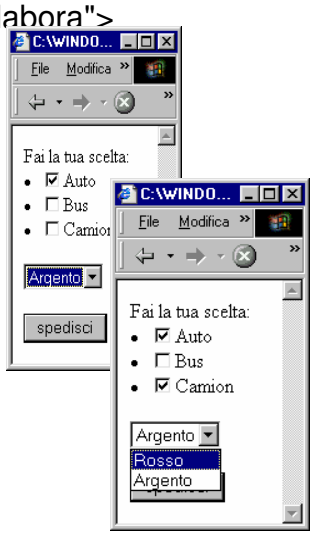
Sends a url of type
http://…/cgi-bin/elabora?a=MarcoRonchetti&b=…

Form – input type="radio"

```
<FORM method="POST" action="/cgi-bin/elabora">
  Fai la tua scelta:
  <LI><Input type="radio" name="tipo"
  value="auto" checked>Auto
  <LI><Input type="radio" name="tipo"
  value="bus">Bus
  <LI><Input type="radio" name="tipo"
  value="camion">Camion
  <P><Input type="radio" name="colore"
  value="rosso">Rosso
  <Input type="radio" name="colore"
   value="argento" checked>Argento</P>
  <Input type="submit" value="spedisci">
</FORM>
```

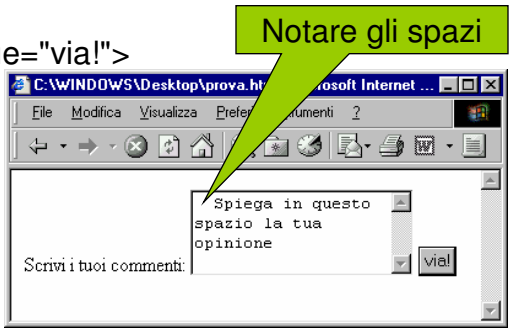
Form – input type="checkbox" - select

```
<FORM method="POST" action="/cgi-bin/elabora">
   Fai la tua scelta:
  <LI><Input type="checkbox"
  name="tipo" value="auto" checked>Auto
  <LI><Input type="checkbox"
  name="tipo" value="bus">Bus
  <LI><Input type="checkbox"
  name="tipo" value="camion">Camion
  <P><Select name="colore">
  <option>Rosso
  <option selected>Argento
  </select></P>
  <Input type="submit" value="spedisci">
</FORM>
```



Form – textarea

```
<FORM method="POST" action="/cgi-bin/elabora">
   Scrivi i tuoi commenti:
  <Textarea
  name="commenti" rows="4" columns="14">
 Spiega in questo spazio la tua opinione
  </TEXTAREA>
  <Input type="submit" value="via!">
</FORM>
```



Example

Esempio: ShowParameters

```
package coreservlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class ShowParameters extends HttpServlet {
  public void doGet(HttpServletRequest request HttpServletResponse
    response) throws ServletException, IOException {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   String title = "Reading All Request Parameters";
   out.println ("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>"
    +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
            "<TABLE BORDER=1 ALIGN=CENTER>\n" +
            "<TR BGCOLOR=\"#FFAD00\">\n" +
            "<TH>Parameter Name<TH>Parameter Value(s)");
```

Esempio: ShowParameters

```
   Enumeration paramNames = request.getParameterNames();
   while(paramNames.hasMoreElements()) {
     String paramName = (String)paramNames.nextElement();
     out.print("<TR><TD>" + paramName + "\n<TD>");
     String[] paramValues = request.getParameterValues(paramName);
     if (paramValues.length == 1) {
       String paramValue = paramValues[0];
       if (paramValue.length() == 0) out.println("<I>No Value</I>");
       else out.println(paramValue);
     } else {
       out.println("<UL>");
       for(int i=0; i<paramValues.length; i++) {out.println("<LI>"
  +paramValues[i]);   }
       out.println("</UL>");
     }
   }
   out.println("</TABLE>\n</BODY></HTML>");
 }
```

Esempio: ShowParameters

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

Esempio: ShowParameters

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>A Sample FORM using POST </TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">A Sample FORM using POST</H1>

<FORM ACTION="/servlet/coreservlets.ShowParameters"
    METHOD="POST">
  Item Number: <INPUT TYPE="TEXT" NAME="itemNum"><BR>
  Quantity: <INPUT TYPE="TEXT" NAME="quantity"><BR>
  Price Each: <INPUT TYPE="TEXT" NAME="price" VALUE="$"><BR>
  <HR>
  First Name: <INPUT TYPE="TEXT" NAME="firstName"><BR>
  Last Name: <INPUT TYPE="TEXT" NAME="lastName"><BR>
  Middle Initial: <INPUT TYPE="TEXT" NAME="initial"><BR>
  Shipping Address:
  <TEXTAREA NAME="address" ROWS=3 COLS=40></TEXTAREA><BR>
```

Esempio: ShowParameters

```
Credit Card:<BR>
  <INPUT TYPE="RADIO" NAME="cardType"
  VALUE="Visa">Visa<BR>
  <INPUT TYPE="RADIO" NAME="cardType"
          VALUE="Master Card">Master Card<BR>
  <INPUT TYPE="RADIO" NAME="cardType"
          VALUE="Amex">American Express<BR>
  <INPUT TYPE="RADIO" NAME="cardType"
  VALUE="Discover">Discover<BR>
  <INPUT TYPE="RADIO" NAME="cardType"
          VALUE="Java SmartCard">Java SmartCard<BR>
Credit Card Number:
<INPUT TYPE="PASSWORD" NAME="cardNum"><BR>
Repeat Credit Card Number:
<INPUT TYPE="PASSWORD" NAME="cardNum"><BR><BR>
<CENTER><INPUT TYPE="SUBMIT" VALUE="Submit
  Order"></CENTER>
</FORM>
</BODY>
</HTML>
```

Accessibility

Accessibility

What is Section 508?
The legislation referred to as "Section 508" is actually an amendment to the Workforce Rehabilitation Act of 1973. The amendment was signed into law by President Clinton on August 7, 1998. Section 508 requires that electronic and information technology that is developed or purchased by the Federal Government is accessible by people with disabilities.

See http://jimthatcher.com/webcourse8.htm for accessibility when using forms

http://jimthatcher.com/webcourse1.htm for accessibility in general.

http://www.innovazione.gov.it/ita/normativa/pubblicazioni/2004_rapporto_comm_acc.pdf

Accessibility in Italy

Legge Stanca 9 gennaio 2004, n. 4

Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici

Testo della legge:
- http://www.pubbliaccesso.gov.it/normative/legge_20040109_n4.htm

Vedi anche:
- http://www.cnipa.gov.it/site/it-IT/Attivit%C3%A0/Commissioni_e_Gruppi_di_Lavoro_interministeriali/Accessibilit%C3%A0/

Rapporto 2004 della commissioneCommissione interministeriale permanente per l'impiego delle ICT a favore delle categorie deboli o svantaggiate
- **http://www.innovazione.gov.it/ita/normativa/pubblicazioni/2004_rapporto_comm_acc.pdf**

WebApps
(Tomcat configuration)

Static pages

To let Tomcat serve static pages, we must define a "Web Application".

That is, in the Tomcat Document Root (by default *$CATALINA_HOME/webapps/*) we must create a folder named after our Web Application (e.g. myApp).

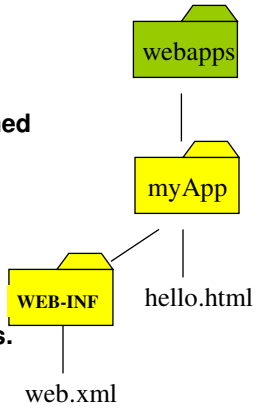
In that "myApp" folder, we MUST create a WEB-INF folder (that can be empy).

In the myApp folder we can then depost the static html files.

On our Tomcat server, the URL for the hello.html file becomes:

http://*machine*/*port*/myApp/hello.html

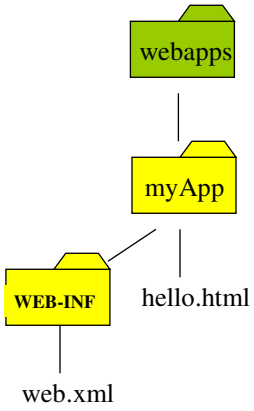
To actually see the webapp, we might have to restart Tomcat



Static pages

A web.xml file MUST be provided:

```
<?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>
</web-app>
```



JSP pages

To let Tomcat serve JSP pages, we follow the same procedure that we described for static pages.

In the myApp folder we can depost the JSP files.

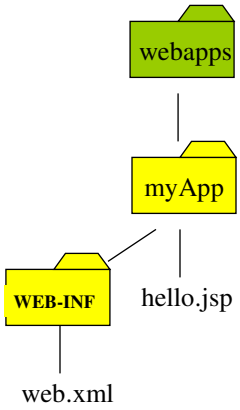
On our Tomcat server, the URL for the hello.jsp file becomes:

http://*machine*/*port*/myApp/hello.jsp

The WEB-INF directory is still empty.

To actually see the webapp, you might have to restart Tomcat (depending on the version you have)

The same web.xml file as in the static case must be provided.



Servlets

To let Tomcat serve servlet, we need add some info. The compiled servlets (.class) must be stored in a "classes" directory in WEB-INF.

Moreover, the web.xml file MUST contain at least:

```
<?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-mapping>
      <servlet-name>invoker</servlet-name>
      <url-pattern>/magic/*</url-pattern>
   </servlet-mapping>
</web-app>
```

The "magic" word is the servlet activation keyword (you can of course customize this word). To execute the servlet called MyServlet.class, the URL will be:

http://*machine*/*port*/myApp/magic/MyServlet

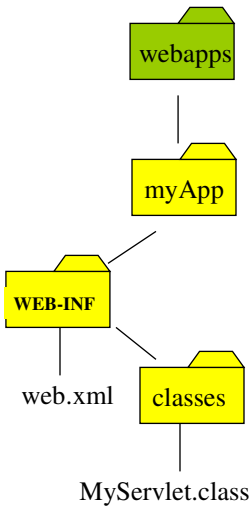
Servlets

The web.xml file CAN contain many additional info.
For instance, it can contain a section defining an alias name for the servlet:

```
…
   <servlet>
      <servlet-name>pippo</servlet-name>
      <servlet-class>Servlet1</servlet-class>
   </servlet>
…
```

In such case, the servlet called MyServlet.class Can be activated ALSO by the URL:

http://*machine*/*port*/myApp/magic/pippo



Dispatching, monitoring etc.

Dispatching

```
RequestDispatcher dispatch =
        cntx.getRequestDispatcher("/SecondServlet");
dispatch.forward(req,res);
```

```
RequestDispatcher dispatch =
        cntx.getRequestDispatcher("/SecondServlet");
dispatch.include(req,res);
```

Dispatching example

```
package servlets;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;

public class SecondServlet extends HttpServlet  {
   public void doGet(HttpServletRequest req,HttpServletResponse res)
   throws IOException,ServletException {
      Printer out=res.getWriter();
      System.out.println("Second Servlet Called");
   }
}
```

Dispatching example

```
package servlets;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.ServletConfig;
   import javax.servlet.ServletContext;
   import java.io.IOException;
   import javax.servlet.ServletException;
   import javax.servlet.ServletContext;
   import javax.servlet.RequestDispatcher;

public class FirstServlet extends HttpServlet  {
   public void doGet(HttpServletRequest req,HttpServletResponse res)
   throws IOException,ServletException {
      Printer out=res.getWriter();
      out.println("First Servlet Called");
      ServletConfig config = getServletConfig();
      ServletContext cntx = config.getServletContext();
      RequestDispatcher dispatch =
            cntx.getRequestDispatcher("/SecondServlet");
      dispatch.forward(req,res);
   }
}
```

Dispatching example

```
<servlet>
<servlet-name>FirstServlet</servlet-name>
<servlet-class>servlets.FirstServlet</servlet-class>
</servlet>

<servlet>
<servlet-name>SecondServlet</servlet-name>
<servlet-class>servlets.SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/firstservlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>SecondServlet</servlet-name>
<url-pattern>/SecondServlet/*</url-pattern>
</servlet-mapping>
```

Monitoring Servlets Lifecycle

Web context	Initialization and Destruction	ServletContextListener	ServletContextEvent
	Attribute added, removed, or replaced	ServletContextAttributeListener	ServletContextAttributeEvent
Session	Creation, invalidation, activation, passivation, and timeout	HttpSessionListenerHttpSessionActivationListener	HttpSessionEvent
	Attribute added, removed, or replaced	HttpSessionAttributeListener	HttpSessionBindingEvent
Request	A servlet request has started being processed by Web components	ServletRequestListener	ServletRequestEvent
	Attribute added, removed, or replaced	ServletRequestAttributeListener	ServletRequestAttributeEvent

Monitoring Servlets Lifecycle - Example

```
/* File : ApplicationWatch.java */
import javax.servlet.ServletContextListener;
import javax.servlet.ServletContextEvent;
public class ApplicationWatch implements
   ServletContextListener {
public static long applicationInitialized = 0L;
/* Application Startup Event */
public void contextInitialized(ServletContextEvent ce) {
   applicationInitialized = System.currentTimeMillis(); }
/* Application Shutdown Event */
public void contextDestroyed(ServletContextEvent ce) {}
}
```

Monitoring Servlets Lifecycle - Example

```
 /* File : SessionCounter.java */
import javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;
public class SessionCounter implements HttpSessionListener
   {
private static int activeSessions = 0;
/* Session Creation Event */
public void sessionCreated(HttpSessionEvent se) {
   activeSessions++; }
/* Session Invalidation Event */
public void sessionDestroyed(HttpSessionEvent se) {
   if(activeSessions > 0) activeSessions--; }
public static int getActiveSessions() { return
   activeSessions; }
}
```

Monitoring Servlets Lifecycle - Example

```
<!-- 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/j2ee/dtds/web-app_2.3.dtd">
<web-app>
<!-- Listeners -->
<listener>
   <listener-class> com.stardeveloper.web.listener.SessionCounter
   </listener-class>
</listener>
<listener>
   <listener-class>
   com.stardeveloper.web.listener.ApplicationWatch </listener-
   class>
</listener>
</web-app>
```

Scope Objects

Web context	ServletContext	Web components within web context servlet.getServletConfig().getServletContext
Session	HttpSession	Web components handling requests that belong to a session
Request	ServletRequest	Web component handling the request
Page	PageContext	Web component in the JSP page

Main Methods:
Object getAttribute(String name)
void setAttribute(String name, Object o)
Enumeration getAttributeNames()

Filters (javax.servlet.filter)

Other classes that preprocess/postprocess request/response

A filter is an object than perform filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the doFilter method. Every Filter has access to a FilterConfig object from which it can obtain its initialization parameters, a reference to the ServletContext which it can use, for example, to load resources needed for filtering tasks.

Filters are configured in the deployment descriptor of a web application

Examples that have been identified for this design are

1) **Authentication Filters**
2) **Logging and Auditing Filters**
3) **Image conversion Filters**
4) **Data compression Filters**
5) **Encryption Filters**
6) **Tokenizing Filters**
7) **Filters that trigger resource access events**
8) **XSL/T filters**
9) **Mime-type chain Filter**