

Web Technology

Programming Web Applications with Servlets

Klaus Ostermann, Uni Marburg

Based on slides by Anders Møller & Michael I. Schwartzbach

Objectives

- How to program Web applications using **servlets**
- Advanced concepts, such as listeners, filters, and request dispatchers
- Running servlets using the **Tomcat** server

Web Applications

- Web servers
 - return files
 - run programs
- Web application: collection of servlets, JSP pages, HTML pages, GIF files, ...
- Servlets: programmed using the servlet API, which is directly based on HTTP
- Notion of lifecycle, possibility to react to lifecycle events
- Different forms of state
 - application (shared state)
 - session (session state)
 - interaction (transient state)

An Example Servlet

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

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>ServletExample</title></head>" +
                    "<body><h1>Hello world!</h1>" +
                    "This page was last updated: " +
                    new java.util.Date() +
                    "</body></html>");
    }
}
```

Hello World!

This page was last updated: Fri Dec 24 19:38:23 CET 2004

Requests

- Methods in `HttpServletRequest`
 - `getHeader`
 - `getParameter`
 - `getInputStream`
 - `getRemoteHost`, `getRemoteAddr`, `getRemotePort`
 - ...

Example: HttpServletRequest (1/2)

```
public class Requests extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Requests</title></head><body>");
        out.println("<h1>Hello, visitor from "+request.getRemoteHost()+"</h1>");
        String useragent = request.getHeader("User-Agent");
        if (useragent!=null)
            out.println("You seem to be using "+useragent+"<p>");
        String name = request.getParameter("name");
        if (name==null)
            out.println("No <tt>name</tt> field was given!");
        else
            out.println("The value of the <tt>name</tt> field is: <tt>" +
                        htmlEscape(name) + "</tt>");
        out.println("</body></html>");
    }
}
```

Example: HttpServletRequest (2/2)

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

private String htmlEscape(String s) {
    StringBuffer b = new StringBuffer();
    for (int i = 0; i < s.length(); i++) {
        char c = s.charAt(i);
        switch (c) {
            case '<': b.append("&lt;"); break;
            case '>': b.append("&gt;"); break;
            case '"': b.append("&quot;"); break;
            case '\\': b.append("&backslash;"); break;
            case '&': b.append("&amp;"); break;
            default: b.append(c);
        }
    }
    return b.toString();
}
```

Hello, visitor from britney.widget.inc

You seem to be using Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.5) Gecko/20031007

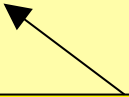
The value of the name field is: John Doe

Responses

- Methods in `HttpServletResponse`
 - `setStatus`
 - `addHeader`, `setHeader`
 - `getOutputStream`, `getWriter`
 - `setContentType`
 - `sendError`, `sendRedirect`
 - ...

Example: BusinessCardServlet

```
public class BusinessCardServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response)  
        throws IOException, ServletException {  
        response.setContentType("text/xml;charset=UTF-8");  
        long expires = new Date().getTime() + 1000*60*60*24;  
        response.addDateHeader("Expires", expires);  
        XMLOutputter outputter = new XMLOutputter();  
        outputter.output(getBusinessCard(),  
                        response.getOutputStream());  
    }  
    ...  
}
```



using JDOM to generate an XML
document with a reference to an
XSLT stylesheet

Servlet Contexts

- One `ServletContext` object for each Web application
- `getServerInfo`
- `getInitParameter`
- ...
- Shared state:
 - `setAttribute("name", value)`
 - `getAttribute("name")`
 - *don't use for mission critical data!*

Example: A Polling Service

A Web application consisting of

- QuickPollQuestion.html
- QuickPollSetup.java
- QuickPollAsk.java
- QuickPollVote.java
- QuickPollResults.java



QuickPoll

What is your question?

To be or not to be ?

Register my question



QuickPoll

To be or not to be?

☒ yes
☐ no

Vote



QuickPoll

To be or not to be?

Yes: 4
No: 1

Example: QuickPollQuestion.html

```
<html>
<head><title>QuickPoll</title></head>
<body>
<h1>QuickPoll</h1>
<form method=post action=setup>
what is your question?<br>
<input name=question type=text size=40>?<br>
<input type=submit name=submit
      value="Register my question">

</form>
</body>
</html>
```

A screenshot of the web form rendered from the HTML code. It features a title "QuickPoll" in a large, bold, black font. Below the title is the text "What is your question?". Underneath this text is a text input field containing the text "To be or not to be" followed by a question mark. At the bottom of the form is a submit button with the text "Register my question".

QuickPoll

What is your question?

To be or not to be ?

Register my question

Example: QuickPollSetup.java

```
public class QuickPollSetup extends HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException {
        String q = request.getParameter("question");
        ServletContext c = getServletContext();
        c.setAttribute("question", q);
        c.setAttribute("yes", new Integer(0));
        c.setAttribute("no", new Integer(0));
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>QuickPoll</title></head><body>" +
            "<h1>QuickPoll</h1>" +
            "Your question has been registered. " +
            "Let the vote begin!" +
            "</body></html>");
    }
}
```

Example: QuickPollAsk.java

```
public class QuickPollAsk extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>QuickPoll</title></head><body>" +
                  "<h1>QuickPoll</h1>" +
                  "<form method=post action=vote>");
        String question =
            (String)getServletContext().getAttribute("question");
        out.print(question+"?<p>");
        out.print("<input name=vote type=radio value=yes> yes<br>" +
                  "<input name=vote type=radio value=no> no<p>" +
                  "<input type=submit name=submit value=Vote>" +
                  "</form>" +
                  "</body></html>");
    }
}
```

QuickPoll

To be or not to be?

☒ yes
☐ no

Vote

Example: QuickPollVote.java (1/2)

```
public class QuickPollVote extends HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException {
        String vote = request.getParameter("vote");
        ServletContext c = getServletContext();
        if (vote.equals("yes")) {
            int yes = ((Integer)c.getAttribute("yes")).intValue();
            yes++;
            c.setAttribute("yes", new Integer(yes));
        } else if (vote.equals("no")) {
            int no = ((Integer)c.getAttribute("no")).intValue();
            no++;
            c.setAttribute("no", new Integer(no));
        }
    }
}
```

Example: QuickPollVote.java (2/2)

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.print("<html><head><title>QuickPoll</title></head><body>" +
        "<h1>QuickPoll</h1>" +
        "Thank you for your vote!" +
        "</body></html>");
    }
}
```


Example: QuickPollResult.java (1/2)

```
public class QuickPollResults extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        ServletContext c = getServletContext();
        String question = (String)c.getAttribute("question");
        int yes = ((Integer)c.getAttribute("yes")).intValue();
        int no = ((Integer)c.getAttribute("no")).intValue();
        int total = yes+no;
        response.setContentType("text/html");
        response.setDateHeader("Expires", 0);
        response.setHeader("Cache-Control",
                           "no-store, no-cache, must-revalidate");
        response.setHeader("Pragma", "no-cache");
        PrintWriter out = response.getWriter();
    }
}
```

Example: QuickPollResult.java (2/2)

```
out.print("<html><head><title>QuickPoll</title></head><body>" +
        "<h1>QuickPoll</h1>");
if (total==0)
    out.print("No votes yet...");
else {
    out.print(question + "?<p>" + "<table border=0>" +
        "<tr><td>Yes:<td>" + drawBar(300*yes/total) + "<td>" + yes +
        "<tr><td>No:<td>" + drawBar(300*no/total) + "<td>" + no +
        "</table>");
}
out.print("</body></html>");
}

String drawBar(int length) {
    return "<table><tr><td bgcolor=black height=20 width=\"" +
        length + "></td></tr></table>";
} }
```

QuickPoll

To be or not to be?

Yes:		4
No:		1

Problems in QuickPoll

- Need **access control** to QuickPollSetup
- No **escaping of special characters**
- Need to **check right order of execution**
- Need to check that expected **form field data** is present
- No **synchronization** in QuickPollVote
- Should store state in **database**
- **Redundancy** in HTML generation

Example: Shopping Cart

Widget Inc. Online Shopping

Item: Amount:

Your shopping cart now contains:

Item	Amount
heavy doodad	1
light gadget	430
small gizmo	15

Sessions

- One `HttpSession` object for each session
 - obtained by `getSession` in the `HttpServletRequest` object
- Session state:
 - `setAttribute("name", value)`
 - `getAttribute("name")`
- Hides the technical details of tracking users with URL rewriting / cookies / SSL sessions

Web Applications

A Web app is structured as a directory:

- *myapp/*
 - contains HTML/CSS/GIF/... files
- *myapp/WEB-INF/*
 - contains the *deployment descriptor* `web.xml`
- *myapp/WEB-INF/classes/*
 - contains servlet class files
(in subdirs corresponding to package names)
- *myapp/WEB-INF/lib/*
 - contains extra jar files

Deployment Descriptors

An XML file `web.xml` describing

- mapping from URIs to application resources
- initialization parameters
- security constraints
- registration of listeners and filters
- Mapping from HTTP error codes to resources

Example web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
          version="2.4">

  <display-name>A Small web Application</display-
  name>

  <servlet>
    <servlet-name>MyFirstServlet</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>MyFirstServlet</servlet-name>
    <url-pattern>/hello/*</url-pattern>
  </servlet-mapping>

</web-app>
```


The Tomcat Server

- Reference Implementation, Open Source
- `common/lib/servlet-api.jar`
- `bin/startup.sh`, `bin/shutdown.sh`
- `conf/server.xml`
- `webapps/myapp`

Advanced Features

- Listeners
- Filters and wrappers
- Request dispatchers
- Security

Listeners

- also called *observers* or *event handlers*
- **ServletContextListener**
 - Web application initialized / shut down
- **ServletRequestListener**
 - request handler starting / finishing
- **HttpSessionListener**
 - session created / invalidated
- **ServletContextAttributeListener**
 - context attribute added / removed / replaced
- **HttpSessionAttributeListener**
 - session attribute added / removed / replaced

Example: SessionMonitor (1/2)

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

public class SessionMonitor
    implements HttpSessionListener, ServletContextListener {
    private int active = 0, max = 0;

    public void contextInitialized(ServletContextEvent sce) {
        store(sce.getServletContext());
    }

    public void contextDestroyed(ServletContextEvent sce) {}

    public void sessionCreated(HttpSessionEvent se) {
        active++;
        if (active > max)
            max = active;
        store(se.getSession().getServletContext());
    }
}
```

Example: SessionMonitor (2/2)

```
public void sessionDestroyed(HttpSessionEvent se) {
    active--;
    store(se.getSession().getServletContext());
}

private void store(ServletContext c) {
    c.setAttribute("sessions_active", new Integer(active));
    c.setAttribute("sessions_max", new Integer(max));
}
}
```

Registration in web.xml:

```
<listener>
  <listener-class>SessionMonitor</listener-class>
</listener>
```

Filters

- Code being executed before and after the servlet
 - executed in stack-like fashion with servlet at the bottom
 - Can **intercept** and **redirect** processing
 - security
 - auditing
 - Can **modify requests and responses**
 - data conversion (XSLT, gzip, ...)
 - specialized caching
- *all without changing the existing servlet code!*

Example: LoggingFilter (1/2)

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

public class LoggingFilter implements Filter {
    ServletContext context;
    int counter;

    public void init(FilterConfig c) throws ServletException {
        context = c.getServletContext();
    }

    public void destroy() {}
}
```

Example: LoggingFilter (2/2)

```
public void doFilter(ServletRequest request,
                    ServletResponse response,
                    FilterChain chain)
    throws IOException, ServletException {
    String uri = ((HttpServletRequest)request).getRequestURI();
    int n = ++counter;
    context.log("starting processing request #" + n + " (" + uri + ")");
    long t1 = System.currentTimeMillis();
    chain.doFilter(request, response);
    long t2 = System.currentTimeMillis();
    context.log("done processing request #" + n + ", " + (t2 - t1) + " ms");
}
}
```


Registration of Filters in web.xml

```
<web-app ...>
  ...
  <filter>
    <filter-name>My Logging Filter</filter-name>
    <filter-class>LoggingFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>My Logging Filter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  ...
</web-app>
```

Wrappers

- Used by filters to modify requests and responses
- `HttpServletRequestWrapper`
- `HttpServletResponseWrapper`
- Example: performing *server-side* XSLT transformation for older browsers

Example: XSLTFilter (1/5)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.jdom.*;
import org.jdom.transform.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class XSLTFilter implements Filter {
    ServletContext context;

    public void init(FilterConfig c) throws ServletException {
        context = c.getServletContext();
    }

    public void destroy() {}
}
```

Example: XSLTFilter (2/5)

```
public void doFilter(ServletRequest request,
                    ServletResponse response,
                    FilterChain chain)
    throws IOException, ServletException {
    HttpServletRequest hreq = (HttpServletRequest)request;
    HttpServletResponse hresp = (HttpServletResponse)response;
    boolean client_capable =
        checkXSLTSupport(hreq.getHeader("User-Agent"));
    ServletResponse res;
    if (client_capable)
        res = response;
    else
        res = new BufferingResponseWrapper(hresp);
    chain.doFilter(request, res);
}
```

Example: XSLTFilter (3/5)

```
if (!client_capable) {
    try {
        hresp.setContentType("application/xhtml+xml");
        transform(((BufferingResponseWrapper)res).getReader(),
            response.getWriter());
    } catch (Throwable e) {
        context.log("XSLT transformation error", e);
        hresp.sendError(500, "XSLT transformation error");
    }
}

boolean checkXSLTSupport(String user_agent) {
    if (user_agent==null)
        return false;
    return
        user_agent.indexOf("MSIE 5.5")!=-1 ||
        user_agent.indexOf("MSIE 6")!=-1 ||
        user_agent.indexOf("Gecko")!=-1;
}
```

Example: XSLTFilter (4/5)

```
void transform(Reader in, Writer out)
    throws JDOMException, IOException {
    System.setProperty("javax.xml.transform.TransformerFactory",
                       "net.sf.saxon.TransformerFactoryImpl");
    SAXBuilder b = new SAXBuilder();
    Document d = b.build(in);
    List pi = d.getContent(new org.jdom.filter.ContentFilter
                           (org.jdom.filter.ContentFilter.PI));
    String xsl = ((ProcessingInstruction)(pi.get(0)))
                 .getPseudoAttributeValue("href");
    XSLTransformer t = new XSLTransformer(xsl);
    Document h = t.transform(d);
    (new XMLOutputter()).output(h, out);
}
}
```

Example: XSLTFilter (5/5)

```
class BufferingResponseWrapper extends HttpServletResponseWrapper {
    CharArrayWriter buffer;
    PrintWriter writer;

    public BufferingResponseWrapper(HttpServletResponse res) {
        super(res);
        buffer = new CharArrayWriter();
        writer = new PrintWriter(buffer);
    }

    public PrintWriter getWriter() {
        return writer;
    }

    Reader getReader() {
        return new CharArrayReader(buffer.toCharArray());
    }
}
```

Request Dispatchers

- Forwarding requests to other resources

```
RequestDispatcher disp = context.getRequestDispatcher  
    ("/shop/buy") ;  
disp.forward(req, resp) ;
```

- Often used with JSP...
- Alternative: sendRedirect method
 - Advantage: no client involvement
 - Disadvantage: Can only redirect on the same server

Security – Roles and Authentication

```
<web-app ...>
  ...
  <security-role>
    <role-name>administrator</role-name>
    <role-name>teacher</role-name>
    <role-name>student</role-name>
  </security-role>

  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Administration</realm-name>
  </login-config>
  ...
</web-app>
```

Security Constraints

```
...  
<security-constraint>  
  <web-resource-collection>  
    <web-resource-name>Restricted Area</web-resource-name>  
    <url-pattern>/restricted/*</url-pattern>  
    <http-method>GET</http-method>  
    <http-method>POST</http-method>  
  </web-resource-collection>  
  <auth-constraint>  
    <role-name>administrator</role-name>  
    <role-name>teacher</role-name>  
  </auth-constraint>  
  <user-data-constraint>  
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>  
  </user-data-constraint>  
</security-constraint>  
...
```

Programmatic Security

Useful request methods:

- `getRemoteUser()`
- `isUserInRole(String role)`
- `isSecure()`
- `getAuthType()`
- `getAttribute("javax.servlet.request.x509Certificate")`

Summary (1/2)

- Servlets closely follow the **request-response** pattern from HTTP
- Features:
 - Multi-threading
 - Declarative configuration
 - Request parsing, including decoding of form data
 - Shared state
 - Session management
 - Advanced code structuring: listeners, filters, wrappers
 - Client authentication, SSL

Summary (2/2)

- Limitations of Servlets
 - Dynamic construction of HTML low-level
 - No compile-time guarantees that the result is well-formed or valid
 - Possibility of CSS attacks
 - Redundancy in HTML code
 - Difficult to separate HTML code and “business logic”
 - Control flow through a session can be difficult to understand
 - Logical flow of a session is severely distorted
 - Untyped “getAttribute/setAttribute” is another source of errors

Essential Online Resources

- The servlet API:
<http://jakarta.apache.org/tomcat/tomcat-5.5-doc/servletapi/>
- Sun's home page for servlets:
<http://java.sun.com/products/servlet/>
- The Tomcat server:
<http://jakarta.apache.org/tomcat/>