An Introduction to XML and Web Technologies

Programming Web Applications with Servlets

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
- Lifecycles
  - 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()+

  }
}
```

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 '>
        case '"
        case '\
        case '&
        default:
    } }
    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

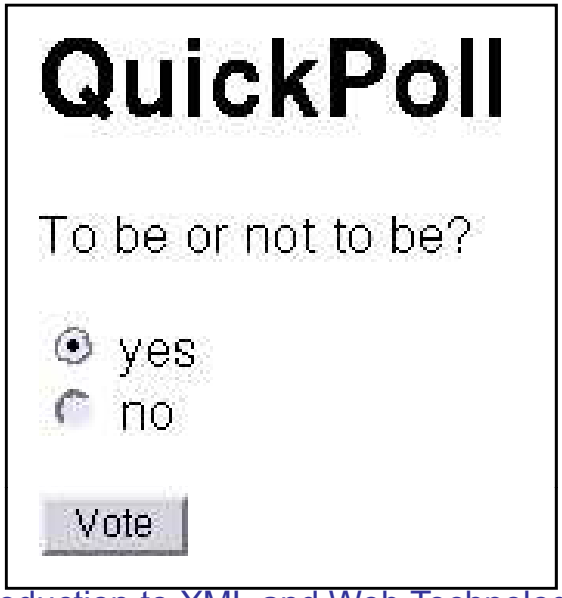
Servlet Contexts

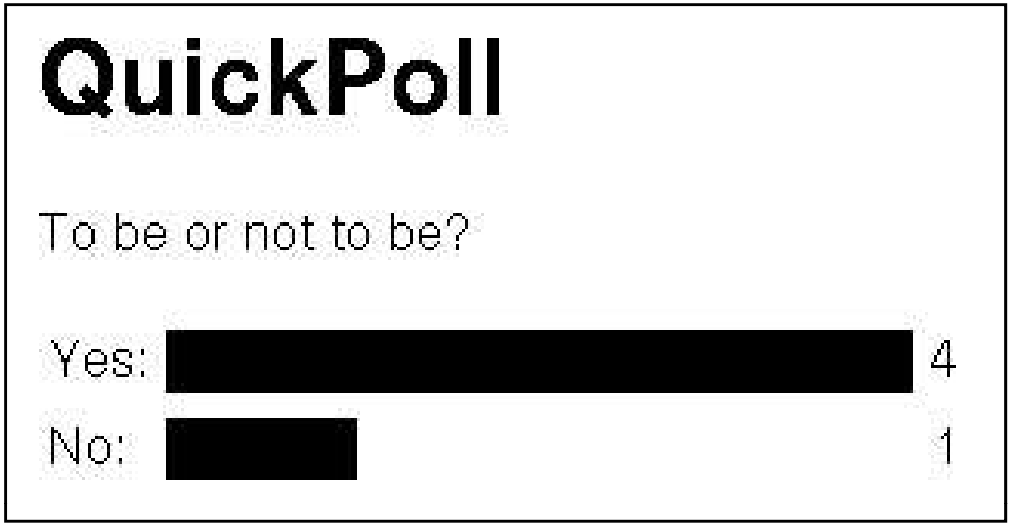
- One `ServletContext` object for each Web application
- `getServerInfo`
- `getInitParameter`
- …
- Shared state:
  - `setAttribute("`*name*`", `*value*`)`
  - `getAttribute("`*name*`")`
  - *don't use for mission critical data!*

Example: A Polling Service

A Web application consisting of

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





Example: QuickPollQuestion.html

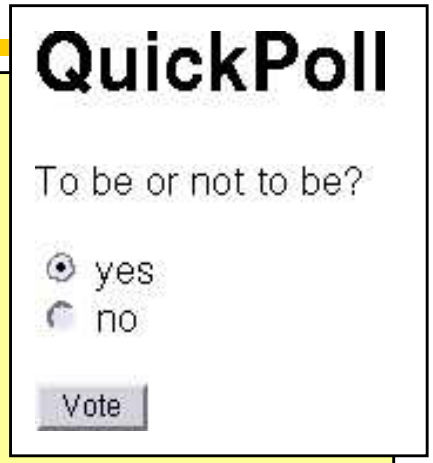
```
<html>
<head><title>QuickPoll</title></head>
<body>
<h1>QuickPoll</h1>
<form method=post action=setup>
What is your question?<br>
<input name=question type=text size=40>?<br>
<input type=submit name=submit
       value="Register my question">
</form>
</body>
</html>
```

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>");
} }
```



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+"></table>";
} }
```

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: Add to shopping cart

Your shopping cart now contains:

Item	Amount
heavy doodad	1
light gadget	430
small gizmo	15

Proceed to cashier

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

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
- Often used with JSP...

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

- 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

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/

Limitations of Servlets

- Low-level **construction of HTML documents**
  - fragments (strings) written to output stream
  - no static well-formedness/validity guarantees

- Low-level **session management**
  - control-flow is often unclear
  - no enforcement of relation between showing a page and receiving form input
  - primitive session state management

JWIG

- Research project (`http://www.jwig.org/`)
- **Session threads**
  - showing a page and receiving form input modeled as a Remote Procedure Call (RPC)
    - explicit control-flow
    - simpler session state management
- **Template**-based page construction using XACT
- **Static checking** of
  - output validity
  - form field consistency

Example: The Guessing Game in JWIG

Please guess a number between 1 and 100:

42 continue

You got it, using **428** guesses.

That makes you the new record holder!

Please enter your name for the hi-score list:

John Doe continue

In 2 plays of this game, the record holder is **John Doe** with **428** guesses.

GuessingGameWrapper.xml

```
<html>
<head><title>The Guessing Game</title></head>
<body bgcolor="aqua">**<[BODY]>**</body>
</html>
```

GuessingGame (1/5)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.jwig.*;
import dk.brics.xact.*;

public class GuessingGamePlay extends SessionThread {
  public XML main() throws IOException, ServletException {
    XML wrapper = XML.loadConstant("GuessingGameWrapper.xml");
    XML form = [[
      <form><input name="guess" type="text" size="2" maxlength="2"/>
      <input type="submit" name="continue" value="continue"/></form>
    ]];
```

GuessingGame (2/5)

```
ServletContext c = getServletContext();
Integer plays = (Integer)c.getAttribute("plays");
if (plays==null)
  plays = new Integer(0);
else
  plays = new Integer(plays.intValue()+1);
c.setAttribute("plays", plays);
int number = (new Random()).nextInt(100)+1;

show(wrapper.plug("BODY",
  [[Please guess a number between 1 and 100: <{form}>]]));
```

GuessingGame (3/5)

```
int guesses = 1;
boolean done = false;
while (!done) {
  int guess = Integer.parseInt(getParameter("guess"));
  if (guess==number)
    done = true;
  else {
    show(wrapper.plug("BODY", [[
      That is not correct. Try a
      <b><{(guess>number)?"lower":"higher"}></b> number: <{form}>
    ]]));
    guesses++;
  }
}
```

GuessingGame (4/5)

```
  XML msg = [[You got it, using <b><{guesses}></b> guesses.]];
  XML thanks = [[Thank you for playing this exciting game!]];
  XML res;
  if (guesses<getCurrentRecord()) {
    show(wrapper.plug("BODY", [[
      <{msg}><p/>
      That makes you the new record holder!<p/>
      Please enter your name for the hi-score list:
      <form><input name="name" type="text" size="20"/>
      <input type="submit" name="continue" value="continue"/></form>
    ]]));
    synchronized(c) {
      if (guesses<getCurrentRecord()) {
        c.setAttribute("holder", getParameter("name"));
        c.setAttribute("record", new Integer(guesses));
      }
    }
    res = wrapper.plug("BODY", thanks);
  } else
    res = wrapper.plug("BODY", [[<{msg}><p/><{thanks}>]]);
  return res;
}
```

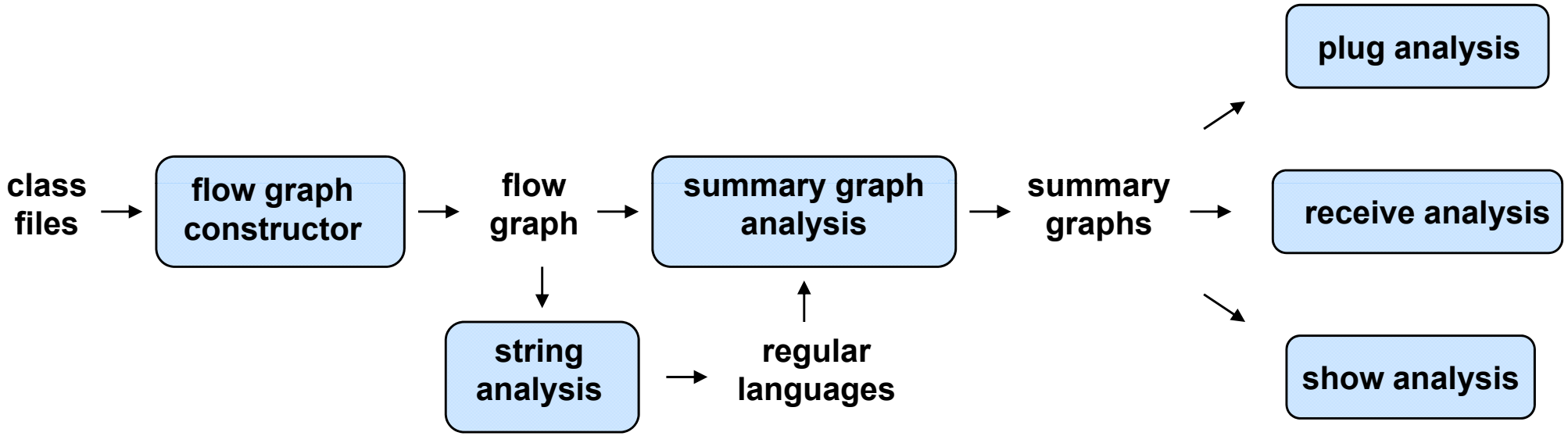
GuessingGame (5/5)

```
  int getCurrentRecord() {
    Integer record = (Integer)c.getAttribute("record");
    if (record!=null)
      return record.intValue();
    else
      return Integer.MAX_VALUE; // no players yet
  }
}
```

GuessingGameHiScore

```
public class GuessingGameHiscore extends HttpServlet {
  public void doGet() throws IOException, ServletException {
    ServletContext c = getServletContext();
    Integer plays = (Integer)c.getAttribute("plays");
    String holder = (String)c.getAttribute("holder");
    Integer record = (Integer)c.getAttribute("record");
    XML body;
    if (record!=null)
      body = [[In <{plays.toString()}> plays of this game,
               the record holder is <b><{holder}></b> with
               <b><{record.toString()}></b> guesses.]];
    else
      body = [[No players yet.]];
    XML.loadConstant("GuessingGameWrapper.xml")
      .plug("BODY", body).write(response.getWriter());
  }
}
```

Static Analysis of JWIG Programs



Catching Errors at Compile Time

```
...
XML ask = [[ <form>Your name? <input name="NAME"/>
             <input type="submit"/></form> ]];
...
```

```
*** Field 'NAME' is never available on line 15
*** Invalid XHTML at line 14
--- element 'input': requirement not satisfied:
<or>
  <attribute name="type">
    <union>
      <string value="submit" />
      <string value="reset" />
    </union>
  </attribute>
  <attribute name="name" />
</or>
```