SOEN 387 Web-based Enterprise Application Design

Stuart Thiel

Concordia University
Department of Computer & Software Engineering

Fall, 2015

Servlets

JSPs

Expression Language (EL)

Basic Taglibs

Outline

Servlets

JSPs

Expression Language (EL)

Basic Taglibs

What Does A Servlet Look Like File-wise?

- `https://tomcat.apache.org/tomcat-8.0-doc/appdev/deployment.html`
- *.html, *.jsp, etc.
- /WEB-INF/web.xml
- /WEB-INF/classes/
- /WEB-INF/lib/

Some Suggestions for file-size?

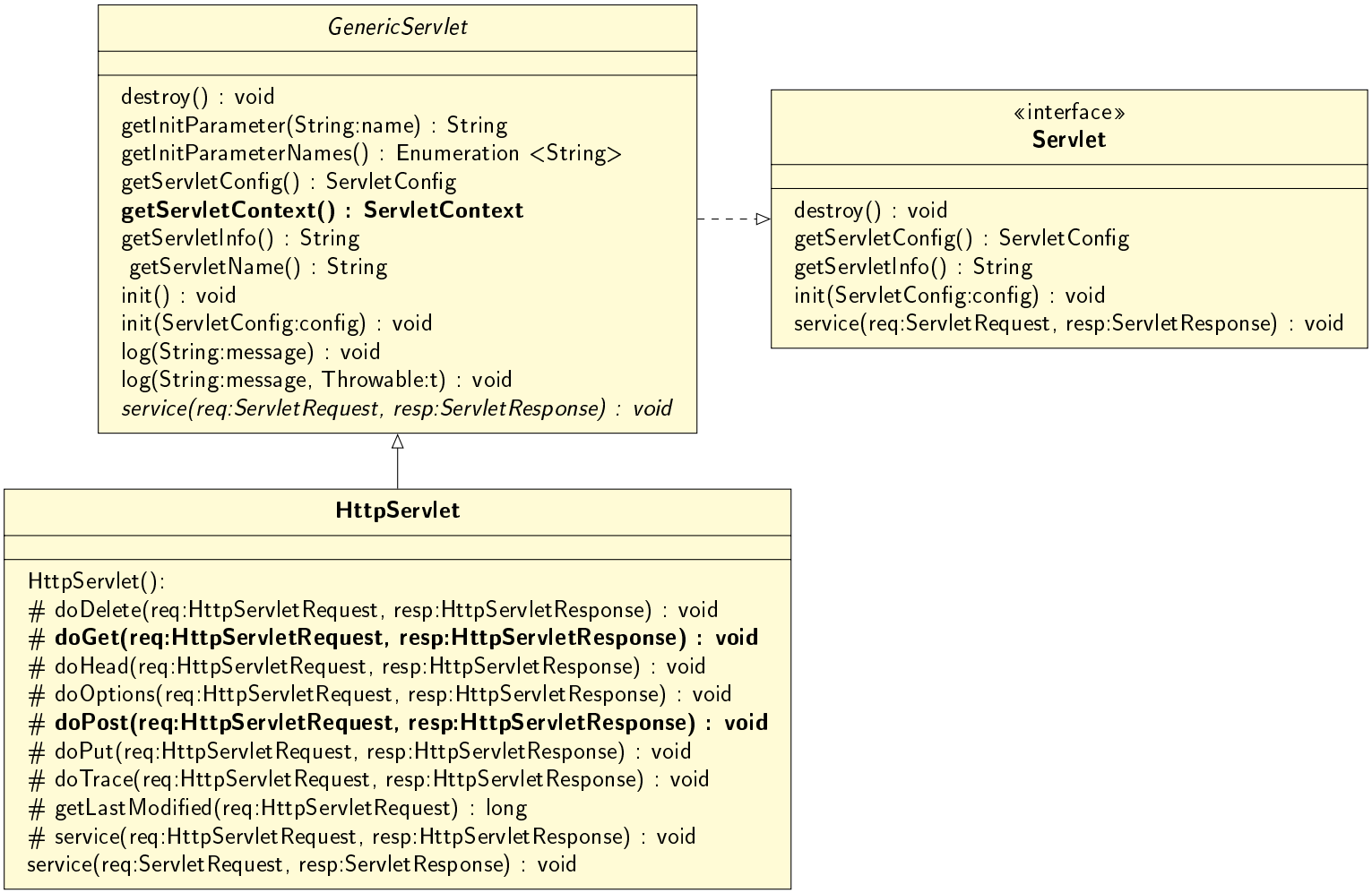
- `https://tomcat.apache.org/tomcat-8.0-doc/appdev/source.html`
- web/ webapp document root
  - under which you find WEB-INF, etc
- src/ keeps the source for your webapp nearby
- docs/ keep any documentation nearby
- Put src/DOCs in SVN (repo)!
- Put document root, save classes in SVN!

How to access JSPs

- I put JSPs where they cannot be directly accessed
  - inside WEB-INF/JSP
  - We'll talk about why this can be useful

A Look at Servlet API

- https://tomcat.apache.org/tomcat-8.0-doc/servletapi/



Presentation in Code is BAD!

```java
package org.cu.soen387;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<body>");
                out.println("Hello " + request.getParameter("name"))
                out.println("</body>");
                out.println("</html>");
                out.close();
        }
}
```

What are JSPs?

- A template language to separate out presentation
- An xml-based servlet language?

How do we get to them?

- Access them directly through servlet container
- Forward from an explicit servlet

Servlet Forwarding

```
package org.cu.soen387;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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

                getServletContext().getRequestDispatcher("/Hello.jsp")
                  .forward(request,response);
        }
}
```

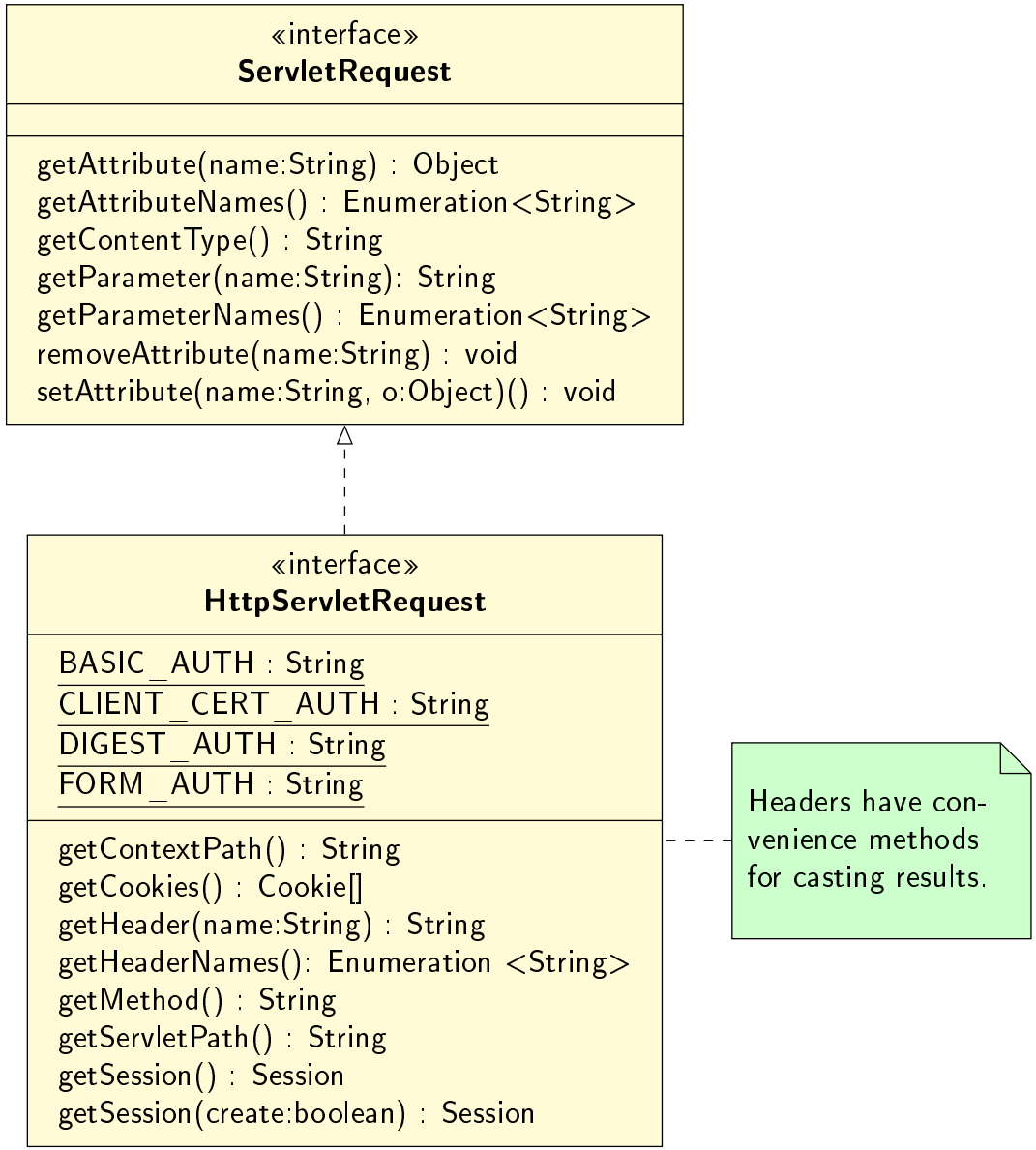
Forwarding

I told you it was messy!

Forward vs. Include

- Forwarding entity is done outputting
  - Forwarded entity closes output
- Can keep including stuff

HttpRequest/Response



Basic Hello World JSP

```
<%@ page
contentType="text/html;
charset=UTF-8" language="java"
%>
<html>
<body>
Hello <%= request.getParameter("name") %>
</body>
</html>
```

Basic Expression Language (EL)

- That's inline code in a JSP
- Avoid code in JSP
- Code confuses non-programmers
- Messy, multiple languages to parse
- Error-prone
- Same reason we don't want presentation in code

Hello World JSP with EL

```
<%@ page
contentType="text/html;
charset=UTF-8" language="java"
%>
<html>
<body>
Hello ${param[name]}
</body>
</html>
```

EL Expressions

- identified by ${}
- resolves many expressions
- knows a few useful operators
  - empty
  - not empty
  - basic arithmetic

EL And Attributes

- Some preset attributes available
  - param
- Always read-only
- Square brackets treat attributes like a map
- Looks up context chain for attribute
  - page
  - request
  - session
  - application
- Prints nothing if it can't find or is null

EL and Attributes

- The act like beans
- can use . notation to dig in
- `${greeting.name}`
  - looks for greeting attribute
  - calls `getName()` on it

Simple Taglibs

```
<%@ page language="java" contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c"
uri="http://java.sun.com/jstl/core_rt" %>
<html>
<body>
<c:if test="${empty param[name]}">
Hello unnamed person
</c:if>
<c:if test="${not empty param[name]}">
Hello ${param[name]}
</c:if>

</body>
</html>
```

Taglibs and Namespace

- That junk at the top says where taglibs are defined
- It also says what letter to use to namespace things "prefix"
- Can define this in web.xml
- JSTL should live somewhere in tomcat, just have to say you're using it
- Other taglibs you need to stick stuff in WEB-INF/lib or /classes

Taglibs Brief Mention

- Can do conditionals, switches, loops
- Note that you can pass labeled parameters
- You can write your own...
- Start with the JSTL Core