Servlets and AJAX

E-Applications Spring 2015

M. Lampis
Michail.lampis@dauphine.fr

Summary

- What we know so far:
  - Running Tomcat
  - Simple JSP applications (one file)
- Today
  - Servlets
  - AJAX (basics)

Servlets

- Basic architecture:
  - Tomcat is a servlet container
  - Servlet = server-side application
  - When Tomcat receives HTTP requests it forwards them to an appropriate servlet
- What about JSP?
  - .jsp pages are automatically compiled into servlets by Tomcat the first file they are loaded
  - Checke the work directory!

Servlets

- In addition to jsp, we can compile and install in Tomcat our own servlets
  - Advantage: separation of program logic from presentation!
    - A servlet is just a Java program. It does not look like an HTML page...
  - Disadvantage: separation of program logic from presentation...
    - Use the most appropriate tool for what you want to do...

Running servlets

- Development cycle
  - Write a Java program (.java file)
    - Your main class must extend HttpServlet
  - Compile it into a class file
  - Place it in an appropriate place in Tomcat, along with necessary libraries
  - Run Tomcat

Writing a servlet

- Simplest possible servlet

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

import javax.servlet.*;

import java.io.*;

public class Pong extends HttpServlet{

public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException

        { ... } //This is where the main body is...

}
```

doGet

- The main work goes in doGet
- This is the method Tomcat calls
  - Takes two parameters: request and response
  - request: similar to jsp, use request.getParameter()
    - Also, request.getSession() gives a session object
  - response: for producing a response
    - response.getWriter() produces an object on which we can .println() the response to the request
- doPost usually just calls doGet (or vice-versa)

Compiling the servlet

- Like any Java program!
  - Except that the servlet-api.jar file must be in the classpath (defines HttpServlet etc.)
- Quick and dirty way:
  - Copy servlet-api.jar to same directory
  - Javac myClass.jave –classpath=servlet-api.jar

Deployment

- Now that I have a .class with my servlet what do I do?
- There is a standard directory structure that Tomcat follows
- Inside webapps create a directory for your app, say, "myapp".

Deployment

- Inside the "myapp" folder
  - Index.html (and other "static" html, css, etc.)
  - A WEB-INF folder containing
    - A web.xml file
    - A classes directory
    - A lib directory
  - We place the class file in the classes directory and any library files we need in lib

Deployment

- We are not done yet!
- We need to tell Tomcat which request to map to which servlet (we could have several servlets)
- Edit the web.xml file for this

web.xml

```
<web-app>
<servlet>
<servlet-name>myservlet</servlet-name>
<servlet-class>myClass</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>myservlet</servlet-name>
<url-pattern>/service</url-pattern>  Note:This can be a regex
</servlet-mapping>
</web-app>
```

This is too much work?

- Convenient way to deploy apps: WAR files
- Web Archives, similar to jar files
- To create a war file
  - Enter directory of deployed app
  - jar cvf myapp.war *
- To deploy an app stored in a war
  - Just copy the war file in the webapps directory!
  - jar tvf file.war list the files
  - jar xvf file.war extracts the files

Running a servlet

- What if I have a new version of a class?
  - Recall: Tomcat automatically re-compiles and re-deploys jsp files when changed
  - Not so with servlets. Once a servlet starts to run, even replacing its class file has no effect
  - We must tell Tomcat to reload it
  - Easiest way: restart Tomcat!!
  - bin/shutdown.sh ; bin/startup.sh

Servlets vs jsp

- Why use servlets instead of jsp?
  - "Cleaner" work environment, writing Java instead of Java+HTML
  - Slightly more cumbersome for small/simple things
  - Basic idea more or less the same
    - Recall: in a JSP we are basically just programming the doGet method. Check out work directory for examples!
  - Use whichever you like!
  - jsp apps can also be packed into war files (in same way)

AJAX

- Asynchronous Javascript And XML
- So far, we have seen web apps where a user has to refresh the page to get new information
  - Not nice!
- In more modern applications we program the client (with javascript) to interact with the server in the background, updating the display as new info arrives
  - No refresh needed!

AJAX methodology

- Create an index.html file with the basic page + javascript
- Create a jsp/servlet that will handle get requests from that page and return an XML file
- The js program periodically (setInterval) sends requests to the servlet and updates the page.

GET requests from javascript

```
var req = new window.XMLHttpRequest();
req.open("GET","servletURL",true);
req.send();
req.onload = refreshState;
```

- Can add parameters to request by encoding them in URL (GET)
- Last parameter of open -> asychronous request
- req.onload = Function that will be executed

HTTP requests

- Can also do POST requests
  - Then parameters are not encoded in URL

    (not today)
- Check for network errors/state
  - req.onreadystatechange

    (not today)
  - Assume network is perfect!

Server side

- The doGet method will receive these requests.
- We must produce an XML response and write it into the response parameter
- Quick and dirty way:
  - Produce a string that contains the XML file
  - println the string

  (good enough for today!)

Example

```
Public void doGet(HttpServletRequest req,HttpServletResponse
res){

res.setContentType("text/xml");

PrintWriter pw=res.getWriter();

pw.println("<replay><t1>blabla</t1> <t2> </t2></reply>");

pw.close();

}
```

Inside the servlet

- The first time a request comes Tomcat creates **one** instance of your class (which extends HttpServlet)
- The init() method is called
- Then, doGet()/doPost() is called for each request
  - For information shared across request, we can add fields to our class. These will be shared.
  - For information relevant to one client use req.getSession() in doGet().