Web APIs
Web APIs - Outline

• Review of Web Software:
  – Browser
    • interprets client side code: HTML, CSS, and JavaScript
    • Makes HTTP requests for pages from Web Servers.
  – Apache (Web Server, serves up HTML pages).
  – Glassfish/Tomcat (JSP application server sw, execute JSP pages)
    • Glassfish is in the NetBeans Bundle on our development machines.
    • Tomcat is installed on the web server (cis-linux2.temple.edu).

• (Server Side) JSP Pages – then and now.
• Simplest (ugly code) Web API & Better Web API (uses gson)
• Organization of Projects that have server side code – enhanced instructions for installing, running, & publishing.
Browsers

• We know that the job of a browser is to request web pages (from web servers) and render web pages.

• We know that a browser interprets three languages:
  – HTML: content and structure,
  – CSS: look and feel, and
  – JavaScript: functionality that lets you interact with the user and/or make AJAX calls (HTTP requests for data)
Apache

• Apache is software that’s installed on a web server. Apache knows how to respond to:
  – HTTP requests for HTML pages (with UI)
  – AJAX calls, e.g., HTTP requests for data (no UI)
• Apache is installed on linux (unix) machines.
• IIS (Internet Information Services) is web server software that’s installed on Windows Servers.
Glassfish and Tomcat

- Glassfish and Tomcat are JSP application server software. They run JSP pages on the server (generating dynamic content, server side) before delivering to the browser.
- Glassfish is part of the NetBeans Bundle that’s installed on your development machine.
- Tomcat is what we happen to have installed on your web server (cis-linux2.temple.edu).
NetBeans Bundle

• **NetBeans** offers a *multi-language context sensitive editor* that can identify errors and provide type-ahead choices for
  – the three client languages just mentioned (HTML, CSS, JavaScript).
  – But also for server side java and JSP (Java Server Pages).
• The NetBeans bundle also provides **Apache**, web server software that serves up HTML pages.
• The NetBeans bundle also provides **Glassfish**, JSP application server software that can execute (generate) JSP pages. web server software that serves up HTML pages.
• Once you publish your web app (for whatever reason), the JSP application server software (on cis-linux2.temple.edu) is tomcat, not glassfish.
JSP Pages

• As just mentioned, JSP pages execute on the web server. The content of the JSP page is not known until it’s run.

• Years ago, JSP pages would contain code like an HTML page (HTML, CSS, JavaScript) but could also include java code that can access a database and output data onto the page.

• See an example of an Old Style JSP page, one that had User Interface (like an HTML page).
Sample Old Style JSP Page (with User Interface)
(color coding is explained on next slide)

```html
<!DOCTYPE HTML>
<html>
  <body>
    <% String strYourName = "";
       String msg = "";
       if (request.getParameter("yourName") != null) { // tests if this is 1st rendering
         strYourName = request.getParameter("yourName");
         msg = "Hello, "+strYourName;
       } // postback
    %>
    <h3>User Interface</h3>
    <form action="old_style_JSP.jsp" method="get">
      What's your name?
      <input name="yourName" type="text" value="<%=strYourName%>"/>
      <input type="submit"/>
    </form>
    <p><%=msg%></p>
  </body>
</html>
```
Notes about the Previous (Old Style) JSP page

- `contentType="text/html"`
  Tells the browser that the page is an HTML page (with UI)

- `<%=msg%>`
  Between `<% ... %>` you can place java code (that can access a database and output data onto the page).

- `<form action="old_style_JSP.jsp" method="get">`
  Upon submit, all data in the form gets sent to the page specified in the action attribute (usually the same page).

- `request.getParameter("yourName")`
  Extracts URL parameter “yourName” (after submit, “postback”). On 1st rendering, `request.getParameter("...")` returns null.
(Old Style) JSP page

NOTE:

• You do not have to memorize anything from the “Old Style” JSP page – you will not be writing any of that code.

• I just want you to have some context about how things used to be...
Using JSP Pages as Web APIs

• Now we use JSP pages for Web APIs, server side pages that can access a database and output data on the page (e.g., JSON format).

• Web APIs do not output an HTML page (HTML/CSS/JavaScript) – just data.

• Web APIs can also be written in other languages like PHP, C# (a java like language using the .NET platform) or python.
Ugly JSP Code that Outputs JSON

```jsp
<%@page contentType="application/json; charset=UTF-8" pageEncoding="UTF-8"%>
<%

// contentType is json, not html...
String myJSON = "{"webUserId": "5", "userEmail": "sallyk@temple.edu"}";
out.print(myJSON); // out.print() prints data into the JSP page
%
```

This page looks ugly because java only uses double quotes for String delimiters and proper JSON includes double quotes. To place a double quote inside those double quotes, you need this: \\

• What’s output by the above code (seen through JSON View):

```
{
  webUserId: "5",
  userEmail: "sallyk@temple.edu"
}
```
Better JSP Page that outputs JSON

```jsp
<%@page contentType="application/json; charset=UTF-8" pageEncoding="UTF-8"%>
<%@page language="java" import="classes.StringData" %>
<%@page language="java" import="com.google.gson.*" %>

// Much better: gson converts Java Object to JSON for you...
StringData sd = new StringData();
sd.webUserId = "5";
sd.userEmail = "sallyk@temple.edu";
Gson gson = new Gson();
out.print(gson.toJson(sd));

• Here’s what it outputs (seen through JSON View):
```
Why that “Better Code” is Better

• We just put what we want to output right into any Java Class.

• We then use the gson Jar file to convert any Java Class to JSON. (A Jar File is just like a zip file that contains compiled java code.)

• We don’t need to worry about writing syntactically correct JSON code.

• We don’t have to use /*" to insert double quotes inside of String literals.
Organization of Projects Containing WebAPIs

- Place your Web APIs in a Web API folder under Web Pages (not required, just to organize).
- The web pages folder will still contain normal files like index.html, css, js, etc.
- Put your java classes in a package under “Source Packages”
- Add the gson Jar file to the Libraries folder of your project. This code can convert a java class to JSON (and vice versa).
Serving Up HTML Pages
(what happens when an HTML or JSON file is requested)
Serving Up Web APIs (JSP Pages)
(what happens when a Web API is invoked)
Enhanced Instructions

• Once you start working with server side code (Java/JSP code), you have to start using *enhanced instructions* for:
  – **Installing** sample code into your NetBeans IDE
  – **Running** your code (must tunnel in if you need database access)
  – **Debugging** your code (system.out.println messages are available in the Glassfish server log accessible from the NetBeans bundle).
  – **Publishing** your code – the class files must be put in the right spot and you must copy/delete/paste the web.xml file afterwards to let the JSP application server (tomcat on cis-linux2.temple.edu) know that you have redeployed new java class files.

• All of these instructions are available from your class web page (section entitled “Enhanced Instructions”, near the “Web API” module).
JSP Pages

• In your Java/JSP Pages (that implement your Web APIs), you will use classes from the `java.sql` package:
  – DriverManager: class that provides access to database drivers,
  – Connection: class that can open and close a database connection,
  – PreparedStatement: class that can create / execute a parameterized SQL statement, and
  – ResultSet: class that can parse through the results generated by a SQL SELECT statement.

• If your Java/JSP code accesses a database, you’ll need to
  – add database drivers to your project (MySQL Jar file)
  – Tunnel into Temple’s network (whenever you are developing from home - so your code can access your database)

• If your Java/JSP code will need to convert between JSON and Java Objects (also called POJOs for Plain Old Java Objects), you need to add the Gson Jar file to your project).
Tips for Writing Client Code with Web APIs

• Do not try to write Client side code (JavaScript) before you know that your server side code is working properly !!!

• To test your Web APIs (JSP Pages), URL tamper, placing the URL of the Web API directly in the browser’s address bar. If you have a JSON View Plugin installed to your browser, you’ll see the data formatted and easy to read.