Servlets

- Generic Java2EE API for invoking and connecting to "mini-servers" (lightweight, short-lived, server-side interactions).
- Mostly HTTPServlets
  - Generate HTTP responses (as for CGI)
  - Embedded via <servlet> tag into HTML pages for server-side includes
  - Servlet code snippets can be embedded directly into HTML pages: JSP = Java Server Pages
    - Competitor: Microsoft ASP = Active Server Pages

Servlet Model

- Pieces:
  - Clients: web browsers ie IE, Netscape
  - Web server: Apache, Netscape Enterprise, IIS
  - CGI Protocol: Specifying what a request/response looks like
  - Servlet container: Running JVM, hooked into the web server, loads and maintains servlets, session info, object store
  - Servlet: A Java class used to handle a specific request.

Servlet Interaction

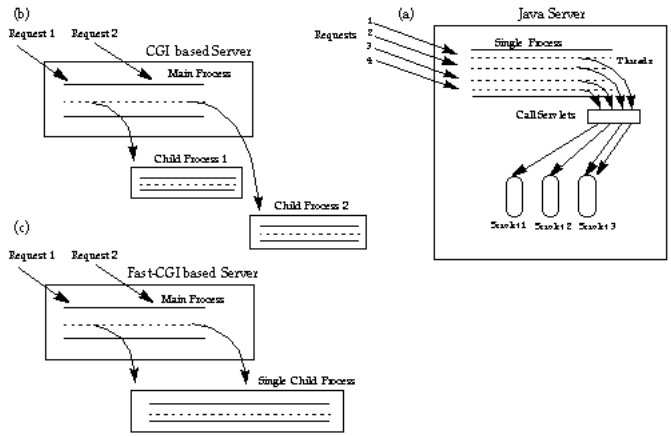
- Client makes a request by specifying a URL+additional info. Basically a method call, the method and arguments.
- The web server (specified in the URL) receives the request.
- The web server identifies the request as a servlet request
- The web server passes the request to the servlet container.
- The servlet container locates the specified servlet (Java code, loaded and running in the container JVM)
- The servlet container feeds the request parameters to the servlet (through HttpServletRequest, HttpServletResponse).
- The servlet executes in a separate thread
- The servlet can store/retrieve objects (possibly session scoped) from the servlet container.
- Output of the servlet is sent via a PrintWriter back to the requesting web browser.
- The servlet continues to be available in the servlet container.

Servlets vs CGI

- Servlets...
  - are generally much faster than CGI scripts because a different process model is used. (but see FastCGI).
  - process model allows resource sharing (ie database connection sharing). Container is always available.
  - uses a standard API that is supported by many web servers.
  - has all the advantages of the Java programming language, including ease of development and platform independence.
  - can access the large set of APIs available for the Java platform.
  - has access to CGI support through the Servlet API
  - Has session awareness, ability to store and retrieve session scoped objects from servlet container.

Servlet Process Model

Servlet, CGI and Fast-CGI process models



Servlet API (javax.Servlet)

- Consists of
  - Servlet Container
  - Servlet Interface (HttpServlet)
  - ServletContext
  - HttpServletResponce
  - HttpServletRequest

Servlet Container

- Is a part of a web server or application server that
  - provides the network services over which requests and responses are sent
  - decodes requests
  - formats MIME based responses
  - contains and manages servlets through their lifecycle

Servlet Interface

- is the central abstraction of the servlet API
- All servlets either implement this interface, or extend a class that implements this interface
- GenericServlet and HttpServlet, two classes in the servlet API that implement the Servlet interface

HttpServlet

- HttpServlet
  - Web developers extend HttpServlet to implement their servlets.
  - Most often override `doGet()` and `doPost()` with code to handle http `GET` and `POST` requests.

ServletContext

- ServletContext
  - interface defines a servlet's view of the web application within which the servlet is running.
  - the Container Provider is responsible for providing an implementation of the ServletContext interface in the servlet container.
  - ServletContext is rooted at a known path in webserver
  - Using the ServletContext object, a servlet can
    - log events
    - obtain URL references to resources
    - set attributes that other servlets in the context can access.

ServletContext Example

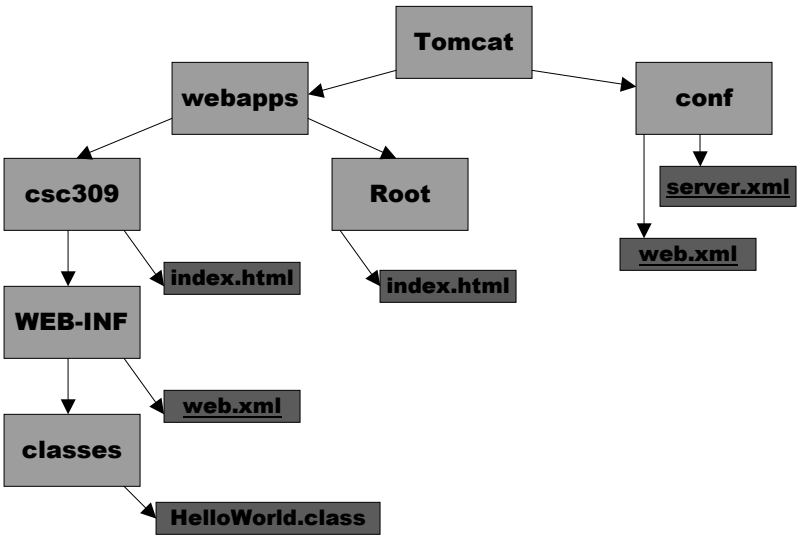
Example:

all servlets associated with this lecture are part of one application, the csc309 servlet context.

References to servlets within that application are via the URL

`http://127.0.0.1:8080/csc309/servlet`

Tomcat Configuration



ServletContext Continued

- One instance of ServletContext per web application per container
- Provides access to Initialization Parameters
  - Defined in the deployment descriptor
    - A file called web.xml in the WEB-INF subdirectory of the app directory
  - Methods:

    `getInitParameter()`, `getInitParameterNames()`

ServletContext Attributes

- Context Attributes
  - Context level shared objects. Can be bound into the context and accessed by any servlet that is part of the application.
  - Methods:

    `setAttribute()`

    `getAttribute()`

    `getAttributeNames()`

    `removeAttributes()`
  - More than 1 thread may be accessing a shared object.
  - Application developer responsible for synchronizing access to shared objects

ServletContext Resources

- Resources
  - Gives servlet access to local documents associated with the application. (i.e. static web pages, images)
  - Methods:

    `getResource()`

    `getResourceAsStream()`

    `getResourcePaths()`

HttpServletResponse

- <u>HttpServletResponse</u>
  - encapsulates all information to be returned from the server to the client
  - In the HTTP protocol, this information is transmitted from the server to the client either by HTTP headers or the message body of the request.
  - Headers: Can set/manipulate http headers
    - primitive manipulation: `setHeader()`, `addHeader()`
    - convenience methods: `setContentType()`, `sendRedirect()`, `sendError()`
  - `getWriter()` - Obtain a PrintWriter, used to return character data to the client

Servlet Examples

Non-interactive web pages

- Hello World Under the URL `http://127.0.0.1:8080/csc309/servlet/HelloWorld` Source: HelloWorld.java
- Flow between many pages (note the different kinds of hrefs):
  - Hello1 Source:Hello1.java
  - Hello2 Source:Hello2.java
- Returning different content: ImageFileServlet Source: ImageFileServlet.java
  How is this different from an *img* tag? The servlet can create the image on the fly.

HttpServletRequest

- HttpServletRequest
  - encapsulates all information from the client request
  - In the HTTP protocol, this information is transmitted from the client to the server in the HTTP headers and the message body of the request.
  - Manipulate 'form' variables: `getParameter()`, `getParameterNames()`, `getParameterValues()`
  - `getCookies()` returns all cookies associated with the request
  - `getSession()` returns the Session instance associated with the request (see below)

Servlet Forms Examples

Example: Using forms and form variables

Servlet: CGILibExample

Source: CGILibExample.java.

Note: override both the `doGet()` and `doPost()` methods since the original CGILibExample used both the post and get methods at the same time.

When to GET/POST

- GET
  - All form parameters are embedded in the URL
  - If you reload, or bookmark and return, the query will get executed a second time with the same parameters
    - Bad if page is a credit card order confirmation
      - 2 charges!
- POST
  - No form parameters are embeddd in the URL
  - On reload
    - Browser will ask if it should re-post
  - On bookmark and return
    - Query will proceed with no parameters
- Therefore: POST to change state, GET to get info

Sessions

- The Hypertext Transfer Protocol (HTTP) is by design a stateless protocol. To build effective web applications, requests from a particular client must be associated with each other. Many strategies for session tracking have evolved over time, but all are difficult or troublesome for the programmer to use directly.
- The Servlet API specification defines a simple HttpSession interface that allows a servlet container to use any of several approaches to track a user's session without involving the Application Developer in the nuances of any one approach.

Session Tracking Mechanisms

- A session token is passed back and forth between the client and servlet container. The token allows the container to associate a session with a particular session.
- The servlet container adapts to whether the client (browser) is accepting cookies.
- Cookies: If the client accepts cookies, sessions will identified by a token stored on the browser as a cookie.
- URL Rewriting: If the client is not accepting cookies, session token is communicated via the URL. Use the HttpServletResponse `encodeURL()` method.

HttpSession

- HttpSession
  - One instance for each active session

```
HttpSession session = request.getSession();
// session=current session or new one, if none exists

Object o = session.getAttribute("name");
// o=the object associated with "name" in this session

session.setAttribute("name",o);
// o is now the object associated with "name" for this session

// NOTE: More than one thread may try to access a resource.
// The application author is responsible for synchronizing
// access to session resources.
```


Session Example

Example: Sessions

Servlet: SessionExample

Source:SessionExample.java.

Note:

Run the above with cookies off and on, notice the URLs.

Shut down and restart your browser between accepting and rejecting cookies.

Servlet Summary

- Web Application
  - Collection of servlets
- ServletContext
  - One object per web application
  - initParameters (from deployment description "web.xml")
  - attributes (name-object pairs associated with the web application)
- HttpSession
  - Associated with one user's interaction
  - Explicit create/delete + timeout capability
  - Session attributes (name-object pairs associated with the app/session)
  - Implemented with cookies or URL re-writing
- HttpServletRequest
  - parameters from GET or POST

JSP

- Java Server Pages
  - http://localhost:8080/csc309/jsp/hello.jsp?name=dave (source)
- Processing (done by a servlet in Tomcat)
  - Read file
  - Construct a .java source file
    - Every not <% line change into out.println() call
    - Remove <% from others
    - Compile
    - Execute as a servlet

Advanced JSP

- Really no different than a servlet
  - Can access other classes
  - http://localhost:8080/examples/jsp/num/numguess.jsp
    - C:\apache\Tomcat4\webapps\examples\jsp\num\numguess.jsp
    - C:\apache\Tomcat4\webapps\examples\WEB-INF\classes\num\NumberGuessBean.java

Tag Libs

- Collection of defined XML tags that take actions when the server encounters them
  - Implemented by Java classes
- Standard jsp taglib
- Can add your own taglib

ASP

- Microsoft-specific technology
- Active Server Pages
- Very similar to JSP
  - Uses VisualBasic instead of Java
  - Can access a vast array of COM and DCOM objects
  - E.g., grocerygateway.com (Nov.23 guest lecture)
    - ASP pages invoke XSLT via COM to gen. HTML output appropriate to the requesting client
    - ASP sends COM messages to C++ servers