Web.xml elements:

* [context-param](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html#1014756)

* [description](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1013555)

* [display-name](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1043408)

* [distributable](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1040893)

* [error-page](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017571)

* [filter](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1015950)

* [filter-mapping](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1039330)

* [icon](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1013332)

* [listener](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1039300)

* [login-config](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1019996)

* [mime-mapping](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017395)

* [resource-env-ref](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017834)

* [resource-ref](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1045815)

* [security-constraint](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017885)

* [security-role](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1025149)

* [servlet](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1039287)

* [servlet-mapping](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1015060)

* [session-config](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017275)

* [taglib](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1017621)

* [welcome-file-list](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1026980)
* [ejb-ref](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html#1020090)

* [ejb-local-ref](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1013984)

* [env-entry](http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html" \l "1025195)

**Q) What is servlet?**   
A) Servlets are modules that extend request/response-oriented servers, such as java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.

**Q) What are the classes and interfaces for servlets?**   
A) There are two packages in servlets and they are javax.servlet and javax.servlet.http.

**Javax.servlet** contains:  
**Interfaces** **Classes**\* Servlet \* Generic Servlet   
\* ServletRequest \* ServletInputStream   
\* ServletResponse \* ServletOutputStream  
\* ServletConfig \* ServletException   
\* ServletContext \* UnavailableException  
\* SingleThreadModel

**Javax.servlet.http** contains:  
**Interfaces** **Classes**   
\* HttpServletRequest \* Cookie   
\* HttpServletResponse \* HttpServlet   
\* HttpSession \* HttpSessionBindingEvent   
\* HttpSessionContext \* HttpUtils   
\* HttpSeesionBindingListener \* HttpSessionEvent

\*HttpSessionListener

**Q) Explain the life cycle methods of a Servlet.**   
A)The javax.servlet.Servlet interface defines the three methods known as life-cycle method.  
public void init(ServletConfig config) throws ServletException  
public void service( ServletRequest req, ServletResponse res) throws ServletException, IOException  
public void destroy()  
First the servlet is constructed, then initialized with the init() method. Any request from client are handled initially by the service() method before delegating to the doXxx() methods in the case of HttpServlet. The servlet is removed from service, destroyed with the destroy() method, then garbage collected and finalized.

**Q) What is the ServletConfig() and why is it useful?**   
A) This interface is implemented by services in order to pass configuration information to a servlet when it is first loaded. A service writer implementing this interface must write methods for the servlet to use to get its initialization parameters and the context in which it is running. public interface ServletConfig.

**Q) What is meant by the ServletContext() and use of the method ?**   
A) public interface ServletContext   
The ServletContext interface gives servlets access to information about their environment, and allows them to log significant events. Servlet writers decide what data to log. The interface is implemented by services, and used by servlets. Different virtual hosts should have different servlet contexts.

**Q) What is the difference between ServletContext and ServletConfig?**

|  |  |
| --- | --- |
| ServletConfig | ServletContext |
| * 1) ServletConfig available in javax.servlet.\*; package * 2) ServletConfig object is one per servlet class * 3) Object of ServletConfig will be created during initialization process of the servlet * 4) This Config object is public to a particular servlet only * *5) Scope*: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed. * 6) We should give request explicitly, in order to create ServletConfig object for the first time * 7) In web.xml – <*init-param*> tag will be appear under <*servlet-class*> tag | * 1) ServletContext available in javax.servlet.\*; pack * 2) ServletContext object is global to entire web application * 3) Object of ServletContext will be created at the time of web application deployment * 4) This context object is public to a particular application. * *5) Scope*: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server. * 6) ServletContext object will be available even before giving the first request * 7) In web.xml – <*context-param*> tag will be appear under <*web-app*> tag |

* Signature: public interface ServletConfig  
  ServletConfig is implemented by the servlet container **to initialize a single servlet** using init(). That is, you can pass initialization parameters to the servlet using the web.xml deployment descriptor. For understanding, this is similar to a constructor in a java class.

Example code:  
<servlet>  
<servlet-name>ServletConfigTest</servlet-name>  
<servlet-class>com.javapapers.ServletConfigTest</servlet-class>  
<init-param>  
<param-name>topic</param-name>  
<param-value>Difference between ServletConfig and ServletContext</param-value>  
</init-param>  
</servlet>

* Signature: public interface ServletContext  
  ServletContext is implemented by the servlet container **for all servlet** to communicate with its servlet container, for example, to get the MIME type of a file, to get dispatch requests, or to write to a log file. That is to get detail about its execution environment. It is applicable only within a single Java Virtual Machine. If a web application is distributed between multiple JVM this will not work. For understanding, this is like a application global variable mechanism for a single web application deployed in only one JVM.

The ServletContext object is contained within the ServletConfig object. That is, the ServletContext can be accessed using the ServletConfig object within a servlet. You can specify param-value pairs for ServletContext object in <context-param> tags in web.xml file.

Example code:  
<context-param>  
<param-name>globalVariable</param-name>  
<param-value>javapapers.com</param-value>  
</context-param>

**Q) What is use of parseQueryString ?**   
A) Parses a query string and builds a hashtable of key-value pairs, where the values are arrays of strings. The query string should have the form of a string packaged by the GET or POST method. (For example, it should have its key-value pairs delimited by ampersands (&) and its keys separated from its values by equal signs (=).)   
Note: public static Hashtable parseQueryString(String s)

**Q) what are the types of servlets.**   
A) Genereic Servlets,HttpServlets.

**Q) What is the difference between HttpServlet and GenericServlet?**   
A) A GenericServlet has a service() method aimed to handle requests. HttpServlet extends GenericServlet and adds support for doGet(), doPost(), doHead() methods (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() methods (HTTP 1.1). Both these classes are abstract.

**Q) What are the different methods in HttpServlet.**   
A) An abstract class that simplifies writing HTTP servlets. It extends the GenericServlet base class and provides a framework for handling the HTTP protocol. Because it is an abstract class, servlet writers must subclass it and override at least one method. The methods normally overridden are:  
\* doGet, if HTTP GET requests are supported. Overriding the doGet method automatically also provides support for the HEAD and conditional GET operations. Where practical, the getLastModified method should also be overridden, to facilitate caching the HTTP response data. This improves performance by enabling smarter conditional GET support.  
doPost, if HTTP POST requests are supported.  
doPut, if HTTP PUT requests are supported.  
\* doDelete, if HTTP DELETE requests are supported.  
\* The lifecycle methods init and destroy, if the servlet writer needs to manage resources that are held for the lifetime of the servlet. Servlets that do not manage resources do not need to specialize these methods.  
\* getServletInfo, to provide descriptive information through a service's administrative interfaces.   
Notice that the service method is not typically overridden. The service method, as provided, supports standard HTTP requests by dispatching them to appropriate methods, such as the methods listed above that have the prefix "do". In addition, the service method also supports the HTTP 1.1 protocol's TRACE and OPTIONS methods by dispatching to the doTrace and doOptions methods. The doTrace and doOptions methods are not typically overridden.

**Q) Why do you need both GET and POST method implementations in Servlet?**   
A) A single servlet can be called from different HTML pages, so different method calls can be possible.

**Q) When init() and Destroy() will be called?**   
A) init() is called whenever the servlet is loaded for the first time into the webserver. Destroy will be called whenever the servlet is removed from the webserver.

**Q) Who is loading the init() method of servlet?**   
A) Web Server

**Q) If you want to modify the servlet, will the Webserver need to be ShutDown.**   
A) No

**Q) What is the advantage of Servlets over other serverside technologies.**   
A) Platform independent, so once compiled can be used in any webserver. For different processes different threads will execute inbuilt multithreaded.

**Q) What is Server-Side Includes (SSI)?**   
A) Server-Side Includes allows embedding servlets within HTML pages using a special servlet tag. In many servlets that support servlets, a page can be processed by the server to include output from servlets at certain points inside the HTML page. This is accomplished using a special internal SSINCLUDE, which processes the servlet tags. SSINCLUDE servlet will be invoked whenever a file with an. shtml extension is requested. So HTML files that include server-side includes must be stored with an .shtml extension.

**Q) What is Single Threaded Model in Servlets and how is it useful give one practical example.**  
A) For every single user a different copy of this servlet is executed. Credit card transactions.

**Q) What is the use of Sessions?**   
A) It’s a part of the SessionTracking and it is for maintaining the client state at server side.

**Q) What are the advantage of using Sessions over Cookies and URLReWriting?**   
A) Sessions are more secure and fast because they are stored at server side. But Sessions has to be used combindly with Cookies or URLReWriting for maintaining the client id that is session id at client side.   
Cookies are stored at client side so some clients may disable cookies so we may not sure that the cookies which we are maintaining may work or not but in sessions cookies are disable we can maintain our session id using URLReWriting.   
In URLReWriting we can't maintain large data because it leads to network traffic and access may be become slow. Where as in sessions will not maintain the data which we have to maintain instead we will maintain only the session id.

**Q) What is session tracking and how do you track a user session in servlets?**   
A) Session tracking is a mechanism that servlets use to maintain state about a series requests from the same user across some period of time. The methods used for session tracking are:   
  
a) User Authentication - occurs when a web server restricts access to some of its resources to only those clients that log in using a recognized username and password   
b) Hidden form fields - fields are added to an HTML form that are not displayed in the client's browser. When the form containing the fields is submitted, the fields are sent back to the server   
c) URL rewriting - every URL that the user clicks on is dynamically modified or rewritten to include extra information. The extra information can be in the form of extra path information, added parameters or some custom, server-specific URL change.   
d) Cookies - a bit of information that is sent by a web server to a browser and which can later be read back from that browser.   
e) HttpSession- places a limit on the number of sessions that can exist in memory. This limit is set in the session.maxresidents property

**Q) What is Cookies and what is the use of Cookies?**   
A) Cookies are used to get user agents (web browsers etc) to hold small amounts of state associated with a user's web browsing. Later that information read by server

**Q) What are cookies and how will you use them?**   
A) Cookies are a mechanism that a servlet uses to have a client hold a small amount of state-information associated with the user.   
  
a) Create a cookie with the Cookie constructor: public Cookie(String name, String value)   
b) A servlet can send a cookie to the client by passing a Cookie object to the addCookie() method of HttpServletResponse:   
public void HttpServletResponse.addCookie(Cookie cookie)   
c) A servlet retrieves cookies by calling the getCookies() method of HttpServletRequest:   
public Cookie[] HttpServletRequest.getCookie().

**Q) What is the use of setComment and getComment methods in Cookies ?**   
A) setComment:If a user agent (web browser) presents this cookie to a user, the cookie's purpose will be described using this comment. This is not supported by version zero cookies.   
public void setComment(String use){   
}   
  
getComment:   
Returns the comment describing the purpose of this cookie, or null if no such comment has been defined.

**Q) Why we are used setMaxAge() and getMaxAge() in Cookies ?**   
A) setMaxAge : public void setMaxAge(int expiry)  
Sets the maximum age of the cookie. The cookie will expire after that many seconds have passed. Negative values indicate the default behaviour:the cookie is not stored persistently, and will be deleted when the user agent exits. zero value causes the cookie to be deleted  
Setting setMaxAge to 0 will delete the cookie. Setting it to -1 will preserve it until the browser is closed.  
getMaxAge(): public int getMaxAge()   
Returns the maximum specified age of the cookie. If none was specified, a negative value is returned, indicating the default behaviour described with setMaxAge.

**Q) Difference between http and https?**

**HTTP(Hpertext Transfer Protocol)** is a protocol used in application layer for transmitting and retrieving information across internet. Http request could be sent and you get a response for that.  
  
**HTTPS(hypertext Transfer Protocol Secure**) works by transmitting normal http interactions through an encrypted system and the information cannot be accessed by any party other than the client and end server. Two types of encryption layer available are **Transport Layer Security (TLS)** and **Secure Sockets Layer (SSL**), both of which encode the data records being exchanged.   
  
Default Port number for**HTTP** is 80 and for **HTTPS** it is 443.

**Q) What is the use of setSecure() and getSecure() in Cookies ?**   
A) setSecure : Indicates to the user agent that the cookie should only be sent using a secure protocol (https). This should only be set when the cookie's originating server used a secure protocol to set the cookie's value.   
public void setSecure(boolean flag)   
getSecure: Returns the value of the 'secure' flag.  
public boolean getSecure()

**Q) What is meant by Httpsession and what is the use of sessions ?**   
A) The HttpSession interface is implemented by services to provide an association between an HTTP client and HTTP server. This session, persists over multiple connections and/or requests during a given time period. Sessions are used to maintain state and user identity across multiple page requests.  
HttpSession session = req.getSession(true);

**Q) What are the methods in HttpSession and use of those methods?**   
A)The following are the methods in HttpSession :  
a) getCreationTime():Returns the time at which this session representation was created.  
b) getId():Returns the identifier assigned to this session.   
c) getLastAccessedTime():Returns the last time the client sent a request carrying the identifier assigned to the session.  
d) getSessionContext():Returns the context in which this session is bound.   
e) getValue(String):Returns the object bound to the given name in the session's application layer data.  
f) getValueNames():Returns an array of the names of all the application layer data objects bound into the session.   
g) invalidate():Causes this representation of the session to be invalidated and removed from its context.  
h) isNew():A session is considered to be "new" if it has been created by the server, but the client has not yet acknowledged joining the session.  
j) putValue(String, Object) :Binds the specified object into the session's application layer data with the given name.  
k) removeValue(String):Removes the object bound to the given name in the session's application layer data.

**Q) How do you communicate between the servlets.**   
A) a)servlet chaining     b)Servlet context(RequestDespatcher interface)

**servlet chaining Example**

ServletConfig config;

RequestDispatcher rd = config.getServletContext().getRequestDispatcher("SecondServlet");  
   request.setAttribute("UserName",name);  
   rd.forward(request , response);

**Servlet context Example**

 ServletContext context=getServletContext();

        String s1=context.getInitParameter("n1");

        String s2=context.getInitParameter("n2");

**In web.xml**

<web-app>

    <context-param>

        <param-name> n1 </param-name>

        <param-value> 100 </param-value>

    </context-param>

    <context-param>

        <param-name> n2  </param-name>

        <param-value> 200 </param-value>

    </context-param>

</web-app>

**Q) Can you send the mail from a servlet, if yes tell how?**   
A) Yes. Using mail API

**Q) How do you access variables across the sessions?**   
A) Through ServletContext.

**Q) What is Servlet Context?**   
A) This object represents resources shared by a group of servlets like servlet's environment, Application attributes shared in the context level.

**Q) Is it possible to communicate from an applet to servlet and how many ways and how?**   
A)Yes, there are three ways to communicate from an applet to servlet and they are: a) HTTP Communication(Text-based and object-based) b) Socket Communication c) RMI Communication (You can say, by using URL object open the connection to server and get the InputStream from URLConnection object). Steps involved for applet-servlet communication: step: 1 Get the server URL. URL url = new URL(); step: 2 Connect to the host URLConnection Con = url.openConnection(); step: 3 Initialize the connection Con.setUseCatches(false): Con.setDoOutput(true); Con.setDoInput(true); step: 4 Data will be written to a byte array buffer so that we can tell the server the length of the data. ByteArrayOutputStream byteout = new ByteArrayOutputStream(); step: 5 Create the OutputStream to be used to write the data to the buffer. DataOutputStream out = new DataOutputStream(byteout);

**Q) Why should we go for interservlet communication?**   
A) Servlets running together in the same server communicate with each other in several ways. The three major reasons to use interservlet communication are: a) Direct servlet manipulation - allows to gain access to the other currently loaded servlets and perform certain tasks (through the ServletContext object) b) Servlet reuse - allows the servlet to reuse the public methods of another servlet. c) Servlet collaboration - requires to communicate with each other by sharing specific information (through method invocation)

**Q) How do u implement threads in servlet?**   
A) Internally implemented

**Q) How do you handle DataBase access and in which method of the servlet do you like to create connection.**   
A) init()

**Q) If you want to improve the performance how do you create connections for multiple users?**   
A) Connection Pooling.

**Q) What is connection pooling?**   
A) Class which manages no of user requests for connections to improve the performance.

**Q) What are the different servers available for developing and deploying Servlets?**   
A) a) JRun2.0--Allaire , b) Apache --jserv , c) jwsdk2.0 --sun , d) servletexec , e) Tomcat webserver--tomcat , f)Weblogic AS--BEA Systems , g)NetDynamics5.0--sun , h)Iplanet--sun&netscape , i)Netscape--netscape , g)IBM websphere--IBM , h)oracle--oracle , i)Proton-Pramati technologies.

**Q) Is it possible to call servlet with parameters in the URL?**   
A) Yes. You can call a servlet with parameters in the syntax as (servletName?Param1 = xxx || m2 = yyy).

**Q) What is Servlet chaining?**   
A) Servlet chaining is a technique in which two or more servlets can cooperate in servicing a single request. In servlet chaining, one servlet's output is piped to the next servlet's input. This process continues until the last servlet is reached. Its output is then sent back to the client.

**Q) How do servlets handle multiple simultaneous requests?**   
A) The server has multiple threads that are available to handle requests. When a request comes in, it is assigned to a thread, which calls a service method (for example: doGet(), doPost( ) and service( ) ) of the servlet. For this reason, a single servlet object can have its service methods called by many threads at once.

**Q) My servlet, which ran correctly under the Servlet 2.0 APIs (Java Web Server 1.1.3) is not running under the Servlet 2.1 APIs (Java Web Server 2.0). What's wrong?**   
A) You might have used servlet to servlet communication by using servletcontext methods like getServlet(), getServlets() which are deprecated and returns null from new release that is from servlet2.1 API.

**Q) How can the data within an HTML form be refreshed automatically whenever there is a change in the database?**  
A) JSP is intended for dynamically generating pages. The generated pages can include wml, html, dhtml or whatever you want.  
When you have a generated page, JSP has already made its work. From this moment you have a page.  
If you want automatic refreshing, then this should be accomplished by the technology included in the generated page (JSP will tell only what to include in the page).  
The browser cannot be loaded by extern factors. The browser is the one who fetches url's since the http protocol is request-response based. If a server can reload a browser without its allow, it implies that we could be receiving pages which we haven't asked for from servers.  
May you could use applets and a ServerSocket for receiving incoming signals from the server for changed data in the DB. This way you can load new information inside the applet or try to force a page reload.  
That's a nice idea -- it could use the showDocument() call to reload the current page. It could also use HTTP polling instead of maintaining an expensive socket connection.   
Perhaps (if possible), could be simpler using an automatic JavaScript refreshing function that force page reload after a specified time interval.

**Q) What are the considerations for servlet clustering?**   
A) The clustering promotes high availability and scalability. The considerations for servlet clustering are:  
  
\*Objects stored in a session should be serializable to support in-memory replication of sessions. Also consider the overhead of serializing very large objects. Test the performance to make sure it is acceptable.  
\*Design for idempotence Failure of a request or impatient users clicking again can result in duplicate requests being submitted. So the Servlets should be able to tolerate duplicate requests.  
\*Avoid using instance and static variables in read and write mode because different instances may exist on different JVMs. Any state should be held in an external resource such as a database.  
\*Avoid storing values in a ServletContext. A ServletContext is not serializable and also the different instances may exist in different JVMs.  
\*Avoid using java.io.\* because the files may not exist on all backend machines. Instead use getResourceAsStream().

**Q) How can a servlet refresh automatically if some new data has entered the database?**   
A) You can use a client-side Refresh or Server Push.

**Q) What is the difference between using getSession (true) and getSession (false) methods?**   
A) GetSession (true) - This method will check whether already a session is existing for the user. If a session is existing, it will return that session object, Otherwise it will create new session object and return that object. getSession(false) - This method will check existence of session. If session exists, then it returns the reference of that session object, if not, this methods will return null.

**Q) What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?**   
A) The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.  
The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must start with a "/" and are interpreted as relative to current context root.

**Q) Explain the directory structure of a web application.**   
A) The directory structure of a web application consists of two parts. A private directory called WEB-INF A public resource directory which contains public resource folder. WEB-INF folder consists of 1. web.xml 2. classes directory 3. lib directory

**Q) What are the common mechanisms used for session tracking?**  
1) By URL rewriting Method

2) Using Session object Getting Session form HttpServletRequest object HttpSession session = request.getSession(true); Get a Value from the session session.getValue(session.getId()); Adding values to session cart = new Cart(); session.putValue(session.getId(), cart); At the end of the session, we can inactivate the session by using the following command session.invalidate(); Method

3) Using cookies Method

4) Using hidden fields

**Q) How Can You invoke other web resources (or other servelt / jsp ) ?**   
A) Servlet can invoke other Web resources in two ways: indirect and direct.   
Indirect Way: Servlet will return the resultant HTML to the browser which will point to another Servlet (Web resource)  
Direct Way : We can call another Web resource (Servelt / Jsp) from Servelt program itself, by using RequestDispatcher object.  
You can get this object using getRequestDispatcher("URL") method. You can get this object from either a request or a Context.  
**Example :**

RequestDispatcher dispatcher = request.getRequestDispatcher("/jspsample.jsp");

if (dispatcher != null)

dispatcher.forward(request, response);

**Q) What is preinitialization of a servlet?**   
A) A container doesn’t initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the < load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

**Q) How do I support both GET and POST protocol from the same Servlet?**   
A) The easy way is, just support POST, then have your doGet method call your doPost method: public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException { doPost(req, res); }

**Q) What is the difference between URL encoding, URL rewriting, HTML escaping, and entity encoding?**   
A) **URL Encoding** is a process of transforming user input to a CGI form so it is fit for travel across the network -- basically, stripping spaces and punctuation and replacing with escape characters. URL Decoding is the reverse process. To perform these operations, call java.net.URLEncoder.encode() and java.net.URLDecoder.decode() (the latter was (finally!) added to JDK 1.2, aka Java 2). Example: changing "We're #1!" into "We%27re+%231%21" **URL Rewriting** is a technique for saving state information on the user's browser between page hits. It's sort of like cookies, only the information gets stored inside the URL, as an additional parameter. The HttpSession API, which is part of the Servlet API, sometimes uses URL Rewriting when cookies are unavailable. Example: changing into (or whatever the actual syntax is; I forget offhand) (Unfortunately, the method in the Servlet API for doing URL rewriting for session management is called encodeURL(). Sigh...) There's also a feature of the Apache web server called URL Rewriting; it is enabled by the mod\_rewrite module. It rewrites URLs on their way in to the server, allowing you to do things like automatically add a trailing slash to a directory name, or to map old file names to new file names. This has nothing to do with servlets. For more information, see the Apache FAQ (http://www.apache.org/docs/misc/FAQ.html#rewrite-more-config) .

**Q) Can a servlet maintain a JTA UserTransaction object across multiple servlet invocations?**   
A) No. A JTA transaction must start and finish within a single invocation (of the service() method). Note that this question does not address servlets that maintain and manipulate JDBC connections, including a connection's transaction handling.

**Q) How does the performance of JSP pages compare with that of servlets? How does it compare with Perl scripts?**   
A) The performance of JSP pages is very close to that of servlets. However, users may experience a perceptible delay when a JSP page is accessed for the very first time. This is because the JSP page undergoes a "translation phase" wherein it is converted into a servlet by the JSP engine. Once this servlet is dynamically compiled and loaded into memory, it follows the servlet life cycle for request processing. Here, the jspInit() method is automatically invoked by the JSP engine upon loading the servlet, followed by the \_jspService() method, which is responsible for request processing and replying to the client. Do note that the lifetime of this servlet is non-deterministic - it may be removed from memory at any time by the JSP engine for resource-related reasons. When this happens, the JSP engine automatically invokes the jspDestroy() method allowing the servlet to free any previously allocated resources. Subsequent client requests to the JSP page do not result in a repeat of the translation phase as long as the servlet is cached in memory, and are directly handled by the servlet's service() method in a concurrent fashion (i.e. the service() method handles each client request within a seperate thread concurrently.) There have been some recent studies contrasting the performance of servlets with Perl scripts running in a "real-life" environment. The results are favorable to servlets, especially when they are running in a clustered environment.

**Q) Should I override the service() method?**   
A) No. It provides a fair bit of housekeeping that you'd just have to do yourself. If you need to do something regardless of whether the request is e.g., a POST or a GET, create a helper method and call that at the beginning of e.g., doPost() and doGet().

**Q) Why should I use JSP when there is already servlet technology available for serving dynamic content?**   
A) While JSP may be great for serving up dynamic Web content and separating content from presentation, some may still wonder why servlets should be cast aside for JSP. The utility of servlets is not in question. They are excellent for server-side processing, and, with their significant installed base, are here to stay. In fact, architecturally speaking, you can view JSP as a high-level abstraction of servlets that is implemented as an extension of the Servlet 2.1 API. Still, you shouldn't use servlets indiscriminately; they may not be appropriate for everyone. For instance, while page designers can easily write a JSP page using conventional HTML or XML tools, servlets are more suited for back-end developers because they are often written using an IDE -- a process that generally requires a higher level of programming expertise. When deploying servlets, even developers have to be careful and ensure that there is no tight coupling between presentation and content. You can usually do this by adding a third-party HTML wrapper package like htmlKona to the mix. But even this approach, though providing some flexibility with simple screen changes, still does not shield you from a change in the presentation format itself. For example, if your presentation changed from HTML to DHTML, you would still need to ensure that wrapper packages were compliant with the new format. In a worst-case scenario, if a wrapper package is not available, you may end up hardcoding the presentation within the dynamic content. So, what is the solution? One approach would be to use both JSP and servlet technologies for building application systems.

**Q) How do I send information and data back and forth between applet and servlet using the HTTP protocol?**   
A) Use the standard java.net.URL class, or "roll your own" using java.net.Socket. See the HTTP spec at W3C for more detail. Note: The servlet cannot initiate this connection! If the servlet needs to asynchronously send a message to the applet, then you must open up a persistent socket using java.net.Socket (on the applet side), and java.net.ServerSocket and Threads (on the server side).

**Q) Can I get the path of the current servlet where it lives on the file system (not its URL)?**   
A) Try using: request.getRealPath(request.getServletPath()) An example may be: out.println(request.getRealPath(request.getServletPath()));

**Q) How do I ensure that my servlet is thread-safe?**   
A) This is actually a very complex issue. A few guidelines: 1. The init() method is guaranteed to be called once per servlet instance, when the servlet is loaded. You don't have to worry about thread safety inside this method, since it is only called by a single thread, and the web server will wait until that thread exits before sending any more threads into your service() method. 2. Every new client request generates (or allocates) a new thread; that thread calls the service() method of your servlet (which may in turn call doPost(), doGet() and so forth). 3. Under most circumstances, there is only one instance of your servlet, no matter how many client requests are in process. That means that at any given moment, there may be many threads running inside the service() method of your solo instance, all sharing the same instance data and potentially stepping on each other's toes. This means that you should be careful to synchronize access to shared data (instance variables) using the synchronized keyword. (Note that the server will also allocate a new instance if you register the servlet with a new name and, e.g., new init parameters.) 4. Note that you need not (and should not) synchronize on local data or parameters. And especially you shouldn't synchronize the service() method! (Or doPost(), doGet() et al.) 5. A simple solution to synchronizing is to always synchronize on the servlet instance itself using &quot;synchronized (this) { ... }&quot;. However, this can lead to performance bottlenecks; you're usually better off synchronizing on the data objects themselves. 6. If you absolutely can't deal with synchronizing, you can declare that your servlet &quot;implements SingleThreadModel&quot;. This empty interface tells the web server to only send one client request at a time into your servlet. From the JavaDoc: &quot;If the target servlet is flagged with this interface, the servlet programmer is guaranteed that no two threads will execute concurrently the service method of that servlet. This guarantee is ensured by maintaining a pool of servlet instances for each such servlet, and dispatching each service call to a free servlet. In essence, if the servlet implements this interface, the servlet will be thread safe.&quot; Note that this is not an ideal solution, since performance may suffer (depending on the size of the instance pool), plus it's more difficult to share data across instances than within a single instance. 7. To share data across successive or concurrent requests, you can use either instance variables or class-static variables, or use Session Tracking. 8. The destroy() method is not necessarily as clean as the init() method. The server calls destroy either after all service calls have been completed, or after a certain number of seconds have passed, whichever comes first. This means that other threads might be running service requests at the same time as your destroy() method is called! So be sure to synchronize, and/or wait for the other requests to quit. Sun's Servlet Tutorial has an example of how to do this with reference counting. 9. destroy() can not throw an exception, so if something bad happens, call log() with a helpful message (like the exception). See the &quot;closing a JDBC connection&quot; example in Sun's Tutorial.

**Q) Why there are no constructors in servlets?**   
A) A servlet is just like an applet in the respect that it has an init() method that acts as a constrcutor. Since the servlet environment takes care of instantiating the servlet, an explicit constructor is not needed. Any initialization code you need to run should be placed in the init() method since it gets called when the servlet is first loaded by the servlet container.

**Q) How to handle multiple concurrent database requests/updates when using JDBC with servlets?**   
A) All the dbms provide the facility of locks whenever the data is being modified. There can be two scenarios: 1. Multiple database updates on different rows, if you are using servlets the servlets will open multiple connections for different users. In this case there is no need to do additional programming. 2. If database updates are on the same row then the rows are locked automatically by the dbms, hence we have to send requests to the dbms repetitively until the lock is released by dbms. This issue is dealt with in the JDBC documentation; look up "Transactions" and "auto-commit". It can get pretty confusing.

**Q) What is the difference between GenericServlet and HttpServlet?**   
A) GenericServlet is for servlets that might not use HTTP, like for instance FTP servlets. Of course, it turns out that there's no such thing as FTP servlets, but they were trying to plan for future growth when they designed the spec. Maybe someday there will be another subclass, but for now, always use HttpServlet.

**Q) How do you share session objects between servlets and JSP?**   
A) Sharing sessions between a servlet and a JSP page is straight forward. JSP makes it a little easy by creating a session object and making it availabe already. In a servlet you would have to do it yourself. This is how: //create a session if one is not created already now

HttpSession session = request.getSession(true); //assign the session variable to a value. session.putValue("variable","value"); in the jsp page this is how you get the session value:

**Q) Is there any method to unload a servlet from Web Server memory without restarting the server?**  
A) There is no standard method/mechanism to unload a servlet from memory. Some servers, like JWS, provide the means to load and unload servlets from their administration module. Others, like Tomcat, require you to just replace the WAR file.

**Q) What distinguishes a JavaBean from a Servlet?**   
A) JavaBeans are a set of rules to follow to create reusable software components, or beans. This contains properties and events. At the end you have a component which could be examined by a program (like an IDE) to allow the user of your JavaBean component to configure it and to run in its Java programs. Servlets are Java classes running in a Servlet engine implementing a particular interface: Servlet, forcing you to implement some methods (service()). The servlet is an extension of your web server where this servlet is running on and only lets you know when a user requests a GET or POST calls from a web page to your servlet. So, both have nothing in common except Java.

**Q) How much data we can store in a session object?**   
A) Any amount of data can be stored there because the session is kept on the server side. The only limitation is sessionId length, which shouldn't exceed ~4000 bytes - this limitation is implied by HTTP header length limitation to 4Kb since sessionId may be stored in the cookie or encoded in URL (using "URL rewriting") and the cookie specification says the size of cookie as well as HTTP request (e.g. GET /document.html\n) cannot be longer then 4kb.

**Q) What is the difference between the doGet and doPost methods?**

|  |  |  |
| --- | --- | --- |
| # | **doGet()** | **doPost()** |
| 1 | In doGet() the parameters are appended to the URL and sent along with header information. | In doPost(), on the other hand will (typically) send the information through a socket back to the webserver and it won't show up in the URL bar. |
| 2 | The amount of information you can send back using a GET is restricted as URLs can only be 1024 characters. | You can send much more information to the server this way - and it's not restricted to textual data either. It is possible to send files and even binary data such as serialized Java objects! |
| 3 | doGet() is a request for information; it does not (or should not) change anything on the server. (doGet() should be idempotent) | doPost() provides information (such as placing an order for merchandise) that the server is expected to remember |
| 4 | Parameters are not encrypted | Parameters are encrypted |
| 5 | doGet() is faster if we set the response content length since the same connection is used. Thus increasing the performance | doPost() is generally used to update or post some information to the server.doPost is slower compared to doGet since doPost does not write the content length |
| 6 | doGet() should be idempotent. i.e. doget should be able to be repeated safely many times | This method does not need to be idempotent. Operations requested through POST can have side effects for which the user can be held accountable. |
| 7 | doGet() should be safe without any side effects for which user is held responsible | This method does not need to be either safe |
| 8 | It allows bookmarks. | It disallows bookmarks. |

**Q) What is the difference between encodeRedirectUrl and encodeURL?**   
A) encodeURL and encodeRedirectURL are methods of the HttpResponse object. Both rewrite a raw URL to include session data if necessary. (If cookies are on, both are no-ops.) encodeURL is for normal links inside your HTML pages. encodeRedirectURL is for a link you're passing to response.sendRedirect(). It has slightly different syntax requirements too gory to get into here.

**Q) I am opening a single JDBC connection in my init() method. Do I need to synchronize on the Connection or the Statement object?**   
A) You shouldn't have to. If your JDBC driver supports multiple connections, then the various createStatement methods will give you a thread-safe, reentrant, independent Statement that should work OK, even if other requests/threads are also accessing other Statements on the same Connection. Of course, crossing your fingers never hurts... Many early JDBC drivers were not re-entrant. The modern versions of JDBC drivers should work OK, but there are never any guarantees. Using connection pooling will avoid the whole issue, plus will lead to improved performance. See this FAQ for more information.

**Q) How can I determine the name and version number of the servlet or JSP engine that I am using?**  
A) From within a servlet, you can invoke the ServletContext.getServerInfo() method as follows: String thisServer= getServletConfig().getServletContext().getServerInfo(); If you are using JSP, you can use this expression:

**Q) How can I get the absolute URL of a servlet/JSP page at runtime ?**   
A) You can get all the necessary information to determine the URL from the request object. To reconstruct the absolute URL from the scheme, server name, port, URI and query string you can use the URL class from java.net. The following code fragment will determine your page's absolute URL:

String file = request.getRequestURI();

if (request.getQueryString() != null) {

file += '?' + request.getQueryString();

}

URL reconstructedURL = new URL(request.getScheme(), request.getServerName(), request.getServerPort(), file);

out.println(URL.toString());

**Q) Why do GenericServlet and HttpServlet implement the Serializable interface?**   
A) GenericServlet and HttpServlet implement the Serializable interface so that servlet engines can "hybernate" the servlet state when the servlet is not in use and reinstance it when needed or to duplicate servlet instances for better load balancing. I don't know if or how current servlet engines do this, and it could have serious implications, like breaking references to objects gotten in the init() method without the programmer knowing it. Programmers should be aware of this pitfall and implement servlets which are stateless as possible, delegating data store to Session objects or to the ServletContext. In general stateless servlets are better because they scale much better and are cleaner code.

**Q) How do I automatically reload servlets?**   
A) Depends upon the server's servlet reload properites.

**Q) What is the difference between an applet and a servlet?**   
A) a) Servlets are to servers what applets are to browsers.   
b) Applets must have graphical user interfaces whereas servlets have no graphical user interfaces.

**Q) What is the difference between GET and POST.**   
A) a) doGet() method is used to get information, while doPost( ) method is used for posting information.   
b) doGet() requests can't send large amount of information and is limited to 240-255 characters. However, doPost( )requests passes all of its data, of unlimited length.   
c) A doGet( ) request is appended to the request URL in a query string and this allows the exchange is visible to the client, whereas a doPost() request passes directly over the socket connection as part of its HTTP request body and the exchange are invisible to the client.

**Q) Can we use the constructor, instead of init(), to initialize servlet?**   
A) Yes , of course you can use the constructor instead of init(). There's nothing to stop you. But you shouldn't. The original reason for init() was that ancient versions of Java couldn't dynamically invoke constructors with arguments, so there was no way to give the constructur a ServletConfig. That no longer applies, but servlet containers still will only call your no-arg constructor. So you won't have access to a ServletConfig or ServletContext.

**Q) How many Cookies is supported to the host ?**   
A) User agents accepted to support twenty per host .And its take four Kilobytes each.

**Q) How do you trap the debug the errors in servlets.**   
A) error log file

**Q) where the session data will store?**   
A) session objects

**Q) what is httptunneling?**   
A) It is mechanism of performing both write and read operations using http protocol. It is extending the functionality of htp protocol.

**Q) List out Differences between CGI Perl and Servlet?**   
A) Servlet is Platform independent and Language dependent whereas CGI is Platform dependent and Language independent.

**Q) How can my application get to know when an HttpSession is removed (when it time-outs)?**   
A) Define a class, say SessionTimeoutNotifier, that implements javax.servlet.http. HttpSessionBindingListener. Create a SessionTimeoutNotifier object and add it to the user session. When the session is removed, SessionTimeoutNotifier.valueUnbound() will be called by the servlet engine. You can implement valueUnbound() to do whatever you want.

**Q) How does one choose between overriding the doGet(), doPost(), and service() methods?**   
A) The differences between the doGet() and doPost() methods are that they are called in the HttpServlet that your servlet extends by its service() method when it receives a GET or a POST request from a HTTP protocol request. A GET request is a request to get a resource from the server. This is the case of a browser requesting a web page. It is also possible to specify parameters in the request, but the length of the parameters on the whole is limited. This is the case of a form in a web page declared this way in html: form method="GET" or form. A POST request is a request to post (to send) form data to a resource on the server. This is the case of a form in a web page declared this way in html: form method="POST". In this case the size of the parameters can be much greater. The GenericServlet has a service() method that gets called when a client request is made. This means that it gets called by both incoming requests and the HTTP requests are given to the servlet as they are (you must do the parsing yourself). The HttpServlet instead has doGet() and doPost() methods that get called when a client request is GET or POST. This means that the parsing of the request is done by the servlet: you have the appropriate method called and have convenience methods to read the request parameters. NOTE: the doGet() and doPost() methods (as well as other HttpServlet methods) are called by the service() method. Concluding, if you must respond to GET or POST requests made by a HTTP protocol client (usually a browser) don't hesitate to extend HttpServlet and use its convenience methods. If you must respond to requests made by a client that is not using the HTTP protocol, you must use service().

**Q) How can we use a servlet as a proxy for communications between two applets?**   
A) One way to accomplish this is to have the applets communicate via TCP/IP sockets to the servlet. The servlet would then use a custom protocol to receive and push information between applets. However, this solution does have firewall problems if the system is to be used over and Internet verses an Intranet.

**Q) How do I deal with multi-valued parameters in a servlet?**   
A) Instead of using getParameter() with the ServletRequest, as you would with single-valued parameters, use the getParameterValues() method. This returns a String array (or null) containing all the values of the parameter requested.

**Q) How can I design my servlet/JSP so that query results get displayed on several pages, like the results of a search engine? Each page should display, say, 10 records each and when the next link is clicked, I should see the next/previous 10 records and so on.**   
A) Use a Java Bean to store the entire result of the search that you have found. The servlet will then set a pointer to the first line to be displayed in the page and the number of lines to display, and force a display of the page. The Action in the form would point back to the servlet in the JSP page which would determine whether a next or previous button has been pressed and reset the pointer to previous pointer + number of lines and redisplay the page. The JSP page would have a scriplet to display data from the Java Bean from the start pointer set to the maximum number of lines with buttons to allow previous or next pages to be selected. These buttons would be displayed based on the page number (i.e. if first then don't display previous button).

**Q) How do servlets differ from RMI? What are the advantages and disadvantages of each technology?**   
A) Servlets extend the server-side functionality of a website. Servlets communicate with other application(s) on that server (or any other server) and perform tasks above and beyond the "normal" static HTML document. A servlet can receive a request to get some information through EJB from one or more databases, then convert this data into a static HTML/WML page for the client to see, for example. Even if the servlet talks to many other applications all over the world to get this information, it still looks like it happened at that website. RMI (Remote Method Invocation) is just that - a way to invoke methods on remote machines. It is way for an application to talk to another remote machine and execute different methods, all the while appearing as if the action was being performed on the local machine. Servlets (or JSP) are mainly used for any web-related activity such as online banking, online grocery stores, stock trading, etc. With servlets, you need only to know the web address and the pages displayed to you take care of calling the different servlets (or actions within a servlet) for you. Using RMI, you must bind the RMI server to an IP and port, and the client who wishes to talk to the remote server must know this IP and port, unless of course you used some kind of in-between lookup utility, which you could do with (of all things) servlets.

**Q) What is HTTP tunneling, in the general sense?**   
A) HTTP tunneling is a general technique whereby arbitrary data may be sent via an HTTP connection to and from CGI scripts or Java Servlets on a Web server. This is done by serializing the data to be transmitted into a stream of bytes, and sending an HTTP message with content type "application/octet-stream". HTTP tunneling is also referred to as Firewall tunneling.

**Q) How do I handle FORMs with multiple form elements (e.g. radio buttons) using the same name?**   
A) For radio buttons, the HTML spec assumes that a given group of buttons will have the same NAME and different VALUEs; the browser makes sure that only one button per group name will be selected (at most). So you can just call request.getParameter("groupname").   
input type="radio" name="topping" value="cheese" checked -Cheese  
input type="radio" name="topping" value="pepperoni" -Pepperoni  
input type="radio" name="topping" value="anchovies" - Anchovies  
If the user selects "Pepperoni" then request.getParameter("topping") will return the string "pepperoni". For lists using the "select multiple" FORM tag, multiple values can be returned for the same parameter name. When that can happen, use request.getParameterValues("param") which returns a String[] you can iterate through.  
It's bad form (so to speak), but you can also duplicate other element types, like   
Name 1: input type="text" name="name" value="Dick"  
Name 2: input type="text" name="name" value="Jane"  
These also get returned in an array by request.getParameterValues().

**Q) What is meant by the term "business logic"?**   
A) "Business logic" is just a fancy way of saying "code." :-) More precisely, in a three-tier architecture, business logic is any code that is not specifically related to storing and retrieving data (that's "data storage code"), or to formatting data for display to the user (that's "presentation logic"). It makes sense, for many reasons, to store this business logic in separate objects; the middle tier comprises these objects. However, the divisions between the three layers are often blurry, and business logic is more of an ideal than a reality in most programs. The main point of the term is, you want somewhere to store the logic and "business rules" (another buzzword) of your application, while keeping the division between tiers clear and clean.

**Q) What is a servlet engine?**   
A) A "servlet engine" is a program that plugs in to a web server and runs servlets. The term is obsolete; the preferred term now is "servlet container" since that applies both to plug-in engines and to stand-alone web servers that support the Servlet API.

**Q) Can I send multiple responses for a single request?**   
A) No. That doesn't even make sense. You can, however, send a "redirect", which tells the user's browser to send another request, possibly to the same servlet with different parameters. Search this FAQ on "redirect" to learn more.

**Q) How do I capture a request and dispatch the exact request (with all the parameters received) to another URL?**  
A) As far as i know it depends on the location of the next target url.   
\* If the next servlet url is in the same host, then you can use the forward method.  
Here is an example code about using forward:   
\* RequestDispatcher rd = null;  
\* String targetURL = "target\_servlet\_name";  
\* ServletContext ctx = this.getServletContext();  
\* rd = ctx.getRequestDispatcher(targetURL);  
\* rd.forward(request, response);

**Q) What is FORM based login and how do I use it? Also, what servlet containers support it?**  
A) Form based login is one of the four known web based login mechanisms. For completeness I list all of them with a description of their nature:   
  
1. HTTP Basic Authentication   
An authentication protocol defined within the HTTP protocol (and based on headers). It indicates the HTTP realm for which access is being negotiated and sends passwords with base64 encoding; therefore it is not very secure. (See RFC2068 for more information.)   
2. HTTP Digest Authentication   
like HTTP Basic Authentication, but with the password transmitted in an encrypted form. It is more secured than Basic, but less then HTTPS Authentication which uses private keys. Yet it is not currently in widespread use.   
3. HTTPS Authentication (SSL Mutual Authentication)  
This security mechanism provides end user authentication using HTTPS (HTTP over SSL). It performs mutual (client & server) certificate based authentication with a set of different cipher suites.  
4. Form Based Login   
A standard HTML form (static, Servlet/JSP or script generated) for logging in. It can be associated with protection or user domains, and is used to authenticate previously unauthenticated users.  
The major advantage is that the look and feel of the login screen can be controlled (in comparison to the HTTP browsers' built in mechanisms). To support 1., 3., and 4. of these authentication mechanisms is a requirement of the J2EE Specification (as of v1.2, 3.4.1.3 Required Login Mechanisms). (HTTP Digest Authentication is not a requirement, but containers are encouraged to support it.) You can also see section 3.3.11.1 of the J2EE Specs. (User Authentication, Web Client) for more detailed descriptions of the mechanisms. Thus any Servlet container that conforms to the J2EE Platform specification should support form based login. To be more specific, the Servlet 2.2 Specification describes/specifies the same mechanisms in 11.5 including form based login in 11.5.3. This section (11.5.3) describes in depth the nature, the requirements and the naming conventions of form based login and I suggest to take a look at it. Here is a sample of a conforming HTML login form:   
'form method="POST" action="j\_security\_check" '  
'input type="text" name="j\_username"'  
'input type="password" name="j\_password">'  
'/form'   
Known Servlet containers that support FORM-based login are:   
\* iPlanet Application Server   
\* Tomcat (the reference implementation of the Java Servlet API)

**Q) What is a web application (or "webapp")?**  
A) A web application is a collection of servlets, html pages, classes, and other resources that can be bundled and run on multiple containers from multiple vendors. A web application is rooted at a specific path within a web server.

**Q) Can there be more than one instance of a servlet at one time ?**   
A) It is important to note that there can be more than one instance of a given Servlet class in the servlet container. For example, this can occur where there was more than one servlet definition that utilized a specific servlet class with different initialization parameters. This can also occur when a servlet implements the SingleThreadModel interface and the container creates a pool of servlet instances to use.

**Q) Is there any way to determine the number of concurrent connections my servlet engine can handle?**   
A) Depends on whether or not your servlet container uses thread pooling. If you do not use a thread pool, the number of concurrent connections accepted by Tomcat 3.1, for example, is 10. This you can see for yourself by testing a servlet with the Apache JMeter tool. However, if your servlet container uses a thread pool, you can specify the number of concurrent connections to be accepted by the container. For Tomcat 3.1, the information on how to do so is supplied with the documentation in the TOMCAT\_HOME/doc/uguide directory.

**Q) How can I explicitly unload a servlet or call the destroy method?**   
A) In general, you can't. The Servlet API does not specify when a servlet is unloaded or how the destroy method is called. Your servlet engine (ie the implementation of the interfaces in the JSDK) might provide a way to do this, probably through its administration interface/tool (like Webshpere or JWS). Most servlet engines will also destroy and reload your servlet if they see that the class file(s) have been modified.

**Q) What is a servlet bean?**   
A) A servlet bean is a serializable servlet that follows the JavaBeans component architecture, basically offering getter/setter methods.   
As long as you subclass GenericServlet/HttpServlet, you are automatically Serializable. If your web server supports them, when you install the servlet in the web server, you can configure it through a property sheet-like interface.

**Q) Why do we need to call super.init(config) in the init method of a servlet?**   
A) Just do as you're told and you won't get hurt! :-) Because if you don't, then the config object will get lost. Just extend HttpServlet, use init() (no parameters) and it'll all work ok. From the Javadoc: init() - A convenience method which can be overridden so that there's no need to call super.init(config).

**Q) How can I change the port of my Java Web Server from 8080 to something else?**   
A) It is very simple. JAVA WEB SERVER comes with remote Web administration tool. You can access this with a web browser. Administration tool is located on port 9090 on your web server. To change port address for web server:   
1. Access tool (http://hostname:9090)  
2. Enter User Id/Password (by default it is admin/admin)  
3. Select service (Web service)  
4. Click on "manage" button. You will get a popup screen with all settings.  
5. Click on network tree node, On right hand side you will get text box for entering port no.  
6. Change port number with desire one.  
7. click on restart button.

**Q) What is the most efficient way of serving pages from a Java object?**  
A) There you have a clear winner in the Servlet. Althought if you are going to change the static content of the page often is going to be a pain because you will have to change Java code. The second place in speed is for JSP pages. But, depending on your application, the difference in speed between JSP pages and raw servlets can be so small that is not worth the extra work of servlet programming. 2-.

**Q) What is the most efficient way of accessing a database from Java?**  
A) If JDBC is the way you want to go the I'd suggest to pick as many drivers as you can (II,III,IV or wathever) and benchmark them. Type I uses a JDBC/ODBC bridge and usually has lousy performance. Again, go for the simplest (usually type IV driver) solution if that meets you performance needs.   
For database applications, the performance bottleneck is usually the database, not the web server/engine. In this case, the use of a package that access JDBC with connection pooling at the application level used from JSP pages (with or withouth beans as middleware) is the usual choice. Of course, your applications requirements may vary.

**Q) For an HTML FORM with multiple SUBMIT buttons, how can a servlet respond differently for each button?**   
A) The servlet will respond differently for each button based on the html that you have placed in the HTML page. Let's explain. For a submit button the HTML looks like ' input type=submit name="Left" value="left" '. A servlet could extract the value of this submit by using the getParameter("Left") from the HttpRequest object. It follows then that if you have HTML within a FORM that appears as:   
input type=submit name="Direction" value="left"  
input type=submit name="Direction" value="right"  
input type=submit name="Direction" value="up"  
input type=submit name="Direction" value="down"  
Then the getParameter("Direction") from the HttpRequest would extract the value pressed by the user, either "left", "right", "up" or "down". A simple comparision in the servlet with the these values could occur and processing based on the submit button would be performed. Similiarly,for submit buttons with different names on a page, each of these values could be extracted using the getParameter() call and acted on. However, in a situation where there are multiple buttons, common practice would be to use one name and multiple values to identify the button pressed.

**Q) How can I call a servlet from a JSP page? How can I pass variables from the JSP that the servlet can access?**  
A) You can use   
'jsp:forward page="/relativepath/YourServlet" ' or response.sendRedirect("http://path/YourServlet").   
Variables can be sent as:   
jsp:forward page=/relativepath/YourServlet  
jsp:param name="name1" value="value1"   
jsp:param name="name2" value="value2"   
/jsp:forward  
You may also pass parameters to your servlet by specifying response.sendRedirect("http://path/YourServlet?param1=val1").

**Q) What are the types of ServletEngines?**   
A) Standalone ServletEngine: A standalone engine is a server that includes built-in support for servlets. Add-on ServletEngine: Its a plug-in to an existing server.It adds servlet support to a server that was not originally designed with servlets in mind. Embedded ServletEngine: it is a lightweight servlet deployment platform that can be embedded in another application.that application become true server.

**Q) How do I separate presentation (HTML) from business logic (Java) when using servlets?**   
A) Almost anybody who has ever written a servlet can identify with this one. We all know it's bad for to embed HTML code in our java source; it's lame to have to recompile and re-deploy every time you want an HTML element to look a bit different. But what are our choices here? There are two basic options;   
1. Use JSP: Java Server Pages allows you to embed Java code or the results of a servlet into your HTML. You could, for instance, define a servlet that gives a stock quote, then use the 'servlet' tag in a JSP page to embed the output. But then, this brings up the same problem; without discipline, your content/presentation and program logic are again meshed. I think the ideal here is to completely separate the two.  
2. Use a templating/parsing system: Hmm...I know you're about to rant about re-inventing the wheel, but it's not that bad (see below). Plus, it really does pay to take this approach; you can have a group of programmers working on the Java code, and a group of HTML producers maintaining the interface. So now you probably want to know how to do it...so read on.  
Use SSI!  
Remember SSI? It hasn't gotten much attention in recent years because of embeddable scripting languages like ASP and JSP, but it still remains a viable option. To leverage it in the servlet world, I believe the best way is to use an API called SSI for Java from Areane. This API will let you emulate SSI commands from a templating system, and much more. It will let you execute any command on any system, including executing java classes! It also comes with several utility classes for creating stateful HTML form elements, tables for use with iteration, and much more. It's also open source, so it's free and you can tweak it to your heart's content! You can read the SSI for Java documentation for detailed info, but the following is an example of its use.  
Here's the servlet:   
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import java.io.\*;  
import java.util.\*;  
import com.areane.www.ssi.\*;  
  
public class SSITemplatingServlet extends HttpServlet {  
private String templateFilesDirectory = "d:\\projects\\idemo\\templates\\";   
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException, FileNotFoundException {doPost(req, res);  
}  
public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException, FileNotFoundException {  
HttpSession ses = req.getSession(true);  
Properties context = null;  
if((context = (Properties)ses.getValue("user.context")) == null) {   
context = new Properties();  
}  
//Write parameters to Properties object  
Enumeration paramNames = req.getParameterNames();  
String curName, curVal;  
while(paramNames.hasMoreElements()) {  
curName = (String)paramNames.nextElement();  
curVal = req.getParameter(curName);  
context.setProperty(curName, curVal);  
}  
//Save the values to the session  
ses.putValue("user.context", context);  
//Parse the page and stream to the client  
String templateName = req.getParameter("template");   
res.setContentType("text/html");  
SsiPage page = SsiParser.parse(this.templateFilesDirectory + templateName);   
page.write(res.getWriter(), context);   
page = null;  
}  
}  
Now, just create a template file, pass the servlet the template file name, and have at it!

**Q) How are Servlets and JSP Pages related?**   
A) JSP pages are focused around HTML (or XML) with Java codes and JSP tags inside them. When a web server that has JSP support is asked for a JSP page, it checks to see if it has already compiled the page into a servlet. Thus, JSP pages become servlets and are transformed into pure Java and then compiled, loaded into the server and executed.

**Q) How can I pass data retrieved from a database by a servlet to a JSP page?**   
A) One of the better approaches for passing data retrieved from a servlet to a JSP is to use the Model 2 architecture as shown below: Basically, you need to first design a bean which can act as a wrapper for storing the resultset returned by the database query within the servlet. Once the bean has been instantiated and initialized by invoking its setter methods by the servlet, it can be placed within the request object and forwarded to a display JSP page as follows:   
com.foo.dbBean bean = new com.foo.dbBean();  
//call setters to initialize bean  
req.setAttribute("dbBean", bean);  
url="..."; //relative url for display jsp page  
ServletContext sc = getServletContext();  
RequestDispatcher rd = sc.getRequestDispatcher(url);  
rd.forward(req, res);  
The bean can then be accessed within the JSP page via the useBean tag as:   
'jsp:useBean id="dbBean" class="com.foo.dbBean" scope="request" '  
...  
<% //iterate through the rows within dbBean and  
//access the values using a scriptlet  
%> Also, it is best to design your application such that you avoid placing beans into the session unless absolutely necessary. Placing large objects within the session imposes a heavy burden on the performance of the servlet engine. Of course, there may be additional design considerations to take care of - especially if your servlets are running under a clustered or fault-tolerant architecture.

**Q) How can I use a servlet to generate a site using frames?**   
A) In general, look at each frame as a unique document capable of sending its own requests and receiving its own responses. You can create a top servlet (say, FrameServlet) that upon invocation creates the frame layout you desire and sets the SRC parameters for the frame tags to be another servlet, a static page or any other legal value for SRC.   
public void doGet(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {   
response.setContentType("text/html");   
PrintWriter out = new PrintWriter (response.getWriter());   
out.println("<html>");  
out.println("<head' Your Title'/head>");  
out.println("<frameset rows=12%,70%,\* cols=\*>");  
out.println("<frame src=/servlets/MenuServlet name=frm\_1>");  
out.println("<frame src=/servlets/DummyServlet?mode=full name=frm\_2>");  
out.println("<frame src=/servlets/DummyServlet?mode=small name=frm\_3>");  
out.println("</frameset'");  
out.println("<body>");  
out.println("</body></html>");  
out.close();}  
Where MenuServlet and DummyServlet provide content and behavior for the frames generated by FrameServlet.

**Q) How can I measure the file downloading time using servlets?**   
A) ServletOutputStream out = response.getOutputStream();  
String filename = getServletContext().getRealPath(request.getQueryString());  
FileInputStream fin = new FileInputStream(filename);  
long start = System.currentTimeMillis();  
byte data[] = new byte[1024]; int len = 0;  
while ((len = fin.read(data)) > 0) {  
out.write(data, 0, len);}  
out.flush();  
long stop = System.currentTimeMills();  
log("took " + (stop - start) + "ms to download " + filename);

**Q) How do I make servlet aliasing work with Apache+Tomcat?**   
A) When you use Tomcat standalone as your web server, you can modify the web.xml in $TOMCAT\_HOME/webapps/myApp/WEB-INF to add a url-pattern:   
< web-app>  
< servlet> < servlet-name> myServlet </servlet-name>  
< servlet-class> myServlet </servlet-class> </servlet>  
< servlet-mapping> <servlet-name> myServlet </servlet-name>  
< url-pattern>/jsp-bin/\* </url-pattern> </servlet-mapping>  
</web-app>  
This will let you use: http://webserver:8080/myApp/jsp-bin/stuff.html instead of: http://webserver:8080/myApp/servlet/myServlet/stuff.html But it won't work on port 80 if you've integrated Tomcat with Apache. Graeme Wallace provided this trick to remedy the situation. Add the following to your tomcat-apache.conf (or to a static version of it, since tomcat re-generates the conf file every time it starts):   
< LocationMatch /myApp/jsp-bin/\* > SetHandler jserv-servlet</LocationMatch>  
This lets Apache turn over handling of the url pattern to your servlet.

**Q) What is inter-servlet communication?**   
A) As the name says it, it is communication between servlets. Servlets talking to each other. [There are many ways to communicate between servlets, including   
\* Request Dispatching   
\* HTTP Redirect   
\* Servlet Chaining   
\* HTTP request (using sockets or the URLConnection class)   
\* Shared session, request, or application objects (beans)   
\* Direct method invocation (deprecated)   
\* Shared static or instance variables (deprecated)   
Basically interServlet communication is acheived through servlet chaining. Which is a process in which you pass the output of one servlet as the input to other. These servlets should be running in the same server.   
e.g. ServletContext.getRequestDispatcher(HttpRequest, HttpResponse).forward("NextServlet") ; You can pass in the current request and response object from the latest form submission to the next servlet/JSP. You can modify these objects and pass them so that the next servlet/JSP can use the results of this servlet.   
There are some Servlet engine specific configurations for servlet chaining. Servlets can also call public functions of other servlets running in the same server. This can be done by obtaining a handle to the desired servlet through the ServletContext Object by passing it the servlet name ( this object can return any servlets running in the server). And then calling the function on the returned Servlet object.   
e.g. TestServlet test= (TestServlet)getServletConfig().getServletContext().getServlet("OtherServlet"); otherServletDetails= Test.getServletDetails(); You must be careful when you call another servlet's methods. If the servlet that you want to call implements the SingleThreadModel interface, your call could conflict with the servlet's single threaded nature. (The server cannot intervene and make sure your call happens when the servlet is not interacting with another client.) In this case, your servlet should make an HTTP request to the other servlet instead of direct calls.   
Servlets could also invoke other servlets programmatically by sending an HTTP request. This could be done by opening a URL connection to the desired Servlet.

**Q) How can I share data between two different web applications?**   
A) Different servlets may share data within one application via ServletContext. If you have a compelling to put the servlets in different applications, you may wanna consider using EJBs.

**Q) What is a Servlet Filter?**   
A) A filter is basically a component that is invoked whenever a resource is invoked for which the filter is mapped. The resource can be something like a servlet, or a URL pattern. A filter normally works on the request, response, or header attributes, and does not itself send a response to the client.

**Q) How can I write an "error page" -- that is, a servlet or JSP to report errors of other servlets?**   
A) The Servlet 2.2 specification allows you to specify an error page (a servlet or a JSP) for different kinds of HTTP errors or ServletExceptions. You can specify this in deployment descriptor of the web application as:   
< error-page>  
< exception-type>FooException</exception-type>  
< location>/error.jsp</location>  
</error-page>  
where FooException is a subclass of ServletException.   
The web container invokes this servlet in case of errors, and you can access the following information from the request object of error servlet/JSP: error code, exception type, and a message.

**Q) How can I pass data from a servlet running in one context (webapp) to a servlet running in another context?**   
A) There are three ways I can think of off the top of my head:   
1. Store the information you want to share in a persistant format, such as in a file system or database. That way, any servlet that is running in a JVM that can "see" these resources can get to this information.   
2. If persisting this information is not an option, you can bind this information to a context that is accessible to all servlet contexts, such as the application server's context. This way, you can keep the data you want to share in memory.   
3. Use the old fashion way of passing information to a servlet - HTTP. One servlet could foward a request to another servlet and include the data that needs to be shared as parameters in the request.

**Q) I have a global variable in a servlet class. What will happen to this global variable if two requests hit on the same time?**   
A) The best way to establish a default occurrence (the servlet handles a request at a time) is to synchronize the access to the global variable or alternatively to create a servlet that implements the SingleThreadModel interface.

**Q) How do I pass a request object of one servlet as a request object to another servlet?**   
A) Use a Request Dispatcher.

**Q) How can I make a POST request through response.sendRedirect() or response.setStatus() and response.setHeader() methods?**   
A) You can't. It's a fundamental limitation of the HTTP protocol. You'll have to figure out some other way to pass the data, such as   
\* Use GET instead   
\* Make the POST from your servlet, not from the client   
\* Store data in cookies instead of passing it via GET/POST

**Q) How can you embed a JavaScript within servlets / JSP pages?**   
A) You don't have to do anything special to include JavaScript in servlets or JSP pages. Just have the servlet/JSP page generate the necessary JavaScript code, just like you would include it in a raw HTML page. The key thing to remember is it won't run in the server. It will run back on the client when the browser loads the generate HTML, with the included JavaScript.

**Q) What is the difference between request attributes, session attributes, and ServletContext attributes?**   
A) A ServletContext attribute is an object bound into a context through ServletContext.setAttribute() method and which is available to ALL servlets (thus JSP) in that context, or to other contexts via the getContext() method. By definition a context attribute exists locally in the VM where they were defined. So, they're unavailable on distributed applications.   
Session attributes are bound to a session, as a mean to provide state to a set of related HTTP requests. Session attributes are available ONLY to those servlets which join the session. They're also unavailable to different JVMs in distributed scenarios. Objects can be notified when they're bound/unbound to the session implementing the HttpSessionBindingListener interface.   
Request attributes are bound to a specific request object, and they last as far as the request is resolved or while it keep dispatched from servlet to servlet. They're used more as comunication channel between Servlets via the RequestDispatcher Interface (since you can't add Parameters...) and by the container. Request attributes are very useful in web apps when you must provide setup information between information providers and the information presentation layer (a JSP) that is bound to a specific request and need not be available any longer, which usually happens with sessions without a rigorous control strategy.   
Thus we can say that context attributes are meant for infra-structure such as shared connection pools, session attributes to contextual information such as user identification, and request attributes are meant to specific request info such as query results.

**Q) Can Tomcat be configured to interpret all, or selected, .html files within a given context as JSP? Or, do JSP files have to end with a .jsp extension?**   
A) yes you can do that by modifying the web.xml file. You will have to invoke the org.apache.jasper.runtime.JspServlet for all the requests having extension .html. You can do that by changing the Servlet mapping code:  
< servlet-mapping>  
< servlet-name> jsp</servlet-name>  
< url> \*.html</url>  
</servlet-mapping>  
And comment out the following block  
< mime-mapping>  
< extension> html</extension>  
< mime-type> text/html</mime-type>  
</mime-mapping>

**Q) How do I write to a log file using JSP under Tomcat? Can I make use of the log() method for this?**   
A) Yes, you can use the Servlet API's log method in Tomcat from within JSPs or servlets. These messages are stored in the server's log directory in a file called servlet.log.

**Q) How can I use a servlet to print a file on a printer attached to the client?**   
A) The security in a browser is designed to restrict you from automating things like this. However, you can use JavaScript in the HTML your servlet returns to print a frame. The browser will still confirm the print job with the user, so you can't completely automate this. Also, you'll be printing whatever the browser is displaying (it will not reliably print plug-ins or applets), so normally you are restricted to HTML and images.   
The JavaScript source code for doing this is:   
< input type="button" onClick="window.print(0)" value="Print This Page">

**Q) What are the different cases for using sendRedirect() vs. getRequestDispatcher()?**   
A) When you want to preserve the current request/response objects and transfer them to another resource WITHIN the context, you must use getRequestDispatcher or getNamedDispatcher. If you want to dispatch to resources OUTSIDE the context, then you must use sendRedirect. In this case you won't be sending the original request/response objects, but you will be sending a header asking to the browser to issue a request to the new URL. If you don't need to preserve the request/response objects, you can use either.

**Q) What is the difference between in-process and out-of-process servlet containers?**  
A) The in-process Servlet containers are the containers which work inside the JVM of Web server, these provides good performance but poor in scalibility.  
The out-of-process containers are the containers which work in the JVM outside the web server. poor in performance but better in scalibility  
In the case of out-of-process containers, web server and container talks with each other by using the some standard mechanism like IPC.  
In addition to these types of containers, there is 3rd type which is stand-alone servlet containers. These are an integral part of the web server.

**Q) Which servlet containers have persistent session support? Specifically, does Tomcat 3.1?**   
A) All servlet containers that implement the Servlet 2.2 API must provide for session tracking through either the use of cookies or through URL rewriting. All Tomcat servlet containers support session tracking.

**Q) How can I set a servlet to load on startup of the container, rather than on the first request?**   
A) The Servlet 2.2 spec defines a load-on-startup element for just this purpose. Put it in the < servlet> section of your web.xml deployment descriptor. It is either empty (< load-on-startup/>) or contains "a positive integer indicating the order in which the servlet should be loaded. Lower integers are loaded before higher integers. If no value is specified, or if the value specified is not a positive integer, the container is free to load it at any time in the startup sequence."   
For example,   
< servlet>  
< servlet-name> foo</servlet-name>  
< servlet-class> com.foo.servlets.Foo</servlet-class>  
< load-on-startup> 5</load-on-startup>  
</servlet>  
Some servlet containers also have their own techniques for configuring this; please submit feedback with information on these.

**Q) Can I use JAAS as the authentication technology for servlets ?**   
A) Yes, JAAS can be used as authentication technology for servlets. One important feature of JAAS is pure Java implementation. The JAAS infrastructure is divided into two main components: an authentication component and an authorization component. The JAAS authentication component provides the ability to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet.

**Q) Does the RequestDispatcher expect a relative URL to be relative to the originally-called servlet or to the current servlet (if different)?**   
A) Since the RequestDispatcher will be passing the control (request object and response object) from the current Servlet, the relative URL must be relative to the current servlet. The originally called servlet has passed the control to the current servlet, and now current servlet is acting as controller to other resourses.

**Q) What is a Servlet Context?**   
A) Servlet Context is a grouping under which related servlets (and JSPs and other web

resources) run. They can share data, URL namespace, and other resources. There can be multiple contexts in a single servlet container. The ServletContext object is used by an individual servlet to "call back" and obtain services from the container (such as a request dispatcher). Read the JavaDoc for javax.servlet.ServletContext for more information. You can maintain "application global" variables by using Servlet Context Attributes.

**Q) Is it possible to write a servlet that acts as a FTP server?**   
A) Yes. It would spawn a thread that opens a ServerSocket, then listens for incoming connections and speaks the FTP protocol.

**Q) I call a servlet as the action in a form, from a jsp. How can I redirect the response from the servlet, back to the JSP? (RequestDispatcher.forward will not help in this case, as I do not know which resource has made the request. request.getRequestURI will return the uri as contained in the action tag of the form, which is not what is needed.)**   
A) You'll have to pass the JSP's URI in to the servlet, and have the servlet call sendRedirect to go back to the JSP. For example:   
< FORM ACTION="/foo/myservlet">  
< INPUT TYPE="HIDDEN" NAME="redirect" VALUE="/foo/thisjsp.jsp">  
Shoe size: <INPUT NAME="shoesize">  
< INPUT TYPE="SUBMIT">  
</FORM>  
Then in the servlet...   
response.sendRedirect(request.getParameter("redirect"));

**Q) Is there a way to disable a user's ability to double-click a submit image/button (and therefore submitting duplicate data -- multiple submits)? Is there a way to do this with Javascript?**   
A) Give the submit image (or button) an onClick() handler. Have the handler check if a flag is set and if not set the flag and submit the form and then clear the form.

**Q) I am using the RequestDispatcher's forward() method to redirect to a JSP. The problem is that the jsp's url is now relative to the servlet's url and all my url's in the jsp such as < img src="pic.gif" > will be corrupt. How do I solve this problem?**   
A) You can use absolute urls like:   
< BODY >   
< % String base = request.getContextPath(); % >   
< IMG src="< %=base% > /img/pic.gif" >   
< /BODY >   
or write out a BASE tag like:   
< % String base = request.getContextPath(); % >  
< HEAD >   
< BASE HREF="< %=base% > " >   
< /HEAD >   
< BODY >   
< IMG src="img/pic.gif" >   
< /BODY >   
That should take care of the problem.

**Q) What is the difference between static variables and instance variables in a servlet?**   
A) According to the Java Language definition, a static variable is shared among all instances of a class, where a non-static variable -- also called an instance variable -- is specific to a single instance of that class.   
According to the Servlet specification, a servlet that does not declare SingleThreadModel usually has one and only one instance, shared among all concurrent requests hitting that servlet.  
That means that, in servlets (and other multithreaded applications), an instance variable behaves very much like a static variable, since it is shared among all threads. You have to be very careful about synchronizing access to shared data.   
The big difference between instance variables and static variables comes when you have configured your servlet engine to instantiate two instances of the same servlet class, but with different init parameters. In this case, there will be two instances of the same servlet class, which means two sets of instance variables, but only one set of static variables.   
Remember that you can store data in lots of different places in a servlet. To with:  
\* Local variables - for loop iterators, result sets, and so forth   
\* Request attributes - for data that must be passed to other servlets invoked with the RequestDispatcher   
\* Session attributes - persists for all future requests from the current user only   
\* Instance variables - for data that persists for the life of the servlet, shared with all concurrent users  
\* Static variables - for data that persists for the life of the application, shared with all concurrent users -- including any other servlet instances that were instantiated with different init parameters   
\* Context attributes - for data that must persist for the life of the application, and be shared with all other servlets

**Q) Suppose I have 2 servers, server1 and server2. How can I take data in a cookie from server1, and send it to server2?**   
A) You'll have to create a (new) similar cookie on server 2.   
Have a ReadCookieServlet running on server1 that-   
\* Reads the cookie, using request.getCookies()   
\* Redirects to WriteCookieServlet running on server2, passing the cookie name, value and expiration date as request parameters, using response.sendRedirect().   
Have a WriteCookieServlet running on server2 that-   
\* Reads the cookie name, value and expiration date request parameters, using request.getParameter().   
\* Creates a similar cookie, using response.addCookie().

**Q) What is the difference between apache webserver, java webserver and tomcat server?**   
A) Apache is an HTTP server written in C that can be compiled and run on many platforms. Java WebServer is an HTTP server from Sun written in Java that also supports Servlets and JSP. Tomcat is an open-source HTTP server from the Apache Foundation, written in Java, that supports Servlets and JSP. It can also be used as a "plug-in" to native-code HTTP servers, such as Apache Web Server and IIS, to provide support for Servlets (while still serving normal HTTP requests from the primary, native-code web server).

**Q) When building web applications, what are some areas where synchronization problems arrise?**   
A) In general, you will run into synchronization issues when you try to access any shared resource. By shared resource, I mean anything which might be used by more than one request.   
Typical examples include:   
\* Connections to external servers, especially if you have any sort of pooling.   
\* Anything which you include in a HttpSession. (Your user could open many browser windows and make many simultaneous requests within the one session.)   
\* Log destinations, if you do your own logging from your servlets.

**Q) What is a request dispatcher and how does it work?**   
A) A RequestDispatcher object can forward a client's request to a resource or include the resource itself in the response back to the client. A resource can be another servlet, or an HTML file, or a JSP file, etc.   
You can also think of a RequestDispatcher object as a wrapper for the resource located at a given path that is supplied as an argument to the getRequestDispatcher method.   
For constructing a RequestDispatcher object, you can use either the ServletRequest.getRequestDispatcher() method or the ServletContext.getRequestDispatcher() method. They both do the same thing, but impose slightly different constraints on the argument path. For the former, it looks for the resource in the same webapp to which the invoking servlet belongs and the pathname specified can be relative to invoking servlet. For the latter, the pathname must begin with '/' and is interpreted relative to the root of the webapp.   
To illustrate, suppose you want Servlet\_A to invoke Servlet\_B. If they are both in the same directory, you could accomplish this by incorporating the following code fragment in either the service method or the doGet method of Servlet\_A:   
RequestDispatcher dispatcher = getRequestDispatcher("Servlet\_B");  
dispatcher.forward( request, response );  
where request, of type HttpServletRequest, is the first parameter of the enclosing service method (or the doGet method) and response, of type HttpServletResponse, the second. You could accomplish the same by   
RequestDispatcher dispatcher=getServletContext().getRequestDispatcher( "/servlet/Servlet\_B" );  
dispatcher.forward( request, response );

**Q) What are the main differences between Servlets and ISAPI?**   
A) The first difference is obviously that Servlets is the technology from Sun Microsystems and ISAPI is from Microsoft. Other Differences are:   
\* Servlet is a simple .class file and ISAPI is a DLL   
\* Servlets run in the Servlet containers and may be in-process or out of process. ISAs run in the same address space as the HTTP server   
\* Servlet container preprocesses and postprocesses the data communication between the client and server. ISAPI Filters provide the capability of pre-processing and post-processing of all data sent between the client and the server   
\* Java is the only choice for writing Servlets, VC++/MFC is used to write ISAPI code   
\* Servlets works on most of the Web servers plus third party containers can be integrated with other web servers to provide servlets on them. ISAPI works on only ISAPI-compliant Web server (for example, Microsoft Internet Information Server)   
\* Servlets can connect to the Databases through JDBC as well as jdbc-odbc bridges. ISAPI can connect to Databases through only ODBC  
\* Servlets have access to many server-side technologies like EJB and etc. ISAPI is limited in scope   
\* Multiple commands can be implemented in a servlet by using pathinfo. ISAPI allows multiple commands in one DLL, implemented as member functions of the CHttpServer object in the DLL.   
\* Content generation and content presentation can be done seperately in Servlets with the help of JSP. ISAPI code has to generate HTML code itself.

**Q) Can I associate a servlet with a particular mime-type, so if the client requests a file of that type, my servlet will be executed?**   
A) In web.xml you can use a mime-mapping to map the type with a certain extension and then map the servlet to that extension.  
e.g.   
< mime-mapping >   
< extension > zzz < /extension >   
< mime-type > text/plain < /mime-type >   
< /mime-mapping >   
< servlet-mapping >   
< url > \*.zzz < /url >   
< servlet-name > MyServlet< /servlet-name >   
< /servlet-mapping >   
So, when a file for type zzz is requested, the servlet gets called.

**Q) I want my servlet page to redirect to a login page if the session has timed out. How can I know if my session has timed out?**   
A) If the servlet engine does the time-out, following code should help you:   
//assume you have a HttpServletRequest request  
if(request.getSession(false)==null) {  
//no valid session (timeouted=invalid)  
//code to redirect to login page  
}

**Q) Are singleton/static objects shared between servlet contexts?**   
A) It depends on from where the class is loaded. The classes loaded from context's WEB-INF directory are not shared by other contexts, whereas classes loaded from CLASSPATH are shared. So if you have exactly the same DBConnection class in WEB-INF/classes directory of two different contexts, each context gets its own copy of the singleton (static) object.

**Q) How do I access the value of a cookie using JavaScript?**   
A) You can manipulate cookies in JavaScript with the document.cookie property. You can set a cookie by assigning this property, and retrieve one by reading its current value.   
The following statement, for example, sets a new cookie with a minimum number of attributes:  
document.cookie = "cookieName=cookieValue";  
And the following statement displays the property's value:  
alert(document.cookie);  
The value of document.cookie is a string containing a list of all cookies that are associated with a web page. It consists, that is, of name=value pairs for each cookie that matches the current domain, path, and date. The value of the document.cookie property, for instance, might be the following string:  
cookieName1=cookieValue1; cookieName2=cookieValue2;

**Q) Under what circumstances will a servlet be reloaded?**  
A) That depends on the Servlet container. Most of the Servlet containers reload the servlet only it detects the code change in the Servlet, not in the referenced classes. In Tomcat's server.xml deployment descriptor, if you have mentioned   
< Context path="/myApp"   
docBase="D:/myApp/webDev"   
crossContext="true"  
debug="0"  
reloadable="true"   
trusted="false" >   
< /Context >   
The reloadable = true makes the magic. Every time the Servlet container detects that the Servlet code is changed, it will call the destroy on the currently loaded Servlet and reload the new code.   
But if the class that is referenced by the Servlet changes, then the Servlet will not get loaded. You will have to change the timestamp of the servlet or stop-start the server to have the new class in the container memory.

**Q) How do you do servlet aliasing with Apache and Tomcat?**   
A)Servlet aliasing is a two part process with Apache and Tomcat. First, you must map the request in Apache to Tomcat with the ApJServMount directive, e.g.,   
ApJServMount /myservlet /ROOT  
Second, you must map that url pattern to a servlet name and then to a servlet class in your web.xml configuration file. Here is a sample exerpt:   
< servlet >   
< servlet-name > myservlet< /servlet-name >   
< servlet-class > com.mypackage.MyServlet< /servlet-class >   
< /servlet >   
< servlet-mapping >   
< servlet-name > myservlet< /servlet-name >   
< url-pattern > /myservlet< /url-pattern >   
< /servlet-mapping >

**Q) How can I return a readily available (static) HTML page to the user instead of generating it in the servlet?**   
A) To solve your problem, you can either send a "Redirect" back to the client or use a RequestDispatcher and forward your request to another page:   
Redirect:  
A redirection is made using the HttpServletResponse object:   
if(condition) {  
response.sendRedirect("page1.html");  
} else {  
response.sendRedirect("page2.html");  
}  
RequestDispatcher:  
A request dispatcher can be obtained through the ServletContext. It can be used to include another page or to forward to it.   
if(condition) {  
this.getServletContext().getRequestDispatcher("page1.html").forward();  
} else {  
this.getServletContext().getRequestDispatcher("page2.html").forward();  
}  
Both solutions require, that the pages are available in you document root. If they are located somewhere else on your filesystem, you have to open the file manually and copy their content to the output writer.   
If your application server is set up in combination with a normal web server like Apache, you should use solution (1), because the the web server usually serves static files much faster than the application server.