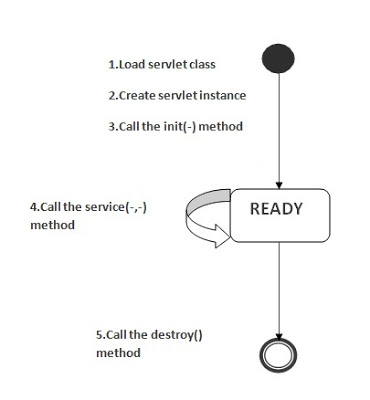
**What is the life-cycle of a servlet?**  
The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:   
Servlet class is loaded.   
Servlet instance is created.   
init method is invoked.   
service method is invoked.   
destroy method is invoked.   
  
[](http://3.bp.blogspot.com/-iOnHr58q3Wo/VWhThR3iFHI/AAAAAAAAAMU/avLC_dETlVI/s1600/servletlife.jpg)  
As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.  
**1) Servlet class is loaded**The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created  
The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

**3) init method is invoked**  
The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:  
public void init(ServletConfig config) throws ServletException   
  
**4) service method is invoked**The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below  
  
public void service(ServletRequest request, ServletResponse response)   
throws ServletException, IOException

**5) destroy method is invoked**The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:  
public void destroy()

**What is a servlet?**

Java Servlet is server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.

All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.

Most of the times, web applications are accessed using HTTP protocol and thats why we mostly extend HttpServlet class.

**What are the new features added to Servlet 2.5?**  
Following are the new features introduced in Servlet 2.5:  
• A new dependency on Java SE 5.0  
• Support for annotations  
• Loading the class  
• Several web.xml conveniences  
• A handful of removed restrictions  
• Some edge case clarifications

**What is different between web server and application server?**

A web server responsibility is to handler HTTP requests from client browsers and respond with HTML response. A web server understands HTTP language and runs on HTTP protocol.

Apache Web Server is kind of a web server and then we have specific containers that can execute servlets and JSPs known as servlet container, for example Tomcat.

Application Servers provide additional features such as Enterprise JavaBeans support, JMS Messaging support, Transaction Management etc. So we can say that Application server is a web server with additional functionalities to help developers with enterprise applications.

**Which HTTP method is non-idempotent?**

A HTTP method is said to be idempotent if it returns the same result every time. HTTP methods GET, PUT, DELETE, HEAD, and OPTIONS are idempotent method and we should implement our application to make sure these methods always return same result. HTTP method POST is non-idempotent method and we should use post method when implementing something that changes with every request.

For example, to access an HTML page or image, we should use GET because it will always return the same object but if we have to save customer information to database, we should use POST method. Idempotent methods are also known as safe methods and we don’t care about the repetitive request from the client for safe methods.

**When is the servlet loaded?**

A servlet can be loaded when:

First request is made.

Server starts up (auto-load).

There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.

Administrator manually loads.

**Why do we need a constructor in a servlet if we use the init method?**

Even though there is an init method in a servlet which gets called to initialize it, a constructor is still required to instantiate the servlet. Even though you as the developer would never need to explicitly call the servlet's constructor, it is still being used by the container (the container still uses the constructor to create an instance of the servlet).

Let us say in a plain java class, you created an init method that gets invoked under some condition, will you be able to invoke it without actually instantiating your class? That is why we need a constructor even for a Servlet.

**What are the types of Servlet?**

There are two types of servlets, GenericServlet and HttpServlet. GenericServlet defines the generic or protocol independent servlet. HttpServlet is subclass of GenericServlet and provides http protocl specific functionality.

**What are the different methods present in a HttpServlet?**

The methods of HttpServlet class are :

doGet() - To handle the GET, conditional GET, and HEAD requests

doPost() - To handle POST requests

doPut() - To handle PUT requests

doDelete() - To handle DELETE requests

doOptions() - To handle the OPTIONS requests and

doTrace() - To handle the TRACE requests

Apart from these, a Servlet also contains init() and destroy() methods that are used to initialize and destroy the servlet respectively. They are not any operation specific and are available in all Servlets for use during their active life-cycle.

**What are the advantages of Servlets over CGI programs?**

Java Servlets have a number of advantages over CGI and other API's. Some are:

1. **Platform Independence** - Java Servlets are 100% pure Java, so it is platform independent. It can run on any Servlet enabled web server. For example if you develop an web application in windows machine running Java web server. You can easily run the same on apache web server without modification code. Platform independency of servlets provide a great advantages over alternatives of servlets.

2. **Performance** - Anyone who has used CGI would agree that Servlets are much more powerful and quicker than CGI. Because the underlying technology is Java, it is fast and can handle multiple request simultaneously. Also, a servlet gets initialized only once in its lifetime and then continues to serve requests without having to be re-initialized again, hence the performance is much higher than CGIs.

3. **Extensibility** - Java Servlets are developed in java which is robust, well-designed and object oriented language which can be extended or polymorphed into new objects. So the java servlets takes all these advantages and can be extended from existing class the provide the ideal solutions.

Also, in terms of Safety & Security Servlets are superior when compared to CGI.

**What are the type of protocols supported by HttpServlet?**

It extends the GenericServlet base class and provides an framework for handling the HTTP protocol. So, HttpServlet only supports HTTP and HTTPS protocol.

**What is meant by Pre-initialization of Servlet?**

When servlet container is loaded, all the servlets defined in the web.xml file do not get initialized by default. When the container receives a request to hit a particular servlet, it loads that servlet. But in some cases if you want your servlet to be initialized when context is loaded, you have to use a concept called pre-initialization of Servlet. In this case, the servlet is loaded when context is loaded. You can specify 1 in between the tag in the Web.xml file in order to pre-initialize your servlet.

**Example.**

<Load-on-startup>1</Load-on-startup>

**What do you understand by servlet mapping?**

Servlet mapping defines an association between a URL pattern and a servlet. You can use one servlet to process a number of url patterns. For example in case of Struts \*.do url patterns are processed by Struts Controller Servlet.

**What interface must be implemented by all Servlets?**

The Servlet Interface must be implemented by all servlets (either the GenericServlet or the HttpServlet)

**What are the uses of Servlets?**

1) Servlets are used to process the client requests.

2) A Servlet can handle multiple request concurrently and be used to develop high performance system

3) A Servlet can be used to load balance among serveral servers, as Servlet can easily forward request.

**What are the difference between the include() and forward() methods?**

The key differences between the two methods are:

a. The include() method inserts the contents of the specified resource directly into the flow of the servlet response, as if it were part of the calling servlet, whereas the forward() is used to show a different resource in place of the servlet that was originally called

b. The include() is often used to include common text or template markup that may be included in many servlets whereas forward() is often used where a servlet plays the role of a controller, processes some input and decides the outcome by returning a particular page response where control is transferred to a different resource

**What is servlet lazy loading?**

Lazy servlet loading means – the container does not initialize the servlets as soon as it starts up. Instead it initializes servlets when they receive their first ever request. This is the standard or default behavior. If you want the container to load your servlet at start-up then use pre-initialization using the load-on-startup element in the deployment descriptor.

**What is Servlet Chaining?**

Servlet Chaining is a method where the output of one servlet is piped or passed onto a second servlet. The output of the second servlet could be passed on to a third servlet, and so on. The last servlet in the chain returns the output to the Web browser.

**What are filters?**

Filters are Java components that are used to intercept an incoming request to a Web resource or the response that is sent back from the resource. It is used to abstract any useful information contained in the request or response. Some of the important functions performed by filters are:

Security checks

Modifying the request or response

Data compression

Logging and auditing

Response compression

Filters are configured in the deployment descriptor of a Web application. Hence, a user is not required to recompile anything to change the input or output of the Web application.

**What are the functions of the Servlet container?**

The functions of the Servlet container are as follows:

Lifecycle management: It manages the life and death of a servlet, such as class loading, instantiation, initialization, service, and making servlet instances eligible for garbage collection.

Communication support: It handles the communication between the servlet and the Web server.

Multithreading support: It automatically creates a new thread for every servlet request received. When the Servlet service() method completes, the thread dies.

Declarative security: It manages the security inside the XML deployment descriptor file.

JSP support: The container is responsible for converting JSPs to servlets and for maintaining them.

**What is the difference between GET and POST method?**

GET is a safe method (idempotent) where POST is non-idempotent method.

We can send limited data with GET method and it’s sent in the header request URL whereas we can send large amount of data with POST because it’s part of the body.

GET method is not secure because data is exposed in the URL and we can easily bookmark it and send similar request again, POST is secure because data is sent in request body and we can’t bookmark it.

GET is t­he default HTTP method whereas we need to specify method as POST to send request with POST method.

Hyperlinks in a page uses GET method.

**What are common tasks performed by Servlet Container?**

Servlet containers are also known as web container, for example Tomcat. Some of the important tasks of servlet container are:

Communication Support: Servlet Container provides easy way of communication between web client (Browsers) and the servlets and JSPs. Because of container, we don’t need to build a server socket to listen for any request from web client, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on business logic for the applications.

Lifecycle and Resource Management: Servlet Container takes care of managing the life cycle of servlet. From the loading of servlets into memory, initializing servlets, invoking servlet methods and to destroy them. Container also provides utility like JNDI for resource pooling and management.

Multithreading Support: Container creates new thread for every request to the servlet and provide them request and response objects to process. So servlets are not initialized for each request and saves time and memory.

JSP Support: JSPs doesn’t look like normal java classes but every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.

Miscellaneous Task: Servlet container manages the resource pool, perform memory optimizations, execute garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes a developer life easier.

**What is ServletConfig object?**

javax.servlet.ServletConfig is used to pass configuration information to Servlet. Every servlet has it’s own ServletConfig object and servlet container is responsible for instantiating this object. We can provide servlet init parameters in web.xml file or through use of WebInitParam annotation. We can use getServletConfig() method to get the ServletConfig object of the servlet.

**What is ServletContext object?**

javax.servlet.ServletContext interface provides access to web application parameters to the servlet. The ServletContext is unique object and available to all the servlets in the web application. When we want some init parameters to be available to multiple or all of the servlets in the web application, we can use ServletContext object and define parameters in web.xml using <context-param> element. We can get the ServletContext object via the getServletContext() method of ServletConfig. Servlet containers may also provide context objects that are unique to a group of servlets and which is tied to a specific portion of the URL path namespace of the host.

ServletContext is enhanced in Servlet Specs 3 to introduce methods through which we can programmatically add Listeners and Filters and Servlet to the application. It also provides some utility methods such as getMimeType(), getResourceAsStream() etc.

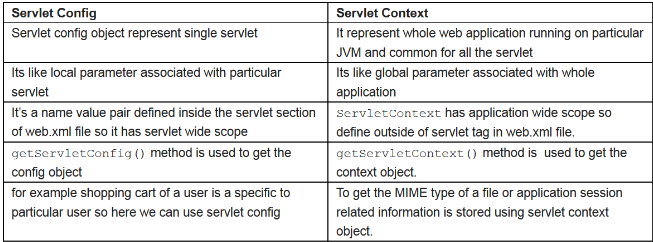
**What is difference between ServletConfig and ServletContext?**

Some of the differences between ServletConfig and ServletContext are:

ServletConfig is a unique object per servlet whereas ServletContext is a unique object for complete application.

ServletConfig is used to provide init parameters to the servlet whereas ServletContext is used to provide application level init parameters that all other servlets can use.

We can’t set attributes in ServletConfig object whereas we can set attributes in ServletContext that other servlets can use in their implementation.



**What is Request Dispatcher?**

RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response. This interface is used for inter-servlet communication in the same context.

There are two methods defined in this interface:

void forward(ServletRequest request, ServletResponse response) – forwards the request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

void include(ServletRequest request, ServletResponse response) – includes the content of a resource (servlet, JSP page, HTML file) in the response.

We can get RequestDispatcher in a servlet using ServletContext getRequestDispatcher(String path) method. The path must begin with a / and is interpreted as relative to the current context root.

**What is difference between PrintWriter and ServletOutputStream?**

PrintWriter is a character-stream class whereas ServletOutputStream is a byte-stream class. We can use PrintWriter to write character based information such as character array and String to the response whereas we can use ServletOutputStream to write byte array data to the response.

We can use ServletResponse getWriter() to get the PrintWriter instance whereas we can use ServletResponse getOutputStream() method to get the ServletOutputStream object reference.

**Can we get PrintWriter and ServletOutputStream both in a servlet?**

We can’t get instances of both PrintWriter and ServletOutputStream in a single servlet method, if we invoke both the methods; getWriter() and getOutputStream() on response; we will get java.lang.IllegalStateException at runtime with message as other method has already been called for this response.

**How can we create deadlock situation in servlet?**

We can create deadlock in servlet by making a loop of method invocation, just call doPost() method from doGet() method and doGet() method to doPost() method to create deadlock situation in servlet.

**What is the use of servlet wrapper classes?**

Servlet HTTP API provides two wrapper classes – HttpServletRequestWrapper and HttpServletResponseWrapper. These wrapper classes are provided to help developers with custom implementation of servlet request and response types. We can extend these classes and override only specific methods we need to implement for custom request and response objects. These classes are not used in normal servlet programming.

**What is SingleThreadModel interface?**

SingleThreadModel interface was provided for thread safety and it guarantees that no two threads will execute concurrently in the servlet’s service method. However SingleThreadModel does not solve all thread safety issues. For example, session attributes and static variables can still be accessed by multiple requests on multiple threads at the same time, even when SingleThreadModel servlets are used. Also it takes out all the benefits of multithreading support of servlets, thats why this interface is Deprecated in Servlet 2.4.

**Do we need to override service() method?**

When servlet container receives client request, it invokes the service() method which in turn invokes the doGet(), doPost() methods based on the HTTP method of request. I don’t see any use case where we would like to override service() method. The whole purpose of service() method is to forward to request to corresponding HTTP method implementations. If we have to do some pre-processing of request, we can always use servlet filters and listeners.

**Is it good idea to create servlet constructor?**

We can define a constructor for servlet but I don’t think its of any use because we won’t be having access to the ServletConfig object until unless servlet is initialized by container. Ideally if we have to initialize any resource for servlet, we should override init() method where we can access servlet init parameters using ServletConfig object.

**What is difference between GenericServlet and HttpServlet?**

GenericServlet is protocol independent implementation of Servlet interface whereas HttpServlet is HTTP protocol specific implementation. Most of the times we use servlet for creating web application and that’s why we extend HttpServlet class. HttpServlet class extends GenericServlet and also provide some other methods specific to HTTP protocol.

**GenericServlet:-**

1. General for all protocol.
2. Implements Servlet Interface.
3. Use Service method.

**HttpServlet:-**

1. Only for HTTP Protocol.
2. Inherit GenericServlet class.
3. Use doPost, doGet method instead of service method.

**What is the purpose of RequestDispatcher Interface?**

The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of antoher resource.

**Can you call a jsp from the servlet?**

Yes, one of the way is RequestDispatcher interface for example:

RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");

rd.forward(request,response);

**What is the inter-servlet communication?**

When we want to invoke another servlet from a servlet service methods, we use inter-servlet communication mechanisms. We can invoke another servlet using RequestDispatcher forward() and include() methods and provide additional attributes in request for other servlet use.

**Are Servlets Thread Safe? How to achieve thread safety in servlets?**

HttpServlet init() method and destroy() method are called only once in servlet life cycle, so we don’t need to worry about their synchronization. But service methods such as doGet() or doPost() are getting called in every client request and since servlet uses multithreading, we should provide thread safety in these methods.

If there are any local variables in service methods, we don’t need to worry about their thread safety because they are specific to each thread but if we have a shared resource then we can use synchronization to achieve thread safety in servlets when working with shared resources.

**What is servlet attributes and their scope?**

Servlet attributes are used for inter-servlet communication, we can set, get and remove attributes in web application. There are three scopes for servlet attributes – request scope, session scope and application scope.

ServletRequest, HttpSession and ServletContext interfaces provide methods to get/set/remove attributes from request, session and application scope respectively.

Servlet attributes are different from init parameters defined in web.xml for ServletConfig or ServletContext.

**How do we call one servlet from another servlet?**

We can use RequestDispatcher forward() method to forward the processing of a request to another servlet. If we want to include the another servlet output to the response, we can use RequestDispatcher include() method.

**How can we invoke another servlet in a different application?**

We can’t use RequestDispatcher to invoke servlet from another application because it’s specific for the application. If we have to forward the request to a resource in another application, we can use ServletResponse sendRedirect() method and provide complete URL of another servlet. This sends the response to client with response code as 302 to forward the request to another URL. If we have to send some data also, we can use cookies that will be part of the servlet response and sent in the request to another servlet.

**What is difference between ServletResponse sendRedirect() and RequestDispatcher forward() method?**

RequestDispatcher forward() is used to forward the same request to another resource whereas ServletResponse sendRedirect() is a two step process. In sendRedirect(), web application returns the response to client with status code 302 (redirect) with URL to send the request. The request sent is a completely new request.

forward() is handled internally by the container whereas sednRedirect() is handled by browser.

We should use forward() when accessing resources in the same application because it’s faster than sendRedirect() method that required an extra network call.

In forward() browser is unaware of the actual processing resource and the URL in address bar remains same whereas in sendRedirect() URL in address bar change to the forwarded resource.

forward() can’t be used to invoke a servlet in another context, we can only use sendRedirect() in this case.

**Why HttpServlet class is declared abstract?**

HttpServlet class provide HTTP protocol implementation of servlet but it’s left abstract because there is no implementation logic in service methods such as doGet() and doPost() and we should override at least one of the service methods. That’s why there is no point in having an instance of HttpServlet and is declared abstract class.

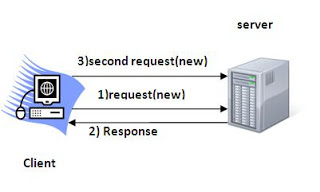
**Why we should override only no-agrs init() method.**

If we have to initialize some resource before we want our servlet to process client requests, we should override init() method. If we override init(ServletConfig config) method, then the first statement should be super(config) to make sure superclass init(ServletConfig config) method is invoked first. That’s why GenericServlet provides another helper init() method without argument that get’s called at the end of init(ServletConfig config) method. We should always utilize this method for overriding init() method to avoid any issues as we may forget to add super() call in overriding init method with ServletConfig argument.

**What is URL Encoding?**

URL Encoding is the process of converting data into CGI form so that it can travel across the network without any issues. URL Encoding strip the white spaces and replace special characters with escape characters. We can use java.net.URLEncoder.encode(String str, String unicode) to encode a String. URL Decoding is the reverse process of encoding and we can use java.net.URLDecoder.decode(String str, String unicode) to decode the encoded string. For example “Pankaj’s Data” is encoded to “Pankaj%27s+Data”.

**What is Session Tracking?**  
Session simply means a particular interval of time.

Session Tracking is a way to maintain state of an user.Http protocol is a stateless protocol.Each time user requests to the server, server treats the request as the new request.So we need to maintain the state of an user to recognize to particular user.  
  
HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:  
  
[](http://3.bp.blogspot.com/-rAg0RPtjAuU/VWl4Vt7pJHI/AAAAAAAAAMo/jifOjIc1MYI/s1600/newrequest.JPG)  
**Why use Session Tracking?**

To recognize the user It is used to recognize the particular user.  
Session Tracking Techniques

**There are four techniques used in Session tracking:**Cookies   
Hidden Form Field   
URL Rewriting   
HttpSession

**What are different methods of session management in servlets?**

Session is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

**Some of the common ways of session management in servlets are:**

User Authentication

HTML Hidden Field

Cookies

URL Rewriting

Session Management API

**What is URL Rewriting?**

We can use HttpSession for session management in servlets but it works with Cookies and we can disable the cookie in client browser. Servlet API provides support for URL rewriting that we can use to manage session in this case.

The best part is that from coding point of view, it’s very easy to use and involves one step – encoding the URL. Another good thing with Servlet URL Encoding is that it’s a fallback approach and it kicks in only if browser cookies are disabled.

We can encode URL with HttpServletResponse encodeURL() method and if we have to redirect the request to another resource and we want to provide session information, we can use encodeRedirectURL() method.

**How does Cookies work in Servlets?**

Cookies are used a lot in web client-server communication, it’s not something specific to java. Cookies are text data sent by server to the client and it gets saved at the client local machine.

Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.

HttpServletRequest getCookies() method is provided to get the array of Cookies from request, since there is no point of adding Cookie to request, there are no methods to set or add cookie to request.

Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookie in response header, there are no getter methods for cookie.

**What are Cookies?**  
There are 2 types of cookies in servlets.   
Non-persistent cookie   
Persistent cookie   
**Non-persistent cookie**

It is valid for single session only. It is removed each time when user closes the browser.  
**Persistent cookie**  
It is valid for multiple session. It is not removed each time when user closes the browser. It is removed only if user logout or signout.

**Advantage of Cookies**  
Simplest technique of maintaining the state.   
Cookies are maintained at client side.   
Disadvantage of Cookies   
It will not work if cookie is disabled from the browser.   
Only textual information can be set in Cookie object.

**What is difference between Cookies and HttpSession?**  
Cookie works at client side whereas HttpSession works at server side.

**How to notify an object in session when session is invalidated or timed-out?**

If we have to make sure an object gets notified when session is destroyed, the object should implement javax.servlet.http.HttpSessionBindingListener interface. This interface defines two callback methods – valueBound() and valueUnbound() that we can define to implement processing logic when the object is added as attribute to the session and when session is destroyed.

**What is the difference between encodeRedirectUrl and encodeURL?**

HttpServletResponse provide method to encode URL in HTML hyperlinks so that the special characters and white spaces are escaped and append session id to the URL. It behaves similar to URLEncoder encode method with additional process to append jsessionid parameter at the end of the URL.

However HttpServletResponse encodeRedirectUrl() method is used specially for encode the redirect URL in response.

So when we are providing URL rewriting support, for hyperlinks in HTML response, we should use encodeURL() method whereas for redirect URL we should use encodeRedirectUrl() method.

**What is filter?**  
A filter is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

**Why do we have servlet filters?**

Servlet Filters are pluggable java components that we can use to intercept and process requests before they are sent to servlets and response after servlet code is finished and before container sends the response back to the client.

**Some common tasks that we can do with filters are:**

Logging request parameters to log files.

Authentication and autherization of request for resources.

Formatting of request body or header before sending it to servlet.

Compressing the response data sent to the client.

Alter response by adding some cookies, header information etc

**What is the effective way to make sure all the servlets are accessible only when user has a valid session?**

We know that servlet filters can be used to intercept request between servlet container and servlet, we can utilize it to create authentication filter and check if request contains a valid session or not.

**Why do we have servlet listeners?**

We know that using ServletContext, we can create an attribute with application scope that all other servlets can access but we can initialize ServletContext init parameters as String only in deployment descriptor (web.xml). What if our application is database oriented and we want to set an attribute in ServletContext for Database Connection.

If you application has a single entry point (user login), then you can do it in the first servlet request but if we have multiple entry points then doing it everywhere will result in a lot of code redundancy. Also if database is down or not configured properly, we won’t know until first client request comes to server. To handle these scenario, servlet API provides Listener interfaces that we can implement and configure to listen to an event and do certain operations.

**How to handle exceptions thrown by application with another servlet?**

If you notice, doGet() and doPost() methods throw ServletException and IOException. Since browser understand only HTML, when our application throw exception, servlet container processes the exception and generate a HTML response. Same goes with other error codes like 404, 403 etc.

Servlet API provides support for custom Exception and Error Handler servlets that we can configure in deployment descriptor, the whole purpose of these servlets are to handle the Exception or Error raised by application and send HTML response that is useful for the user. We can provide link to application home page or some details to let user know what went wrong.

We can configure them in web.xml like below:

<error-page>

<error-code>404</error-code>

<location>/AppExceptionHandler</location>

</error-page>

<error-page>

<exception-type>javax.servlet.ServletException</exception-type>

<location>/AppExceptionHandler</location>

</error-page>

**What is a deployment descriptor?**

Deployment descriptor is a configuration file for the web application and it’s name is web.xml and it resides in WEB-INF directory. Servlet container use this file to configure web application servlets, servlet config params, context init params, filters, listeners, welcome pages and error handlers.

With servlet 3.0 annotations, we can remove a lot of clutter from web.xml by configuring servlets, filters and listeners using annotations.

**How to make sure a servlet is loaded at the application startup?**

Usually servlet container loads a servlet on the first client request but sometimes when the servlet is heavy and takes time to loads, we might want to load it on application startup. We can use load-on-startup element with servlet configuration in web.xml file or use WebServlet annotation loadOnStartup variable to tell container to load the servlet on system startup.

<servlet>

<servlet-name>foo</servlet-name>

<servlet-class>com.foo.servlets.Foo</servlet-class>

<load-on-startup>5</load-on-startup>

</servlet>

The load-on-startup value should be int, if it’s negative integer then servlet container will load the servlet based on client requests and requirement but if it’s 0 or positive, then container will load it on application startup.

If there are multiple servlets with load-on-startup value such as 0,1,2,3 then lower integer value servlet will be loaded first.

**How to get the actual path of servlet in server?**

We can use following code snippet to get the actual path of the servlet in file system.

getServletContext().getRealPath(request.getServletPath())

**How to get the server information in a servlet?**

We can use below code snippet to get the servlet information in a servlet through servlet context object.

getServletContext().getServerInfo()

**Write a servlet to upload file on server.**

File Upload and Download and common tasks in a java web application. Unfortunately Servlet API doesn’t provide easy methods to upload file on server, so we can use Apache FileUpload jar to make our life easier.

**How do we go with database connection and log4j integration in servlet?**

If you work with database connection a lot in your web application, its best to initialize it in a servlet context listener and set it as a context attribute for other servlets to use.

Integrating Log4j is also very easy in web applications, all we need is a log4j configuration XML or property file and then configure it in a servlet context listener.

**How to get the IP address of client in servlet?**

We can use request.getRemoteAddr() to get the client IP address in servlet.

**What are important features of Servlet 3?**

Servlet Specs 3.0 was a major release and some of the important features are:

**Servlet Annotations**: Prior to Servlet 3, all the servlet mapping and it’s init parameters were used to defined in web.xml, this was not convenient and more error prone when number of servlets are huge in an application.

Servlet 3 introduced use of java annotations to define a servlet, filter and listener servlets and init parameters. Some of the important Servlet API annotations are WebServlet, WebInitParam, WebFilter and WebListener. Read more about them at Servlet 3 annotations.

**Web Fragments**: Prior to servlet specs 3.0, all the web application configurations are required to be present in the web.xml that makes it cluttered with lot of elements and chances of error increases. So servlet 3 specs introduced web fragments where we can have multiple modules in a single web application, all these modules should have web-fragment.xml file in META-INF directory. We can include all the elements of web.xml inside the web-fragment.xml too. This helps us in dividing our web application into separate modules that are included as JAR file in the web application lib directory.

Adding Web Components dynamically: We can use ServletContext object to add servlets, filters and listeners programmatically. This helps us in building dynamic system where we are loading a component only if we need it. These methods are addServlet(), addFilter() and addListener() defined in the servlet context object.

**Asynchronous Processing:** Asynchronous support was added to delegate the request processing to another thread rather than keeping the servlet thread busy. It can increase the throughput performance of the application.

**What are different ways for servlet authentication?**

**Servlet Container provides different ways of login based servlet authentication:**

HTTP Basic Authentication

HTTP Digest Authentication

HTTPS Authentication

Form Based Login: A standard HTML form for authentication, advantage is that we can change the login page layout as our application requirements rather than using HTTP built-in login mechanisms.

**How can we achieve transport layer security for our web application?**

We can configure our servlet container to use SSL for message communication over the network. To configure SSL on Tomcat, we need a digital certificate that can be created using Java keytool for development environment. For production environment, you should get the digital certificate from SSL certificate providers, for example, Verisign or Entrust.

In web.xml file <load-on-startup>1</load-on-startup> is defined between <servlet></servlet> tag what does it means.

**Important points on load-on-startup element**

1. If <load-on-startup> value is same for two servlet than they will be loaded in an order on which they are declared inside web.xml file.

2. if <load-on-startup> is 0 or negative integer than Servlet will be loaded when Container feels to load them.

3. <load-on-startup> guarantees loading, initialization and call to init() method of servlet by web container.

4. If there is no <load-on-startup> element for any servlet than they will be loaded when web container decides to load them.

**When to use <load-on-startup> in web.xml**

<load-on-startup> is suitable for those servlet which performs time consuming jobs e.g. Creating Database Connection pool, downloading files or data from network or prepare environment ready for servicing client in terms of initializing cache , clearing pipelines and loading important data in memory. If any of your servlet performs these jobs then declare them using <load-on-startup> element and specify order as per your business logic or what suites your application. Remember lower the value of <load-on-startup>, servlet will be loaded first. You can also check your web container documentation on how exactly load on start-up is supported.

**How can you get the information about one servlet context in another servlet?**

In context object we can set the attribute which we want on another servlet and we can get that attribute using their name on another servlet.

Context.setAttribute (“name”,” value”)

Context.getAttribute (“name”)

**What is war file?**

A war (web archive) file specifies the web elements. A servlet or jsp project can be converted into a war file. Moving one servlet project from one place to another will be fast as it is combined into a single file.

**How to create war file?**

The war file can be created using jar tool found in jdk/bin directory. If you are using Eclipse or Netbeans IDE, you can export your project as a war file.

To create war file from console, you can write following code.

jar -cvf abc.war \*

**What are the annotations used in Servlet 3?**

There are mainly 3 annotations used for the servlet.

@WebServlet : for servlet class.

@WebListener : for listener class.

@WebFilter : for filter class.

**Which event is fired at the time of project deployment and undeployment?**

ServletContextEvent.

**Which event is fired at the time of session creation and destroy?**

HttpSessionEvent.

**Which event is fired at the time of setting, getting or removing attribute from application scope?**

ServletContextAttributeEvent.

**What is the use of welcome-file-list?**

It is used to specify the welcome file for the project.

**What is the use of attribute in servlets?**

Attribute is a map object that can be used to set, get or remove in request, session or application scope. It is mainly used to share information between one servlet to another.