1. [Difference between Apache and Tomcat](#diff_Apache_Tomcat)
2. [What is servlet](#servlet)
3. [How to make the site https](#website_secure_with_https)
4. [Difference Between GenericServlet and HttpServlet?](#generic_serv_httpservlet)

public interface **Servlet**

Defines methods that all servlets must implement.

**A servlet is** a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

To implement this interface, you can write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extendsjavax.servlet.http.HttpServlet.

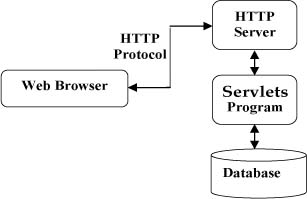
This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:

1. The servlet is constructed, then initialized with the init method.
2. Any calls from clients to the service method are handled.
3. The servlet is taken out of service, then destroyed with the destroy method, then garbage collected and finalized.

In addition to the life-cycle methods, this interface provides the getServletConfig method, which the servlet can use to get any startup information, and thegetServletInfo method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

## Servlets Architecture:

Following diagram shows the position of Servelts in a Web Application.



## Servlets Packages:

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages,

[**Difference between Filter and Listener in Servlet (Java EE)**](http://stackoverflow.com/questions/4720942/difference-between-filter-and-listener-in-servlet-java-ee)

<http://stackoverflow.com/questions/4720942/difference-between-filter-and-listener-in-servlet-java-ee>

Servlet Filter is used for monitoring request and response from client to the servlet, or to modify the request and response, or to audit and log.

Servlet Listener is used for listening to events in a web containers, such as when you create a session, or place an attribute in an session or if you passivate and activate in another container, to subscribe to these events you can configure listener in web.xml, for example HttpSessionListener.

And it is also worth nothing that listeners implements javax.servlet.ServletContextListener while filters implement javax.servlet.Filter

# [How do I import the javax.servlet API in my Eclipse project?](http://stackoverflow.com/questions/4076601/how-do-i-import-the-javax-servlet-api-in-my-eclipse-project)

Ensure that you're using at least [Eclipse IDE for **Java EE** developers](http://www.eclipse.org/downloads/) (with the **EE**). It contains development tools to create dynamic web projects and easily integrate servletcontainers (those tools are part of Web Tools Platform, WTP). You also need to ensure that you already have a servletcontainer installed which implements at least the same Servlet API version as the servletcontainer in the production environment, for example [Apache Tomcat](http://tomcat.apache.org/),[Oracle GlassFish](http://glassfish.java.net/), [JBoss AS](http://www.jboss.org/jbossas)/[WildFly](http://www.wildfly.org/), etc.

A servletcontainer is a concrete implementation of the Servlet API. Note that the [Java EE SDK](http://www.oracle.com/technetwork/java/javaee/downloads/index.html) download at Oracle.com basically contains GlassFish. So if you happen to already have downloaded Java EE SDK, then you already have GlassFish. Also note that for example GlassFish and JBoss AS/WildFly are *more* than just a servletcontainer, they also supports JSF, EJB, JPA and all other Java EE fanciness.

Once having installed both Eclipse and a servletcontainer, do the following steps in Eclipse:

1. Integrate servletcontainer in Eclipse. Open the *Servers* view in the bottom box, rightclick there and choose*New > Server*. Pick the appropriate servletcontainer make and version and walk through the wizard.
2. Create new dynamic web project in Eclipse which is associated with the integrated servletcontainer. Open the *Project Navigator* on the left hand side. Rightclick there and choose *New > Project* and then in menu*Web > Dynamic Web Project* and set the *Target Runtime* to the integrated server. Eclipse will then automatically take the servletcontainer's libraries in the build path. This way you'll be able to import and use the Servlet API.

Or if it's an existing project, you can set/change the server by *Targeted Runtimes* in project's properties.

You should above all **never** manually copy/download/move/include the individual servletcontainer-specific libraries like servlet-api.jar, jsp-api.jar, el-api.jar, j2ee.jar, javaee.jar, etc. It would only lead to future portability, compatibility, classpath and maintainability troubles, because your webapp would not work when it's deployed to a servletcontainer of a different make/version than where those libraries are originally obtained from.

Here are some typical exceptions which you can get when you litter the /WEB-INF/lib or even /JRE/lib, /JRE/lib/ext, etc with servletcontainer-specific libraries in a careless attempt to fix the compilation errors:

* [java.lang.NullPointerException at org.apache.jsp.index\_jsp.\_jspInit](http://stackoverflow.com/questions/4886196/spring-mvc-jsp-and-nullpointerexception)
* [java.lang.NoClassDefFoundError: javax/el/ELResolver](http://stackoverflow.com/questions/6360121/java-lang-noclassdeffounderror-javax-el-elresolver-when-running-simplehellobyent)
* [java.lang.NoSuchFieldError: IS\_DIR](http://stackoverflow.com/questions/4214826/error-java-lang-nosuchfielderror-is-dir-while-starting-apache-tomcat-from-ecli)
* [java.lang.NoSuchMethodError: javax.servlet.jsp.PageContext.getELContext()Ljavax/el/ELContext;](http://stackoverflow.com/questions/7274675/getting-error-while-accessing-the-map-from-spring-controller-to-jsp-using-jstl/7274928#7274928)
* [java.lang.AbstractMethodError: javax.servlet.jsp.JspFactory.getJspApplicationContext(Ljavax/servlet/ServletContext;)Ljavax/servlet/jsp/JspApplicationContext;](http://stackoverflow.com/questions/6998747/tomcat-6-http-authentication)
* [org.apache.jasper.JasperException: The method getJspApplicationContext(ServletContext) is undefined for the type JspFactory](http://stackoverflow.com/questions/2327118/howto-solve-jasper-exception-problem)
* [java.lang.VerifyError: (class: org/apache/jasper/runtime/JspApplicationContextImpl, method: createELResolver signature: ()Ljavax/el/ELResolver;) Incompatible argument to function](http://stackoverflow.com/questions/7310968/web-app-working-in-eclipse-gives-verify-el-error-org-apache-catalina-core-standar)
* [jar not loaded. See Servlet Spec 2.3, section 9.7.2. Offending class: javax/servlet/Servlet.class](http://stackoverflow.com/questions/2641338/how-to-resolve-error-listenerstart-when-deploying-web-app-in-tomcat-5-5)

# [Difference between applicationContext.xml and spring-servlet.xml in Spring](http://stackoverflow.com/questions/3652090/difference-between-applicationcontext-xml-and-spring-servlet-xml-in-spring)

Spring lets you define multiple contexts in a parent-child hierarchy.

The applicationContext.xml defines the beans for the "root webapp context", i.e. the context associated with the webapp.

The spring-servlet.xml (or whatever else you call it) defines the beans for one servlet's app context. There can be many of these in a webapp, one per Spring servlet (e.g. spring1-servlet.xml for servlet spring1, spring2-servlet.xml for servlet spring2).

Beans in spring-servlet.xml can reference beans in applicationContext.xml, but not vice versa.

All Spring MVC controllers must go in the spring-servlet.xml context.

In most simple cases, the applicationContext.xml context is unnecessary. It is generally used to contain beans that are shared between all servlets in a webapp. If you only have one servlet, then there's not really much point, unless you have a specific use for it.

# [what is the difference between a portlet and a servlet?](http://stackoverflow.com/questions/1480528/what-is-the-difference-between-a-portlet-and-a-servlet)

Portlets are part of JSR-168 standard that regulates portal containers and components. This is different standard from standards for web containers (and servlets). Though there are definitely strong parallels between these two standards they differ in containers, APIs, life cycle, configuration, deployment, etc.

The main difference between portlet vs. servlet could be that while servlet always responds to single type of action - request, portlet (due to nature of its life cycle and stronger container bindings) has to respond to two types of actions: render and request. There are of course more to it but I found this as the core difference between the two when I studied portal development.

<http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html>

Top 10 Servlet Interview Question Answers - J2EE  
**Question 1: In web.xml file   <load-on-startup>1</load-on-startup> is defined between <servlet></servlet> tag what does it means.**

Ans**:**whenever we request for any servlet the servlet container will initialize the servlet and load it which is defined in our config file called web.xml by default it will not initialize when our context is loaded .defining like this <load-on-startup>1</load-on-startup> is also known as pre initialization of servlet means now the servlet for which we have define this tag has been initialized in starting when context is loaded before getting any request.When this servlet question was asked to me in an interview few years back , I was not even aware of this element but this questions pointed me to look DTD of web.xml and understand other elements as well.**.**

Read more: <http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html#ixzz3CgeHbnAP>

**Question 2: How can we create deadlock condition on our servlet?**

Ans: one simple way to call doPost() method inside doGet() and doGet()method inside doPost() it will create deadlock situation for a servlet. This is rather simple servlet interview questions but yet tricky if you don’t think of it 

Read more: <http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html#ixzz3CgeZv4Xi>

**Question 3: For initializing a servlet can we use constructor in place of init ().**

Ans: No, we can not use constructor for initializing a servlet because for initialization we need an object of servletConfig using this object we get all the parameter which are defined in deployment descriptor for initializing a servlet and in servlet class we have only default constructor according to older version of java so if we want to pass a

Config object we don’t have parametrized constructor and apart from this servlet is loaded and initialized by container so ots a job of container to call the method according to servlet specification they have lifecycle method so init() method is called firstly.  
  
Read more: <http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html#ixzz3CgekrLOB>

# [How to implement a HTTPS login page in a web application?](http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application)

<http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11down vote[favorite](http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application)  **18** | I want to create a secure login/logout mechanism. I started reading the following articles to get an idea of things to take into account:   * [Solving the Logout Problem Properly and Elegantly](http://www.javaworld.com/javaworld/jw-09-2004/jw-0927-logout.html) * [Revisiting the logout problem](http://www.javaworld.com/javaworld/jw-10-2006/jw-1006-logout.html)   These articles make some good points, but I was thinking in using HTTPS in a similar way as the Yahoo mail login page. You know... you type [http://mail.yahoo.com](http://mail.yahoo.com/) and you are redirected to a HTTPS page like**https://**login.yahoo.com/config/login where you insert your username and password and after your credentials are verified you are redirected back to a HTTP page with a generated session\_id cookie and all communications from there on are on HTTP using the cookie.  What do I need to implement this behavior?  I want to do this for two Java web apps (one with Spring framework and one with Struts 1) but don’t know exactly how to integrate that HTTPS part into the application (I have never worked with HTTPS before).  [java](http://stackoverflow.com/questions/tagged/java) [login](http://stackoverflow.com/questions/tagged/login) [https](http://stackoverflow.com/questions/tagged/https)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/q/1454021/3426143)|[edit](http://stackoverflow.com/posts/1454021/edit) | [edited Sep 24 '09 at 8:18](http://stackoverflow.com/posts/1454021/revisions) | asked Sep 21 '09 at 11:14  user159088 | |
|  | add a comment |

## 6 Answers

[active](http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/1454021/how-to-implement-a-https-login-page-in-a-web-application?answertab=votes#tab-top)

|  |  |
| --- | --- |
| up vote15down voteaccepted  +100 | First of all you need to enable SSL for your server. For Tomcat you need to generate an openSSL keystore and add the following connector to server.xml:  <Connector port="8443" scheme="https" secure="true" SSLEnabled="true"  keystoreFile="mykeystore" sslProtocol="TLS"  keystorePass="keystore password" />  To integrate SSL into your application I recommend Spring Security. It offers exactly what you want (login over HTTPS, then redirected to HTTP). All you have to do to implement it, is to set forceHTTPS to true:  <bean id="authenticationProcessingFilterEntryPoint"  class="org.springframework.security.ui.webapp.AuthenticationProcessingFilterEntryPoint">  <property name="loginFormUrl" value="/pages/login.jsp" />  <property name="forceHttps" value="true"/>  </bean>  Of course Spring and Spring security do have a rather steep learning curve, but it is totally worth it. Do it once and then you can apply it to new apps in less than an hour. You can use Spring Security in both the Spring and Struts application.  Spring security used to be Acegi security. This is an [article](http://www.javaworld.com/javaworld/jw-10-2007/jw-10-acegi2.html?page=1) that will get you started. |

Not sure about any Java or spring specifics, but in general:

1) Set up an SSL cert on your server.

2) Forward or Link to an absolute URL (with https:// at the beginning) when going to login page

3) Forward to an absolute URL (with http://) after successful authentication.

4) Include a check in the login page code to only accept https connections.

Of course there may be framework specific ways of doing the http/https redirect without resorting to explicitly specifying the full URL.

Apache is the Apache Web Server also known as httpd. See [http://httpd.apache.org](http://httpd.apache.org/)

# [What is the difference between apache and tomcat in the xampp control panel? [duplicate]](http://programmers.stackexchange.com/questions/221030/what-is-the-difference-between-apache-and-tomcat-in-the-xampp-control-panel)

Tomcat is Apache Tomcat. See [http://tomcat.apache.org](http://tomcat.apache.org/)

httpd is a web server whereas Tomcat is a Servlet Container. While at the simplest level both can be viewed as web servers that serve static files they have very different focuses. httpd is typically used to host static content and dynamic content written in perl, php amongst others. Tomcat is primarily intended to be used to serve web applications developed in Java to the Java Servlet specification.

If you are going to serve only static(such as HTML) pages then Apache webserver would suffice but if the pages being (dynamically) served are in JSP/Servlet then you need a container that can interpret them, for which we use Tomcat.Similarly, if you will be using PHP then you need to add its support in Webserver.  
Webserver is used to just respond to the requests with feature such as load balancing, whereas containers( such as Tomcat) are used to manage the lifecycle of the pages generated using [JSP](http://www.coderanch.com/forums/f-50/JSP) and servlet..  
If you need more info on the container specific responsibilty then you can check out either the JSP or Servlet specification.

In general, the Apache HTTP server is just a plain old web server designed to serve static web pages. There are plenty of modules which can be installed to enhance Apache's abilities so that it can serve dynamic webpages using various technologies such as PHP, CGI or whatever, but the core of Apache is just a plain old HTTP server.

Tomcat, on the other hand, is specifically designed from the ground-up to serve as a [Java Servlet](http://en.wikipedia.org/wiki/Java_Servlet) engine. It's primary purpose is to implement the Java Servlet API and execute Java servlets for the purpose of building dynamic websites. Tomcat can also be used as a regular HTTP server that serves static pages, but that is not its primary purpose. (Also, Tomcat is allegedly slower than Apache httpd when it comes to serving static pages.)

The two technologies can be used together through a connector module called [mod\_jk](http://tomcat.apache.org/connectors-doc-archive/jk2/jk/aphowto.html). This will allow you to use the Apache HTTP server to serve regular static webpages, and the Tomcat Servlet engine to execute servlets.

### [How to make a website secured with https - Stack Overflow](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB4QFjAA&url=http%3A%2F%2Fstackoverflow.com%2Fquestions%2F2205325%2Fhow-to-make-a-website-secured-with-https&ei=XmXFVIOLGqHfsASMnAI&usg=AFQjCNFEqnOqe6CWeDlwsqX0VjdtWPLQsA&bvm=bv.84349003,d.aWw)

### Difference Between GenericServlet and HttpServlet?

Generic Servlet:

* GenericServlet class is direct subclass of Servlet interface.
* Generic Servlet is protocol independent.It handles all types  of protocol  like http, smtp, ftp etc.
* Generic Servlet only supports  service() method.It handles only simple request

public void service(ServletRequest req,ServletResponse res ).

* Generic Servlet only supports  service() method.

HttpServlet:

* HttpServlet class is the direct subclass of Generic Servlet.
* HttpServlet is protocol dependent. It handles only http protocol.
* HttpServlet  supports public void service(ServletRequest req,ServletResponse res ) and protected void service(HttpServletRequest req,HttpServletResponse res).
* HttpServlet supports also   doGet(),doPost(),doPut(),doDelete(),doHead(),doTrace(),doOptions()etc.