Tomcat 5 Startup Sequence

Sequence 1. Start from Command Line
Class: org.apache.catalina.startup.Bootstrap
What it does:
        a) Set up classloaders
                commonLoader (common)-> System Loader
                sharedLoader (shared)-> commonLoader -> System Loader
                catalinaLoader(server) -> commonLoader -> System Loader
        b) Load startup class (reflection)
                org.apache.catalina.startup.Catalina
                setParentClassloader -> sharedLoader
                Thread.contextClassloader -> catalinaLoader
        c) Bootstrap.daemon.init() complete

Sequence 2. Process command line argument (start, startd, stop, stopd)
Class: org.apache.catalina.startup.Bootstrap (assume command->start)
What it does:
        a) Catalina.setAwait(true);
        b) Catalina.load()
                b1) initDirs() -> set properties like
                                  catalina.home
                                  catalina.base == catalina.home (most cases)
                b2) initNaming
                        setProperty(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
                                    org.apache.naming.java.javaURLContextFactory ->default)
                b3) createStartDigester()
                        Configures a digester for the main server.xml elements like
                        org.apache.catalina.core.StandardServer (can change of course :)
                        org.apache.catalina.deploy.NamingResources
                                Stores naming resources in the J2EE JNDI tree
                        org.apache.catalina.LifecycleListener
                                implements events for start/stop of major components
                        org.apache.catalina.core.StandardService
                                The single entry for a set of connectors,
                                so that a container can listen to multiple connectors
                                ie, single entry
                        org.apache.coyote.tomcat5.CoyoteConnector
                                Connectors to listen for incoming requests only
                        It also adds the following rulesets to the digester
                                NamingRuleSet
                                EngineRuleSet
                                HostRuleSet
                                ContextRuleSet
                b4) Load the server.xml and parse it using the digester
                    Parsing the server.xml using the digester is an automatic
                    XML-object mapping tool, that will create the objects defined in server.xml
                    Startup of the actual container has not started yet.
                b5) Assigns System.out and System.err to the SystemLogHandler class

```
                b6) Calls initialize on all components, this makes each object register itself with
the
                    JMX agent.
                    During the process call the Connectors also initialize the adapters.
                    The adapters are the components that do the request pre-processing.
                    Typical adapters are HTTP1.1 (default if no protocol is specified,
                    org.apache.coyote.http11.Http11Protocol)
                    AJP1.3 for mod_jk etc.

        c) Catalina.start()
                c1) Starts the NamingContext and binds all JNDI references into it
                c2) Starts the services under <Server> which are:
                        StandardService -> starts Engine (ContainerBase ->Logger,Loader,Realm,Cluster
etc)
                c3) StandardHost (started by the service)
                                Configures a ErrorReportValvem to do proper HTML output for different
HTTP
                                errors codes
                                Starts the Valves in the pipeline (at least the ErrorReportValve)
                                Configures the StandardHostValve,
                                        this valves ties the Webapp Class loader to the thread
context
                                        it also finds the session for the request
                                        and invokes the context pipeline
                                Starts the HostConfig component
                                        This component deploys all the webapps
                                                (webapps & conf/Catalina/localhost/*.xml)
                                        Webapps are installed using the deployer
(StandardHostDeployer)
                                        The deployer will create a Digester for your context, this
digester
                                        will then invoke ContextConfig.start()
                                                The ContextConfig.start() will process the default
web.xml (conf/web.xml)
                                                and then process the applications web.xml (WEB-
INF/web.xml)

                c4) During the lifetime of the container (StandardEngine) there is a background
thread that
                    keeps checking if the context has changed. If a context changes (timestamp of war
file,
                    context xml file, web.xml) then a reload is issued (stop/remove/deploy/start)

        d) Tomcat receives a request on an HTTP port
            d1) The request is received by a separate thread which is waiting in the PoolTcpEndPoint
                 class. It is waiting for a request in a regular ServerSocket.accept() method.
                 When a request is received, this thread wakes up.
            d2) The PoolTcpEndPoint assigns the a TcpConnection to handle the request.
                It also supplies a JMX object name to the catalina container (not used I believe)
            d3) The processor to handle the request in this case is Coyote Http11Processor,
                and the process method is invoked.
                This same processor is also continuing to check the input stream of the socket
                until the keep alive point is reached or the connection is disconnected.
            d4) The HTTP request is parsed using an internal buffer class (Coyote Http11 Internal
Buffer)
                The buffer class parses the request line, the headers, etc and store the result in a
                Coyote request (not an HTTP request) This request contains all the HTTP info, such
                as servername, port, scheme, etc.
            d5) The processor contains a reference to an Adapter, in this case it is the
                Coyote Tomcat 5 Adapter. Once the request has been parsed, the Http11 processor
                invokes service() on the adapter. In the service method, the Request contains a
                CoyoteRequest and CoyoteRespons (null for the first time)
                The CoyoteRequest(Response) implements HttpRequest(Response) and
HttpServletRequest(Response)
                The adapter parses and associates everything with the request, cookies, the context
```

```
through a
                Mapper, etc
            d6) When the parsing is finished, the CoyoteAdapter invokes its container
(StandardEngine)
                and invokes the invoke(request,response) method.
                This initiates the HTTP request into the Catalina container starting at the engine
level
            d7) The StandardEngine.invoke() simply invokes the container pipeline.invoke()
            d8) By default the engine only has one valve the StandardEngineValve, this valve simply
                invokes the invoke() method on the Host pipeline (StandardHost.getPipeLine())
            d9) the StandardHost has two valves by default, the StandardHostValve and the
ErrorReportValve
            d10) The standard host valve associates the correct class loader with the current thread
                 It also retrieves the Manager and the session associated with the request (if there
is one)
                 If there is a session access() is called to keep the session alive
            d11) After that the StandardHostValve invokes the pipeline on the context associated
                 with the request.
            d12) The first valve that gets invoked by the Context pipeline is the FormAuthenticator
                 valve. Then the StandardContextValve gets invoke.
                 The StandardContextValve invokes any context listeners associated with the context.
                 Next it invokes the pipeline on the Wrapper component (StandardWrapperValve)
            d13) During the invocation of the StandardWrapperValve, the JSP wrapper (Jasper) gets
invoked
                 This results in the actual compilation of the JSP.
                 And then invokes the actual servlet.
        e) Invocation of the servlet class
```