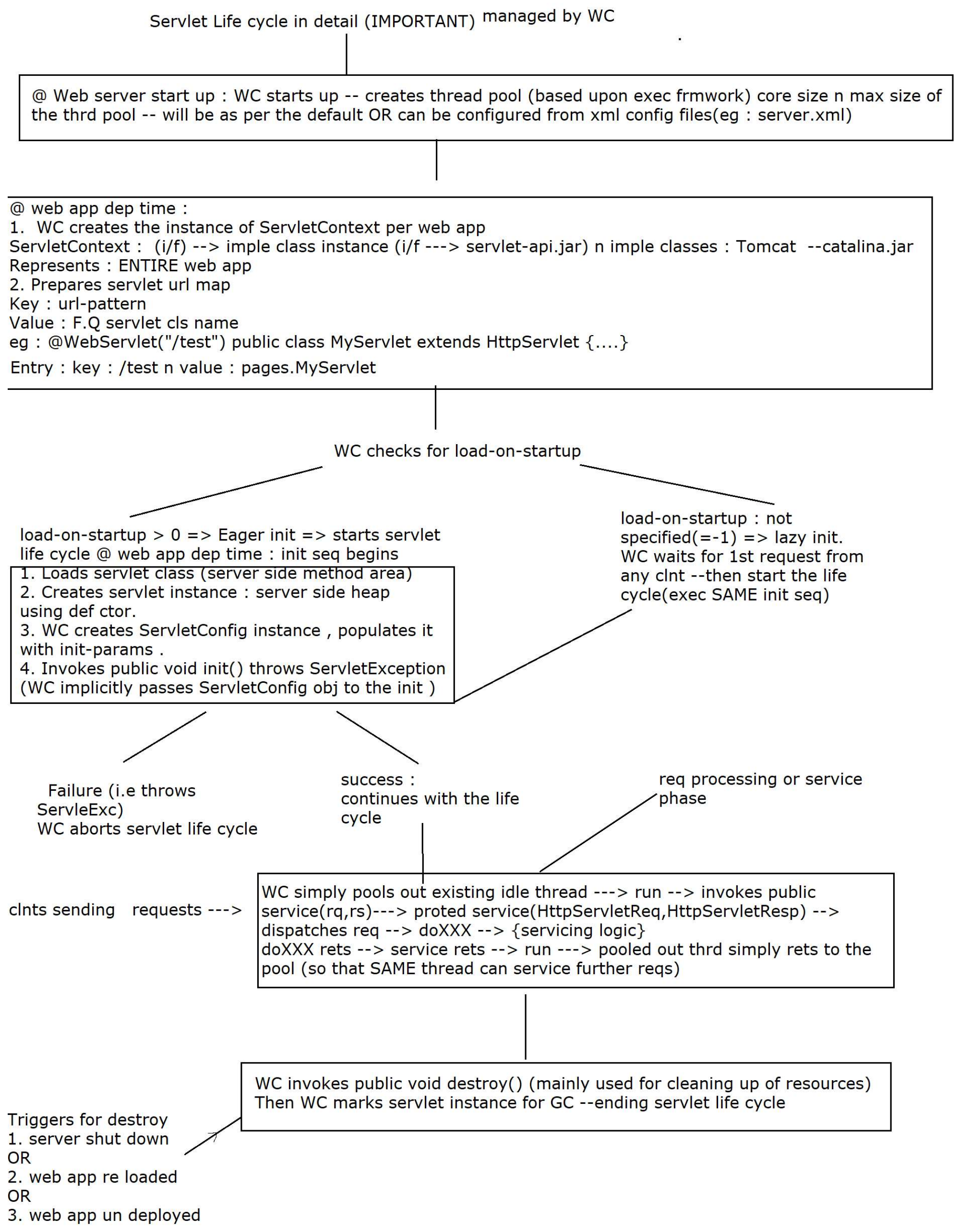

Servlet Life cycle in detail (IMPORTANT) managed by WC

@ Web server start up : WC starts up -- creates thread pool (based upon exec frmwork) core size n max size of the thrd pool -- will be as per the default OR can be configured from xml config files(eg : server.xml)

@ web app dep time :
1. WC creates the instance of ServletContext per web app
ServletContext : (i/f) --> imple class instance (i/f ---> servlet-api.jar) n imple classes : Tomcat --catalina.jar
Represents : ENTIRE web app
2. Prepares servlet url map
Key : url-pattern
Value : F.Q servlet cls name
eg : @WebServlet("/test") public class MyServlet extends HttpServlet {....}
Entry : key : /test n value : pages.MyServlet

WC checks for load-on-startup

load-on-startup > 0 => Eager init => starts servlet life cycle @ web app dep time : init seq begins

1. Loads servlet class (server side method area)
2. Creates servlet instance : server side heap using def ctor.
3. WC creates ServletConfig instance , populates it with init-params .
4. Invokes public void init() throws ServletException (WC implicitly passes ServletConfig obj to the init)

load-on-startup : not specified(=-1) => lazy init.
WC waits for 1st request from any clnt --then start the life cycle(exec SAME init seq)

Failure (i.e throws ServleExc)
WC aborts servlet life cycle

success :
continues with the life cycle

req processing or service phase

clnts sending requests --->

WC simply pools out existing idle thread ---> run --> invokes public service(rq,rs)---> proted service(HttpServletReq,HttpServletResp) --> dispatches req --> doXXX --> {servicing logic}
doXXX rets --> service rets --> run ---> pooled out thrd simply rets to the pool (so that SAME thread can service further reqs)

WC invokes public void destroy() (mainly used for cleaning up of resources)
Then WC marks servlet instance for GC --ending servlet life cycle

Triggers for destroy
1. server shut down
OR
2. web app re loaded
OR
3. web app un deployed