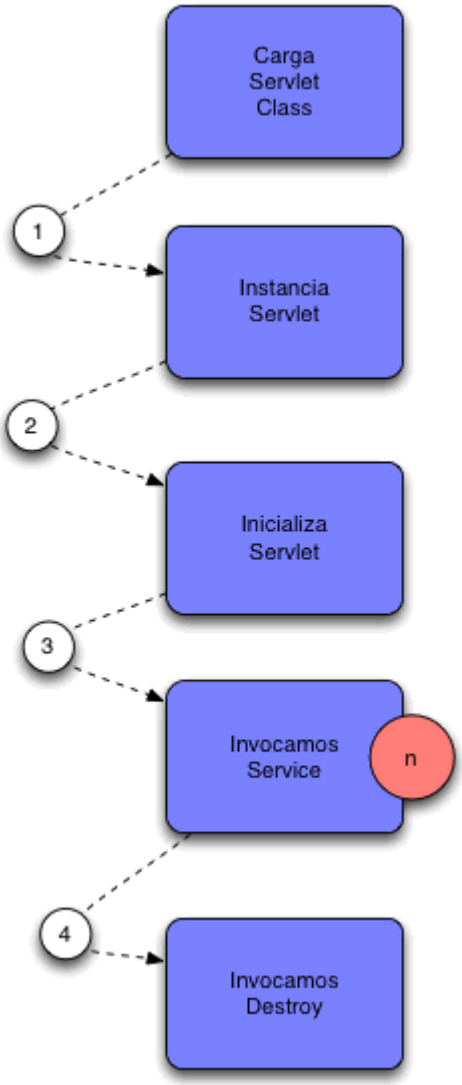
Trabajamos con Servlets todos los días pero muchas veces se nos olvida como funcionan exactamente y cual es el Servlet lifecycle o ciclo de vida, vamos a verlo un poco más a detalle. Para ello primero tenemos que ver cuales son los distintos estados por los que un Servlet pasa.

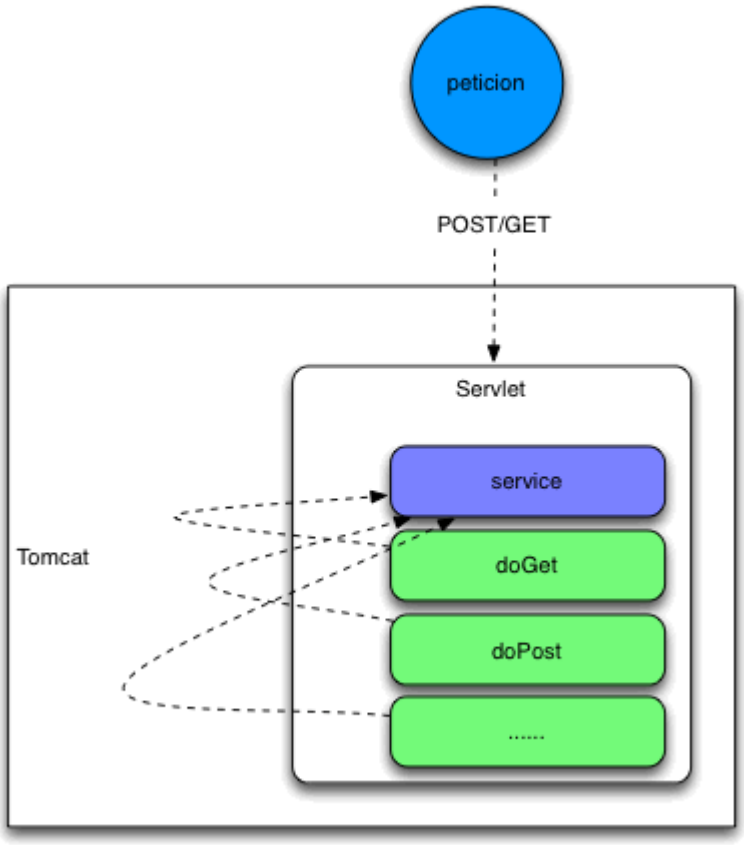


Los estados por los que un Servlet pasa son los siguientes:

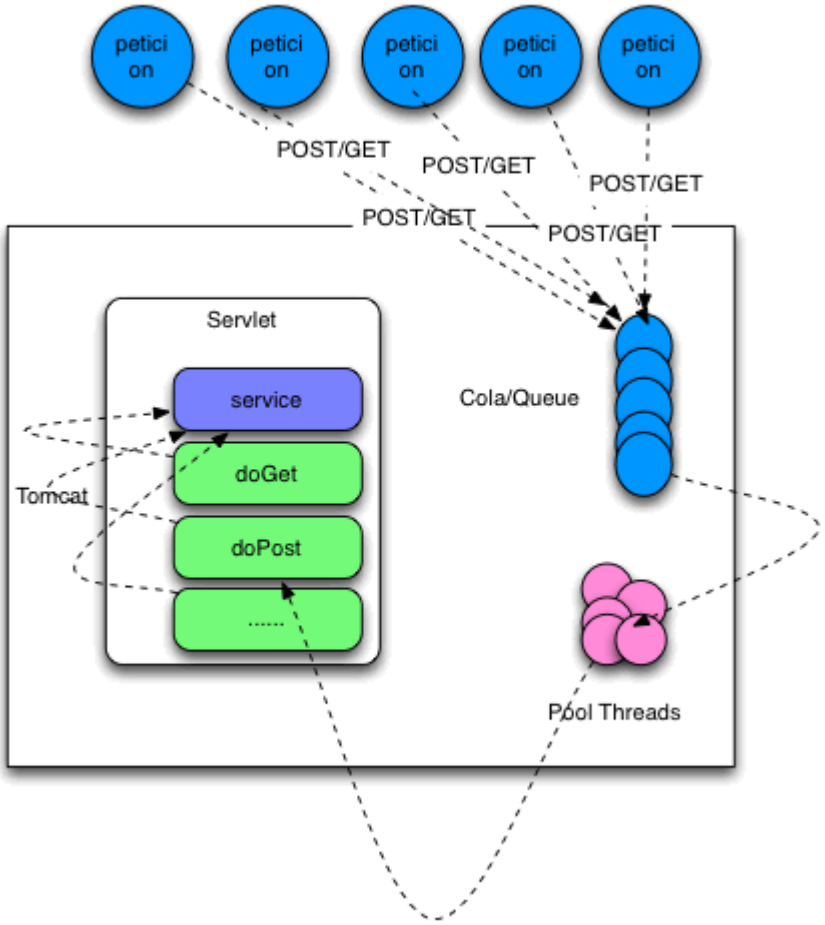
1. Cargar el Servlet:El primer paso que realizaremos será cargar la clase Servlet que queramos instanciar en memoria a través de su clase ServletHelloWorld.class por ejemplo.
2. Instanciar el Servlet:El segundo paso será instanciar un Servlet como objeto para poder utilizarlo.
3. Invocar init():El tercer paso será invocar a su método init() para registrarlo correctamente en el Servlet Container (se ejecuta una sola vez)
4. Invocar service():El cuarto paso es el más habitual que será invocar repetidamente al método service() a través de doGet() o doPost() para que el Servlet ejecute la funcionalidad solicitada.
5. Invocar destroy():Por último cuando el Servlet Container (Tomcat por ejemplo) considere que este Servlet no se necesita mas puede invocar al método destroy y eliminar el Servlet de memoria liberando recursos.

Servlet LifeCycle y Contenedores

Una vez que tenemos claro cual es el ciclo de vida tenemos que ver como un contenedor de Servlets tipo Tomcat lo implementa. Habitualmente un contenedor de Servlets recibe peticiones HTTP para acceder a un Servlet concreto por lo tanto instancia el Servlet y lo pone a disposición de cada una de las peticiones a través del método service() al cual doPost() doGet() etc delegan.



Para realizar esta operación el Servlet Container usa un pool de Threads que se encarga de gestionar las peticiones que han sido encoladas.



De esta forma es como un Servlet Container habitualmente trabaja con un Servlet a la hora de gestionar las peticiones que le llegan. Por ello los Servlets por defecto no son Thread Safe.

Otros artículos relacionados : Servlet3.0 ,ServletContext ServletFilters