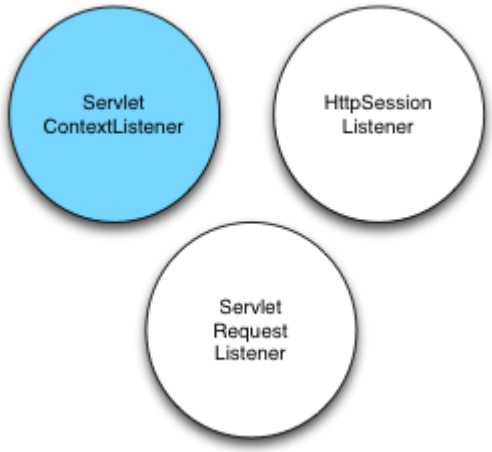
Ya hemos hablado del concepto de ServletContext en un articulo anterior .Vamos a complementar aquel articulo centrandonos en otro concepto importante de JEE a nivel web los listeners y en concreto en este caso el ServletContextListener . Estos listener estan diseñados para escuchar los diferentes eventos que se producen en el ciclo de vida de la aplicacion web. Los listener mas importantes son los siguientes .



ServletContextListener: Listener que se encarga de gestionar los eventos generales de la aplicación como son arranque y parada.

HttpSessionListener :Listener que se encarga de gestionar los propios eventos de la sessión como creación ,invalidación y destrucción de sessiones.

ServletRequestListener : Listener que se encarga de los eventos de creación y destruccion de peticiones.

Usando ServletContextListener

En nuestro caso vamos ha hacer un ejemplo elemental de ServletContextListener de tal forma que nos avise por consola cuando la aplicación se arranca y se para.

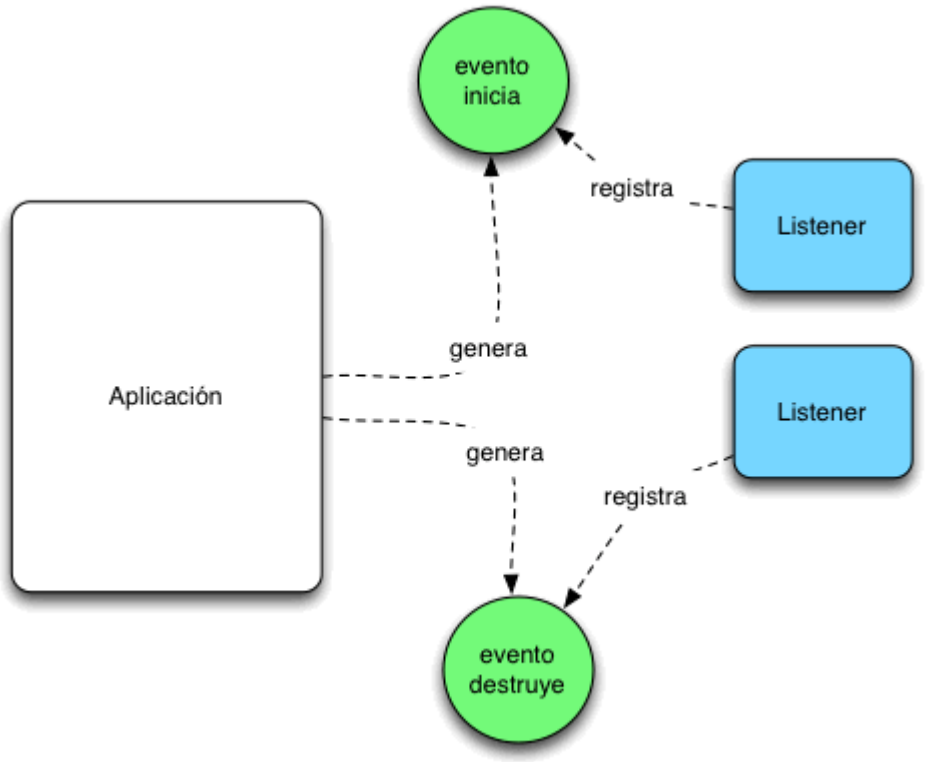
```
package com.arquitecturajava;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

@WebListener
public class MiListener implements ServletContextListener {
 @Override
 public void contextInitialized(ServletContextEvent sce) {
 System.out.print("aplicacion web arrancada");
 }

 @Override
 public void contextDestroyed(ServletContextEvent sce) {
 System.out.print("aplicacion web parada");
 }
 }
```

Este listener registra dos eventos fundamentales contextInitialized que se ejecuta al arrancar la aplicación y contextDestroyed que se ejecuta cuando la aplicación finaliza.



Podemos usar estos eventos para cargar datos en memoria cuando arranca la aplicación enviarnos un correo informándonos de inicio o parada ,o para otras muchas necesidades que se nos ocurran.

ServletContextListener y @WebListener

Hemos hecho uso de la anotación WebListener de Servlet 3.0 por simplicidad .Sin embargo podríamos haber registrado el listener a nivel de web.xml que es algo mas clásico y que permite un control mas exhaustivo . Ya que podriamos declarar varios y definir el orden entre ellos en el fichero.

```
</pre>
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
 <display-name>ServletContextListener</display-name>
<listener>
<listener-class>com.arquitecturajava.MiListener</listener-class>
</listener>
</web-app>
<pre>
```