Java Servlet Listener

- **Why do we have Servlet Listener?**
  We know that using ServletContext, we can create an attribute with application scope that all other servlets can access but we can initialize **ServletContext init parameters as String only** in deployment descriptor (web.xml). What if our application is database oriented and we want to set an attribute in ServletContext for Database Connection. If you application has a single entry point (user login), then you can do it in the first servlet request but if we have multiple entry points then doing it everywhere will result in a lot of code redundancy. Also if database is down or not configured properly, we won't know until first client request comes to server.
  Event is occurrence of something, in web application world an event can be initialization of application, destroying an application, request from client, creating/destroying a session, attribute modification in session etc.
  Servlet API provides different types of Listener interfaces that we can implement and configure in web.xml to process something when a particular event occurs. For example, in above scenario we can create a Listener for the application startup event to read context init parameters and create a database connection and set it to context attribute for use by other resources.

- **Servlet Listener Interfaces and Event Objects**
  Servlet API provides different kind of listeners for different types of Events. Listener interfaces declare methods to work with a group of similar events, for example we have ServletContext Listener to listen to startup and shutdown event of context. Every method in listener interface takes Event object as input. Event object works as a wrapper to provide specific object to the listeners.
  Servlet API provides following event objects.

1. **javax.servlet.AsyncEvent** – Event that gets fired when the asynchronous operation initiated on a ServletRequest (via a call to ServletRequest#startAsync or ServletRequest#startAsync(ServletRequest, ServletResponse)) has completed, timed out, or produced an error.
2. **javax.servlet.http.HttpSessionBindingEvent** – Events of this type are either sent to an object that implements HttpSessionBindingListener when it is bound or unbound from a session, or to a HttpSessionAttributeListener that has been configured in the web.xml when any attribute is bound, unbound or replaced in a session.
3. **javax.servlet.http.HttpSessionEvent** – This is the class representing event notifications for changes to sessions within a web application.
4. **javax.servlet.ServletContextAttributeEvent** – Event class for notifications about changes to the attributes of the ServletContext of a web application.
5. **javax.servlet.ServletContextEvent** – This is the event class for notifications about changes to the servlet context of a web application.
6. **javax.servlet.ServletRequestEvent** – Events of this kind indicate lifecycle events for a ServletRequest. The source of the event is the ServletContext of this web application.
7. **javax.servlet.ServletRequestAttributeEvent** – This is the event class for notifications of changes to the attributes of the servlet request in an application.

Servlet API provides following Listener interfaces.

1. **javax.servlet.AsyncListener** – Listener that will be notified in the event that an asynchronous operation initiated on a ServletRequest to which the listener had been added has completed, timed out, or resulted in an error.
2. **javax.servlet.ServletContextListener** – Interface for receiving notification events about ServletContext lifecycle changes.
3. **javax.servlet.ServletContextAttributeListener** – Interface for receiving notification events about ServletContext attribute changes.
4. **javax.servlet.ServletRequestListener** – Interface for receiving notification events about requests coming into and going out of scope of a web application.
5. **javax.servlet.ServletRequestAttributeListener** – Interface for receiving notification events about ServletRequest attribute changes.
6. **javax.servlet.http.HttpSessionListener** – Interface for receiving notification events about HttpSession lifecycle changes.
7. **javax.servlet.http.HttpSessionBindingListener** – Causes an object to be notified when it is bound to or unbound from a session.
8. **javax.servlet.http.HttpSessionAttributeListener** – Interface for receiving notification events about HttpSession attribute changes.
9. **javax.servlet.http.HttpSessionActivationListener** – Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated.

- **Servlet Listener Configuration**
  We can define listener in web.xml as:

```
<listener>
    <listener-class>
    com.journaldev.listener.AppContextListener
    </listener-class>
</listener>
```

- **ServletContextListener**
  We will read servlet context init parameters to create the DBConnectionManager object and set it as attribute to the ServletContext object.

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

import com.journaldev.db.DBConnectionManager;

@WebListener
public class AppContextListener implements ServletContextListener {

    public void contextInitialized(ServletContextEvent servletContextEvent) {
        ServletContext ctx = servletContextEvent.getServletContext();

        String url = ctx.getInitParameter("DBURL");
        String u = ctx.getInitParameter("DBUSER");
        String p = ctx.getInitParameter("DBPWD");

        //create database connection from init parameters and set it to context
        DBConnectionManager dbManager = new DBConnectionManager(url, u, p);
        ctx.setAttribute("DBManager", dbManager);
        System.out.println("Database connection initialized for Application.");
    }

    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        ServletContext ctx = servletContextEvent.getServletContext();
        DBConnectionManager dbManager = (DBConnectionManager) ctx.getAttribute("DBManager");
        dbManager.closeConnection();
        System.out.println("Database connection closed for Application.");

    }

}
```

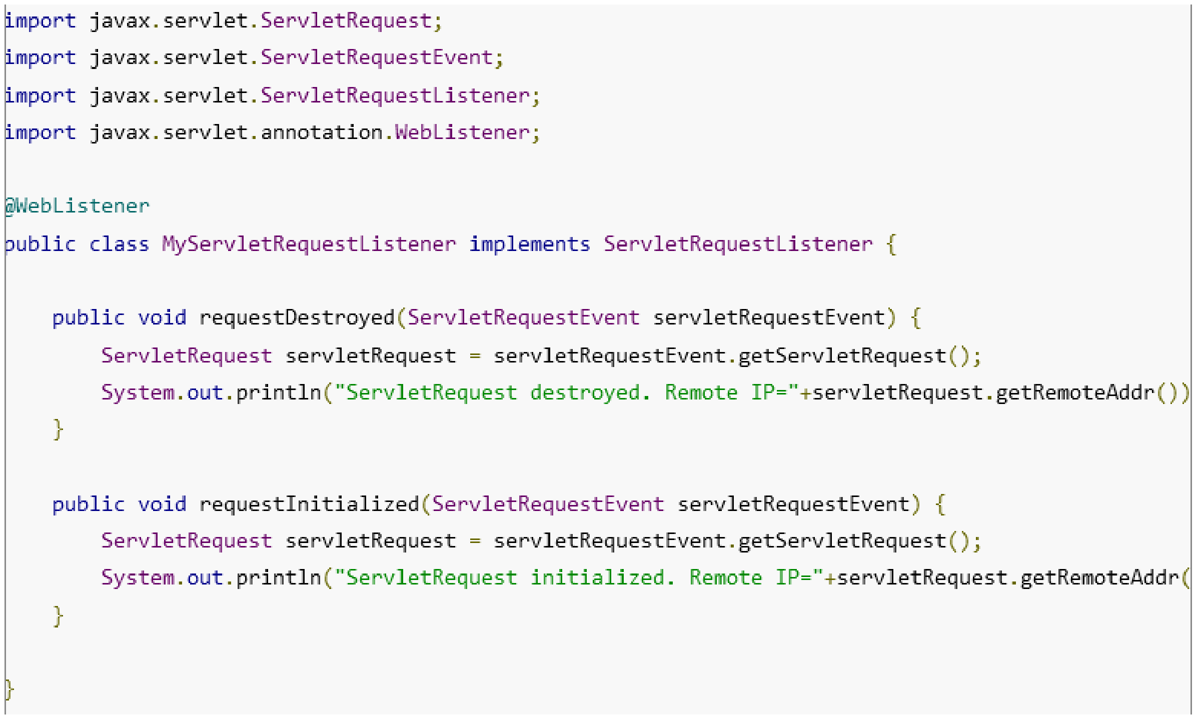
- **ServletContextAttributeListener**
  A simple implementation to log the event when attribute is added, removed or replaced in servlet context.

```
import javax.servlet.ServletContextAttributeEvent;
import javax.servlet.ServletContextAttributeListener;
import javax.servlet.annotation.WebListener;

@WebListener
public class AppContextAttributeListener implements
ServletContextAttributeListener {

    public void attributeAdded(ServletContextAttributeEvent
servletContextAttributeEvent) {
        System.out.println("ServletContext attribute
added::{"+servletContextAttributeEvent.getName()+","+servletContextAttribu
teEvent.getValue()+"}");
    }

    public void attributeReplaced(ServletContextAttributeEvent
servletContextAttributeEvent) {
        System.out.println("ServletContext attribute
replaced::{"+servletContextAttributeEvent.getName()+","+servletContextAttr
ibuteEvent.getValue()+"}");
    }

    public void attributeRemoved(ServletContextAttributeEvent
servletContextAttributeEvent) {
        System.out.println("ServletContext attribute
removed::{"+servletContextAttributeEvent.getName()+","+servletContextAttri
buteEvent.getValue()+"}");
    }

}
```

- **HttpSessionListener**
  A simple implementation to log the event when session is created or destroyed.

```
import javax.servlet.annotation.WebListener;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

@WebListener
public class MySessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent sessionEvent) {
        System.out.println("Session Created:: ID="+sessionEvent.getSession().getId());
    }

    public void sessionDestroyed(HttpSessionEvent sessionEvent) {
        System.out.println("Session Destroyed:: ID="+sessionEvent.getSession().getId());
    }

}
```

- **ServletRequestListener**
  A simple implementation of ServletRequestListener interface to log the ServletRequest IP address when request is initialized and destroyed.



```
import javax.servlet.ServletRequest;
import javax.servlet.ServletRequestEvent;
import javax.servlet.ServletRequestListener;
import javax.servlet.annotation.WebListener;

@WebListener
public class MyServletRequestListener implements ServletRequestListener {

    public void requestDestroyed(ServletRequestEvent servletRequestEvent) {
        ServletRequest servletRequest = servletRequestEvent.getServletRequest();
        System.out.println("ServletRequest destroyed. Remote IP="+servletRequest.getRemoteAddr())
    }

    public void requestInitialized(ServletRequestEvent servletRequestEvent) {
        ServletRequest servletRequest = servletRequestEvent.getServletRequest();
        System.out.println("ServletRequest initialized. Remote IP="+servletRequest.getRemoteAddr(
    }

}
```