Servlet Listener Example – ServletContextListener, HttpSessionListener and ServletRequestListener

This is the fifth article in the series of **Java Web Application**, you might want to check out earlier four articles too.

1. Java Web Application
2. Servlets in Java
3. Servlet Session Management
4. Servlet Filter

In this tutorial, we will look into **servlet listener**, benefits of listeners, some common tasks that we can do with listeners, servlet API listener interfaces and Event objects. In the end we will create a simple web project to show example of commonly used Listener implementation for **ServletContext**, **Session** and **ServletRequest**.

1. 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. To handle these scenario, servlet API provides Listener interfaces that we can implement and configure to listen to an event and do certain operations.

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.

2. 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.

A. **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.
B. **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.
The session binds the object by a call to HttpSession.setAttribute and unbinds the object by a call to HttpSession.removeAttribute.
We can use this event for cleanup activities when object is removed from session.
C. **javax.servlet.http.HttpSessionEvent** – This is the class representing event notifications for changes to sessions within a web application.
D. **javax.servlet.ServletContextAttributeEvent** – Event class for notifications about changes to the attributes of the ServletContext of a web application.
E. **javax.servlet.ServletContextEvent** – This is the event class for notifications about changes to the servlet context of a web application.
F. **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.
G. **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.

A. **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.

B. **javax.servlet.ServletContextListener** – Interface for receiving notification events about ServletContext lifecycle changes.
C. **javax.servlet.ServletContextAttributeListener** – Interface for receiving notification events about ServletContext attribute changes.
D. **javax.servlet.ServletRequestListener** – Interface for receiving notification events about requests coming into and going out of scope of a web application.
E. **javax.servlet.ServletRequestAttributeListener** – Interface for receiving notification events about ServletRequest attribute changes.
F. **javax.servlet.http.HttpSessionListener** – Interface for receiving notification events about HttpSession lifecycle changes.
G. **javax.servlet.http.HttpSessionBindingListener** – Causes an object to be notified when it is bound to or unbound from a session.
H. **javax.servlet.http.HttpSessionAttributeListener** – Interface for receiving notification events about HttpSession attribute changes.
I. **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. A container that migrates session between VMs or persists sessions is required to notify all attributes bound to sessions implementing HttpSessionActivationListener.

3. Servlet Listener Configuration

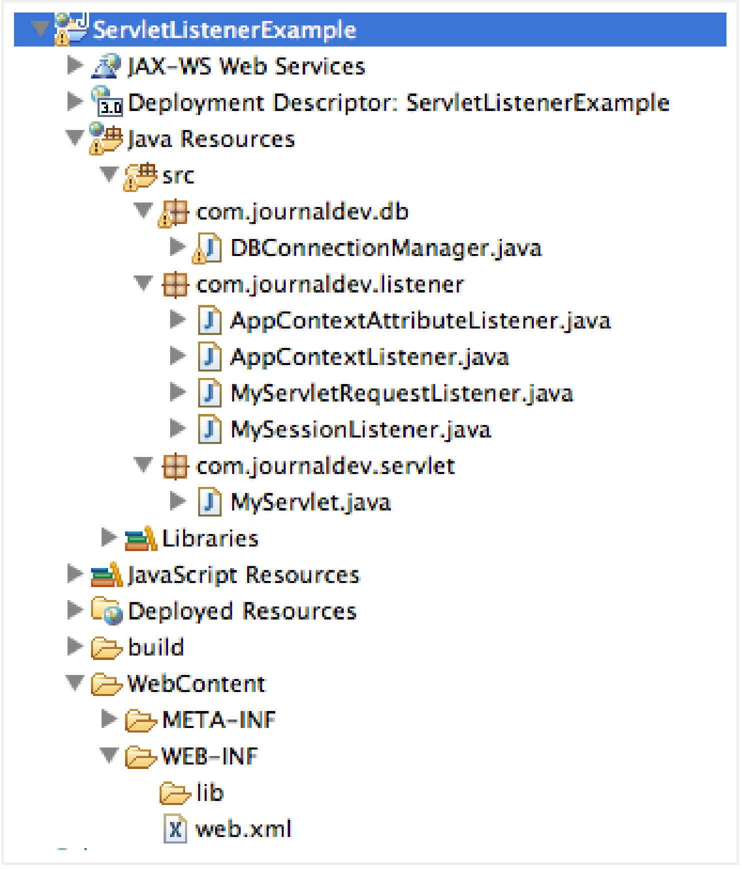
We can use **@WebListener** annotation to declare a class as Listener, however the class should implement one or more of the Listener interfaces.

We can define listener in web.xml as:

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

4. Servlet Listener Example

Let’s create a simple web application to see listeners in action. We will create dynamic web project in Eclipse **ServletListenerExample** those project structure will look like below image.



```
ServletListenerExample
  JAX-WS Web Services
  Deployment Descriptor: ServletListenerExample
  Java Resources
    src
      com.journaldev.db
        DBConnectionManager.java
      com.journaldev.listener
        AppContextAttributeListener.java
        AppContextListener.java
        MyServletRequestListener.java
        MySessionListener.java
      com.journaldev.servlet
        MyServlet.java
    Libraries
  JavaScript Resources
  Deployed Resources
  build
  WebContent
    META-INF
    WEB-INF
      lib
      web.xml
```

web.xml: In deployment descriptor, I will define some context init params and listener configuration.

web.xml

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

  <context-param>
    <param-name>DBUSER</param-name>
    <param-value>pankaj</param-value>
  </context-param>
  <context-param>
    <param-name>DBPWD</param-name>
    <param-value>password</param-value>
  </context-param>
  <context-param>
    <param-name>DBURL</param-name>
    <param-value>jdbc:mysql://localhost/mysql_db</param-value>
  </context-param>

  <listener>
    <listener-class>com.journaldev.listener.AppContextListener</listener-c
  </listener>
  <listener>
    <listener-class>com.journaldev.listener.AppContextAttributeListener</l
  </listener>
  <listener>
    <listener-class>com.journaldev.listener.MySessionListener</listener-cl
  </listener>
  <listener>
    <listener-class>com.journaldev.listener.MyServletRequestListener</list
  </listener>
</web-app>
```

DBConnectionManager: This is the class for database connectivity, for simplicity I am not providing code for actual database connection. We will set this object as attribute to servlet context.

DBConnectionManager.java

```java
package com.journaldev.db;

import java.sql.Connection;

public class DBConnectionManager {

    private String dbURL;
    private String user;
    private String password;
    private Connection con;

    public DBConnectionManager(String url, String u, String p){
        this.dbURL=url;
        this.user=u;
        this.password=p;
        //create db connection now

    }

    public Connection getConnection(){
        return this.con;
    }

    public void closeConnection(){
        //close DB connection here
    }
}
```

MyServlet: A simple servlet class where I will work with session, attributes etc.

MyServlet.java

```java
package com.journaldev.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/MyServlet")
public class MyServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse r
            ServletContext ctx = request.getServletContext();
            ctx.setAttribute("User", "Pankaj");
            String user = (String) ctx.getAttribute("User");
            ctx.removeAttribute("User");

            HttpSession session = request.getSession();
            session.invalidate();

            PrintWriter out = response.getWriter();
            out.write("Hi "+user);
    }

}
```

Now we will implement listener classes, I am providing sample listener classes for commonly used listeners – ServletContextListener, ServletContextAttributeListener, ServletRequestListener and HttpSessionListener.

A. ServletContextListener implementation

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

AppContextListener.java

```
package com.journaldev.listener;

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 servletContextE
        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 t
        DBConnectionManager dbManager = new DBConnectionManager(url, u
        ctx.setAttribute("DBManager", dbManager);
        System.out.println("Database connection initialized for Applic
    }

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

    }

}
```

B. ServletContextAttributeListener implementation

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

AppContextAttributeListener.java

```java
package com.journaldev.listener;

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

@WebListener
public class AppContextAttributeListener implements ServletContextAttr

    public void attributeAdded(ServletContextAttributeEvent servletCon
        System.out.println("ServletContext attribute added::{"+servlet
    }

    public void attributeReplaced(ServletContextAttributeEvent servlet
        System.out.println("ServletContext attribute replaced::{"+serv
    }

    public void attributeRemoved(ServletContextAttributeEvent servletC
        System.out.println("ServletContext attribute removed::{"+servl
    }

}
```

C. HttpSessionListener implementation

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

MySessionListener.java

```java
package com.journaldev.listener;

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.getSes
    }

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

}
```

D. ServletRequestListener implementation

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

MyServletRequestListener.java

```
package com.journaldev.listener;

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

@WebListener
public class MyServletRequestListener implements ServletRequestListene

    public void requestDestroyed(ServletRequestEvent servletRequestEve
        ServletRequest servletRequest = servletRequestEvent.getServlet
        System.out.println("ServletRequest destroyed. Remote IP="+serv
    }

    public void requestInitialized(ServletRequestEvent servletRequestE
        ServletRequest servletRequest = servletRequestEvent.getServlet
        System.out.println("ServletRequest initialized. Remote IP="+se
    }

}
```

Now when we will deploy our application and access MyServlet in browser with URL `http://localhost:8080/ServletListenerExample/MyServlet`, we will see following logs in the server log file.

```
ServletContext attribute added::{DBManager,com.journaldev.db.DBConnectionM
Database connection initialized for Application.
ServletContext attribute added::{org.apache.jasper.compiler.TldLocationsCa

ServletRequest initialized. Remote IP=0:0:0:0:0:0:0:1%0
ServletContext attribute added::{User,Pankaj}
ServletContext attribute removed::{User,Pankaj}
Session Created:: ID=8805E7AE4CCCF98AFD60142A6B300CD6
Session Destroyed:: ID=8805E7AE4CCCF98AFD60142A6B300CD6
ServletRequest destroyed. Remote IP=0:0:0:0:0:0:0:1%0


ServletRequest initialized. Remote IP=0:0:0:0:0:0:0:1%0
ServletContext attribute added::{User,Pankaj}
ServletContext attribute removed::{User,Pankaj}
Session Created:: ID=88A7A1388AB96F611840886012A4475F
Session Destroyed:: ID=88A7A1388AB96F611840886012A4475F
ServletRequest destroyed. Remote IP=0:0:0:0:0:0:0:1%0


Database connection closed for Application.
```

Notice the sequence of logs and it's in the order of execution. The last log will appear when you will shutdown the application or shutdown the container.

Thats all for listener in servlet, we will look into cookies and some common servlet examples next.

Please show your love by sharing or comments.

Download Project:

Download Servlet Listener Example Project
1365 downloads

Check out next article in the series about **Cookies in Servlet**.