UNIT : 9

Reference :-Marty Hall, Larry Brown, "Core Servlets and JavaServer Pages Volume – 2", Pearson Education, 2nd ed.(2004)
Chapter :- 06 : The Application Event Framework

Monitoring Creation & Destruction of Servlet Context

- The ServletContextListener class responds to **the initialization and destruction of the servlet context.**
- These events correspond to the **creation and shutdown of the Web application** itself.
- The **ServletContextListener** is most commonly used for :
  - to set up application-wide resources like database connection pools
  - to read the initial values of application-wide data that will be used by multiple servlets and JSP pages.
- To use listener there are **6 steps.**

Monitoring Creation & Destruction of Servlet Context

- **Step 1 → Implement ServletContextListener Interface.** (Available in javax.servlet package).
- **Step 2 → Implement contextInitialized & contextDestroyed**.

  contextInitialized
  - is triggered when the Web application is first loaded and the servlet context is created.
  - two most common tasks performed by this method are :
  - 1) creating application-wide data
  - 2) storing that data in an easily accessible location.

Monitoring Creation & Destruction of Servlet Context

contextDestroyed
- Is triggered when the Web application is being shut down and the servlet context is about to be destroyed.
- task performed by this method is the releasing of resources.
- used to close database connections associated with a now-obsolete connection pool

- **Step 3 → Obtain a reference to servlet context.**
  - Both methods contextInitialized and contextDestroyed takes **ServletContextEvent** as argument.
  - ServletContextEvent class has **getServletContext()** method.
  - Returns servlet context.

Monitoring Creation & Destruction of Servlet Context

- **Step 4 → Use servlet context.**
  - Read initialization parameter with **getInitParameter()** method.
  - Store data with **setAttribute()** method.
  - Make log file entries with log.
- **Step 5 → Declare the listener.**
  - Use the listener and listener-class elements to simply list the fully qualified name of the listener class. [Assume that it will be done in web.xml file] [in case of tag library same can be done in tld file]

```
<listener>
  <listener-class>somePackage.SomeListener</listener-class>
</listener>
```


Monitoring Creation & Destruction of Servlet Context

- **Step 6 → Provide any needed initialization parameter**
  - use the **get- InitParameter()** method to read context initialization parameters as the basis of data that will be made available to all servlets and JSP pages.
  - use the **context-param web.xml element** to provide the names and values of these initialization parameters.

```
<context-param>
  <param-name>name</param-name>
  <param-value>value</param-value>
</context-param>
```

Monitoring Creation & Destruction of Servlet Context

- **Step 6 → Provide any needed initialization parameter**
  - use the **get- InitParameter()** method to read context initialization parameters as the basis of data that will be made available to all servlets and JSP pages.
  - use the **context-param web.xml element** to provide the names and values of these initialization parameters.

```
<context-param>
   <param-name>name</param-name>
   <param-value>value</param-value>
</context-param>
```


Detecting changes in Servlet Context Attribute

- Setting up initial values of resources and store references to them in the servlet context when web application is loaded.
- **To get notification if resource changes** e.g. if value of Resource B depends upon value of Resource A, if resource A changes you need to automatically update value of Resource B.
- Handling this situation is the job of servlet context attribute listeners.
- It is a **five step process :**

Detecting changes in Servlet Context Attribute

- **Step 1 → Implement the ServletContextAttributeListener interface.**(Available in javax.servlet package).
- **Step 2 → Implement attributeAdded, attributeReplaced, and attributeRemoved.**

attributeAdded
  - Is triggered when a new attribute is added to the servlet context.

attributeReplaced
  - When a new value is assigned to an existing servlet context attribute, attribute- Added is triggered with the new value and ttributeReplaced is triggered with the old value

Detecting changes in Servlet Context Attribute

attributeRemoved
  - The attributeRemoved method is triggered when a servlet context attribute is removed altogether.

- **Step 3 → Obtain references to the attribute name, attribute value, and servlet context.**
  - Each of above three methods takes **ServletContextAttributeEvent** as argument.
  - The ServletContextAttributeEvent class has three useful methods:
  - **getName()** → Name of the attribute that has been changed.
  - **getValue()** → value of the changed attribute
  - **getServletContext()** → servletContext.

Detecting changes in Servlet Context Attribute

- **Step 4 → Use the objects.**
  - Normally comparison is made for attribute name to a stored name to see if it is the one you are monitoring.
  - The attribute value is used in an application-specific manner.
  - The servlet context is usually used to
  - 1) read previously stored attributes (getAttribute),
  - 2) store new or changed attributes (setAttribute), and
  - 3) make entries in the log file (log).
- **Step 5 → Declare the listener**
  - Use the listener and listener-class elements to simply list the fully qualified name of the listener class.

```
<listener>
  <listener-class>somePackage.SomeListener</listener-class>
</listener>
```


Recognizing Session Creation and Destruction

- session tracking, data is stored in per-user HttpSession objects, not in the servlet context.
- Where as Classes that implement the ServletContextListener and ServletContext- AttributeListener interfaces respond to creation, destruction, and changes in the servlet context, which is shared by all servlets and JSP pages in the Web application.
- To monitor changes to this user-specific data, is a job of the HttpSessionListener and HttpSessionAttributeListener interfaces.

Recognizing Session Creation and Destruction

- Http-SessionListener, the listener that is notified when a session is created or destroyed (either deliberately with invalidate or by timing out).
- HttpSessionAttribute- Listener, the listener that is notified when session attributes are added, replaced, or removed.
- HttpSessionListener involves **five steps.**

Monitoring Creation & Destruction of Servlet Context

- **Step 1 → Implement the HttpSessionListener interface.**(Available in javax.servlet.http package).
- **Step 2 → Implement sessionCreated and sessionDestroyed**

sessionCreated

  - is triggered when a new session is created.

sessionDestroyed

  - is triggered when a session is destroyed.
  - This destruction could be due to an explicit call to the invalidate method or because the elapsed time since the last client access exceeds the session timeout

Monitoring Creation & Destruction of Servlet Context

- **Step 3 → Obtain a reference to the session and possibly to the servlet context.**
  - Each of above two methods takes HttpSessionEvent as an argument.
  - HttpSessionEvent has getSession() method → provides access to session object.
  - occasionally also want a reference to the servlet context. If so, first obtain the session object and then call getServletContext on it.
- **Step 4 → Use the objects**
  - the only methods you usually call on the session object is the setAttribute method.

Monitoring Creation & Destruction of Servlet Context

- **Step 5 → Declare Listener**
  - In the web.xml or TLD file, use the listener and listener-class elements to simply list the fully qualified name of the listener class
  - <listener>
    <listener-class>somePackage.SomeListener</listener-class>
  - </listener>