ServletContextEvent Ve ServletContextListener Sınıflarının Kullanımı

Event(Olay): bir şeyin meydana gelmesi anlamına gelmektedir. Web uygulamalarında, bir event şu türlere sahip olabilir: uygulamanın **başlaması** (initialization of application), uygulamanın **silinmesi** (destroying an application), kullanıcıdan **istek** gelmesi(request from client), bir **session yaratma/yok etme,** session'da **attribute değişimleri** vs.

Servlet API farklı türlere sahip Listener interface'ler sağlar. Bu sayede özel bir event meydana geldiğinde bir işlem yaptırmak istiyorsak bu interface'leri kullanabiliriz. Örneğin, uygulama başlatıldığı zaman veritabanı bağlantısı yaratabiliriz.

Bir web uygulamasını sunucuya(Tomcat, Jetty vs) deploy ederken, **ServletContextEvent** olayı meydana gelir. Bu olayı yakalamak için **ServletContextListener** interface'i implement eden bir sınıf yaratıp, bu sınıfı **web.xml** dosyasının `<listener>` elementini kullanarak kayıt etmek gereklidir.

Not: ServletContextEvent bir web uygulamasının servlet context'inde meydana gelen olayları temsil eder.

Uygulamamızı **deploy** ederken bazı işlemlerin yapılmasını sağlayabiliriz. Örneğin **veritabanı** bağlantısı **yaratma,** dosya yaratma vs gibi işlemleri yapabiliriz.

Örnek Uygulama

web.xml

```xml
<context-param>
      <param-name>dbUrl</param-name>
      <param-value>jdbc:postgresql://localhost:5432/jbossdb</param-value>
  </context-param>
  <context-param>
      <param-name>dbUserName</param-name>
      <param-value>qwerty</param-value>
  </context-param>
  <context-param>
      <param-name>dbPassword</param-name>
      <param-value>qwerty</param-value>
  </context-param>
  <listener>
      <listener-class>mucayufa.java.web.MyServletListener</listener-class>
  </listener>
```

MyServletListener

```java
public class MyServletListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        ServletContext context=servletContextEvent.getServletContext();
        String dburl=context.getInitParameter("dbUrl");
        String dbusername=context.getInitParameter("dbUserName");
        String dbpassword=context.getInitParameter("dbPassword");

        DBConnector.createConnection(dburl, dbusername, dbpassword);
        System.out.println("Connection Establised.........");

    }

    @Override
    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        DBConnector.closeConnection();
    }
}
```

DBConnector

```java
public class DBConnector {

    private static Connection con;

    public static void createConnection(String dbUrl,String dbusername,String dbPassword){
        try {
            Class.forName("org.postgresql.Driver");
            con= DriverManager.getConnection(dbUrl, dbusername, dbPassword);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public static Connection getConnection(){
```

```
15          return con;
16      }
17
18      public static void closeConnection(){
19          if(con!=null){
20              try {
21                  con.close();
22              } catch (SQLException ex) {
23                  ex.printStackTrace();
24              }
25          }
26
27      }
28  }
```

Veritabanı bağlantısı açma-kapama işlemleri ile ilgili detaylı bilgi için **tıklayınız** (www.softwarevol.com/en/tutorial/Java-Connection-Pooling)