20장 ServletContextListener 구현

- 서블릿은 다양한 시점에서 발생하는 이벤트와 이벤트를 처리하기 위한 인터페이스를 정의하고 있다.

- 이들 이벤트와 인터페이스를 이용하면 웹 어플리케이션에서 필요로 하는 데이터의 초기화나 요청 처리 등을 추적할 수 있게 된다.

20.1 ServletContextListener를 이용한 이벤트 처리

- 웹 컨테이너는 웹 어플리케이션이 시작되거나 종료되는 시점에 특정 클래스의 메서드를 실행할 수 있는 기능을 제공하고 있다. 이 기능을 사용하면 웹 어플리케이션을 실행할 때 필요한 초기화 작업이나 종료된 후 사용된 자원을 반환하는 등의 작업을 수행할 수 있다.

- 웹 어플리케이션이 시작되고 종료될 때 특정한 기능을 실행하려면 다음과 같이 코드를 작성한다.
  1. javax.servlet.ServletContextListener 인터페이스를 구현한 클래스를 작성한다.
  2. web.xml 파일에 1번에서 작성한 클래스를 등록한다.

[chap20/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

         <listener>
                  <listener-class>jdbc.DBCPInitListener</listener-class>
         </listener>

         <context-param>
                  <param-name>poolConfig</param-name>
                  <param-value>
                           jdbcdriver=com.mysql.jdbc.Driver
                           jdbcUrl=jdbc:mysql://localhost:3306/guestbook?characterEncoding=utf8
                           dbUser=jspexam
                           dbPass=jsppw
                           poolName=guestbook
                  </param-value>
         </context-param>

</web-app>
```

[chap20/jdbc/DBCPInitListener.java]

```
package jdbc;

import java.io.IOException;
import java.io.StringReader;
import java.sql.DriverManager;
import java.util.Properties;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

```

```
11  import org.apache.commons.dbcp2.ConnectionFactory;
12  import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
13  import org.apache.commons.dbcp2.PoolableConnection;
14  import org.apache.commons.dbcp2.PoolableConnectionFactory;
15  import org.apache.commons.dbcp2.PoolingDriver;
16  import org.apache.commons.pool2.impl.GenericObjectPool;
17  import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
18
19  public class DBCPInitListener implements ServletContextListener {
20
21          @Override
22          public void contextInitialized(ServletContextEvent sce) {
23                  String poolConfig =
24                                  sce.getServletContext().getInitParameter("poolConfig");
25                  Properties prop = new Properties();
26                  try {
27                          prop.load(new StringReader(poolConfig));
28                  } catch (IOException e) {
29                          throw new RuntimeException("config load fail", e);
30                  }
31                  loadJDBCDriver(prop);
32                  initConnectionPool(prop);
33          }
34
35          private void loadJDBCDriver(Properties prop) {
36                  String driverClass = prop.getProperty("jdbcdriver");
37                  try {
38                          Class.forName(driverClass);
39                  } catch (ClassNotFoundException ex) {
40                          throw new RuntimeException("fail to load JDBC Driver", ex);
41                  }
42          }
43
44          private void initConnectionPool(Properties prop) {
45                  try {
46                          String jdbcUrl = prop.getProperty("jdbcUrl");
47                          String username = prop.getProperty("dbUser");
48                          String pw = prop.getProperty("dbPass");
49
50                          ConnectionFactory connFactory =
51                                          new  DriverManagerConnectionFactory(jdbcUrl,  username,
52  pw);
53
54                          PoolableConnectionFactory poolableConnFactory =
55                                          new PoolableConnectionFactory(connFactory, null);
56                          poolableConnFactory.setValidationQuery("select 1");
57
58                          GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
59                          poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60L * 5L);
60                          poolConfig.setTestWhileIdle(true);
61                          poolConfig.setMinIdle(5);
62                          poolConfig.setMaxTotal(50);
63
64                          GenericObjectPool<PoolableConnection> connectionPool =
65                                          new           GenericObjectPool<>(poolableConnFactory,
66  poolConfig);
67                          poolableConnFactory.setPool(connectionPool);
68
69                          Class.forName("org.apache.commons.dbcp2.PoolingDriver");
70                          PoolingDriver driver = (PoolingDriver)
71                                  DriverManager.getDriver("jdbc:apache:commons:dbcp:");
72                          String poolName = prop.getProperty("poolName");
73                          driver.registerPool(poolName, connectionPool);
74                  } catch (Exception e) {
75                          throw new RuntimeException(e);
```

```
76                          }
77              }
78
79              @Override
80              public void contextDestroyed(ServletContextEvent sce) {
81              }
82
83    }
```

20.1.1 리스너의 실행순서

- 한 개 이상의 리스너 등록 가능
  - contextInitialized() 메서드는 등록된 순서대로 실행
  - contextDestroyed() 메서드는 등록된 반대 순서대로 실행

20.1.2 리스너에서의 익셉션 처리

- 리스너의 메서드에 try - catch로 예외를 잡은 뒤, RuntimeException을 발생시키도록 함
  - 리스너가 예외를 발생할 경우 웹 어플리케이션 시작에 실패함

```
public void contextInitialized(ServletContextEvent sce) {
    try {
        ...
        ...
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
```

20.1.3 애노테이션을 이용한 리스너 등록

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

@WebListener
public class CListener implements ServletContextListener {
   ...
}
```