서블릿 이벤트

서블릿은 다양한 시점에 발생되는 이벤트와 이벤트를 처리하기 위한 인터페이스를 정의하고 있다. 이들 이벤트와 인터페이스를 이용하면 웹 어플리케이션에서 필요로 하는 데이터의 초기화나 요청 처리 등을 추적할 수 있게 된다.
서블릿 규약은 다양한 이벤트를 처리할 수 있는 인터페이스를 정의하고 있는데, 그 중에 **ServletContextListener** 를 살펴본다.
데이터 저장 영역(application, session, request)에 데이터가 들어가고 나가는 혹은 그 객체가 생성되고 소멸되는 일련의 작업들에 대해 컨트롤한다.

1. 이벤트가 작동되는 경우

1) 컨텍스트(웹 어플리케이션)가 초기화되는 경우
2) 세션이 생기거나 소멸되는 경우
3) 속성이 바뀌는 경우

이런 이벤트에 대해서 미리 web.xml 파일에 등록해 두면 웹 서버는 자동으로 이벤트를 감지하고 우리가 지정한 클래스 내의 메서드를 실행해 준다.
이벤트는 외부의 변화보다는 내부의 변화를 처리하는 경우가 많다.
서블릿에서 이벤트 클래스를 실행되게 하려면 무엇보다 **web.xml** 파일의 설정이 중요하다.
초기에 **웹 서버가 구동하면서 해당 이벤트를 대기 상태로 두기 때문이다**.

2. ServletContextListener 를 이용한 이벤트 처리

웹 컨테이너는 웹 어플리케이션(컨텍스트)이 시작되거나 종료되는 시점에 특정 클래스의 메서드를 실행할 수 있는 기능을 제공하고 있다.
이 기능을 사용하면 웹 어플리케이션을 실행하는 데 필요한 초기화 작업이나 웹 애플리케이션이 종료된 후 사용된 자원을 반환하는 등의 작업을 수행할 수 있다.
웹 어플리케이션이 시작되고 종료될 때 특정한 기능을 실행하려면 다음과 같이 코드를 작성해야 한다.

1. javax.servlet.**ServletContextListener** 인터페이스를 구현한 클래스를 작성한다.
2. **web.xml** 파일에 1 에서 작성한 클래스를 등록한다.

javax.servlet.ServletContextListener 인터페이스는 웹 어플리케이션이 시작되거나 종료될 때 호출되는 메서드를 정의한 인터페이스로서, 다음과 같은 두 개의 메서드가 정의되어 있다.

public void **contextInitialized**(SerlvetContextEvent sce) //웹 애플리케이션이 초기화될 때 호출된다.
public void **contextDestoryed**(ServletContextEvent sce) //웹 애플리케이션이 종료될 때 호출된다.

ServletContextListener 인터페이스를 구현한 클래스가 웹 애플리케이션이 시작되거나 종료될 때 실행되도록 하려면 web.xml **<listener> 태그와 </listener-class>** 태그를 사용해서 완전한 클래스 이름을 명시해주면 된다.

```
<web-app>
	<listener>
		<listener-class>mvcjsp.jdbc.loader.DBCPInitiListener</listener-class>
	</listener>
	<listener>
	<!-- 웹 애플리케이션의 시작/종료 이벤트를 처리할 리스너 클래스의 완전한 이름을 값으로 갖는다. -->
		<listener-class>mvcjsp.jdbc.CodeInitListener</listener-class>
	</listener>
</web-app>
```

ServletContextListener 인터페이스에 정의된 두 메서드는 모두 파라미터로 javax.servlet.ServletContextEvent 타입의 객체를 전달 받는데, ServletContextEvent 클래스는 시작되거나 종료된 웹 애플리케이션 컨텍스트를 얻을 수 있는 getServletContext() 메서드를 제공하고 있다.

```
public ServletContext getServletContext()
```

```
<web-app>
	<context-param>
		<param-name>jdbcdriver</param-name>
		<param-value>com.sql.jdbc.Driver</param-value>
	</context-param>
</web-app>
```

getServletContext() 메서드가 리턴하는 ServletContext 객체는 JSP 의 application 기본 객체와 동일한 객체로서, ServletContext 객체를 이용하면 web.xml 파일에 설정된 어플리케이션 초기화 파라미터를 얻을 수 있다. 어플리케이션 초기화 파라미터는 <context-param> 태그를 사용해서 설정한다.

(1) ServletContext 가 제공하는 초기화 파라미터 관련 메서드

1) String **getInitParameter**(String name)

지정한 이름을 갖는 초기화 파라미터의 값을 리턴한다. 존재하지 않을 경우 null 을 리턴한다. name 파라미터에는 <param-name> 태그로 지정한 이름을 입력한다.

2) java.util.Enumeration.getInitParameterNames()

web.xml 파일에 정의된 초기화 파라미터의 이름 목록을 Enumeration 타입으로 리턴한다.
--> 초기화 파라미터는 주로 웹 어플리케이션을 구동하는 데 필요한 초기화 작업을 수행하는데 필요한 값을 설정하는 데 사용한다.

※ 리스너의 실행 순서

```
<listener>
        <listener-class>mvcjsp.jdbc.loader.AListener</listener-class>
</listener>
<listener>
        <listener-class>mvcjsp.jdbc.loader.BListener</listener-class>
</listener>
```

web.xml 에 한 개 이상의 리스너가 등록되어 있는 경우, contextInitialized() 메서드는 등록된 순서대로 실행되고 contextDestoryed() 메서드는 등록된 반대 순서대로 실행된다. 즉, 위 코드의 경우 웹 애플리케이션이 시작될 때는 AListener 가 먼저 실행되고 그 다음에 BListenter 가 실행된다. 반대로 웹 어플리케이션이 종료될 때는 BListener 가 종료되고 그 다음에 AListener 가 종료된다.

3. web.xml 에서 사용되는 태그

(1) <listener> 태그는 서블릿의 이벤트를 대기 상태로 둔다.

<listener> 태그를 가지고 설정할 수 있는 이벤트는 다음과 같다.

- ServletContextListener 이벤트

ServletContext 객체가 초기화되거나 소멸될 때 발생하는 이벤트이다. **contextInitialized**(), **contextDestoryed**() 메서드가 있다.

- ServletContextAttributeListener 이벤트

ServletContext 객체에 속성이 추가되거나 삭제되거나, 수정될 때 발생하는 이벤트이다.
attributeAdded(), **attributeRemoved**(), **attributeReplaced**() 메서드가 있다.

- HttpSessionListener 이벤트

HttpSession 객체가 생성되거나 소멸될 때 발생하는 이벤트이다.
sessionCreated(), **sessionDestroyed**() 메서드가 있다.

- HttpSessionAttributeListener 이벤트

ServletRequest 객체가 초기화되거나 소멸될 때 발생하는 이벤트이다.
requestInitialized(), **requestDestroyed**() 메서드가 있다.

- ServletRequestAttributedListener 이벤트

ServletRequest 객체에 속성이 추가되거나 삭제되거나, 수정될 때 발생하는 이벤트이다.
attributeAdded(), **attributeRemoved**(), **attributeReplaced**() 메서드가 있다.

(2) <context-param> 태그는 초기화 매개 변수가 된다.

(3) 예제 1

1) web.xml

```
<display-name>testProject</display-name>

<context-param>
	<param-name>co_name</param-name>
	<param-value>스마트 주식회사</param-value>
</context-param>
<context-param>
	<param-name>co_tel</param-name>
	<param-value>02-1234-5678</param-value>
</context-param>
<context-param>
	<param-name>admin_email</param-name>
	<param-value>test@smart.com</param-value>
</context-param>
```

2) 자바코드

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TestServlet extends HttpServlet {
	public void doGet(HttpServletRequest request, HttpServletResponse response)
				throwsServletException, IOException {

		ServletContext context = this.getServletContext();

		String co_name = context.getInitParameter("co_name");
		String co_tel = context.getInitParameter("co_tel");
		String admin_email = context.getInitParameter("admin_email");

		response.setContentType("text/html;charset=utf-8");
		PrintWriter out = response.getWriter();
```

```
                    out.println("<html><head>");
                    out.println("<style type='text/css'>");
                    out.println(".n_bo { border:none }");
                    out.println("</style>");
                    out.println("</head><body><center>");
                    out.println("회사상호 : ");
                    out.println("<input type='text' class='n_bo' value='" + co_name + "'/><br/>");
                    out.println("회사전화 : ");
                    out.println("<input type='text' class='n_bo' value='" + co_tel + "'/><br/>");
                    out.println("대표메일 : ");
                    out.println("<input type='text' class='n_bo' value='" + admin_email +
"'/><br/>");
                    out.println("</center></body></html>");
          }
}
```

화면 출력)
회사상호: 스마트 주식회사
회사전번: 02-1234-5678
대표메일: test@smart.com

이렇게 ServletContext 객체를 가지고 web.xml 파일에 등록된 초기화 매개변수들을 가져와서 사용할 수 있다. 그러나 ServletContext 객체와 관련된 이벤트에서는 이들 값의 추가, 삭제, 수정에 대한 정보를 확인할 수 없다. 이벤트가 관리하는 값들은 오직 해당 객체의 속성들뿐이며, 매개변수는 관리 대상에서 제외된다.

(4) 예제 2

이제 ServletContext 객체와 관련된 이벤트를 처리해 보도록 하자.
ServletContext 객체의 변화에 대해 반응하는 메서드들은 **ServletContextListener** 와 **ServletContextAttributeListener** 인터페이스에 정의되어 있기 때문에 이를 구현해 주어야 한다.

1) web.xml : 태그 순서 주의함!

```
<listener>
          <display-name>Context Listener</display-name>
          <listener-class>Round18_04_Servlet_Listener</listener-class>
</listener>
<context-param>
          <param-name>co_name</param-name>
          <param-value>승현 주식회사</param-value>
</context-param>
```

2) 자바코드

```
import javax.servlet.*;

public class Test2Servlet_Listener implements
ServletContextListener, ServletContextAttributeListener {

    public void contextInitialized(ServletContextEvent e) {
  // 톰캣이 구동될 때 실행된다.
        System.out.println("ServletContext 가 초기화 되었습니다.");
        System.out.println("init context = " + e.getServletContext());
    }
    public void contextDestroyed(ServletContextEvent e) {
  // 톰캣이 종료될 때 실행된다.
        System.out.println("ServletContext 가 소멸 되었습니다.");
        System.out.println("dest context = " + e.getServletContext());
    }
    public void attributeAdded(ServletContextAttributeEvent e) {
  // ServletContext 객체에 속성이 새로 추가될 때 실행된다.
        System.out.println("Context 영역에 값이 추가 되었습니다.");
        System.out.println("added = " + e.getName() + " : " + e.getValue());
    }
    public void attributeRemoved(ServletContextAttributeEvent e) {
  // ServletContext 객체의 속성이 삭제될 때 실행된다.
        System.out.println("Context 영역에 값이 삭제 되었습니다.");
        System.out.println("removed = " + e.getName() + " : " + e.getValue());
    }
    public void attributeReplaced(ServletContextAttributeEvent e) {
  // ServletContext 객체의 속성이 수정될 때 수정 직전에 실행된다.
        System.out.println("Context 영역에 값이 변경 되었습니다.");
        System.out.println("replaced = " + e.getName() + " : " + e.getValue());
    }
}
```

(5) 예제 3

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class Test3Servlet extends HttpServlet {
	public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		response.setContentType("text/html;charset=utf-8");
		PrintWriter out = response.getWriter();
		out.println("<html><body><center>");
		out.println("<form method='post'>");
		out.println("<input type='submit' value='Context 값 할당 하기'/>");
		out.println("</form></center></body></html>");
	}

	public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		ServletContext context = this.getServletContext();
		context.setAttribute("my_name", "홍길동");
		context.setAttribute("my_name", "길동");
		context.removeAttribute("my_name");

		response.setContentType("text/html;charset=utf-8");
		PrintWriter out = response.getWriter();
		out.println("<html><body><center>");
		out.println("Context 값 추가, 삭제, 변경 성공!");
		out.println("</center></body></html>");
	}
}
```

ServletContext 객체의 속성을 바꾸는 서블릿 코드이다.
웹서버가 구동 시점에는 최초로 contextInitialzed() 메서드가 실행되어 ServletContext 객체가 초기화된다.
URL 패턴에 맞추어 서블릿을 실행하면 페이지가 열리고 아무런 이벤트도 발생하지 않는다.
여기에서 Context 값 할당하기 단추를 누르면 doPost() 메서드가 실행되면서 ServletContextAttributeListener 이벤트가 발생하여 차례로 추가, 삭제, 수정 메시지를 확인할 수 있다.
이상과 같이 이벤트를 web.xml 파일에 등록하면 웹에서 특정 동작에 이벤트가 발생해서 해당 메서드를 실행하게 된다.