Listener

-Servlet API가 제공하는 Event class와 Listener interface를 이용하면 Servlet/JSP application에 event-driven programming을 도입할 수 있다.
-모든 event class는 java.util.EventObject class를 확장한다.
-Listener는 ServletContext, HttpSession, ServletRequest의 3가지 계층으로 구분되어 제공된다.

1. Listener interface 등록
1)javax.servlet과 javax.servlet.http package에 포함된 listener interface를 이용하면 listener를 만들 수 있다.

a. javax.servlet.ServletContextListener interface : Servlet Context 의 생명주기 event listener 로서, Servlet Context 가 생성될 때 호출되는 method와 종료될 때 호출되는 method 를 가진다.
b. javax.servlet.ServletContextAttributeListener : Servlet Context attribute 의 추가, 제거, 변경 event listener.
c. javax.servlet.http.HttpSessionListener : HttpSession 의 생성, 타임 아웃, 제거 event 의 listener.
d. javax.servlet.http.HttpSessionAttributeListener : Session 속성의 추가, 제거, 변경 event listener.
e. javax.servlet.http.HttpSessionActivationListener : HttpSession 의 활성화/비활성화 event listener
f. javax.servlet.http.HttpSessionBindingListener : HttpSession 속성으로 저장될 class 를 대상으로 하는 interface 로, 이 interface 를 구현한 class 가 HttpSession 에 추가되거나 삭제되면 HttpSessionBindingListener 로 event 가 전달된다.
g. javax.servlet.ServletRequestListener : ServletRequest 의 생성, 제거 event listener.
h. javax.servlet.ServletRequestAttributeListener : ServletRequest 속성의 추가, 제거, 변경 event listener.
i. javax.servlet.AsyncListener : 비동기 작업용 listener.

2)listener는 관련 interface를 구현하는 Java class로 간단히 생성한다.
3)Servlet container가 listener를 인지하려면 먼저 listener를 등록해야 하는데, Servlet 3.0에서는 두가지 등록 방법을 제공한다.

a. WebListener annotation을 class에 명시하는 방법
```
@WebListener
public class ListenerClass implements ListenerInterface {}
```

b. web.xml의 listener 요소에 listener를 등록하는 방법
```
<listener>
    <listener-class>fully qualified listener class</listener-class>
</listener>
```

4)listener method는 동기적으로 호출된다.

2. Servlet Context Listener
1)ServletContext 계층의 listener interface로는 ServletContextListener와 ServletContextAttributeListener가 있다.
2)ServletContextListener
a. ServletContext 가 초기활 될 때와 제거될 때 호출
b. 초기화될 때 Container 는 등록된 ServletContextListener 들의 contextInitialized method를 호출한다.
```
void contextInitialized(ServletContextEvent sce)
```

c. ServletContext가 제거될 때 등록된 모든 ServletContextListener의 contextDestroyed method를 호출한다.

```
void contextDestroyed(ServletContextEvent sce)
```

d. 이 두 method는 Servlet Container로 부터 ServletContextEvent 객체를 받는다.
e. javax.servlet.ServletContextEvent class 는 java.util.EventObject 의 자식 클래스로서, 다음과 같은 method 를 가진다.

```
public ServletContext getServletContext()
```

f. getServletContext() 만이 ServletContext 에 손쉽게 접근할 수 있는 유일한 방법을 제공
g. 대부분의 ServletContextListener 는 ServletContext 에 속성을 저장할 목적으로 만들어진다.

[AppListener.java]
```
package com.example.libs;
import java.util.HashMap;
import java.util.Map;
```

```
import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class AppListener implements ServletContextListener {
   @Override
   public void contextDestroyed(ServletContextEvent sce) {}

   @Override
   public void contextInitialized(ServletContextEvent sce) {
      ServletContext servletContext = sce.getServletContext();

      Map<String, String> countries = new HashMap<String, String>();
      countries.put("kr", "대한민국");
      countries.put("us", "미국");
      servletContext.setAttribute("countries", countries);
   }
}

[web.xml]
<listener>
   <listener-class>com.example.libs.AppListener</listener-class>
</listener>

[counties.jsp]
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>국가 목록</title>
</head>
<body>
지원 국가 목록:
<ul>
   <c:forEach items="${countries}" var="country">
      <li>${country.value}</li>
   </c:forEach>
</ul>
</body>
</html>
```

3. ServletContextAttributeListener
   1)ServletContext의 속성이 추가, 제거, 변경될 때 호출
   2)void attributeAdded(ServletContextAttributeEvent event)
   3)void attributeRemoved(ServletContextAttributeEvent event)
   4)void attributeReplaced(ServletContextAttributeEvent event)
   5)3개의 method는 모두 ServletContextAttributeEvent를 받는다.
   6)ServletContextAttributeEvent는
      a. java.lang.String getName()
      b. java.lang.Object getValue()

4. Session Listener
   1)HttpSession과 관련된 listener interface는 HttpSessionActivationListener, HttpSessionAttributeListener, HttpSessionBindingListener, HttpSessionListener가 있다.
   2)HttpSessionListener
      a. HttpSession이 생성되거나 제거될 때 등록된 모든 HttpSessionListener를 호출한다.
         -void sessionCreated(HttpSessionEvent se)
         -void sessionDestroyed(HttpSessionEvent se)

      b. HttpSessionEvent 를 인자로 받는다.
         -public HttpSession getSession()

   3)HttpSessionAttributeListener
      a. ServletContextAttributeListener와 비슷하지만, HttpSession에 속성이 추가, 제거, 변경될 때 호출
      b. void attributeAdded(HttpSessionBindingEvent event)

c. void attributeRemoved(HttpSessionBindingEvent event)
d. void attributeReplaced(HttpSessionBindingEvent event)
e. HttpSessionBindingEvent 를 인자로 받는다.
-public HttpSession getSession()
-java.lang.String getName()
-java.lang.Object getValue()

[SessionListener.java]

```
package com.example.libs;

import java.util.concurrent.atomic.AtomicInteger;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

public class SessionListener implements HttpSessionListener,
        ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        ServletContext servletContext = sce.getServletContext();
        servletContext.setAttribute("userCounter", new AtomicInteger());
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {}

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        ServletContext servletContext = session.getServletContext();
        AtomicInteger userCounter = (AtomicInteger) servletContext.getAttribute("userCounter");
        int userCount = userCounter.incrementAndGet();
        System.out.println("userCount가 다음으로 증가 :" + userCount);
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        ServletContext servletContext = session.getServletContext();
        AtomicInteger userCounter = (AtomicInteger) servletContext.getAttribute("userCounter");
        int userCount = userCounter.decrementAndGet();
        System.out.println("---------- userCount가 다음으로 감소 :" + userCount);
    }
}
```

-http://...../counties.jsp 호출할 것.

5. HttpSessionActivationListener

1)처리 용량을 확장할 목적으로 여러 서블릿 컨테이너를 이용해 구성한 분산 환경에서는 한 Servlet Container의 Session을 다른 Servlet Container로 이관하거나 메모리를 확보하기 위해 Session 속성을 직렬화해야 할 수도 있다.
2)일반적으로 메모리가 부족할 경우에는 메모리를 확보하기 위해 덜 사용되는 객체를 직렬화해 2차 저장소에 저장한다.
3)이때 세션 속성의 클래스가 HttpSessionActivationListener interface를 구현한다면 Servlet Container는 interface를 호출한다.
4)void sessionWillPassivate(HttpSessionEvent se)
-Listener 를 담은 HttpSession 이 비활성화(passivate)되려고 할 때 호출
-Servlet Container 가 전달하는 HttpSessionEvent 를 이용하면 활성화된 HttpSession 에 접근 가능

5)void sessionDidActivate(HttpSessionEvent se)
-Listener 객체를 가진 HttpSession 이 활성화된 다음 호출
-Servlet Container 가 전달하는 HttpSessionEvent 를 이용하면 활성화된 HttpSession 에 접근 가능

6. HttpSessionBindingListener
1)구현체가 HttpSession 에 추가(bound)되거나 제거(unbound)될 때 호출
2)class의 instance가 session 속성으로 저장되거나 제거될 시점에 특정 동작을 할 경우 사용

7. ServletRequest Listener
1)ServletRequest 계층에는 AsyncListener, ServletRequestListener, ServletRequestAttributeListener의 3가지 listener interface가 제공된다.

8. ServletRequestListener
1)ServletRequest가 생성, 제거될 때 호출
2)Servlet Container가 pool을 이용해 ServletRequest를 재사용한다면 풀에서 ServletRequest를 가져올 때가 생성시점이고, 풀로 반환할 때가 제거 시점이다.
3)void requestDestroyed(ServletRequestEvent sre)
-ServletRequest 가 제거될 때(또는 풀로 반환할 때) 호출

4)void requestInitialized(ServletRequestEvent sre)
-ServletRequest 가 생성될 때(또는 풀에서 가져올 때) 호출

5)ServletRequestEvent 를 인자로 받는다.
6)public ServletRequest getServletRequest() : ServletRequest instance 에 접근
7)public ServletContext getServletContext() : ServletContext 를 반환하는 메소드

[PerformanceDemo.java]

```
package com.example.libs;

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

@WebListener
public class PerformanceDemo implements ServletRequestListener {

    @Override
    public void requestInitialized(ServletRequestEvent sre) {
        ServletRequest servletRequest = sre.getServletRequest();
        servletRequest.setAttribute("start", System.nanoTime());
    }

    @Override
    public void requestDestroyed(ServletRequestEvent sre) {
        ServletRequest servletRequest = sre.getServletRequest();
        Long start = (Long) servletRequest.getAttribute("start");
        Long end = System.nanoTime();
        HttpServletRequest httpServletRequest = (HttpServletRequest) servletRequest;
        String uri = httpServletRequest.getRequestURI();
        System.out.println( uri + " 처리시간" +
                ": "  + ((end - start) / 1000) + "마이크로초");
    }
}
```

-http://...../counties.jsp 호출할 것.