1 ServletContextAttributeListener--了解

作用 : 可以检测ServletContext域中属性的变化.

具体为:

- 将某个属性添加到ServletContext域中
- ServletContext域中某个属性值被替换
- 将某个属性从ServletContext域中移除

1.1 API介绍

```
void attributeAdded(ServletContextAttributeEvent scab) 监听属性添加到servletcontext中
```

```
void attributeRemoved(ServletContextAttributeEvent scab) 监听属性从servletcontext中移除
```

```
void attributeReplaced(ServletContextAttributeEvent scab) 监听属性从servletcontext中被替换
```

1.2 使用步骤

1.创建一个类实现ServletContextAttributeListenner接口

2.给这个类添加注解@WebListener,(也可以使用配置文件的方式,只需要在web.xml中使用listener标签配置一下)

3.实现ServletContextAttributeListenner接口的方法

4.创建一个servlet，doGet方法中分别向servletContext对象中添加、替换、删除属性

1.3 案例代码

MyServletContextAttributeListenner代码:

```
package com.itheima.listenner;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextAttributeEvent;
import javax.servlet.ServletContextAttributeListener;
import javax.servlet.annotation.WebListener;

@WebListener
public class MyServletContextAttributeListenner implements
ServletContextAttributeListener {
    @Override
    public void attributeAdded(ServletContextAttributeEvent
servletContextAttributeEvent) {
            //获取被监听的对象
            ServletContext servletContext =
servletContextAttributeEvent.getServletContext();
            //获取被添加到servletContext对象中的属性名
```

```
            String name = servletContextAttributeEvent.getName();
            //获取被添加到servletContext对象中的属性值
            String value = (String)servletContextAttributeEvent.getValue();
            System.out.println("被添加到servletContext对象中的属性
是："+name+"="+value);
    }

    @Override
    public void attributeRemoved(ServletContextAttributeEvent
servletContextAttributeEvent) {
        //获取被监听的对象
        ServletContext servletContext =
servletContextAttributeEvent.getServletContext();
        //获取被移出servletContext对象中的属性名
        String name = servletContextAttributeEvent.getName();
        //获取被移出servletContext对象中的属性值
        String value = (String)servletContextAttributeEvent.getValue();
        System.out.println("被移出servletContext对象中的属性是："+name+"="+value);
    }

    @Override
    public void attributeReplaced(ServletContextAttributeEvent
servletContextAttributeEvent) {
        //获取被监听的对象
        ServletContext servletContext =
servletContextAttributeEvent.getServletContext();
        //获取servletContext对象中被替换的属性名
        String name = servletContextAttributeEvent.getName();
        //获取servletContext对象中被替换的属性值
        String value = (String)servletContextAttributeEvent.getValue();
        System.out.println("servletContext对象中的被替换前属性是："+name+"="+value);
    }
}
```

创建一个servlet,测试往ServletContext中添加属性,替换属性和删除属性：

```
package com.itheima.servlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ServletDemo",urlPatterns = "/servletDemo")
public class ServletDemo extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        ServletContext servletContext = getServletContext();
        //向servletContext对象中添加属性username:zhangsan
```

```
        servletContext.setAttribute("username","zhangsan");
        //替换servletContext对象中的属性username:lisi
        servletContext.setAttribute("username","lisi");
        //移除servletContext对象中的属性username
        servletContext.removeAttribute("username");
    }
}
```

同理：使用如下接口以相同的方式也可以监听到session对象和request对象域中属性的变化--了解

HttpSessionAttributeListener监听器：监听Httpsession域中属性的变化

ServletRequestAriibuteLister监听器：监听ServletRequest域中属性的变化

2 与session中的绑定的对象相关的监听器（对象感知监听器）--了解

```
即将要被绑定到session中的对象有几种状态
绑定状态：就一个对象被放到session域中
解绑状态：就是这个对象从session域中移除了
钝化状态：是将session内存中的对象持久化（序列化）到磁盘
活化状态：就是将磁盘上的对象再次恢复到session内存中
```

绑定与解绑的监听器

```
//感知user被绑定到session中的方法
@Override
public void valueBound(HttpSessionBindingEvent event) {
    System.out.println("user被绑定到session域中了");
    System.out.println(event.getName());
}

//感知user从session中解绑的方法
@Override
public void valueUnbound(HttpSessionBindingEvent event) {
    System.out.println("user从session域中解绑了");
    System.out.println(event.getName());
}
```

钝化与活化的监听器HttpSessionActivationListenerxt

```java
//钝化
@Override
public void sessionWillPassivate(HttpSessionEvent se) {
    System.out.println("costomer被钝化了....");
}
//活化
@Override
public void sessionDidActivate(HttpSessionEvent se) {
    System.out.println("costomer被活化了....");
}
```

注意：被序列化的对象要实现序列化Serializable接口