

3.18日上课内容

一、作业评讲

作业1：过滤器，初始化多个参数

过程

代码

java代码

结果

作业2：/* 换成/ 或 * 作用和区别？

过程

二、上课内容-1

监听器

监听器是什么？

监听器入门示例

ServletContextListener 监听器的接口

步骤

项目中的session如何监控？

三、上课内容-2

监听器

监听session

案例

问题：如何把session时间从xml文件设置转换到servlet中

作用：如何知道一个运行的网页session什么时候销毁？

监听request

案例

四、上课内容-3

监听器

监听attributeList,监听器获取不同作用域的数据

案例1.测验context获取

练习:测试获取request属性

思路：

具体实现

监听作用域的目的(作用)

案例:测试获取监听session对象属性

作业

把今天所讲的监听器+过滤器用在新闻发布系统案例

例如:所有的编码集问题可以用过滤器(request,response)

监听器：监听request,session[登录进去后,3分钟后把当前用户注销]

预习

servlet3.0 注解

3.18日上课内容

一、作业评讲

作业1：过滤器，初始化多个参数

过程

代码

```
//获取到 所有 KEY
Enumeration<String> names = arg0.getInitParameterNames();
//遍历所有的 key 输出所有的 value
while(names.hasMoreElements()){
    System.out.println(arg0.getInitParameter(names.nextElement()));
}

```

```
<init-param>
  <param-name>charset</param-name>
  <param-value>utf-8</param-value>
</init-param>
<init-param>
  <param-name>a1</param-name>
  <param-value>小明</param-value>
</init-param>

```

java代码

```
1 public void init(Filterconfig arg0){
2     Enumeration<String> enumeration=arg0.getInitParameterNames();
3     while(enumeration.hasMoreElements()){
4         System.out.println(arg0.getInitParameter(enumeration.nextElement()));
5     }
6 }

```

结果

```
警告: Creation of SecureRandom instance for session ID generation
----->过滤器初始化。。。
----->获取初始化的value:utf-8
=====
小明
utf-8
小红
三月 18, 2020 2:13:21 下午 org.apache.catalina.startup.HostConfig d

```

作业2:/* 换成/ 或 * 作用和区别?

过程

1. 编写两个jsp页面
2. 新建一个servlet

```
1 System.out.println("进入到loginServlet中");

```

3. 新建一个filter

```
1 //在doFilter中编写代码
2 System.out.println("-----》拦截一些请求，停在当前位置");

```

4. 测试结果

1. /:代表根目录下面的文件,类似于全局变量,根目录下面的请求,都可以直接访问,不会进入filter
2. *:等同于 /*

二、上课内容-1

监听器

监听器是什么?

监听器概述

- 监听器是Web应用程序事件模型的一部分
 - ◆ Web应用中的某些状态发生改变时会产生相应的事件
 - ServletContext、HttpSession、ServletRequest三个域对象引发的事件
 - 域对象中的属性引发的事件
 - ◆ 监听器可以接收这些事件,以便在事件发生时做出相关处理

监听器入门示例

ServletContextListener 监听器的接口

步骤

1. 创建项目
2. 创建LoginServlet
3. 在doGet中输出一句话

```
1 System.out.println("进入了loginservlet");
```

4. 新建一个监听器放在listener包下,名字叫MyListener
5. 实现接口ServletContextListener,添加未实现方法

```
1 public void contextDestroyed(ServletContextEvent arg0){
2     System.out.println("监听器销毁了");
3 }
4 public void contextInitialized(ServletContextEvent arg0){
5     System.out.println("监听器初始化了");
6 }
```

6. 配置监听器的xml文件

```

1 <listener>
2 <listener-class>包名+类名全路径</listener-class>
3 </listener>

```

7. 运行结果

```

三月十八, 2020 2:51:00 下午 org.apache.catalina.startup.
信息: Initialization processed in 4909 ms
三月十八, 2020 2:51:00 下午 org.apache.catalina.core.St
信息: Starting service Catalina
三月十八, 2020 2:51:00 下午 org.apache.catalina.core.St
信息: Starting Servlet Engine: Apache Tomcat/7.0.96
ServletContext初始化!
三月十八, 2020 2:51:03 下午 org.apache.catalina.util.Se
警告: Creation of SecureRandom instance for session
三月十八, 2020 2:53:16 下午 org.apache.catalina.core.Ap
信息: ContextListener: contextDestroyed()
ServletContext监听销毁了!
三月十八, 2020 2:53:17 下午 org.apache.coyote.AbstractP
信息: Stopping ProtocolHandler [http-bio-8080]

```

8. 总结:ServletContextEvent监听项目运行(启动,停止)

项目中的session如何监控?

1. 监控是否注销?
2. 监控是否启动?

三、上课内容-2

监听器

监听session

案例

1. 新建一个MySessionListener实现HttpSessionListener
2. 结果

```

/**
 * Default constructor.
 */
public MySessionListener() {
    // TODO Auto-generated constructor stub
}

/**
 * @see HttpSessionListener#sessionCreated(HttpSessionEvent)
 */
public void sessionCreated(HttpSessionEvent arg0) {
    // TODO Auto-generated method stub
    System.out.println("session被创建!");
}

/**
 * @see HttpSessionListener#sessionDestroyed(HttpSessionEvent)
 */
public void sessionDestroyed(HttpSessionEvent arg0) {
    // TODO Auto-generated method stub
    System.out.println("session被销毁!");
}

```

3. 在servlet中创建一个session

```
1 HttpSession session=request.getSession()  
2     System.out.println("session被创建出来，监控器监控session1分钟后销毁");
```

4. xml配置文件

```
<listener>  
    <listener-class>com.dt.listener.MyListener</listener-class>  
</listener>  
<session-config>  
    <session-timeout>1</session-timeout>  
</session-config>  
<listener>  
    <listener-class>com.dt.listener.MySessionListener</listener-class>  
</listener>
```

5. 执行测试结果

```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\JRE8\bin\javaw.exe (2020年3月18日 下午3:24:56)  
进入到LoginServlet中。。。。。  
session被创建!  
selvet----> session被创建出来，监控器监控session1分钟后销毁。。。。。
```

6. 一分钟后结果

```
进入到LoginServlet中。。。。。  
session被创建!  
selvet----> session被创建出来，监控器监控session1分钟后销毁。。。。。  
session被销毁!
```

7. 小结:session的创建和销毁由监控器帮我们处理了

问题：如何把session时间从xml文件设置转换到servlet中

在servlet加上如下代码

```
1 session.setMaxInactiveInterval(60); //单位秒
```

作用：如何知道一个运行的网页session什么时候销毁？

使用监听器就可以

例如：一个注册的用户，过半个小时后，要注销掉当前的用户，可以给session设置时间

监听request

案例

1. 新建Listener实现ServletRequestListener

```

public class MyRequestListener implements ServletRequestListener {

    /**
     * Default constructor.
     */
    public MyRequestListener() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see ServletRequestListener#requestDestroyed(ServletRequestEvent)
     */
    public void requestDestroyed(ServletRequestEvent arg0) {
        // TODO Auto-generated method stub
        System.out.println("request被销毁了!");
    }

    /**
     * @see ServletRequestListener#requestInitialized(ServletRequestEvent)
     */
    public void requestInitialized(ServletRequestEvent arg0) {
        // TODO Auto-generated method stub
        System.out.println("request初始化创建了!");
    }

}

```

2. 在servlet中进行测试

```

request初始化创建了!
进入到LoginServlet中.....
session被创建!
selvet----> session被创建出来, 监控器监控session1分钟后销毁.....
request被销毁了!

```

- 问题: 为什么request, 还没等到session销毁, request就销毁了?
 - request只要到达服务端后, 使命就完成了!
- session的默认的生命时间是多少呢?
 - 一次会话期

四、上课内容-3

监听器

监听attributeList,监听器获取不同作用域的数据

案例1.测验context获取

1. 创建Listener,实现接口ServletContextAttribute,ServletRequestAttribute

```

1 public class AttributeListener implements ServletContextAttributeListener, HttpSessionAttributeLi:
2
3     /**
4      * Default constructor.
5      */
6     public AttributeListener() {
7         // TODO Auto-generated constructor stub
8     }
9
10    /**
11     * @see ServletContextAttributeListener#attributeAdded(ServletContextAttributeEvent)
12     */
13    public void attributeAdded(ServletContextAttributeEvent arg0) {
14        // TODO Auto-generated method stub
15        System.out.println("向ServletContext添加了: "+arg0.getName()+"|值: "+arg0.getValue());
16    }
17
18    /**
19     * @see ServletContextAttributeListener#attributeRemoved(ServletContextAttributeEvent)
20     */
21    public void attributeRemoved(ServletContextAttributeEvent arg0) {
22        // TODO Auto-generated method stub
23        System.out.println("向ServletContext移除了: "+arg0.getName()+"|值: "+arg0.getValue());
24    }
25
26    public void attributeReplaced(ServletContextAttributeEvent arg0) {
27        // TODO Auto-generated method stub
28        System.out.println("向ServletContext修改了: "+arg0.getName()+"|916649855正在观看(本人回复));
29    }
30

```


2. 新建一个ContentServlet

```
1 System.out.println("进入了ContextServlet");
2 ServletContext servletContext=getServletContext();//得到
  servletContext对象
3 servletContext.setAttribute("aa","小明");
```

```
// TODO Auto-generated method stub
//response.getWriter().append("Served at: ").append(re
System.out.println("----->进入了ContextServlet");
ServletContext sc=this.getServletContext();//得到servle
sc.setAttribute("aaa", "小明");
sc.setAttribute("aaa", "小红");
sc.removeAttribute("aaa");
```

3. 获取结果

```
26 * @see HttpServlet#doGet(HttpServletRequest request, HttpServlet
27 */
28 protected void doGet(HttpServletRequest request, HttpServletResponse
29 // TODO Auto-generated method stub
30 //response.getWriter().append("Served at: ").append(request.g
31 System.out.println("----->进入了ContextServlet");
32 ServletContext sc=this.getServletContext();//得到servletContext
33 sc.setAttribute("aaa", "小明");
34 sc.setAttribute("aaa", "小红");
35 sc.removeAttribute("aaa");
36
37
```

Tomcat v7.0 Server at localhost [Apache Tomcat] D:\JRE8\bin\javaw.exe (2020年3月18日 下午4:31:14)

request初始化创建了!
----->进入了ContextServlet
向ServletContext添加了: aaa | 值: 小明
向ServletContext修改了: aaa | 值: 小明
向ServletContext移除了: aaa | 值: 小红
request被销毁了!

练习:测试获取request属性

思路:

1. 创建项目
2. 创建一个Listener监听器
3. 创建一个servlet编写测试代码
4. 启动项目进行测验
5. 观看测验结果

具体实现

1. 创建项目,新建Listener监听器

```
1 /*
2  * 版权所有(C), 2020, 所有权利保留。
3  * 项目名: Jsp_listener_01
4  * 文件名: MyListener.java
5  * 模块说明:
6  * 修改历史:
7  * 2020-3-18 16:37:47 - weiBin - 创建。
8  */
```

```
9
10 package com.wb.listener; /**
11  * Create By WeiBin on 2020/3/18 16:37
12  */
13
14 import javax.servlet.ServletContextAttributeEvent;
15 import javax.servlet.ServletContextAttributeListener;
16 import javax.servlet.ServletRequestAttributeEvent;
17 import javax.servlet.ServletRequestAttributeListener;
18 import javax.servlet.annotation.WebListener;
19 import javax.servlet.http.HttpSessionAttributeListener;
20 import javax.servlet.http.HttpSessionBindingEvent;
21
22 @WebListener()
23 public class MyListener implements ServletContextAttributeListener,
24     HttpSessionAttributeListener,
25     ServletRequestAttributeListener {
26
27     @Override
28     public void attributeAdded(ServletContextAttributeEvent
29     servletContextAttributeEvent) {
30         System.out.println("context添加了attribute名字
31         是"+servletContextAttributeEvent.getName()+"\t|值
32         是"+servletContextAttributeEvent.getValue());
33     }
34
35     @Override
36     public void attributeRemoved(ServletContextAttributeEvent
37     servletContextAttributeEvent) {
38         System.out.println("context移除了attribute名字
39         是"+servletContextAttributeEvent.getName()+"\t|值
40         是"+servletContextAttributeEvent.getValue());
41     }
42
43     @Override
44     public void attributeReplaced(ServletContextAttributeEvent
45     servletContextAttributeEvent) {
46         System.out.println("context替换了attribute名字
47         是"+servletContextAttributeEvent.getName()+"\t|值
48         是"+servletContextAttributeEvent.getValue());
49     }
50
51     @Override
52     public void attributeAdded(ServletRequestAttributeEvent
53     servletRequestAttributeEvent) {
54         System.out.println("request添加了attribute名字
55         是"+servletRequestAttributeEvent.getName()+"\t|值
56         是"+servletRequestAttributeEvent.getValue());
57     }
58
59     @Override
60     public void attributeRemoved(ServletRequestAttributeEvent
61     servletRequestAttributeEvent) {
62         System.out.println("request移除了attribute名字
63         是"+servletRequestAttributeEvent.getName()+"\t|值
64         是"+servletRequestAttributeEvent.getValue());
65     }
66
67     @Override
68     public void attributeBinding(ServletRequestAttributeEvent
69     servletRequestAttributeEvent) {
70         System.out.println("request绑定了attribute名字
71         是"+servletRequestAttributeEvent.getName()+"\t|值
72         是"+servletRequestAttributeEvent.getValue());
73     }
74
75     @Override
76     public void attributeUnbinding(ServletRequestAttributeEvent
77     servletRequestAttributeEvent) {
78         System.out.println("request解绑了attribute名字
79         是"+servletRequestAttributeEvent.getName()+"\t|值
80         是"+servletRequestAttributeEvent.getValue());
81     }
82 }
```



```

51
52     @Override
53     public void attributeReplaced(ServletRequestAttributeEvent
ServletRequestAttributeEvent) {
54         System.out.println("request替换了attribute名字
是"+ServletRequestAttributeEvent.getName()+"\t|值
是"+ServletRequestAttributeEvent.getValue());
55     }
56
57     @Override
58     public void attributeAdded(HttpSessionBindingEvent
HttpSessionBindingEvent) {
59         System.out.println("session添加了attribute名字
是"+HttpSessionBindingEvent.getName()+"\t|值
是"+HttpSessionBindingEvent.getValue());
60     }
61
62     @Override
63     public void attributeRemoved(HttpSessionBindingEvent
HttpSessionBindingEvent) {
64         System.out.println("session移除了attribute名字
是"+HttpSessionBindingEvent.getName()+"\t|值
是"+HttpSessionBindingEvent.getValue());
65     }
66
67     @Override
68     public void attributeReplaced(HttpSessionBindingEvent
HttpSessionBindingEvent) {
69         System.out.println("session替换了attribute名字
是"+HttpSessionBindingEvent.getName()+"\t|值
是"+HttpSessionBindingEvent.getValue());
70     }
71 }
72

```

2. 创建servlet

```

1  /*
2   * 版权所有(C)，2020，所有权利保留。
3   * 项目名: Jsp_listener_01
4   * 文件名: RequestServlet.java
5   * 模块说明:
6   * 修改历史:
7   * 2020-3-18 16:35:35 - weiBin - 创建。
8   */
9
10 package com.wb.servlet;
11
12 import javax.servlet.http.HttpSession;
13 import java.io.IOException;
14
15 /**
16  * Create By weiBin on 2020/3/18 16:35
17  */

```

```

18 @javax.servlet.annotation.WebServlet(name =
    "RequestServlet",urlPatterns = "/request")
19 public class RequestServlet extends javax.servlet.http.HttpServlet
    {
20     protected void doPost(javax.servlet.http.HttpServletRequest
        request, javax.servlet.http.HttpServletResponse response) throws
        javax.servlet.ServletException, IOException {
21
22     }
23
24     protected void doGet(javax.servlet.http.HttpServletRequest
        request, javax.servlet.http.HttpServletResponse response) throws
        javax.servlet.ServletException, IOException {
25         System.out.println("进入了request");
26         request.setAttribute("bbb", "大宝");
27         request.setAttribute("bbb", "小宝");
28         request.removeAttribute("bbb");
29         System.out.println("-----");
30         HttpSession session=request.getSession();
31         session.setAttribute("cccc", "张三");
32         session.setAttribute("cccc", "李四");
33         session.removeAttribute("cccc");
34         System.out.println("-----");
35     }
36 }

```

3. 测试结果

```

protected void doGet(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServlet
    System.out.println("进入了request");
    request.setAttribute(s: "bbb", o: "大宝");
    request.setAttribute(s: "bbb", o: "小宝");
    request.removeAttribute(s: "bbb");
    System.out.println("-----");
    HttpSession session=request.getSession();
    session.setAttribute(s: "cccc", o: "张三");
    session.setAttribute(s: "cccc", o: "李四");
    session.removeAttribute(s: "cccc");
    System.out.println("-----");
}

```

istServlet > doGet()

7.0.96 x

mcatal Localhost Log x Tomcat Catalina Log x

Output

请求地址: Deployment or web application directory D:\apache-tomcat-7.0.96\apache

request替换了attribute名字是org.apache.catalina.ASYNC_SUPPORTED |值是true

进入了request

request添加了attribute名字是bbb |值是大宝

request替换了attribute名字是bbb |值是大宝

request移除了attribute名字是bbb |值是小宝

session添加了attribute名字是cccc |值是张三

session替换了attribute名字是cccc |值是张三

session移除了attribute名字是cccc |值是李四

监听作用域的目的(作用)

- 跟踪作用域(这种情况), 用监听器.存值, 存对象

案例:测试获取监听session对象属性

1. 被监听的类需要实现接口

2. 效果

作业

把今天所讲的监听器+过滤器用在新闻发布系统案例

例如:所有的编码集问题可以用过滤器(request,response)

监听器: 监听request,session[登录进去后,3分钟后把当前用户注销]

预习

servlet3.0 注解