第17天 Listener与Filter

【教学内容】

Listener

- Listener监听器介绍
- Web监听器介绍
- 演示监听对象的创建与销毁
- 演示监听属性的变化
- 案例-定时销毁session分析
- 案例-定时销毁session实现
- 演示监听session绑定javaBean对象

Filter介绍

- Filter介绍
- Filter快速入门
- Filter链与生命周期
- FilterConfig对象介绍
- Filter配置详解

Filter案例

- 案例-自动登录功能分析
- 案例-自动登录功能实现
- Md5加密
- Java中对方法进行增强介绍
- 案例-全站编码过滤器案例分析
- 案例-全站编码过滤器案例实现

【教学总结】

【第一阶段】

【学习目标】

了解web开发中监听器的作用

【内容:Listener】

监听器介绍

1. 什么是监听器

Javaweb中的监听器是用于监听web常见对象

HttpServletRequest,HttpSession,ServletContext

2.监听器的作用

1.监听web对象创建与销毁.

2.监听web对象的属性变化

3.监听session绑定javaBean操作.

3.监听机制相关概念

1.事件----一件事情

2.事件源---产生这件事情的源头

3.注册监听---将监听器与事件绑定，当事件产生时，监听器可以知道，并进行处理。

4.监听器---对某件事情进行处理监听的一个对象

web监听器介绍

1.javaweb监听器介绍

1.监听web对象创建与销毁的监听器

ServletContextListener

HttpSessionListener

ServletRequestListener

2.监听web对象属性变化

ServletContextAttributeListener

HttpSessionAttributeListener

ServletRequestAttributeListener

3.监听session绑定javaBean

HttpSessionBindingListener

HttpSessionActivationListener

2.javaweb监听器创建步骤

Ø 创建一个类，实现指定的监听器接口

Ø 重写接口中的方法.

Ø 在web.xml文件中配置监听

演示监听对象创建与销毁

1.ServletContext对象的创建与销毁监听

Ø ServletContext对象的创建与销毁分析

ServletContext对象是服务器开启时创建。服务器关闭时销毁。

Ø 监听ServletContext对象的创建与销毁

2.HttpSession对象的创建与销毁监听

Ø 1.HttpSession对象的创建与销毁分析

session对象创建:取决于请求中是否有jsessinid，如果有，可能会获取一个已经存在的session对象。如果没有，会创建一个新的session对象.

销毁session：

1.默认超时　30分钟

2.关闭服务器

3.invalidate()方法

4.setMaxInactiveInterval(int interval) 可以设置超时时间

Ø 2.监听HttpSession对象的创建与销毁

3.HttpServletRequest对象的创建与销毁监听

Ø 1.HttpServletRequest对象的创建与销毁分析

request对象是发送请求时创建，当响应产生时，销毁.

Ø 2.监听HttpServletRequest对象的创建与销毁

演示监听属性变化

1.javaweb中监听属性变化API介绍

ServletContextAttributeListener

void	**attributeAdded**(ServletContextAttributeEvent scab) Notification that a new attribute was added to the servlet context.
void	**attributeRemoved**(ServletContextAttributeEvent scab) Notification that an existing attribute has been removed from the servlet context.
void	**attributeReplaced**(ServletContextAttributeEvent scab) Notification that an attribute on the servlet context has been replaced.

HttpSessionAttributeListener

void	**attributeAdded**(HttpSessionBindingEvent se) Notification that an attribute has been added to a session.
void	**attributeRemoved**(HttpSessionBindingEvent se) Notification that an attribute has been removed from a session.
void	**attributeReplaced**(HttpSessionBindingEvent se) Notification that an attribute has been replaced in a session.

ServletRequestAttributeListener

void	**attributeAdded**(ServletRequestAttributeEvent srae) Notification that a new attribute was added to the servlet request.
void	**attributeRemoved**(ServletRequestAttributeEvent srae) Notification that an existing attribute has been removed from the servlet request
void	**attributeReplaced**(ServletRequestAttributeEvent srae) Notification that an attribute was replaced on the servlet request.

3. 演示监听session中属性变化

演示监听session绑定javaBean

1. HttpSessionBindingListener监听器功能介绍

使javaBean对象在被绑定到会话或从会话中取消对它的绑定时得到通知

2. HttpSessionActivationListener监听器功能介绍

绑定到会话的对象可以侦听通知它们会话将被钝化和会话将被激活的容器事件

3. 注意事项

这两个监听器比较特殊,它是由javaBean来实现的，并且不需要在web.xml文件中注册监听.

javaBean必须是序列化的.

4. 演示HttpSessionBindingListener功能

javaBean自动感知绑定到session中以及从session中移除.

```
public void valueBound(HttpSessionBindingEvent event) {
System.out.println("绑定了user对象到了session");
}

public void valueUnbound(HttpSessionBindingEvent event) {
System.out.println("从session中移除了user对象");
}
```

5. 演示HttpSessionActivationListener功能

1. bean1.jsp中向session存储一个user对象
2. bean2.jsp中从session中获取user对象的name属性
3. User类实现接口HttpSessionActivationListener,可以监听到活化与钝化操作
4. 做一个配置文件，来设定当活化与钝化操作时文件的位置.

在meta-inf目录下创建一个context.xml文件

```
<Context>
<Manager className="org.apache.catalina.session.PersistentManager" maxIdleSwap="1">
<Store className="org.apache.catalina.session.FileStore" directory="it315"/>
</Manager>
```

</Context>

监听器案例--定时销毁过期session分析

1. 功能描述

如果session对象超过五秒没有使用就销毁。

2. 分析

怎样可以将每一个创建的session全都保存起来？

我们可以做一个HttpSessionListener，当session对象创建时，就将这个session对象装入到一个集合中.将集合保存到ServletContext域中。

怎样可以判断session过期了？

在HttpSession中有一个方法public long getLastAccessedTime()

它可以得到session对象最后使用的时间。

可以使用invalidate方法销毁。

我们上面的操作需要使用任务调度功能.

在java中有一个Timer定时器类

```
public class TimerDemo {

    public static void main(String[] args) {
        Timer t = new Timer(); // 创建了一个定时器

        t.schedule(new TimerTask() {

            @Override
            public void run() {

                System.out.println(new Date().toLocaleString());
            }
        }, 1000, 1000);
    }
}
```

关于三个域对象获取

如果在Servlet中要获取　request在方法上就有，request.getSession()

getServletContext();

如果我们有request对象了，request.getSession()　request.getSession().getServletCotnext();

程序在使用时，需要考虑并发问题,因为我们在web中，它一定是一个多线程的，那么我们的程序对集合进行了添加，还有移除操作。

4. 步骤分析

1. 创建一个ServletContext创建与销毁监听器，在ServletContext对象创建时，将集合创建出来并保存到ServletContext中

2. 创建一个HttpSession创建与销毁监听器，当创建一个HttpSession对象时，将session对象获取到保存到从ServletContext获取的集合中

3. 在ServletContext的监听器内部创建定时器来完成定时销毁session操作.

监听器案例--定时销毁过期session实现

【笔试面试题】

无

【重点总结】

关于ServletContext域对象的监听器大家重点掌握，其它做为了解

【第二阶段】

【学习目标】

掌握Filter在开发中的作用

掌握Filter的创建及基本流程

【内容:Filter介绍】

Filter介绍

1. 什么是Filter及其作用介绍

Filter是sun公司中servlet2.3后增加的一个新功能.

Servlet规范中三个技术　Servlet Listener Filter

在javaEE中定义了一个接口 javax.servlet.Filter来描述过滤器

通过Filter可以拦截访问web资源的请求与响应操作.

WEB开发人员通过Filter技术，对web服务器管理的所有web资源：例如Jsp, Servlet, 静态图片文件或静态 html 文件等进行拦截，从而实现一些特殊的功能。例如实现URL级别的权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能。

2. Filter API介绍

Filter是javax.servlet包下的一个接口主要有以下三个方法

void	**destroy**() Called by the web container to indicate to a filter that it is being taken out of service.
void	**doFilter**(ServletRequest request, ServletResponse response, FilterChain chain) The doFilter method of the Filter is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
void	**init**(FilterConfig filterConfig) Called by the web container to indicate to a filter that it is being placed into service.

Filter的快速入门

1.Filter创建步骤介绍

Filter创建步骤:

1.创建一个类实现javax.servlet.Filter接口

2.得写接口方法

3.在web.xml文件中配置Filter

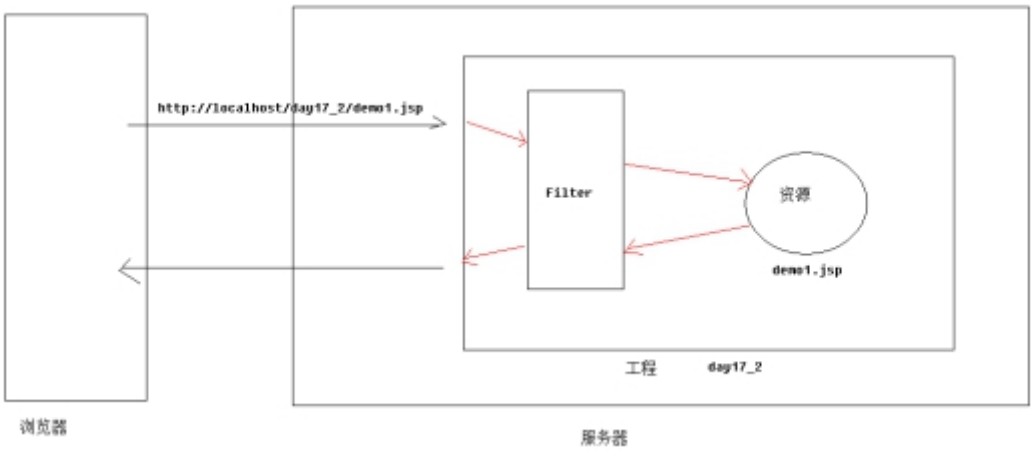
Filter在web.xml文件中配置的目的:

1.配置拦截什么样的资源。

2.Filter初始化

```
<filter>
<filter-name>demo1Filter</filter-name>
<filter-class>cn.itcast.web.filter.Demo1Filter</filter-class>
</filter>
<filter-mapping>
<filter-name>demo1Filter</filter-name>
<url-pattern>/demo1</url-pattern>
</filter-mapping>
```

2.演示Filter拦截操作



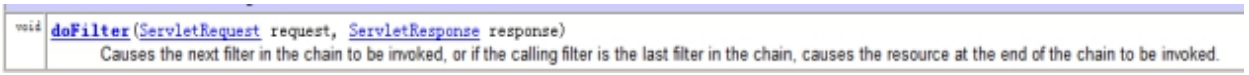
注意:在Filter的doFilter方法内如果没有执行

```
chain.doFilter(request, response);
```

那么资源是不会被访问到的。

3.FilterChain功能介绍

FilterChain 是 servlet 容器为开发人员提供的对象，它提供了对某一资源的已过滤请求调用链的视图。过滤器使用 FilterChain 调用链中的下一个过滤器，如果调用的过滤器是链中的最后一个过滤器，则调用链末尾的资源。



Filter链与Filter生命周期

1.Filter链介绍

多个Filter对同一个资源进行了拦截，那么当我们在开始的Filter中执行chain.doFilter(request,response)时，是访问下一下Filter,直到最后一个Filter执行时，它后面没有了Filter,才会访问web资源。

Ø 关于多个FIlter的访问顺序问题

如果有多个Filter形成了Filter链，那么它们的执行顺序是怎样确定的？

它们的执行顺序取决于<filter-mapping>在web.xml文件中配置的先后顺序。

2.Filter生命周期

当服务器启动，会创建Filter对象，并调用init方法，只调用一次.

当访问资源时，路径与Filter的拦截路径匹配，会执行Filter中的doFilter方法，这个方法是真正拦截操作的方法.

当服务器关闭时，会调用Filter的destroy方法来进行销毁操作.

FilterConfig介绍

1.FilterConfig功能介绍

在Filter中的init方法上有一个参数叫FilterConfig

FilterConfig是Filter的配置对象

它的作用:

1. 获取Filter的名称
2. 获取初始化参数
3. 获取ServletContext对象

2.FilterConfig常用API

String	**getFilterName**() Returns the filter-name of this filter as defined in the deployment descriptor.
String	**getInitParameter**(String name) Returns a String containing the value of the named initialization parameter, or nu
Enumeration	**getInitParameterNames**() Returns the names of the filter's initialization parameters as an Enumeration of St parameters.
ServletContext	**getServletContext**() Returns a reference to the ServletContext in which the caller is executing.

3.FilterConfig功能演示

Filter配置详解

1.Filter基本配置介绍

<filter>

<filter-name>filter名称</filter-name>

<filter-class>filter类全名</filter-class>

</filter>

<filter-mapping>

<filter-name>filter名称</filter-name>

<url-pattern>映射路径</url-pattern>

</filter-mapping>

2. 关于url-pattern配置

1. 完全匹配

要求必须以"/"开始.

2. 目录匹配

要求必须以"/"开始，以*结束.

3. 扩展名匹配

不能以"/"开始，以*.xxx结束.

3. 关于servlet-name配置

针对于servlet拦截的配置　<servlet-name>配置

在Filter中它的配置项上有一个标签

<servlet-name>它用于设置当前Filter拦截哪一个servlet。

是通过servlet的name来确定的。

5. 关于dispatcher配置

可以取的值有　REQUEST　FORWARD　ERROR　INCLUDE

它的作用是：当以什么方式去访问web资源时，进行拦截操作.

1. REQUEST 当是从浏览器直接访问资源，或是重定向到某个资源时进行拦截方式配置的 它也是默认值

2. FORWARD 它描述的是请求转发的拦截方式配置

3. ERROR 如果目标资源是通过声明式异常处理机制调用时，那么该过滤器将被调用。除此之外，过滤器不会被调用。

4. INCLUDE 如果目标资源是通过RequestDispatcher的include()方法访问时，那么该过滤器将被调用。除此之外，该过滤器不会被调用

【笔试面试题】

描述java中Filter的作用

【重点总结】

关于Filter大家需要掌握Filter的作用及创建方式

重点了解Filter的配置方式

【第三阶段】

【学习目标】

【内容:Filter案例】

Filter案例--自动登录分析

1. 功能描述

2. 画图分析

3. 步骤分析

Filter案例--自动登录实现

md5加密介绍

1. md5加密介绍

Message Digest Algorithm MD5（中文名为消息摘要算法第五版）为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护。

主流编程语言普遍已有MD5实现

2. mysql中的md5加密实现

md5(字段)

例如 update user set password=md5(password);

3. java中的md5加密实现

```
/**
  * 使用md5的算法进行加密
 */
public static String md5(String plainText) {
byte[] secretBytes = null;
try {
secretBytes = MessageDigest.getInstance("md5").digest(
plainText.getBytes());
} catch (NoSuchAlgorithmException e) {
throw new RuntimeException("没有md5这个算法！");
}
String md5code = new BigInteger(1, secretBytes).toString(16);
for (int i = 0; i < 32 - md5code.length(); i++) {
md5code = "0" + md5code;
}
return md5code;
```

}

java中功能增强介绍

1. 继承方式进行增强
2. 装饰模式进行增强

Filter案例--解决全站乱码分析

1. 功能描述
2. 画图分析
3. 步骤分析

Filter案例--解决全站乱码实现

【笔试面试题】

无

【重点总结】

掌握两个案例

【总结】

对于Listener,大家只需要了解，知道Listener的作用及创建方式。

对于Filter大家需要重点掌握，知道Filter的作用是用于拦截浏览器请求与响应web资源，

并知道Filter的创建方式与配置。

对于Filter的两个案例要求大家掌握

【课后作业】

问答题

简述Listener的作用及创建方式

简述javaweb开发有哪些监听器作用是什么

简述Filter的作用及创建方式

简述Filter的生命周期

简述FilterConfig对象的作用

简述Filter的配置

操作题

定时销毁session

自动登陆案例

全局编码过滤器