

3.16作业

作业1：有多个参数的情况下,那么init()方法如何获取？

思路

代码

观察发现config有getInitParameterNames()方法,返回的是一个枚举集合,所以得出也可以遍历枚举来取得初始化参数的值

作业2：如果请求/* 换成单独的/ 或*,那么/ 或 * 有什么区别？ 请用实验说明描述？

思路

代码

小结

3.16作业

作业1：有多个参数的情况下,那么init()方法如何获取？

思路

1. 创建web项目
2. 创建filter过滤器
3. 定义多个属性并且xml配置多个属性
4. 在init方法中获取属性
5. 打印在控制台即可

代码

1. 创建web项目,生成filter,在Filter中自定义2个以上属性



2. 配置xml文件

```

<filter>
  <filter-name>MyFilter</filter-name>
  <filter-class>com.wb.filter.MyFilter</filter-class>
  <!-- 配置初始化参数
      多个参数,配置多个<init-param>
  -->
  <init-param>
    <param-name>charset</param-name>
    <param-value>utf-8</param-value>
  </init-param>
  <init-param>
    <param-name>date</param-name>
    <param-value>3月16日</param-value>
  </init-param>
  <init-param>
    <param-name>age</param-name>
    <param-value>18</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/*</url-pattern>

```

3. 在init方法中编写代码测试能否获取到初始化的参数

```

public void init(javax.servlet.FilterConfig config) throws javax.servlet.ServletException {
    //config.getInitParameter(" 对应初始化参数的键");此方法的结果返回的是键所对应的值
    String charset = config.getInitParameter( S: "charset");
    String date = config.getInitParameter( S: "date");
    String age = config.getInitParameter( S: "age");
    System.out.println("charset="+charset+"\t date="+date+"\t age"+age);
    // 给成员属性赋值
    this.charset=charset;
    this.date=date;
    this.age=Integer.parseInt(age);
}

```

4. 启动web项目,观察控制台结果

```

    system.out.println("编码是"+charset+"\t时间为"+date+"\t年龄为"+age);
    chain.doFilter(req, resp);
}

public void init(javax.servlet.FilterConfig config) throws javax.servlet.ServletException {
    //config.getInitParameter(" 对应初始化参数的键");此方法的结果返回的是键所对应的值
    String charset = config.getInitParameter( S: "charset");
    String date = config.getInitParameter( S: "date");
    String age = config.getInitParameter( S: "age");
    System.out.println("charset="+charset+"\t date="+date+"\t age"+age);
    // 给成员属性赋值
    this.charset=charset;
    this.date=date;
    this.age=Integer.parseInt(age);
}

```

ter > init()
 7.0.96 x
 mcat Localhost Log x Tomcat Catalina Log x
 chartswar exploded
 terwar exploded
 Output
 [2020-03-16 08:46:44,892] Artifact Ajax_echarts:war exploded: Artifact is deployed successfully
 [2020-03-16 08:46:44,896] Artifact Ajax_echarts:war exploded: Deploy took 1,630 milliseconds
 消息源 16, 2020 8:46:44 消息源 org.apache.catalina.deploy.WebXml setVersion
 瑞一操: Unknown version string [4.0]. Default version will be used.
 charset=utf-8 date=3月16日 age18
 [2020-03-16 08:46:44,987] Artifact Jsp_filter:war exploded: Artifact is deployed successfully
 [2020-03-16 08:46:44,987] Artifact Jsp_filter:war exploded: Deploy took 1,720 milliseconds
 编码是utf-8 时间为3月16日 年龄为18

5. 小结:上述案例可以看出,多个参数可以用FilterConfig多次调用getInitParameter()方法来取值

观察发现config有getInitParameterNames()方法,返回的是一个枚举集合,所以得出也可以遍历枚举来取得初始化参数的值

此方法并没有跑起来, 不知道什么原因

```
public void init(javax.servlet.FilterConfig config) throws javax.servlet.ServletException {
    // 获取枚举集合
    Enumeration<String> initParameterNames = config.getInitParameterNames();
    // 遍历枚举集合
    while (initParameterNames.hasMoreElements()){
        System.out.println("此对象的值为"+initParameterNames.nextElement());
        if (initParameterNames.nextElement().equals("date")){
            System.out.println("正在给date赋值");
            this.date=config.getInitParameter(initParameterNames.nextElement());
        }
        if (initParameterNames.nextElement().equals("age")){
            System.out.println("正在给age赋值");
            this.age=Integer.valueOf(config.getInitParameter(initParameterNames.nextElement()));
        }
        if (initParameterNames.nextElement().equals("charset")){
            System.out.println("正在给charset赋值");
            this.charset=config.getInitParameter(initParameterNames.nextElement());
        }
    }
}
```

MyFilter -> doFilter()

Server: Tomcat 7.0.96

Deployment: Jsp_filterwar exploded

Output:

```
漏洞: Context [] startup failed due to previous errors
[2020-03-16 08:55:55.377] Artifact Jsp_filter:war exploded: Error during artifact deployment. See serve
洞父深 16, 2020 8:56:04 洞姬峰 org.apache.catalina.startup.HostConfig deployDirectory
淇℃佬: Deploying web application directory D:\apache-tomcat-7.0.96\apache-tomcat-7.0.96\webapps\manager
洞父深 16, 2020 8:56:05 洞姬峰 org.apache.catalina.startup.HostConfig deployDirectory
```

作业2: 如果请求/* 换成单独的/ 或*,那么/ 或 * 有什么区别? 请用实验说明描述?

思路

1. 创建web项目
2. 编写filter
3. 修改filter的url-pattern
4. 分别以a标记,表单,get,post 等不同请求去测试
5. 然后观察区别 (我们都知道/*是过滤所有请求)

代码

1. 创建web项目,生成filter,在Filter中写好输出语句来作为是否进入了过滤器的判断

```
16  * Create By WeiBin on 2020/3/16 20:59
17  */
18  public class MyFilter implements Filter {
19      public void destroy() {
20          System.out.println("过滤器被销毁了");
21      }
22
23      public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws javax.servlet.ServletException {
24          System.out.println("进入了doFilter方法");
25          chain.doFilter(req, resp);
26      }
27
28      public void init(FilterConfig config) throws ServletException {
29          System.out.println("过滤器初始化了");
30      }
31
32  }
33
```

Project Structure:

- web
 - WEB-INF
 - web.xml
 - index.jsp
 - MyFilter_01.html

2. 编写一个servlet用来接收请求(没有其他要求)

3. 编写xml配置文件

1. /的配置

```

        version="4.0">
        <filter>
            <filter-name>MyFilter</filter-name>
            <filter-class>com.wb.filter.MyFilter</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>MyFilter</filter-name>
            <url-pattern>/</url-pattern>
        </filter-mapping>
    
```

2. *的配置

```

        version="4.0"
        <filter>
            <filter-name>MyFilter</filter-name>
            <filter-class>com.wb.filter.MyFilter</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>MyFilter</filter-name>
            <url-pattern>*</url-pattern>
        </filter-mapping>
    
```

4. 分别查看/和*的结果

	直接网址访问	a标记	get请求	post请求
/	没有进入dofilter	没有进入dofilter	没有进入dofilter	没有进入dofilter
*	进入了dofilter	进入了dofilter	进入了dofilter	进入了dofilter

小结

1. 看出/并没有过滤任何请求,而*则过滤了所有请求
2. 测试时候,如果直接写地址+端口号+"/",后台表示进入了dofilter, 代表"/"匹配的是单个/,并且自己测验时发现并不能匹配/test这种servlet请求
3. *可以匹配你所有,类似模糊匹配,可以写成*.jsp,*.html,*.js等形式,即过滤这种后缀的文件和请求
4. /是精确匹配,只匹配"/"