关于servlet

笔记本：	关于servlet		
创建时间：	2018/1/26/周五 13:38	**更新时间：**	2018/1/26/周五 13:41
作者：	1634896520@qq.com		

关于servlet

一,什么是Servlet

主要是要先讲明白Servlet是干嘛的.
Servlet是JavaWeb的 三大组件 之一,它属于动态资源,Servlet的作用是处理请求,服务器会把接收到的请求交给Servlet来处理,在Servlet中通常需要:
- 接收请求数据;
- 处理请求;
- 完成响应;

例如客户端发出登录请求,或者输出注册请求,这些请求都应该由Servlet来完成处理!Servlet需要我们自己来编写,每个Servlet必须实现javax.servlet.Servlet接口.

二,实现Servlet的方式

实现Servlet的三种方式:(由我们来写)
- 实现javax.servlet.Servlet接口;
- 继承javax.servlet.GenericServlet类;
- 继承javax.servlet.http.HttpServlet类,会专门对http的请求提供一些支持;

一般情况都是去继承HttpServlet类来完成Servlet(方便),但是学习Servlet的起始阶段,还要从javax.servlet.Servlet接口开始学习.

服务器:相当于10086,Servlet就相当于10086中的话务员就等着人给打电话,服务器就是等着客户端来访问,根据访问的请求,将对应数据传递给客户端.
每个Servlet都是不同的,它们能处理的请求都是不同的.
Tomcat会将不同的请求,发送给对应的Servlet.
网络请求都是异步的.
注: Servlet中的生命周期等回调方法,都是服务器自动调用的.
注: Servlet的对象也是由服务器来创建.(比如Tomcat);

三,如何让浏览器访问Servlet

1,给Servlet指定一个Servlet路径,让Servlet与这个路径绑定在一起.
2,通过浏览器访问Servlet路径
3,需要在web.xml文件中对Servlet进行配置

```
<servlet>
  <servlet-name>名字</servlet-name>
  <servlet-class>包名.Servlet类名</servlet-class>
</servlet>

<servlet-mapping>
  <!-- 此处名字与上一处相同 -->
  <servlet-name>名字</servlet-name>
  <!-- .servlet扩展名可以不写,也可以直接写:XXXServlet -->
  <!-- 此处应该写的就是要在浏览器中输入的路径 -->
  <url-pattern>/名字.servlet</url-pattern>
</servlet-mapping>
```

四,生命周期方法

•void init(ServletConfig sc);//第一次被访问时会创建对象,创建对象后马上执行该方法
•void service(ServletRequest srq,SerletResponse srp);每次处理请求时都会执行该方法
•void destroy();即将销毁之前会执行该方法,一般为服务器被关闭之前会执行.

五,特性

•每个Servlet类都是单例模式,但可以存在多个Servlet类.
•线程不安全,效率高.
•Servlet类由我们来写,对象由服务器创建,并且由服务器调用对应的方法.

六,ServletConfig接口

web.xml中的配置信息,会被加载进内存,在内存中以ServletConfig的实现类对象的形式存在.
所以如果想要获得Servlet的配置信息,可以直接通过ServletConfig的实现类对象来获得.
一个ServletConfig对象,对应一段web.xml中Servlet的配置信息.
方法介绍:
- String getServletName();获取的是中的内容;
- ServletContext getServletContext();获取Servlet上下文对象;
- String getInitParameter(String name);通过名称(key)获取指定初始化参数的值(value),初始化参数在web.xml中配置;
- Enumeration getInitParameterNames();获得所有初始化参数的名称,就是获取所有的key(在web.xml中配置);

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <servlet>
        <servlet-name>Servlet1</servlet-name>
        <servlet-class>MyServlet</servlet-class>
        <!--初始化参数-->
        <init-param>
            <param-name>k1</param-name>
            <param-value>v1</param-value>
        </init-param>

        <init-param>
            <param-name>k2</param-name>
            <param-value>v2</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>Servlet1</servlet-name>
        <url-pattern>/HelloServlet</url-pattern>
    </servlet-mapping>


</web-app>
```

七,GenericServlet抽象类

该抽象类就是实现类部分Servlet接口的抽象方法,暴露出一个抽象方法Service();
很少会用.

八,HttpServlet

继承javax.servlet.http.HttpServlet类,会专门对http的请求提供一些支持;
能方便的获取请求方式(get或post等);

流程
// 使用方式:
// 1,在service生命周期中,将参数中的ServletRequest强转成HttpServletRequest,
// 将ServletResponse强转成HttpServletResponse.
// 2,在service生命周期中,调用非生命周期service方法,将强转后的参数传入
// 3,非生命周期的service方法,会根据传入的参数判断当前的请求方式(get或post)
// 4,如果为get,则会调用doGet方法,如果为post,则会调用doPost方法
// 5,所以我们需要复写doGet或doPost方法来做我们想做的事

// 6,如果不覆盖doGet或doPost方法,访问时会405,服务器默认表示不支持该请求方式.
// 7,doGet和doPost不是抽象方法,有实体,实体就是给客户端返回一个405

九,关于线程安全

Servlet不是线程安全的,优势是效率高速度快.但是也存在线程安全问题,很可能出现多个线程同时对同一个对象进行读写操作.
注: 不要在Servlet中创建成员变量,创建局部变量即可.
注: 可以创建不存储数据的成员变量,该变量没有涉及到set/get等方法.
注: 可以创建存储数据的成员变量,但是该变量存储的数据必须是只读的.

十,在web.xml中对Servlet配置,使得Servlet在服务器启动时就创建

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <servlet>
        <servlet-name>httpservlet</servlet-name>
        <servlet-class >MyHttpServlet</servlet-class>
        <!-- 数值代表创建顺序 -->
        <load-on-startup>0</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>httpservlet</servlet-name>
        <url-pattern>/wudi</url-pattern>
    </servlet-mapping>


</web-app>
```

十一,关于url-pattern&

标签是标签的子标签,用来定义浏览器访问Servlet时输入的url地址.
标签标签可以有多个,也就是可以输入多种url地址来访问同一个Servlet.
可以使用通配符*表示任意数量的任意字符

```
<url-mapping>
    <servlet-name>httpservlet</servlet-name>
    <url-pattern>/wudi</url-pattern>
    <url-pattern>/heheda</url-pattern>
    <!-- 路径匹配,就是只要路径对了就ok -->
    <url-pattern>/heheda/*</url-pattern>
    <!-- 拓展名匹配,.do是Struts1推荐使用的,Struts2推荐使用.action -->
    <url-pattern>*.do</url-pattern>
    <!-- 匹配所有的url -->
    <url-pattern>/*</url-pattern>

    <!-- 如果上述匹配方式同时存在,则谁匹配的url越多,则谁优先级越低 -->
```

```
    <!-- 下面这种优先级最高 -->
    <url-pattern>/heheda</url-pattern>
</url-mapping>
```

注: 通配符*只能出现在两端,不能出现在url中间且只能存在一个通配符.