servlet_part2.txt

笔记本:	杰普实训三_笔记		
创建时间:	2018/8/15 14:56	**更新时间:**	2018/8/18 17:31
作者:	家龙		
URL:	file://D:\资料\Desktop\servlet资料1\servlet_part2.txt		

C/S和B/S的比较

1) c/s架构的系统在使用的时候必须安装一个客户端软件，如果服务器端升级之后，客户端也需要进行升级.所以对于这样一种类型的软件可维护性比较差。

2) b/s架构的系统在使用的时候客户端不需要安装特殊的软件，只需要一个浏览器就可以了，服务器端升级完毕之后，客户端不需要作任何改变，就可以使用升级之后的软件，所以对于这样一种类型的软件可维护性比较好，应用广泛。

1.按照JavaEE规范编写servlet。

```
1.怎么在Javaee规范下创建Servlet
    实现指定的接口 或者继承父类，实现指定的方法
    1.javax.servlet.Servlet
        5个抽象方法
        init,service,destroy,getServletInfo,getServletConfig
    2.javax.Servlet.GenericServlet  implmenets Servlet,ServletConfig
        1个抽象方法
        service
    3.javax.Servlet.http.HttpServlet extends GenericServlet
        0个抽象方法
        servcie--service--[doXxx]
```

1.1实现javax.servlet.Servlet接口，实现其中的五个抽象方法。主要关注service方法，servlet容器接收到请求后，会根据请求映射加载对应的servlet类，创建其对象，经过一些步骤后，调用service方法。

1.2继承javax.servlet.GenericServlet抽象类，GenericServlet实现了Servlet接口，实现了四个抽象方法（init，getServletConfig，getServletInfo，destory），其service方法由自定义servlet实现，servlet容器接收到请求后，会创建对应servlet对象，调用init方法进行初始化，然后调用其service方法。

注：在GenericServlet中有两个init方法：

init(ServletConfig config)与init()方法，在有参数的init方法中，servlet容器在调用的时候会给当前servlet对象的config对象赋值，config对象用来获取该servlet在wen.xml中配置的初始化参数，然后调用无参的init方法。即我们要重写init方法的话，需要重写无参的init方法。

```
public void init(ServletConfig config) throws ServletException {
        this.config = config;
        this.init();
    }
public void init() throws ServletException {
```

```
        // NOOP by default
    }
```

1.3继承javax.servlet.http.HttpServlet抽象类，HttpServlet中没有抽象方法，继承自GenericServlet，且实现了GenericServlet中的service方法。servlet容器接收到请求后，创建对应servlet对象，调用其service方法：

注：HttpServlet中有两个Service方法，service(ServletRequest req, ServletResponse res)与service(HttpServletRequest req, HttpServletResponse resp)，在servlet容器接收到请求后，创建对应servlet对象，调用init方法进行初始化后，调用参数为ServletRequest ServletResponse的service方法，这个方法中将请求与响应对象类型转换为HttpServletRequest与HttpServletResponse，然后调用第二个service方法，在第二个service方法中，通过HttpServletRequest中封装的方法获取请求方式，如Get请求方式，则，调用对应的doGet方法，Post请求方式，调用对应的doPost方法，所以我们只需要重写doGet或者doPost方法即可。

```
367         3.HttpServlet  没有抽象方法的
368             使用集成的4个方法，自己实现了servcie
369             service(ServletRequest req, ServletResponse res){}:实现的
370                 :进行类型转换，调用自定义的重载的servcie
371             service(HttpServletRequest req, HttpServletResponse resp):自定义的
372                 :进行请求方法的判断，根据请求方法，执行对应的doXxx();
373             doXxx
374
375     Servlet的生命周期：servlet容器
376     初始化：创建(构造器)，初始化(init(ServletConfig config))
377
378     运行：service(ServletRequest )-->servcie(HttpServletReuqest,)-->doXxx(
        HttpServletReuqest)
379
380
381     销毁
382
```

注1：由于客户端是通过URL地址访问web服务器中的资源，所以Servlet程序若想被外界 访问，必须把servlet程序映射到一个URL地址上，这个工作在web.xml文件中使用<servlet>元素和<servlet-mapping>元素完成。

<servlet>元素用于注册Servlet，它包含有两个主要的子元素：<servlet-name>和<servlet-class>，分别用于设置Servlet的注册名称和Servlet的完整类名。

一个<servlet-mapping>元素用于映射一个已注册的Servlet的一个对外访问路径，它包含有两个子元素：<servlet-name>和<url-pattern>，分别用于指定Servlet的注册名称和Servlet的对外访问路径。

注2：

同一个Servlet可以被映射到多个URL上，即多个<servlet-mapping>元素的<servlet-name>子元素的设置值可以是同一个Servlet的注册名。

注3：在Servlet映射到的URL中也可以使用*通配符，但是只能有两种固定的格式：一种格式是"*.扩展名"，另一种格式是以正斜杠（/）开头并以"/*"结尾，否则报错。

注4：在eclipse中写javaee项目的目录结构，与部署到Tomcat服务器下webapps中的应用程序目录结构是不同的。

2.servlet的生命周期

Servlet的生命周期是由servlet的容器来控制的，可以分为初始化、运行和销毁阶段三个阶段。

初始化阶段

1）Web服务器首先检查是否已经装载并创建了该Servlet的实例对象。如果是，则直接执行第6）步（可以配置），否则，执行第2）步。默认在第一次访问servlet的时候创建

2）servlet容器加载servlet类，把servlet类的.class文件中的数据读到内存中。

3）servlet容器创建一个ServletConfig对象，ServletConfig对象包含了servlet的初始化配置信息（web.xml文件中可以配置）。

4）Servlet容器创建一个servlet实例对象。

5）Servlet容器调用Servlet实例对象的init()方法。init方法可以重写无参的。

运行阶段：

6）当servlet容器接受到一个请求时，创建一个用于封装HTTP请求消息的HttpServletRequest对象和一个代表HTTP响应消息的HttpServletResponse对象，然后调用Servlet的service()方法并将请求和响应对象作为参数传递进去，执行service方法，处理请求。

销毁阶段：

7）WEB应用程序被停止或重新加载之前，Servlet引擎将卸载Servlet，并在卸载之前调用Servlet的destroy()方法。

注1：可在web.xml中通过<load-on-startup>进行配置，修改servlet对象创建时机。

1)load-on-startup 元素标记容器是否应该在启动的时候加载这个servlet，(实例化并调用其init()方法)。

2)它的值必须是一个整数，表示servlet应该被载入的顺序

3)如果该元素不存在或者这个数为负时，则容器会当该Servlet被请求时，再加载。

4)当值为0或者大于0时，表示容器在应用启动时就加载并初始化这个servlet；

5)正数的值越小，该servlet的优先级越高，应用启动时就越先加载。

当值相同时，容器自己选择顺序来加载

注2：servlet是单例模式,在web项目运行期间,一个servlet只会创建一个对象，

web项目本身就需要在多线程的环境中运行,tomcat服务器会提供这样的多线程环境,当多个客户端并发访问同一个Servlet时，web服务器会为每一个客户端的访问

请求创建一个线程，并在这个线程上调用Servlet的service方法，因此service方法内如果访问了同一个资源的话，就有可能引发线程安全问题。所以在servlet中声明的成员变量,就会有线程安全的问题。所以我们应该尽量少的在servlet中定义成员变量。

3.servlet接口中的方法

```
//初始化servlet对象的时候被调用
void      init(ServletConfig config)
//销毁servlet对象的时候被调用
void      destroy()
//访问servlet对象的时候被调用
void      service(ServletRequest req, ServletResponse res)
```

//返回servlet相关信息,比如作者、版本、版权等父类中(GenericServlet)默认返回一个空字符串 ""，如果需要返回信息的话,程序员可以自己重写这个方法

```
String  getServletInfo()
//返回ServletConfig对象
ServletConfig  getServletConfig()
```

4.ServletConfig中的方法

在Servlet的配置文件web.xml中，可以使用一个或多个<init-param>标签为servlet配置一些初始化参数。

```
<init-param>
    <param-name></param-name>
    <param-value></param-value>
</init-param>
```

当servlet配置了初始化参数后，web容器在创建servlet实例对象时，会自动将这些初始化参数封装到ServletConfig对象中，并在调用servlet的init方法时，将ServletConfig对象传递给servlet。进而，我们通过ServletConfig对象就可以得到当前servlet的初始化参数信息。

```
//返回servlet在web.xml文件中所配置的注册名称，也就是<servlet-name>这个标签中的值
String      getServletName()
//获得在web.xml中所配置的指定名字的参数值
String      getInitParameter(String name)
//获得给当前servlet传的所有参数的名字
Enumeration      getInitParameterNames()
//获得ServletContext类型对象，ServletContext是web项目中非常重要的一个类型对象
ServletContext      getServletContext()
```

5.servlet的访问

使用web.xml文件里面的这个<url-pattern>标签中的映射路径,来访问servlet

5.1 在浏览器的地址栏中,直接输入servlet映射的路径来访问

这时候是get方式的访问,servlet中的doGet方法最终会被调用

5.2 在页面中,可以使用超链接来访问servlet

这时候是get方式的访问,servlet中的doGet方法最终会被调用

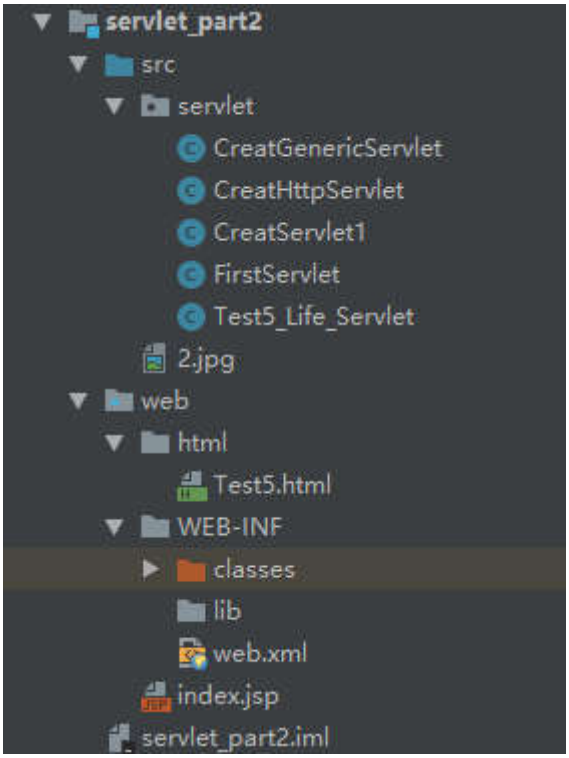
5.3 在页面中,可以使用表单发送请求去访问servlet

这时候默认情况下是get方式访问,但是可以通过表单里的属性值进行设置为get或者post方式

5.4 将来还可以使用javascript或者在ajax中发请求访问servlet,这个时候也可以进行设置使用get方式还是post方式进行访问servlet

其他的情况:

还可以使用标签语言来访问servlet




```
package servlet;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;

import java.io.IOException;
import java.io.PrintWriter;

//第一个servlet类，继承抽象父类HttoSerblet
public class FirstServlet extends HttpServlet {

    //请求URL--执行一段java代码(作用：处理这个请求)(哪段：service)
    //service-->service-->doXxx
    //方法重写：方法名一样，参数列表一样，访问权限不能被缩小，异常不能被扩大
    protected void doGet(HttpServletRequest request,HttpServletResponse response) throws
ServletException,IOException{
```

```
        //这个URL请求的处理过程就在这里写
        //获取请求中的相关信息[参数]，进行一系列的处理[接收参数]
        //注册:....数据库交互.....
        //返回响应
        PrintWriter pw = response.getWriter();
        pw.print("hello world");
        pw.flush();
        pw.close();
    }
}
```

```
package servlet;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.io.PrintWriter;

/**
* Created by TjI on 2018/8/17 8:30.
* 第一种创建servlet 的方式,实现javax.servlet.Servlet接口
* Servlet接口中有5个抽象方法,所以自定义Servlet需要全部实现
*/
public class CreatServlet1 implements Servlet {

    //初始化:servlet对象创建后进行初始化调用,需要传参:参数类型:servletConfig
    @Override
    public void init(ServletConfig servletConfig) throws ServletException {

    }
    //返回包含此Servlet相关的配置信息的ServletConfig对象
    @Override
    public ServletConfig getServletConfig() {
        return null;
    }
    //运行时接收到请求,就会执行Service
    @Override
    public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException,
IOException {
        System.out.println("接收到请求");
        //类型转换
        String method = ((HttpServletRequest) servletRequest).getMethod();
        //接收到请求后执行以下代码
        System.out.println("接收到请求:"+method);
        //返回响应,返回一个html页面,设置响应内容编码格式;
        servletRequest.setCharacterEncoding("UTF-8");
        //contentType:响应的文件类型是html,编码格式
        servletResponse.setContentType("text/html;charset=utf-8");
        //获取响应输出流
        PrintWriter pw = servletResponse.getWriter();
        pw.print("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"
\"http://www.w3.org/TR/html4/loose.dtd\">");
        pw.print("<html>");
        pw.println("<head>");
        pw.print("<title>实现Serclet接口的方式</title>");
        pw.println("</head>");
        pw.print("<body><h1 style='color:red;'>hello,方式1:实现javax.servlet.Servlet</h1></body>");
        pw.print("你好");
        pw.print("hello");
        pw.print("</html>");


        pw.flush();
```

```
        pw.close();

    }
    //返回Servlet的字符串描述信息 作者 版本 版权
    @Override
    public String getServletInfo() {
        return null;
    }
    //servlet对象销毁的时候调用
    @Override
    public void destroy() {

    }
}
```

```
package servlet;

import javax.servlet.*;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;

/**
* Created by Tjl on 2018/8/17 9:04.
* GenericServlet 实现了Servlet,实现了4个方法,有一个抽象Service
* 有两个init方法,一个是实现了ServletCOnfig,调用重载的init 重载无参的构造方法
*/
public class CreatGenericServlet extends GenericServlet {
    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
        System.out.println("CreatGenericServlet service...");
        //1.找到图片,创建图片的字节输入流
        //相对当前运行时时类的路径
        InputStream is = this.getClass().getResourceAsStream("../2.jpg");
        res.setContentType("image/jpeg");
        //2获取响应字节输出流
        ServletOutputStream os = res.getOutputStream();
        //3.输出(边读边写,读多少写多少)
        byte[] bs = new byte[100];
        //数字len:记录每次读取了多少字节
        int len = -1;
        while ((len = is.read(bs))!=-1) {
            os.write(bs,0,len);
        }
        os.flush();
        is.close();
        os.close();


    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        System.out.println("此时是否赋值:"+getServletConfig());
        super.init(config);
    }

    @Override
    public void init() throws ServletException {
        System.out.println(getServletConfig());
        super.init();
    }
```

```
    @Override
    public void destroy() {
        System.out.println("GenericServlet实现的destroy没有任何操作,要重写");
        super.destroy();
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/**
* Created by TjI on 2018/8/17 9:59.
* test3:实现servlet的第三种,也是最常用的方式,继承自javax.servlet.http.HttpServlet
* HttpServlet继承自GenericServlet,实现了Servlet
* HttpServlet:5个方法都实现了
*/
public class CreatHttpServlet extends HttpServlet {

    public CreatHttpServlet() {
        System.out.println("创建CreatHttpServlet实例");
    }
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到Get方式请求");
        //返回响应:边看边写(println),边读边写(io)
        resp.setContentType("text/plain;charset=utf-8");
        PrintWriter pw = resp.getWriter();
        pw.print("这是响应内容");
        pw.flush();
        pw.close();
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到post方式请求");
    }

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        super.service(req, resp);
        System.out.println("httpservlet service..");
    }

    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
        super.service(req, res);
        System.out.println("servlet servic");
    }
}
```

```
package servlet;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

import javax.servlet.*;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
* day2_test5:测试Servlet的生命周期
*            1.初始化:构造器+init[只执行一次]
*                  初始化时机:默认:
*                        配置:
*            2.运行:service
*               接收到请求，执行
*            3.销毁:destory
*                  web应用停止或者重新加载
*
*
* */
public class Test5_Life_Servlet extends HttpServlet{
    public Test5_Life_Servlet() {
        System.out.println("1.执行构造器");
    }


    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        System.out.println("2.默认初始化，赋值ServletConfig对象");
    }


    @Override
    public void init() throws ServletException {
        super.init();
        System.out.println("3.无参init，重写此方法，进行初始化");
    }


    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        super.service(req, resp);
        System.out.println("5.判断请求方法，调用doXxx方法");
    }


    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
        super.service(req, res);
        System.out.println("4.运行，进行类型强制准换，调用另一个service");
    }


    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("执行doGet方法");
        resp.setContentType("text/plain;charset=utf-8");
        PrintWriter pw = resp.getWriter();
        /*
        GenericServlet已经实现了ServletConfig接口中定义的方法了
         */
        pw.println("输出:");
```

```
        pw.println(getServletName());
        pw.println(getInitParameter("password"));
        pw.println(getInitParameterNames());
        pw.println(getServletContext());
        pw.println("*****************************************");

        ServletConfig servletConfig = getServletConfig();
        Enumeration<String> initParameterNames = servletConfig.getInitParameterNames();
        pw.println("如果不知道参数名,可以获取所有参数名然后遍历");
        while (initParameterNames.hasMoreElements()) {
            String key = initParameterNames.nextElement();
            String value = servletConfig.getInitParameter(key);
            pw.println(key+" : "+value);
        }
        pw.println("*****************************************");
//        应用上下文
        ServletContext servletContext = servletConfig.getServletContext();
        pw.println("应用上下文:" +servletContext.getContextPath());
        pw.println(servletContext.getRealPath("/"));


        pw.flush();
        pw.close();
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("6.执行doPost方法");
        doPost(req, resp);
    }

    @SuppressWarnings("unused")
    private void destory() {
        System.out.println("7.销毁:程序停止或者重新加载");
    }

}
```

Test5.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>测试请求方式</title>
</head>
<body>
    <!--相对路径-->
    <a href="../LifeServlet.servlet">超链接访问方式:GET</a>

</body>
</html>
```

web.xml

```
<?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_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>FirstServlet</servlet-name>
        <servlet-class>servlet.FirstServlet</servlet-class>
    </servlet>
    <servlet-mapping>
```

```
    <servlet-name>FirstServlet</servlet-name>
    <url-pattern>/first.servlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>CreatServlet1</servlet-name>
    <servlet-class>servlet.CreatServlet1</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CreatServlet1</servlet-name>
    <url-pattern>/CreatServlet1.servlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>CreatGenericServlet</servlet-name>
    <servlet-class>servlet.CreatGenericServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CreatGenericServlet</servlet-name>
    <url-pattern>/CGS.servlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>CreatHttpServlet</servlet-name>
    <servlet-class>servlet.CreatHttpServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CreatHttpServlet</servlet-name>
    <url-pattern>/CHS.servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>CreatHttpServlet</servlet-name>
    <url-pattern>*.servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>CreatHttpServlet</servlet-name>
    <url-pattern>/111</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>Test5LifeServlet</servlet-name>
    <servlet-class>servlet.Test5_Life_Servlet</servlet-class>
    <init-param>
      <param-name>username</param-name>
      <param-value>tom</param-value>
    </init-param>
    <init-param>
      <param-name>password</param-name>
      <param-value>123</param-value>
    </init-param>
    <load-on-startup>0</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Test5LifeServlet</servlet-name>
    <url-pattern>/LifeServlet.servlet</url-pattern>
  </servlet-mapping>
</web-app>
```