本章内容

- Servlet
- ServletConfig
- ServletContext

01_Servlet介绍及入门

- A.servlet是什么，它有什么作用?
    - Servlet是运算在服务器上的一个java程序，简单说，它就是一个java类。我们要使用servlet,需要导入servlet的api.
    - Servlet主要功能在于能够和浏览器进行交互。是一个动态资源.
- B.快速入门
    - 在src下创建一个java类。HelloworldServlet,并让这个类继承HttpServlet
    - 重写了doGet方法、doPost方法
    - 在的web.xml文件中对servlet进行配置

02_Servlet详解

- Servlet是一个运行在服务器上java动态资源。Sun对于servlet定义，就是一个javax.servlet.Servlet接口。
- A.Servlet是如何处理请求?
    - Servlet接口下有一个实现类叫GenericServlet,GenericServlet有一个子类HttpServlet.
    - 在Servlet接口中定义了一个方法service,它的主要作用是处理来自浏览器的请求操作。在service方法的重载的方法中，对请求方式进行判断，如果是GET就会调用doGet方法，如果是POST就会调用doPost方法。
- B.创建Servlet的三种方式
    - implements Servlet接口
    - extends GenericServlet类
    - extends HttpServlet类
    - 在开发中，一般应用比较多的是使用extends HttpServlet，优点是它是与http协议相关的，封装了http协议相关的操作。

03_Servlet生命周期

- 在javax.servlet.Servlet接口中定义了三个方法init service destroy它们就是servlet的生命周期方法
- A.什么是生命周期?

    - 简单理解成是servlet是什么时候创建的，它是什么时候销毁的。

- B.演示servlet生命周期:

- C.Servlet生命周期总结:

    - 第一次访问servlet,servlet会被创建，并将servlet对象常驻内存，调用init方法进行初始化操作，init方法中执行一次。
    - 调用service方法，用于处理来自浏览器端的请求，以后都是开启一个线程来处理浏览器端请求。
    - 当tomcat服务器正常关闭时，会调用destroy方法将servlet销毁。

04_load-on-startup配置

- 可以让servlet跟随服务器的启动而启动
- 对于load-on-startup它的可以配置的值有10个，1代表优先级最高，数值越大，优先级越低。

05_Servlet配置详解

- 对于servlet,我们需要在web.xml文件中对其进行配置
    - 在web.xml中声明Servlet
    - 在web.xml中给Servlet映射访问路径
- A.问题1：对于一个servlet我们是否可以有多条路径访问?
    - 可以,一个<servlet>可有多个<servlet-mapping>与其对应.
- B.问题2:url-pattern的它书写规则?
    - 完全匹配 要求以”/”开始
    - 目录匹配 要求以”/”开始，以*结束
        - 比如:/a/b/c/*
        - 只要前面是/a/b/c后面是啥无所谓,就可以访问到对应的servlet
    - 扩展名匹配 要求不能以”/”开始，以*.xxx结束
        - 比如:*.abc,
        - 那么前面怎么写无所谓,但是必须以abc结尾才能访问到对应的servlet，
        - 最经典错误 /*.xxx

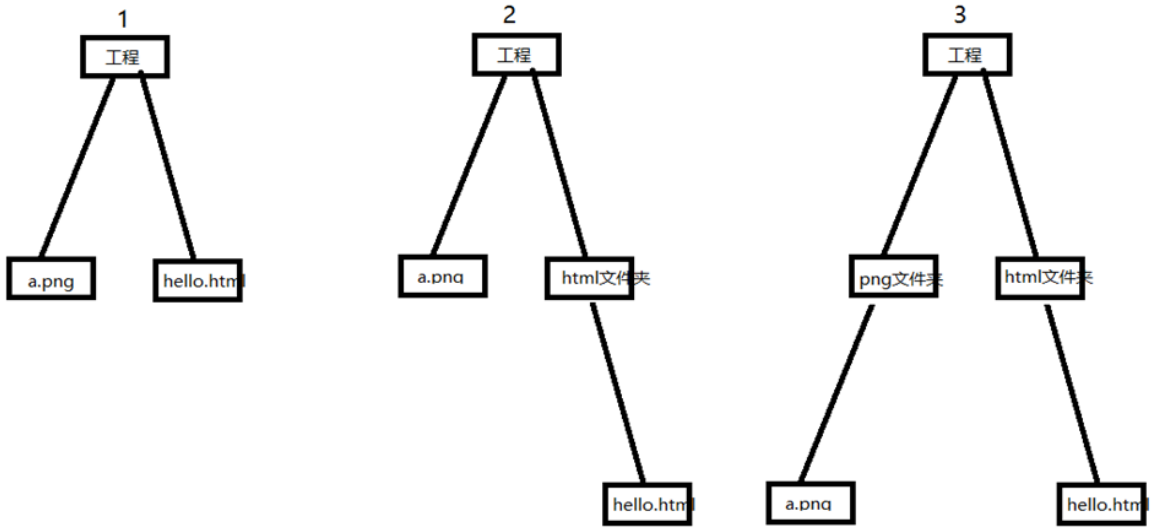
06_缺省servlet

- A.概念
    - 创建一个servlet时，如果它的url-pattern的配置值为”/”这时这个servlet就是一个缺省的servlet，tomcat服务器中默认就有缺省Servlet
- B.缺省Servlet的作用
    - 凡是在web.xml文件总找不到匹配的<servlet-mapping>元素的URL，它们的请求都将交给缺省Servlet处理。也就是说，缺省的servlet用于处理其他Servlet处理不了的请求。
    - 当访问tomcat服务中的静态资源(html、图片等等)时，实际上是在访问这个缺省的servlet。
- C.案例演示

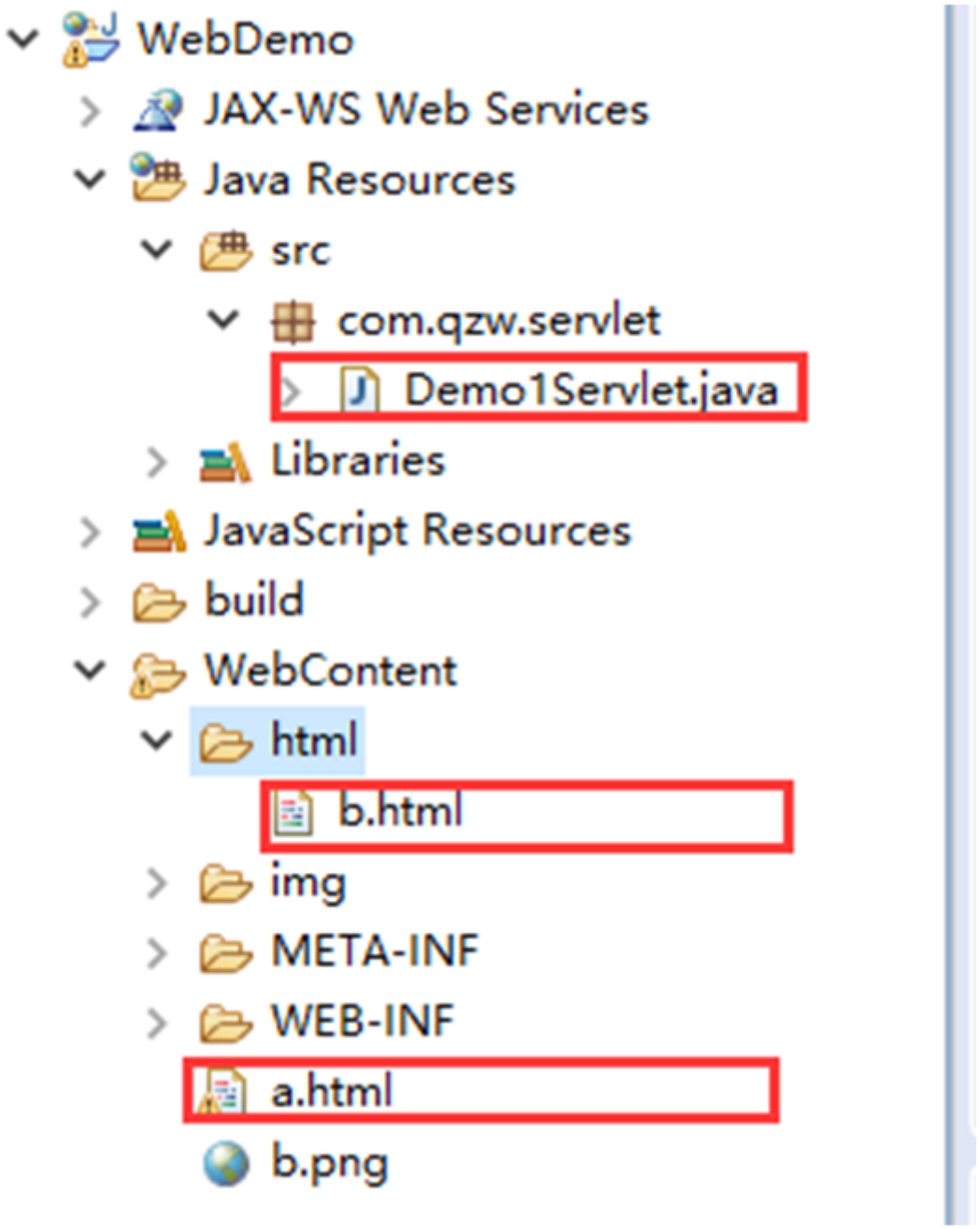
- 缺省Servlet的两个作用

07_服务器的路径问题

- A.案例1:



- 分别使用绝对路径和相对路径完成以上三种情况,在hello.html中访问a.png.
- B.案例2:



- 分别使用绝对路径和相对路径在html文件中访问Demo1Servlet

08_ServletConfig对象

- A.ServletConfig介绍
  - ServletConfig是javax.servlet.包下的一个接口。ServletConfig它是Servlet的一个配置对象。对于ServletConfig对象我们学习它，主要明白以下三个问题
- B.问题1:ServletConfig是由谁创建的，它怎样传递到Servlet中?
  - ServletConfig对象是由服务器创建的，它是通过Servlet的init方法传递到Servlet中。
- C.问题2:ServletConfig对象，它的作用是什么?
  - 获取Servlet名称 getServletName
  - 获取Servlet初始化参数 getInitParameter getInitParameterNames
  - 获取ServletContext对象。

- D.问题3：如何获取一个ServletConfig对象?
  - Servlet接口下的getServletConfig方法
- E.案例演示
  - ServletConfig对象的作用

09_ServletContext对象

- A.ServletContext介绍
  - ServletContext它是javax.servlet包下的一个接口。
  - 当服务器启动时，会为服务器中的每一个web应用程序创建一个ServletContext对象，一个ServletContext对象对应的就是一个web应用程序。
  - 对于ServletContext,我们叫它上下文对象，ServletConfig对象中维护了ServletContext对象，也就是说，我们可以通过ServletConfig对象来获取ServletContext对象。
  - 在web应用中的servlet要想实现资源的共享，可以通过ServletContext来完成，ServletContext也叫做域对象。
- B.ServletContext对象作用
  - a.实现Servlet资源共享
    - ServletContext也叫做域对象，可以将它想像成一个Map<String,Object>，可以通过它实现Servlet资源共享。
    - public Object getAttribute(String name)
    - public void removeAttribute(String name)
    - public void setAttribute(String name, Object object)
  - b.获取全局初始化参数
    - 在web.xml中配置的全局初始化参数，可以通过ServletContext对象获取
    - public String getInitParameter(String name)
    - public java.util.Enumeration<E> getInitParameterNames()
  - c.获取资源在服务器上的真实磁盘路径
    - public String getRealPath(String path)
    - 传入的参数是从 当前servlet 部署在tomcat中的文件夹算起的相对路径.
  - d.案例演示
    - ServletContext对象的作用

11_ServletContext综合案例

- 需求:统计站点访问次数

```
//1,获取ServletContext对象
ServletContext servletContext = getServletContext();
//2,判断是否第一次
Object count = servletContext.getAttribute("count");
if(null == count){
    //2.1,第一次访问
    servletContext.setAttribute("count", 1);
}else {
    //2.2,非第一次访问
    Integer existCount = (Integer) servletContext.getAttribute("count");
```

```
    existCount++;
    servletContext.setAttribute("count", existCount);
}
System.out.println(servletContext.getAttribute("count"));
```