JavaWeb核心之Servlet

1.Servlet简介

1.1 什么是Servlet

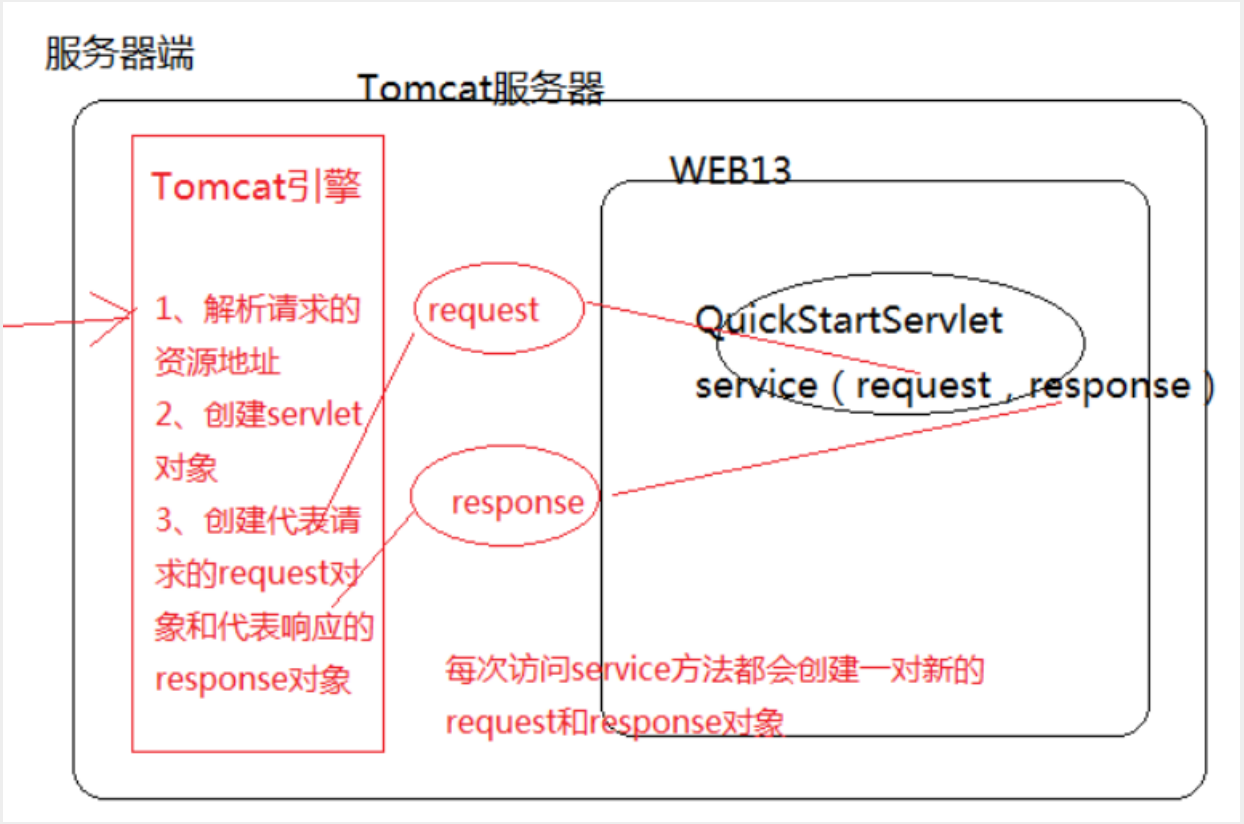
Servlet 运行在服务端的Java小程序，是sun公司提供一套规范（接口），用来处理客户端请求、响应给浏览器的动态资源。但servlet的实质就是java代码，通过java的API 动态的向客户端输出内容。
Servlet规范：包括三个技术点：
1. servlet技术
2. filter技术
3. listener技术

1.2 Servlet快速入门

实现步骤：
1. 创建类实现Servlet接口
2. 覆盖尚未实现的方法---service方法
3. 在web.xml进行servlet的配置

但在实际开发中，我们不会直接去实现Servlet接口，因为那样需要覆盖的方法太多，我们一般创建类继承HttpServlet
实现步骤：
1. 创建类继承HttpServlet类
2. 覆盖doGet和doPost

3. 在web.xml中进行servlet的配置



3. Servlet的API

3.1 Servlet接口中的方法

3.1.1 init(ServletConfig config)

何时执行：Servlet对象创建时执行。
ServletConfig：代表的是该servlet对象的配置信息。

3.1.2 service(ServletRequest request,ServletResponse response)

何时执行：每次请求都会执行
ServletRequest ： 代表请求 认为ServletRequest 内部封装的http请求的信息
ServletResponse ： 代表响应 认为要封装的是响应的信息

3.1.3 destroy()

何时执行：servlet对象销毁的时候执行

3.2 HttpServlet类中的方法

```
1）init()
2）doGet(HttpServletRequest request,HttpServletResponse response)
```

```
3）doPost(HttpServletRequest request,HttpServletResponse response)
4）destroy()
```

3.3 Servlet的生命周期

1. Servlet何时创建
   默认第一次访问servlet时创建该对象
   如果在web.xml中配置<load-on-startup></load-on-startup>，标签内传入表示优先级的正整数，那么Servlet在服务器启动时就创建了。

2. Servlet何时销毁
   服务器关闭servlet就销毁了

3. 每次访问必然执行的方法
   service(ServletRequest req, ServletResponse res)方法

4. Servlet的配置

4.1 基本配置

```
<!--servlet类的配置-->
<servlet>
    <servlet-name>QuickStartServlet</servlet-name>
    <servlet-class>com.scct.servlet.QuickStartServlet</servlet-class>
</servlet>
<!--servlet虚拟路径的配置-->
<servlet-mapping>
    <servlet-name>QuickStartServlet</servlet-name>
    <url-pattern>/testServlet</url-pattern>
</servlet-mapping>
```

4.1.1 servlet-name

可以随便取，但是servlet类的配置及其虚拟路径的配置中的servlet-name需要对应。

4.1.2 servlet-class

必须是servlet类的全包名

4.1.3 url-pattern

明白了上面两个基本参数的意思之后，接下来是的配置方式

1. 完全匹配，访问的资源与配置的资源完全相同时才可以访问

```
例：<url-pattern>/testServlet</url-pattern>
    URL：http://localhost:8080/WEB13/testServlet
```

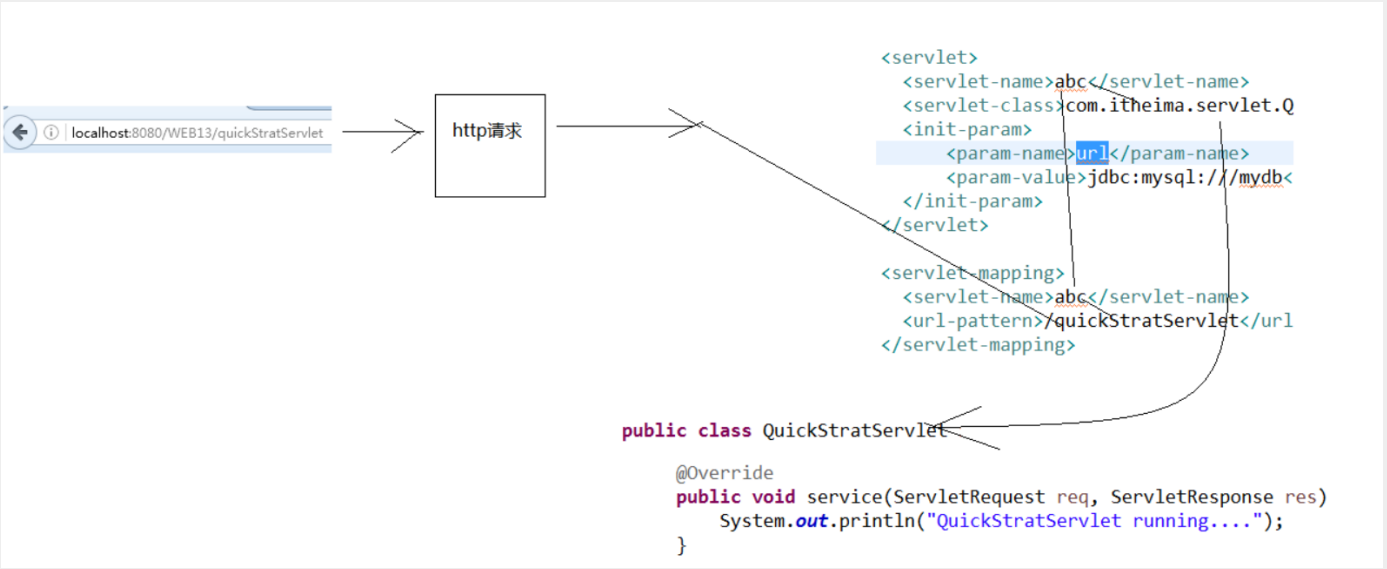
2. 目录匹配，只要URL路径前面的目录与配置的目录相同就可以访问，与目录后面的内容无关

```
格式：<url-pattern>/虚拟路径/*</url-pattern>
例：<url-pattern>/aaa/bbb/ccc/*</url-pattern>
URL：http://localhost:8080/WEB13/aaa/bbb/ccc/fsfddfdf.html
```

3. 扩展名匹配，只要扩展名相同就可以访问资源

```
格式：<url-pattern>*.扩展名</url-pattern>
例：<url-pattern>*.aaa</url-pattern>
URL：http://localhost:8080/WEB13/.aaa
```

※：第二种与第三中不能混用，/aaa/bbb/*.abcd（错误的）



4.2 缺省配置

1. 如果某个Servlet的仅仅为一个正斜杠(/)，那么这个Servlet就成为当前web应用程序的缺省servlet。

2. 默认（缺省）servlet是用于处理别人处理不了的请求。(处理写错的或者没有匹配的路径)

3. 凡是在web.xml文件中找不到匹配的元素的URL，它们的访问请求都将交给缺省servlet处理，也就是说，缺省servlet用于处理所有其他servlet都不处理的访问请求。

4. 在<tomcat的安装目录>\conf\web.xml文件（这是服务器的配置文件，自己所有工程的web.xml文件都相当于继承了这个文件）中，注册了一个名称为org.apache.catalina.servlets.De

faultServlet的Servlet，并将这个Servlet设置为了缺省Servlet。

5. 当访问Tomcat服务器中的某个静态HTML文件和图片时，实际上是访问这个缺省Servlet。
6. 当访问web工程下内容下，都是web.xml文件中的去查找资源。而我们在访问a.txt时，在xml文件中没有与其对应的资源，这时，是默认的servlet处理。默认servlet搜索一下，有没有a.txt，如果有，就读出来，显示在浏览器中。没有就404。我们看到的404都是默认servlet处理的。默认servlet怎么读，怎么写另学。
7. 如果自己写了默认servlet，Tomcat中的servlet失效。

4.3 欢迎页面

当我们访问到项目目录，不再深入访问别的资源时，会有一个欢迎页面，前提是该欢迎页面存在，否则报404错误。下面的welcome-file-list配置欢迎页面。

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

如果项目目录下，本文即http://localhost:8080/WEB13下有这里面的某个资源，那么，在该目录下，不访问任何资源将会有一个欢迎页面出现，其顺序如上，可以自定义配置。即按照自定义的配置顺序找到第一个存在的资源并显示。

5. ServletContext对象

5.1 ServletContext对象概念

ServletContext代表一个web应用环境对象，即一个web环境。也就是说ServletContext对象内部封装了该web应用的信息。因此一个web应用只有一个ServletContext对象。
我们知道Servlet对象的生命周期是：默认第一次访问服务器时创建Servlet对象，服务器关闭时销毁Servlet对象。那么ServletContext对象的生命周期又是怎样的呢?
创建时间：web应用被加载（即服务器启动或发布web应用）时；
销毁时间：web应用被卸载（即服务器关闭或者该应用被移除）时；

5.2 ServletContext对象获取

1. Servlet类中有个init(ServletConfig config)方法，可以通过该方法获得ServletContext对象；

```
例：ServletContext servletContext=config.getServletContext();
```

2. 在Servlet类中我们还可以通过该类的另一个方法来获得ServletContext对象；

```
例：ServletContext servletContext=this.getServletContext();
```

5.3 ServletContext对象作用

5.3.1 获得web应用全局的初始化参数

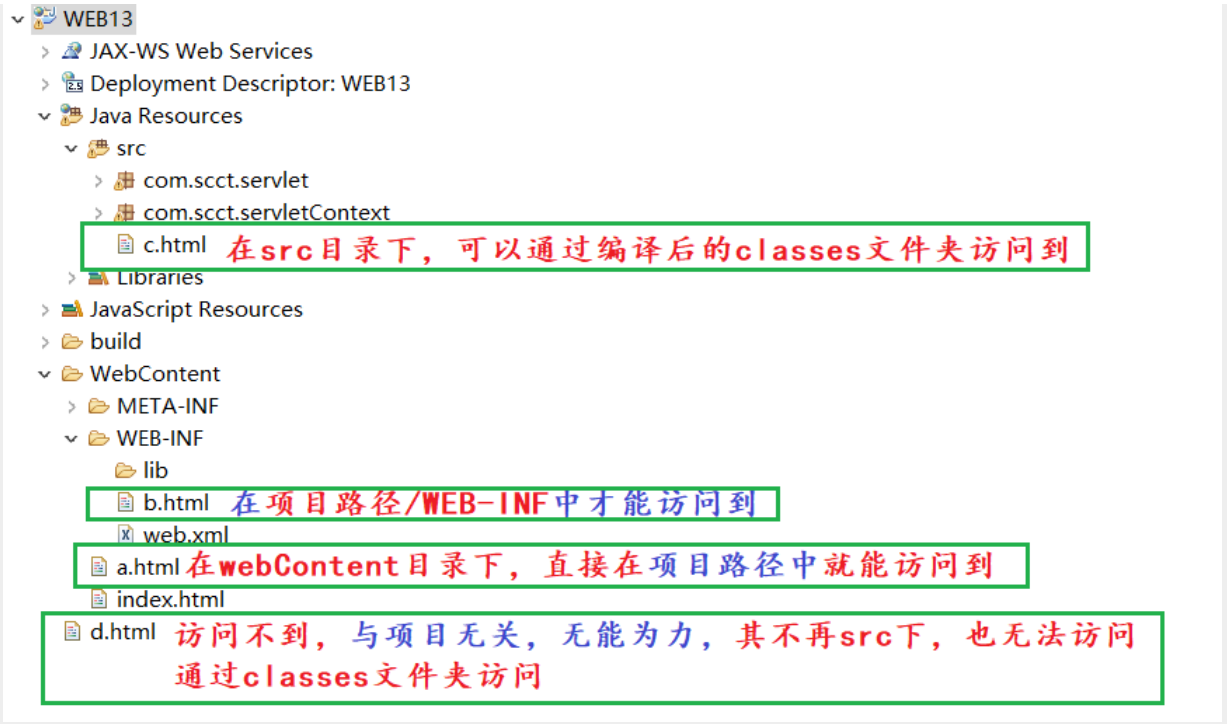
首先看一个配置文件：web.xml 该文件配置了web应用程序的信息，如：

```
<context-param>
    <param-name>Driver</param-name>
    <param-value>com.mysql.jdbc.Driver</param-value>
</context-param>
```

这是一个数据库连接的信息，我们可以通过ServletContext对象来获得该信息。

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
//获得web应用全局的初始化信息
        ServletContext servletContext = this.getServletContext();
        String initParameter = servletContext.getInitParameter("Driver");
        System.out.println("initParameter:"+initParameter);//输出结果：initP
arameter:com.mysql.jdbc.Driver
        }
```

5.3.2 获得web应用中资源绝对路径



用ServletContext对象.getRealPath("相对路径名")；即可访问。

```
        //2、获得a b c d.txt的绝对路径
        //2.1 获得a.txt
        String realPath_A = context.getRealPath("a.txt");
        System.out.println(realPath_A);
        //2.2 获得b.txt
        String realPath_B = context.getRealPath("WEB-INF/b.txt");
        System.out.println(realPath_B);
        //2.3 获得c.txt
        String realPath_C = context.getRealPath("WEB-INF/classes/c.txt");
        System.out.println(realPath_C);
        //2.4 获得d.txt----获取不到
```

结果如下：

```
D:\tomcat\apache-tomcat-7.0.52\webapps\WEB13\a.txt
D:\tomcat\apache-tomcat-7.0.52\webapps\WEB13\WEB-INF\b.txt
D:\tomcat\apache-tomcat-7.0.52\webapps\WEB13\WEB-INF\classes\c.txt
```

5.3.3 ServletContext是一个域对象

什么是域对象？ 存储数据的区域就是域对象。
ServletContext域对象的作用范围：整个web应用(所有的web资源都可以随意向ServletContext域对象中存取数据，数据可以共享)。
域对象的通用的方法：

1. setAtrribute(String name,Object obj);
2. getAttribute(String name);
3. removeAttribute(String name);

```
        //3、域对象---向servletContext中存数据
        context.setAttribute("name", "zhangsan");
```

访问第二个servlet资源，取出第一个servlet对象创建的数据：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    //域对象---从servletContext中取数据
    String attribute = (String) this.getServletContext().getAttribute("name");
    System.out.println(attribute);
}
```

则结果：

```
zhangsan
```