## [01] Servlet 是什么

# 1、Servlet 是什么

Servlet（Server Applet），全称 Java Servlet，是用 Java 编写的服务器端程序。其主要功能在于交互式地浏览和修改数据，生成动态 Web 内容。

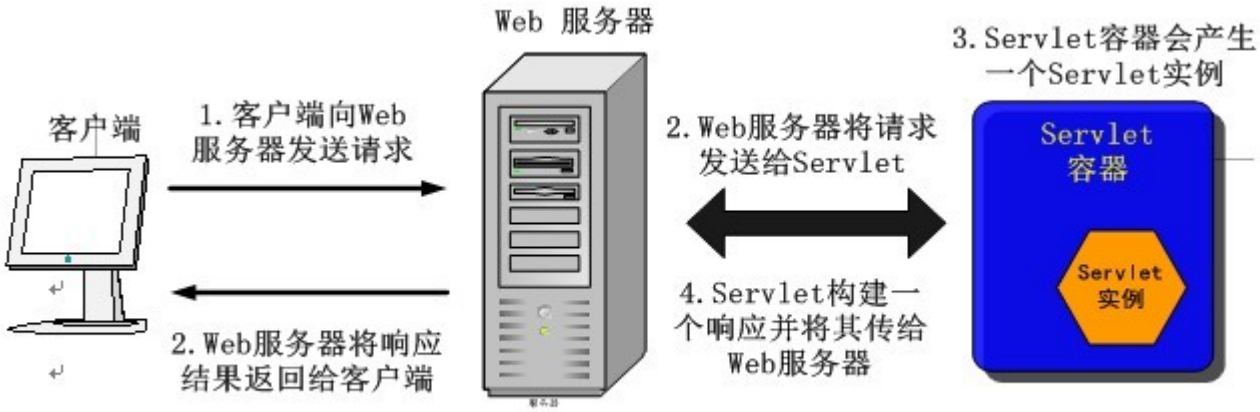
## 1.1 广义上来讲

从本质上来讲，**Servlet 就是一个特殊的 Java 类**，说它特殊是因为这个 Java 类**必须直接或间接地实现 Servlet 接口**（javax.servlet.Servlet），我们自定义的 Servlet 更多是采用继承 HttpServlet 的方式，以达到间接实现 Servlet 接口的目的。

Servlet 还有一个比较特殊的地方：**必须运行在 Web 容器（服务器）中**，不能像普通类使用 main 方法访问，它的方法由服务器在相应情况下调用执行。一般来说自定义 Servlet 要覆盖其 doGet 和 doPost 方法，根据请求是 GET/POST 方式会自动调用相应的 doGet/doPost 方法。

又要实现接口，又只能在 Web 容器中运行，那么 Servlet 的任务就来了：

- 获取请求数据
- 处理请求
- 完成响应




## 1.2 狭义上来讲

首先要明白的是，JavaEE 实际上是 sun 公司定义的一系列技术标准所组成的平台，用来 B/S 结构的应用程序。所以它实质是一套体系或者说是一套标准（其中如我们所熟知的 JSP、JSTL 和 Servlet 都隶属其中），所以有了以下 Servlet 的狭义说法。

所以，狭义的 Servlet 是指 Java 语言实现的一个**接口**，再白话一点，你可以理解为，Servlet 就是一套**规范**，不过是以接口的形式展现。下面摘自 J2EE 的 API：

*Defines methods that all servlets must implement.*

*A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.*

*To implement this interface, you can write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extends javax.servlet.http.HttpServlet.*

*This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:*

- *The servlet is constructed, then initialized with the init method.*
- *Any calls from clients to the service method are handled.*
- *The servlet is taken out of service, then destroyed with the destroy method, then garbage collected and finalized.*

*In addition to the life-cycle methods, this interface provides the getServletConfig method, which the servlet can use to get any startup information, and the getServletInfo method, which allows the servlet to return basic information about itself, such as author, version, and copyright.*

可以看到，在 J2EE 这套大标准中，**Servlet 是作为服务器端用以交互请求和响应的一套标准**。而符合 J2EE 这套标准（包括 Servlet 标准）的产品叫“实现”，不同服务器对待这套标准的方式不同，我们也可以说，“实现”是不同的，所以像 JBoss、Tomcat、WebLogic 都是 J2EE 标准的“实现”。

## 2、Servlet 的配置

访问 Servlet 之前，必须配置 Servlet，否则不同的请求，如何对应不同的 Servlet 来进行处理呢？

在 web.xml 中配置 Servlet 的主要信息，示例如下：

```
<servlet>
    <!-- 自定义的名字，在 web.xml 中不重复 --->
    <servlet-name>MyServlet</servlet-name>
    <!-- Servlet 类的完整名字 -->
    <servlet-class>j2ee.servlet.MyServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <!-- 与之前自定义的名字相对应 -->
    <servlet-name>MyServlet</servlet-name>
    <!-- 自定义的逻辑地址，必须以/开头 -->
    <url-pattern>/doServlet</url-pattern>
</servlet-mapping>
```

这里重点说明一下 url-pattern，自定义的逻辑地址，也就是说如果访问的 url 中，根路径之后符合自定义的逻辑要求，则跳转到对应的 Servlet，如上 /doServlet，意味着如果我的 url 是诸如 http://localhost:8080/doServlet，那么访问会跳转到 j2ee.servlet.MyServlet 类中去，根据 GET 或 POST 调用相应的 doGet 或 doPost 方法。

其中，自定义的逻辑地址，有以下几种匹配方式：

- **完全匹配**
  - 如<url-pattern>/doServlet</url-pattern>
  - 即如 http://localhost:8080/doServlet 才能访问， .../doServlet.do 或 .../doServlet/xxx 等都无法访问到该 Servlet
- **路径匹配**
  - 如<url-pattern>/*</url-pattern>
  - 即根路径下所有请求都会访问到对应 Servlet
- **扩展名匹配**
  - 如<url-pattern>*.do</url-pattern> 以 .do 结尾的请求
  - 如<url-pattern>*.html</url-pattern> 以 .html 结尾的请求

## 3、Servlet 的作用

最基本的网页是静态的，也就是最普通的 html 形式，为了能编写服务器动态网页，Servlet 接下了重任，用来输出 HTML 标签和内容：

```
public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\">");
        out.println("<HTML>");
        out.println("  <HEAD><TITLE>This is my
Servlet</TITLE></HEAD>");
        out.println("  <BODY>");
        out.print("  This is ");
        out.print(this.getClass());
```

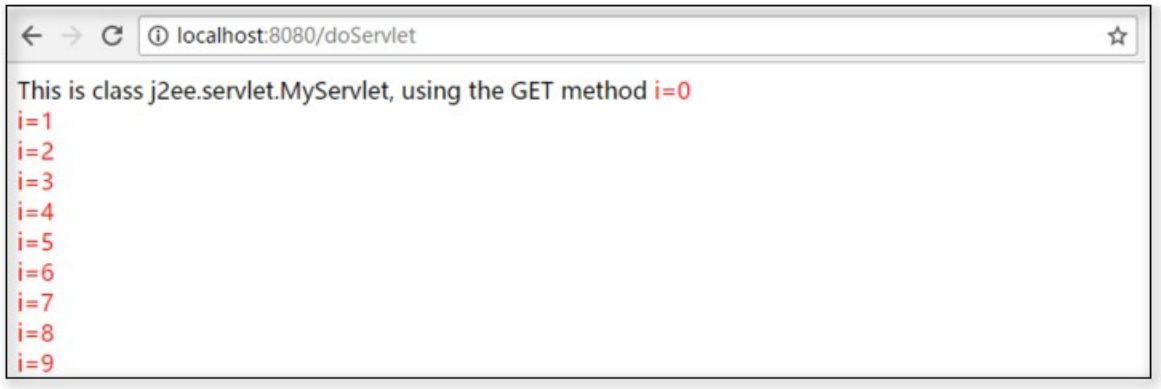
```
        out.println(", using the GET method");
        out.println("  </BODY>");
        for (int i = 0; i < 10; i++) {
            out.println("<font color='red'>i=" + i + "</font><br>");
        }
        out.println("</HTML>");
        out.flush();
        out.close();
    }

    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

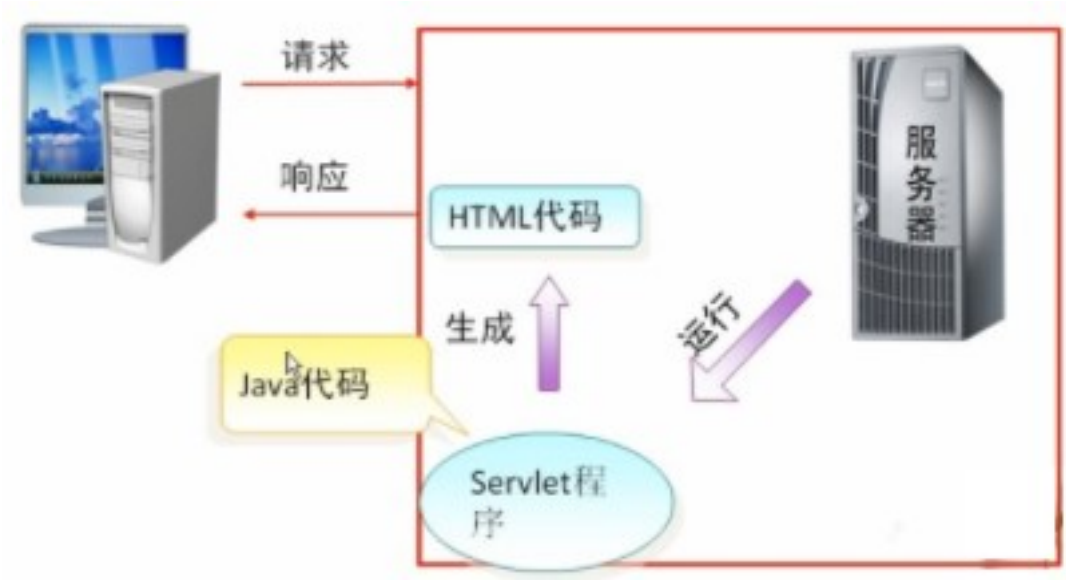
如 web.xml 中配置 Servlet 如下：

```
<servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>j2ee.servlet.MyServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/doServlet</url-pattern>
</servlet-mapping>
```

那么我们访问符合的路径时，我们得到的是：




可以看到，我们通过 Servlet 中编写网页内容，最终得到了相应的网页内容，而且该页面的内容根据我们代码中的逻辑不同会动态发生改变，也就是我们说的能编写动态页面。




但是真的太麻烦了，我们要写大量诸如 out.print 的重复代码，而且格式杂乱也不便于修改。

直到 JSP 的出现，需要输出动态内容的功能，都在 JSP 中实现了（至于 JSP 是什么，以后会提到），Servlet 不用再去写那些痛苦的 out.print 代码了。所以，现在 Servlet 的作用是**用来做控制器使用，用来接收请求，处理请求，跳转到不同的页面**。

**Servlet –** Fast but Strict Guideline




**JSP –** Flexible, Easy but Slow


**JSP** **Web Server** **Servlet**

i