Servlet 学习笔记

From W3C

Servlet 是什么

Java servlet 是运行在 web 服务器或者应用服务器上的程序，它是作为来自 web 浏览器或者其他 HTTP 客服端的请求和 HTTP 服务器上的数据库或者应用程序之间的**中间层**。

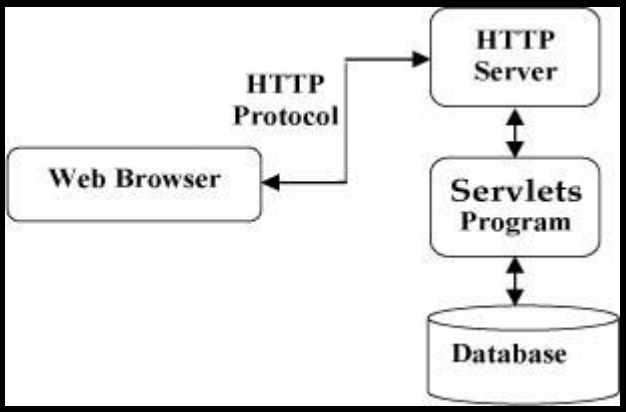
使用 Servlet，可以收集来自网页表单的用户输入，呈现来自数据库或者其他源的记录，还可以动态创建网页。

Java Servlet 通常可以与使用 CGI: Common gateway interface 公共网关接口 实现相同的功能。

相比于 CGI，有如下优点：

1. 性能更好
2. Servlet 在 web 服务器中执行，没有必要再创建一个单独的今次那个来处理每一个客户端请求。
3. 独立于平台—java 编写的
4. 服务器上的 java 安全管理器执行一些列的限制，更加安全可信
5. Java 类库支持

Servlet 在 web 程序中的位置：

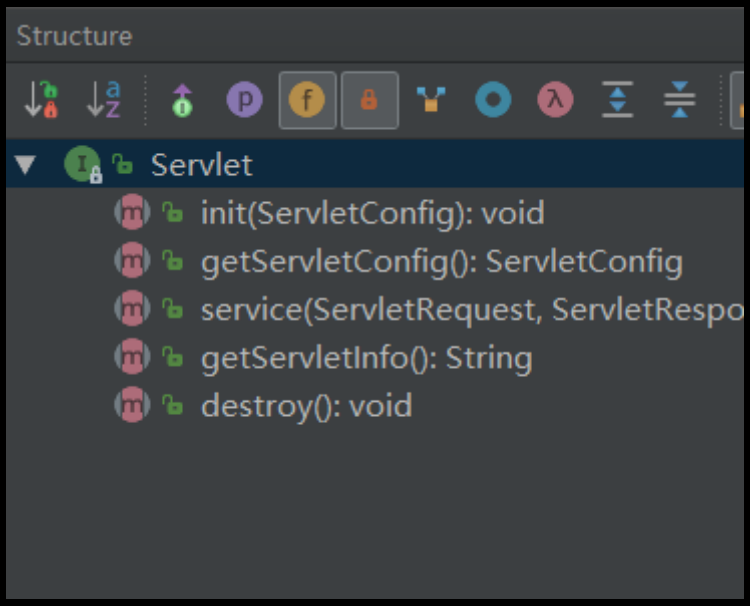


Servlet 的功能/任务：

1. 读取客户端/浏览器发送的显式数据。这包括网页上的 HTML 表单
2. 读取客户端发送的隐式的 HTTP 请求数据，包括 cookies 等
3. 处理数据并生成结果
4. 发送数据到客户端

Servlet 的本质

servlet 就是一个 Java 接口



网络协议、http 什么的，servlet 根本不管

那 servlet 是干嘛的？很简单，接口的作用是什么？规范呗！

servlet 接口定义的是一套处理网络请求的规范，所有实现 servlet 的类，都需要实现它那五个方法，其中最主要的是两个生命周期方法 init()和 destroy()，还有一个处理请求的 service()，也就是说，所有实现 servlet 接口的类，或者说，所有想要处理网络请求的类，都需要回答这三个问题：

1. 你初始化时要做什么
2. 你销毁时要做什么
3. 你接受到请求时要做什么

这是 Java 给的一种规范！

servlet 是一个规范，那实现了 servlet 的类，就能处理请求了吗？ 不能。

你从来不会在 servlet 中写什么监听 8080 端口的代码，**servlet 不会直接和客户端打交道**！

那请求怎么来到 servlet 呢？答案是 servlet 容器，比如我们最常用的 tomcat

tomcat 才是与客户端直接打交道的家伙，他监听了端口，请求过来后，根据 url 等信息，确定要将请求交给哪个 servlet 去处理，然后调用那个 servlet 的 service 方法，service 方法返回一个 response 对象，tomcat 再把这个 response 返回给客户端。

Servlet 的前世今生

什么是 Web 服务器？

Web 服务器的作用说穿了就是：**将某个主机上的资源映射为一个 URL 供外界访问。**

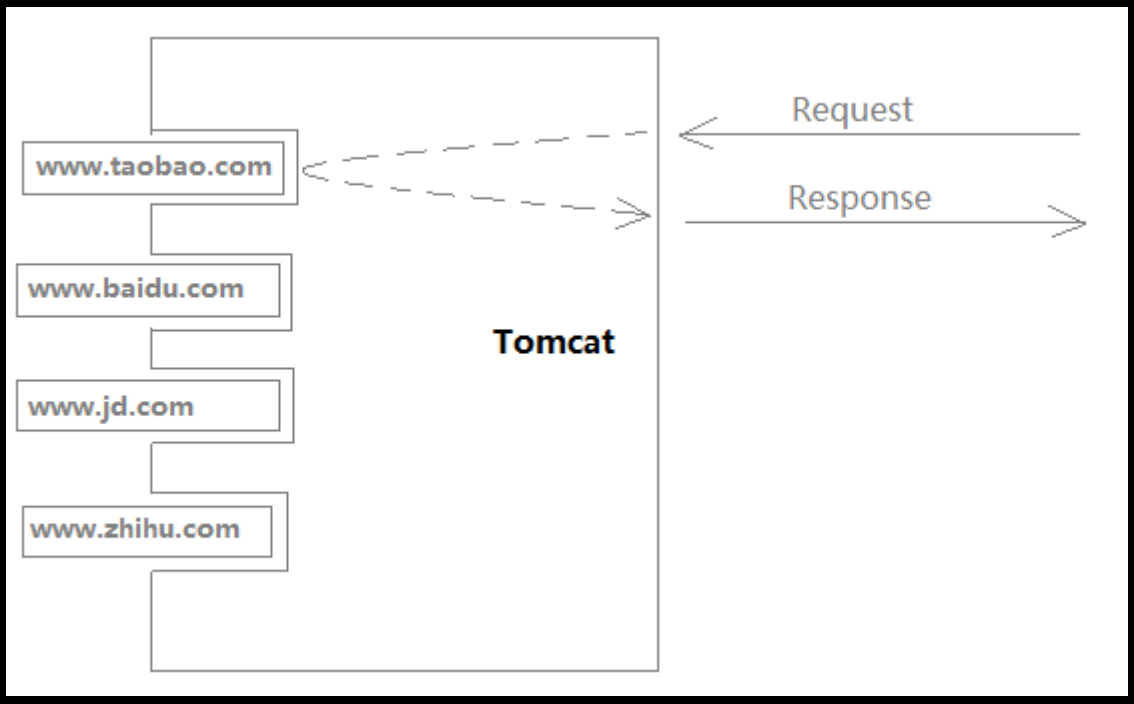
什么是 Servlet 容器？

顾名思义，里面存放 Servlet 对象

为什么能通过 web 服务器映射的 URL 访问资源？ 肯定需要写程序处理请求，主要三个过程：

1. 接收请求
2. 处理请求
3. 响应请求

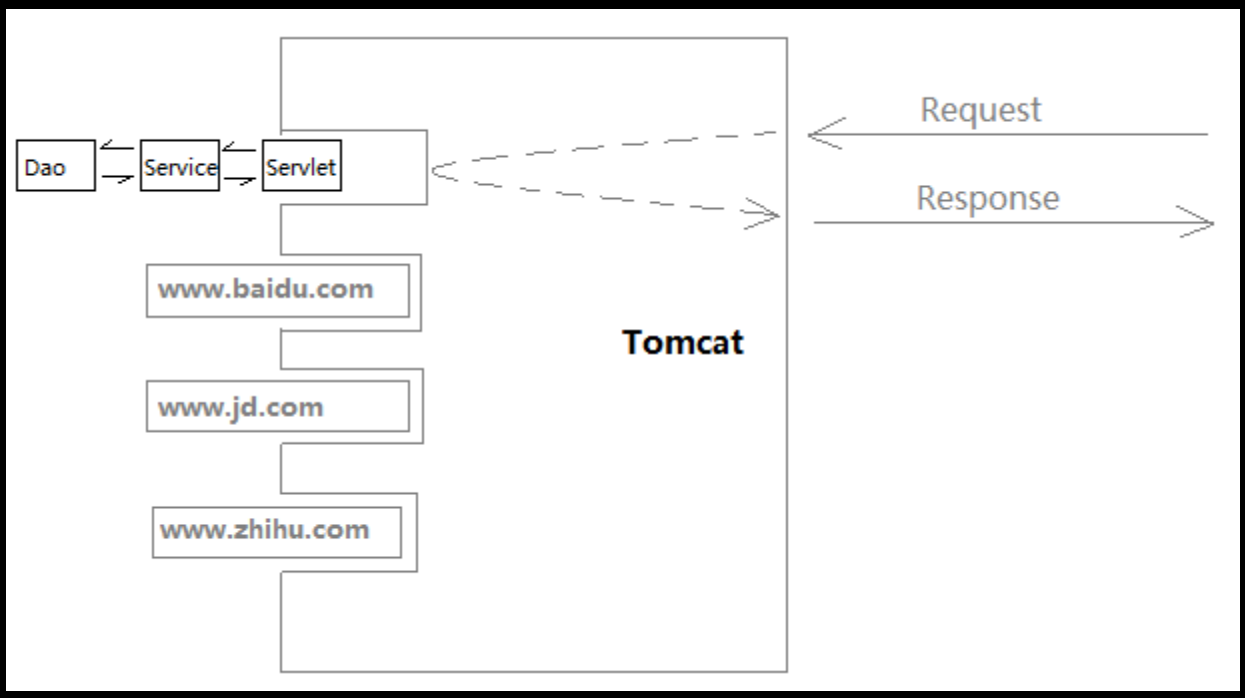
接收和响应是共性的功能，没有差异性，于是大家就把接收和相应两个步骤抽取成 web 服务器。



但是请求的逻辑是不同的，怎么办？

抽取出来做成 Servlet，交给程序员自己编写。

当然，后来出现了三层架构，一些逻辑从 Servlet 中抽离出来，分担到 service 和 dao。



但是 Servlet 并不擅长往浏览器输出 HTML 页面，所以出现了 JSP

等 Spring 家族出现后，Servlet 开始退居幕后，取而代之的是方便的 SpringMVC。

SpringMVC 的核心组件 DispatcherServlet 其实本质就是一个 Servlet。

但它已经自立门户，在原来 HttpServlet 的基础上，又封装了一条逻辑。

Servlet 生命周期

Servlet 遵循的过程：

1. Servlet 通过调用 init () 方法进行初始化。
2. Servlet 调用 service() 方法来处理客户端的请求。
3. Servlet 通过调用 destroy() 方法终止（结束）。

init() 方法

```
public void init() throws ServletException {
  // 初始化代码...
}
```

init 方法被设计成只调用一次。它在第一次创建 Servlet 时被调用，在后续每次用户请求时不再调用。因此，它是用于一次性初始化，就像 Applet 的 init 方法一样。

Servlet 创建于用户第一次调用对应于该 Servlet 的 URL 时，但是您也可以指定 Servlet 在服务器第一次启动时被加载。

当用户调用一个 Servlet 时，就会创建一个 Servlet 实例，每一个用户请求都会产生一个新的线程，适当的时候移交给 doGet 或 doPost 方法。init() 方法简单地创建或加载一些数据，这些数据将被用于 Servlet 的整个生命周期。

service() 方法

service() 方法是执行实际任务的主要方法。Servlet 容器（即 Web 服务器）调用 service() 方法来处理来自客户端（浏览器）的请求，并把格式化的响应写回给客户端。

每次服务器接收到一个 Servlet 请求时，服务器会产生一个新的线程并调用服务。service() 方法检查 HTTP 请求类型（GET、POST、PUT、DELETE 等），并在适当的时候调用 doGet、doPost、doPut，doDelete 等方法。

```
public void service(ServletRequest request,
                    ServletResponse response)
     throws ServletException, IOException{
}
```

service() 方法由容器调用，service 方法在适当的时候调用 doGet、doPost、doPut、doDelete 等方法。所以，您不用对 service() 方法做任何动作，您**只需要根据来自客户端的请求类型来重载 doGet() 或 doPost() 即可**。

doGet() 方法

GET 请求来自于一个 URL 的正常请求，或者来自于一个未指定 METHOD 的 HTML 表单，它由 doGet() 方法处理。

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
    // Servlet 代码
}
```

doPost() 方法

```
public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
    throws ServletException, IOException {
    // Servlet 代码
}
```

destroy() 方法

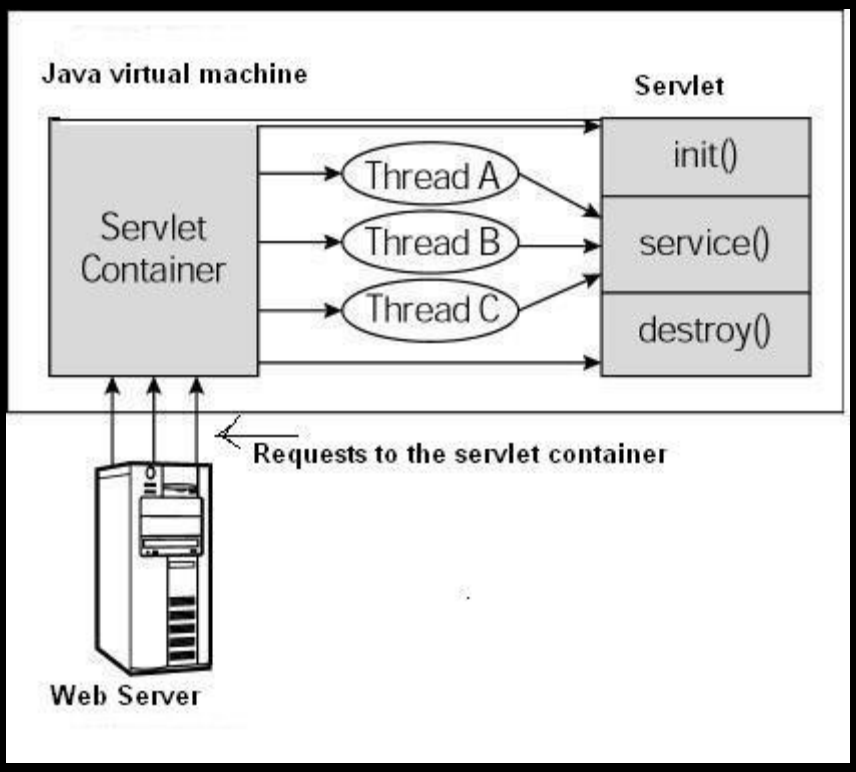
destroy() 方法只会被调用一次，在 Servlet 生命周期结束时被调用。destroy() 方法可以让您的 Servlet 关闭数据库连接、停止后台线程、把 Cookie 列表或点击计数器写入到磁盘，并执行其他类似的清理活动。

```
  public void destroy() {
    // 终止化代码...
  }
```

架构图

下图显示了一个典型的 Servlet 生命周期方案。

1. 第一个到达服务器的 HTTP 请求被委派到 Servlet 容器。
2. Servlet 容器在调用 service() 方法之前加载 Servlet。
3. 然后 Servlet 容器处理由多个线程产生的多个请求，每个线程执行一个单一的 Servlet 实例的 service() 方法。



Servlet 实例

Servlet 是服务 HTTP 请求并**实现 javax.servlet.Servlet 接口**的 Java 类。Web 应用程序开发人员通常**编写 Servlet 来扩展 javax.servlet.http.HttpServlet**，并实现 Servlet 接口的抽象类专门用来处理 HTTP 请求。

```
// 导入必需的 java 库
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// 扩展 HttpServlet 类
public class HelloWorld extends HttpServlet {

  private String message;

  public void init() throws ServletException
  {
      // 执行必需的初始化
      message = "Hello World";
  }

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
      // 设置响应内容类型
      response.setContentType("text/html");

      // 实际的逻辑是在这里
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
  }

  public void destroy()
  {
      // 什么也不做
  }
}
```

Servlet 部署

默认情况下，Servlet 应用程序位于路径 <Tomcat-installation-directory>/webapps/ROOT 下，且类文件放在 <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes 中。

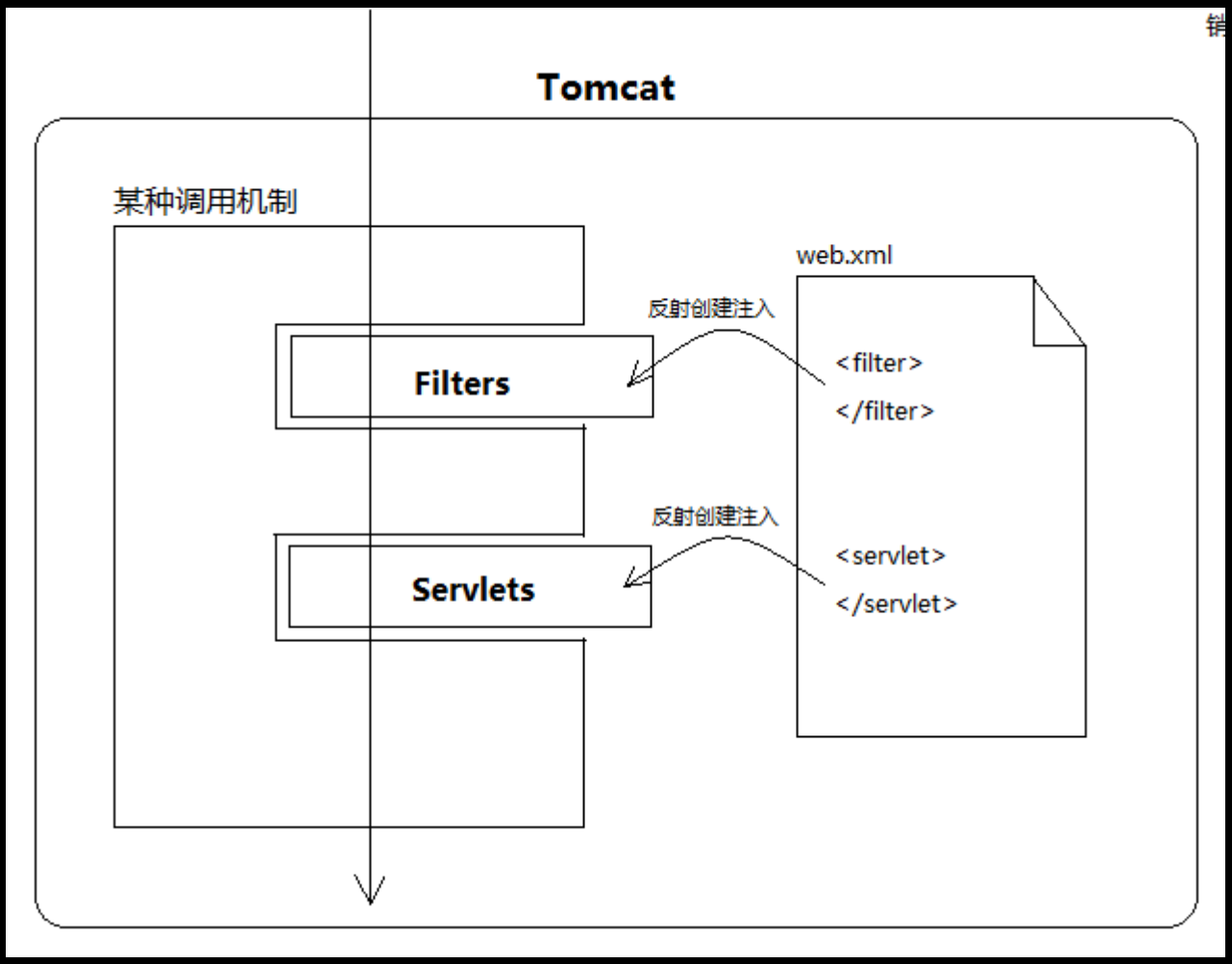
如果您有一个完全合格的类名称 com.myorg.MyServlet，那么这个 Servlet 类必须位于 WEB-INF/classes/com/myorg/MyServlet.class 中。

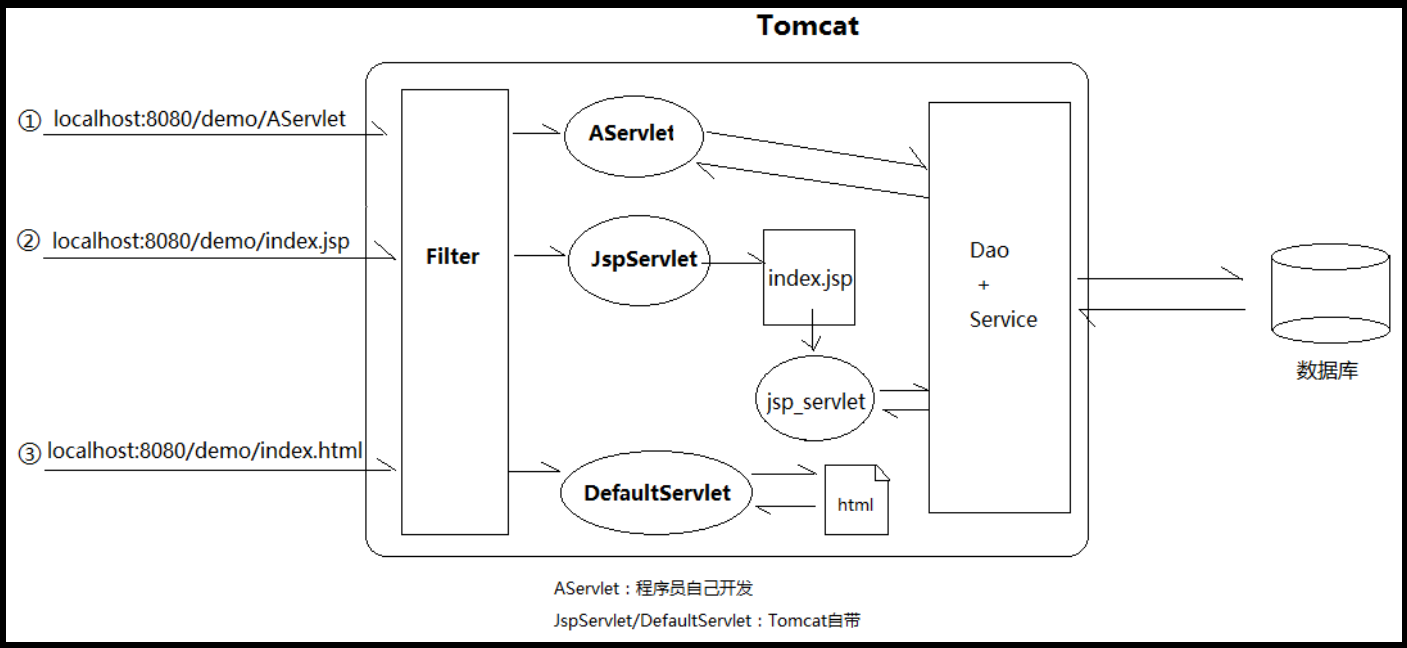
现在，让我们把 HelloWorld.class 复制到 <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes 中，并在位于 <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/ 的 web.xml 文件中创建以下条目：

```
  <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>HelloWorld</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/HelloWorld</url-pattern>
    </servlet-mapping>
```

Apache Tomcat 是一款 Java Servlet 和 JavaServer Pages 技术的开源软件实现，可以作为测试 Servlet 的独立服务器，而且可以集成到 Apache Web 服务器。





注入+回调

其实，编程学习越往后越是如此，我们能做的其实很有限。大部分工作，框架都已经帮我们做了。

Edwin Xu

很多时候，框架就像一个傀儡师，我们写的程序是傀儡，顶多就是给傀儡化化妆、打扮打扮，实际的运作全是傀儡师搞的。

CGI 与 Servlet 的区别和联系

1．定义：

CGI(Common Gateway Interface 公共网关接口)是 HTTP 服务器与你的或其它机器上的程序进行“交谈”的一种工具，其程序须运行在网络服务器上。

2．功能：

绝大多数的 CGI 程序被用来解释处理来自表单的输入信息，并在服务器产生相应的处理，或将相应的信息反馈给浏览器。CGI 程序使网页具有交互功能。

3．运行环境：

CGI 程序在 UNIX 操作系统上 CERN 或 NCSA 格式的服务器上运行。 在其它操作系统（如：windows NT 及 windows95 等）的服务器上 也广泛地使用 CGI 程序，同时它也适用于各种类型机器。

4．CGI 处理步骤：

⑴通过 Internet 把用户请求送到服务器。

⑵服务器接收用户请求并交给 CGI 程序处理。

⑶CGI 程序把处理结果传送给服务器。

⑷服务器把结果送回到用户。

Edwin Xu

Servlet 是一种服务器端的 Java 应用程序，具有独立于平台和协议的特性,可以生成动态的 Web 页面。 它担当客户请求（Web 浏览器或其他 HTTP 客户程序）与服务器响应（HTTP 服务器上的数据库或应用程序）的中间层。 Servlet 是位于 Web 服务器内部的服务器端的 Java 应用程序，与传统的从命令行启动的 Java 应用程序不同，Servlet 由 Web 服务器进行加载，该 Web 服务器必须包含支持 Servlet 的 Java 虚拟机。

工作模式：客户端发送请求至服务器；服务器启动并调用 Servlet，Servlet 根据客户端请求生成响应内容并将其传给服务器；服务器将响应返回客户端。

Java Servlet 与 CGI (Common Gateway Interface 公共网关接口)的比较：

与传统的 CGI 和许多其他类似 CGI 的技术相比，Java Servlet 具有更高的效率，更容易使用，功能更强大，具有更好的可移植性，更节省投资。在未来的技术发展过程中，Servlet 有可能彻底取代 CGI。

在传统的 CGI 中，每个请求都要启动一个新的进程，如果 CGI 程序本身的执行时间较短，启动进程所需要的开销很可能反而超过实际执行时间。而在 Servlet 中，每个请求由一个轻量级的 Java 线程处理(而不是重量级的操作系统进程)。

在传统 CGI 中，如果有 N 个并发的对同一 CGI 程序的请求，则该 CGI 程序的代码在内存中重复装载了 N 次；而对于 Servlet，处理请求的是 N 个线程，只需要一份 Servlet 类代码。在性能优化方面，Servlet 也比 CGI 有着更多的选择。

*** 方便**

Servlet 提供了大量的实用工具例程，例如自动地解析和解码 HTML 表单数据、读取和设置 HTTP 头、处理 Cookie、跟踪会话状态等。

*** 功能强大**

在 Servlet 中，许多使用传统 CGI 程序很难完成的任务都可以轻松地完成。例如，Servlet 能够直接和 Web 服务器交互，而普通的 CGI 程序不能。Servlet 还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

*** 可移植性好**

Servlet 用 Java 编写，Servlet API 具有完善的标准。因此，为 IPlanet Enterprise Server 写的 Servlet 无需任何实质上的改动即可移植到 Apache、Microsoft IIS 或者 WebStar。几乎所有的主流服务器都直接或通过插件支持 Servlet。

Edwin Xu