**Servlet 是什么？**

Java Servlet是运行在Web服务器或应用服务器上的程序, 它是作为来自Web浏览器或其他HTTP客户端的请求和HTTP服务器上的数据库或应用程序之间的中间层。

使用Servlet, 您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 通常情况下与使用 CGI（Common Gateway Interface，公共网关接口）实现的程序可以达到异曲同工的效果。

但是相比于CGI, Servlet有以下几点优势:

性能明显更好;

Servlet在Web服务器的地址空间内执行, 这样它就没有必要再创建一个单独的进程来处理每个客户端请求;

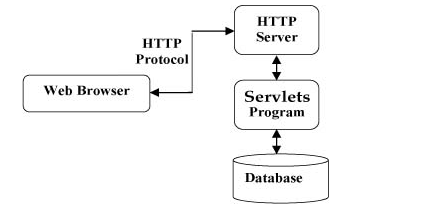
Servlet是独立于平台的, 因为它们是用 Java 编写的;

服务器上的Java安全管理器执行了一系列限制, 以保护服务器计算机上的资源, 因此, Servlet是可信的;

Java类库的全部功能对Servlet来说都是可用的, 它可以通过sockets和RMI机制与applets、数据库或其他软件进行交互。

## Servlet 架构

下图显示了 Servlet 在 Web 应用程序中的位置。



## Servlet 任务

Servlet 执行以下主要任务:

(1).读取客户端(浏览器)发送的显式的数据, 这包括网页上的HTML表单, 或者也可以是来自applet或自定义的HTTP客户端程序的表单;

(2).读取客户端(浏览器)发送的隐式的HTTP请求数据, 这包括cookies、媒体类型和浏览器能理解的压缩格式等等;

(3).处理数据并生成结果, 这个过程可能需要访问数据库, 执行RMI或CORBA调用, 调用Web服务, 或者直接计算得出对应的响应;

(4).发送显式的数据(即文档)到客户端(浏览器), 该文档的格式可以是多种多样的, 包括文本文件(HTML或XML）、二进制文件(GIF图像)、Excel等。

(5).发送隐式的HTTP响应到客户端(浏览器), 这包括告诉浏览器或其他客户端被返回的文档类型(例如HTML), 设置cookies和缓存参数, 以及其他类似的任务.

## Servlet包

Java Servlet是运行在带有支持Java Servlet规范的解释器的web服务器上的Java类;

Servlet可以使用**javax.servlet** 和 **javax.servlet.http**包创建, 它是Java企业版的标准组成部分, Java企业版是支持大型开发项目的Java类库的扩展版本。

这些类实现 Java Servlet 和 JSP 规范。

在写本教程的时候, 二者相应的版本分别是Java Servlet 2.5和JSP 2.1。

Java Servlet 就像任何其他的 Java 类一样已经被创建和编译。

在您安装Servlet包并把它们添加到您的计算机上的Classpath类路径中之后, 您就可以通过JDK的Java编译器或任何其他编译器来编译Servlet。

## 下一步呢？

接下来, 本教程会带你一步一步地设置您的Servlet环境, 以便开始后续的Servlet使用。

因此, 请系紧您的安全带, 随我们一起开始Servlet的学习之旅吧, 相信您会很喜欢这个教程的!!!

Servlet 创有三种方式。

(1).**实现 Servlet 接口**

因为是实现Servlet接口, 所以我们需要实现接口里的方法。

下面我们也说明了Servlet的执行过程, 也就是Servlet的生命周期。

// Servlet的生命周期:从Servlet被创建到Servlet被销毁的过程

// 一次创建，到处服务

// 一个Servlet只会有一个对象，服务所有的请求

/\*

\* 1.实例化（使用构造方法创建对象）

\* 2.初始化 执行init方法

\* 3.服务 执行service方法

\* 4.销毁 执行destroy方法

\*/

public class ServletDemo1 implements Servlet {

//public ServletDemo1(){}

//生命周期方法:当Servlet第一次被创建对象时执行该方法,该方法在整个生命周期中只执行一次

public void init(ServletConfig arg0) throws ServletException {

System.out.println("=======init=========");

}

//生命周期方法:对客户端响应的方法,该方法会被执行多次，每次请求该servlet都会执行该方法

public void service(ServletRequest arg0, ServletResponse arg1)

throws ServletException, IOException {

System.out.println("hehe");

}

//生命周期方法:当Servlet被销毁时执行该方法

public void destroy() {

System.out.println("\*\*\*\*\*\*destroy\*\*\*\*\*\*\*\*\*\*");

}

//当停止tomcat时也就销毁的servlet。

public ServletConfig getServletConfig() {

return null;

}

public String getServletInfo() {

return null;

}

}

(2).**继承 GenericServlet 类**

它实现了Servlet接口除了service的方法, 不过这种方法我们极少用。

public class ServletDemo2 extends GenericServlet {

@Override

public void service(ServletRequest arg0, ServletResponse arg1)

throws ServletException, IOException {

System.out.println("heihei");

}

}

(3).**继承 HttpServlet 方法**

public class ServletDemo3 extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

System.out.println("haha");

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

System.out.println("ee");

doGet(req,resp);

}

}

创建Servlet的第三种方法, 也是我们经常用的方法。

这里只简单讲Servlet的三种创建方式, 关于更详细的应用我们后面再说。