一个简单的请求到响应的过程

1、一个客户端，通常是浏览器或者一些应用发送请求到服务器地址，比如访问一个 Url 或者其他的东西。而我们的服务器通常要分为两个部分，一部分是服务器硬件，有了硬件之后还要有硬件上对应运行的软件。即服务器软件

2、其次，服务器的硬件部分接收到了这一段请求（URL），将其递交给对应的进程，服务器软件。此时这个服务器软件即为 Web 服务器，比如说 Apache。

3、再然后，这些 Web 服务器本身不一定具备提供动态页面的访问能力，所以对 jsp 或者一些其他的访问需要某些东西来辅助实现，这个东西类似于插件（客观来讲 Tomcat 不是 Apache 的一个插件，在这里暂且理解为插件）。在本例中，这个插件就是 Tomcat。

4、Tomcat 是一个运行环境，或者叫容器。Tomcat 负责实例化 jsp 并且处理请求（通俗的讲就是“运行”jsp，得到对应的响应信息），并且返回一个响应到 apache，apache 再调度硬件资源，继续将响应返回到客户端。

5、至此，一个响应完成。

服务器：我们常常提到的服务器从硬件角度上说就是一台高性能的 Computer。我们通常指的服务器其实应该是装有能够处理具体请求事务的服务器软件的 Computer。比如最常见的 www 服务器、mail 服务器、计费服务器、ftp 服务器等等。很多时候人们常把诸如 Tomcat、IIS、Weblogic 也称之为 web 服务器，其实这些只是用于开发、集成、部署和管理 Web 应用、网络应用和数据库应用的应用服务器软件。

Web 服务器：web 服务器主要作用是处理客户请求，并作出响应。当浏览器发送一个 HTTP 请求到 web 服务器，web 服务器解析请求，在内部做处理，返回一个 HTTP 响应给浏览器，浏览器解析此响应，并作出相应动作（例如：显示 HTML 页面，展示图片等等。）

浏览器和 web 服务器之间主要是通过 http 协议来进行交互。

Web 服务器的限制：web 服务器擅长提供静态页面。静态页面只是原封不动的呆在目录中，服务器找到静态页面，并把它原封不动的传回给客户，每个客户看到的东西都一样。但如果想要提供动态页面和动态数据，例如，淘宝网站中实时更新的商品，不同用户购物车中的商品，在结算后写入数据库的数据等等。web 服务器就显得有些力不从心了，这时就需要一个辅助应用，能够生成动态页面，而且这个应用能与 web 服务器通信，并且能够和后端 java 语言进行交互，从而从数据库存取数据。这个辅助应用就是 servlet。

辅助应用 servlet：servlet 是 sun 公司提供的一门用于开发动态 web 资源的技术，我们一般把实现了 servlet 接口的 java 类也称之为 servlet。在 MVC 设计模式中，servlet 扮演着 C（控制器）的角色，当客户请求到来，进入到 servlet，servlet 调用 M（业务逻辑）从数据库中存取数据，并返回页面或参数给浏览器。

但 servlet 也需要帮助。当请求到来时，必须有人加载、初始化和实例化 Servlet，或者创建或分配一个新的线程处理这个请求，调用 servlet 的 doGet（）、doPost（）方法。并需要有人创建 servlet 必须的参数 HTTPServletRequest 和 HTTPServletResponse。当请求处理结束后，有人销毁 servlet，管理着 servlet 的生命周期。这个人就是 web 容器。

Web 容器：servlet 没有 main()方法，它们受控于另一个 Java 应用，这个 Java 应用称为容器。

Tomcat 就是这样一个容器，如果 web 服务器应用（如 Apache）得到一个指向某 servlet 的请求（而不是其他请求，如请求一个普通的静态页面），此时服务器不是把这个请求交给 servlet 本身，而是交给部署该 servlet 的容器，要由容器向 servlet 提供 HTTP 请求和响应（对象），而且要由容器调用 servlet 的方法，如 doGet()和 doPost()。

容器能提供什么？

（1） 通信支持

利用容器提供的方法，你能轻松地让 servlet 与 web 服务器对话。无需自己建立 ServerSocket、监听端口、创建流等等。容器知道自己与 web 服务器之间的协议，所以你的 servlet 不必担心 web 服务器（Apache）和你自己的 web 代码之间的 API。你要考虑的只是如何在 servlet 中实现业务逻辑。

（2）生命周期管理

容器控制着 servlet 的生与死。它会负责加载类、实例化和初始化 servlet、调用 servlet 方法、并使 servlet 对象能够被垃圾回收。有了容器的控制，你就不必考虑太多的资源管理了。

（3）多线程支持

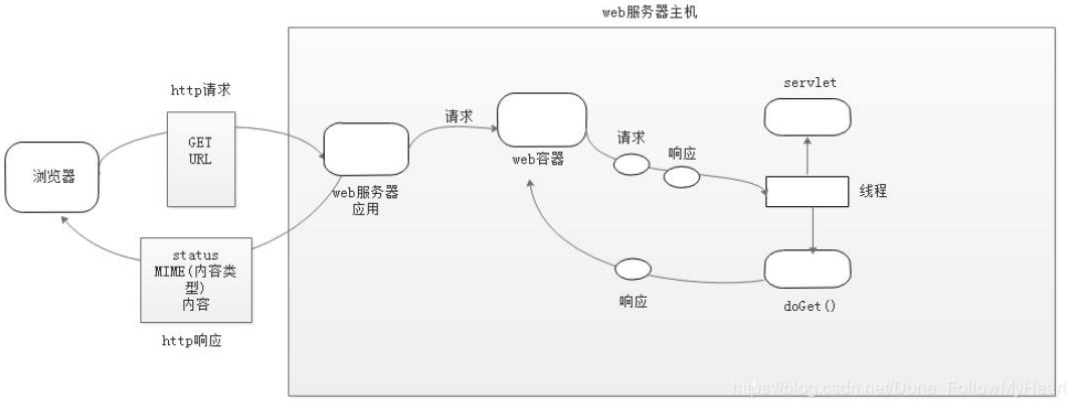
容器会自动为它接受的每个 servlet 创建一个新的线程。针对客户的请求，如果 servlet 已经运行完相应的 HTTP 服务方法，这个线程就会结束（也就是会死掉）。这并不是说不用考虑线程安全性，还是会遇到同步问题的。不过，由服务器创建和管理多个线程来处理多个请求，这样确实能让你少做很多工作。

（4）声明方式实现安全

利用容器，可以使用 xml 部署描述文件来配置（和修改）安全性，而不必将其硬编码写到 servlet（或其他类代码中）。不用去修改你的 java 源文件，也不用重新编译，你就能管理和修改安全性配置。

（5）JSP 支持

JSP 能提供动态页面，jsp 会被翻译为 java 代码，写入到响应流中，web 服务将响应对象转换为 HTTP 响应并返回给浏览器，浏览器解析 HTTP 响应，进而形成动态页面。而 web 容器负责将 jsp 翻译为 java 代码。



图解分析：

（1）用户点击一个链接，其 URL 指向一个 servlet 而不是静态页面。

（2）请求到达 web 服务器，web 服务器识别该请求为 servlet 请求，将请求送到 web 容器。

（3）容器识别出该请求要的是一个 servlet，所以容器创建 2 个对象 HTTPServletRequest 和 HTTPServletResponse。

（4）容器根据请求中的 URL 找到正确的 servlet，为这个请求创建或分配一个线程，并把请求对象和响应对象传给这个线程。

（5）容器调用 servlet 的 service（）方法。根据请求的不同类型，service（）会调用 doGet（）或 doPost（）方法。

（6）doGet（）或 doPost（）方法生成动态页面或数据，并把动态页面或数据写入到响应对象。

（7）线程结束，容器将响应对象转换为一个 HTTP 响应，把它发回给浏览器，然后删除请求和响应对象。

web 容器和 web 服务器的区别和联系：

我们常常将 web 容器和 web 服务器和概念和作用混合，对这 2 个应用总是很模糊。web 容器（Tomcat）和 web 服务器（Apache）都可以作为一个独立的应用。web 服务器擅长处理 HTTP 服务，但它不能处理动态页面并和数据库进行交互。这些功能 web 容器都可以做到，但 web 容器没有 web 服务器更擅长处理 HTTP 服务，所以常见的 HTTPweb 服务器应用经常会结合使用 Apache 和 Tomcat，充分发挥 2 者的长处，Apache 作为 HTTPweb 服务器，Tomcat 作为 web 容器。这就是为什么我们下载的 tomcat 的全名为“ apache-tomcat-版本号 ”的原因所在。

中间件：提供系统软件和应用软件之间连接的软件，以便于软件各部件之间的沟通。中间件处在操作系统和更高一级应用程序之间。他充当的功能是：将应用程序运行环境与操作系统隔离，从而实现应用程序开发者不必为更多系统问题忧虑，而直接关注该应用程序在解决问题上的能力 。容器就是中间件的一种。

常见的 web 服务器：Unix 和 Linux 平台下常用的服务器有 Apache、Nginx、Lighttpd、Tomcat、IBM WebSphere 等，其中应用最广泛的是 Apache。而 Window 平台下最常用的服务器是微软公司的 IIS。