Java服务端开发与Servlet

时间 2014-11-22 23:58:36  [三石·道](http://www.tuicool.com/sites/AF7juu)

原文  [http://www.molotang.com/articles/1749.html](http://www.molotang.com/articles/1749.html?utm_source=tuicool&utm_medium=referral)

主题 [Servlet](http://www.tuicool.com/topics/11070064)[Web服务器](http://www.tuicool.com/topics/11000096)

由于最近相当长一段时间都是忙于iOS相关的开发，很久没有整理Java的文章了。iOS接触得久了，有时候发现有些东西和之前了解到的Java很相似，或关系密切，可以互相借鉴。这很有意思，正所谓“他山之石，可以攻玉”，无论哪门语言，很多技术都是相通的。 接下来一段时间，会将过去所做过的东西和零散的技术笔记整理出来，晒在Blog上和大家分享交流。如果能帮到新从事这个领域开发的朋友，那自然是更好不过了。

之前Java的文章已经把Java语言本身特点的基本东西整理过了，接下来的文章主要则是与Java具体应用相关的技术方面。

根据本人参与开发的经历，我认为Java当前应用最广泛的领域应当算Web网站应用和后端应用服务相关的开发。而在这些开发重一个很基本的技术点，那就就是Java的Servlet。说它是最基本的技术点的原因就在于，它在Socket等Java网络通信的基础上，把IO、并发等技术内容整合并包装起来，对业务逻辑提供了上层抽象的API，满足了很多复杂的应用场景，是Java EE的重要组成部分。现在的Spring MVC、Struts也都是在Servlet之上封装起来。 **这里，我就 从对Java Servlet的理解开始整理起来。** 这也为后面整理其它Java技术文章做一个铺垫。

**0. 起源**

通过本人小站早期的一些写Java的文章，我们知道，大约在1995年前后，Java正式向这个世界say hello。1995年那个时候，互联网也已经逐步流行起来，面向对象语言的概念也很时髦，Java应运而生。那个时候，Java有个东西叫applet，就是在浏览器端运行的应用。

Applet这个想法是好的，但是想通过它来填平各个平台的差异是太难了。 比如有一次，微软在IE中用了一套未遵循规范的浏览器支持，使得很多开发出来的applet只能在微软平台上正常运行。虽然Sun当时起诉了微软也赢得了诉讼，但我们知道，由于各种各样的原因，applet这种浏览器端形式的应用并没有真正成功起来。

“东方不亮西方亮”，Java的applet在浏览器开发领域没有流行起来，但是Java在服务端的应用越来越广泛，并成为主流。而相对浏览器端的applet，服务端也有类似的概念，那正是servlet。最终在1997年，Servlet API1.0发布。

**1. 什么是Java Servlet**

互联网刚刚兴起时，一个很主要的应用就是使用浏览器，通过HTTP协议，访问互联网上的超文本内容资源。提供超文本内容资源的计算机可以认为是服务器，而在这个服务器上还有一个专门负责提供服务的软件，也叫“服务器”（Server）。当前比较主流的服务器，基本上就是Apache http server和Nginx了。微软有过自己的IIS，现在的一些大公司也都有自己定制化过的服务器，比如淘宝的Tengien、百度的BWS。

最初，这些Http server不过是把一些固定的（静态的）的文档放在上面，供用户访问和查阅。可是，随着需求的变化、技术的发展，静态资源已经远远不够了。于是服务器端需要根据不同的访问需求返回对应的不同的结果。这就是程序需要做的事儿了！我们可以用各种开发工具各种编程语言来满足这个需求，只要能输出符合html格式的字符串，最后通过CGI脚本返回给Http server，再由Http server返回给浏览器UA。

刚刚提到了生成超文本Web内容可以使用各种语言，Java当然也包括在内。而Java在服务端处理服务器请求的部分，就是Java servlet。

可以认为Java是在Http Server后面的支持CGI的动态内容生成程序中的一种语言。但Java提供的技术方案不仅如此，相比于Perl等，Java提供的方案不必为每一个请求创建一个重量级的进程，自然就更高效。Java语言本身的语法特性，也降低了开发成本。

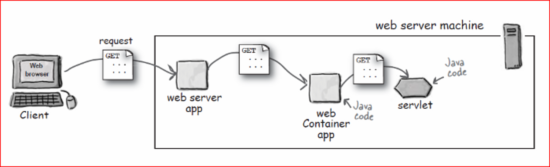
这里需要注意的是，Servlet只是服务器端的应用中处理具体业务逻辑的部分，而Servlet本身是需要生存在一个大的环境当中的，那就是Servlet容器。

Java的Servlet API约定了一个统一的接口层，其下是Servlet容器需要提供的公共服务，其上则是对应具体业务逻辑的Servlet实现。Java Servlet 和其它技术一起，构成了Java EE，Servlet API是Java EE API重要的一部分。下面就说说这个Servlet容器。

**2. Servlet容器**

Servlet容器，就是Servlet的生存和运行环境。对于一个Java之外的CGI程序来说，通常是在一个请求来临时开启一个进程来利用系统资源处理请求，给出响应，但Java的Servlet则不是，它会一直“保持”对请求的高度关注，必要时用一个线程来解决请求。

简单来想，我们一个简单的基于Java SE API的程序，只需要把逻辑写在main方法里，直接执行，到任务完成为止，main方法退出。这是一个线性的执行过程。而如果时时保持对请求的及时响应，显然这个程序是没有退出执行的，是一个非线性的，类似于环形的执行过程。而这个非线性的运行着的环境，就是Servlet Container提供的，普通的Servlet则不需要关注这些，只需要提供具体的业务内容处理逻辑即可。



Servlet容器

对于一个典型的Servlet容器实现，比如Tomcat，就是用了一个while(running)循环保持着持续运行，请求来了的时候呼唤Servlet。但这只是Servlet容器提供Servlet运行环境的一个方面而已。

刚刚说容器在Servlet API之下提供了基础服务，那我们就详细看它提供了什么样的服务：

* 通信支持。不管是作为Http server的一个后台支持也好，还是直接面对浏览器请求，网络通信都是基于socket进行的，而这些通信的具体细节，都由Servlet容器来负责的，Servlet无需关注。
* 生命周期管理。上面提到的例子就是生命周期管理的一个方面体现，除此之外Servlet的加载、实例化也都有容器统一管理。
* 多线程并发。作为一个比较权威的网站，可想其访问量会是相当大的，如果没有良好的并发支持，用户使用浏览器进行访问必然不能得到很好的响应。而无论是BIO，还是NIO，抑或AIO，都是与服务器操作系统相关的，与具体逻辑相关度不大，Servlet容器也将这个公共部分抽取出来进行实现。
* 安全。
* 除此之外，有些容器也要对 JSP技术进行支持。

Tomcat是一个典型的Servlet容器实现，目前已经到了8.0版本，包罗了各项功能，可谓成熟、完善。其它的实现也有Jetty、Jboss等。

（接下来如果有时间有机会，我会以Tomcat源码为例整理一篇文章出来，分析下Servlet容器是如何实现的）

**3. Servlet API**

Servlet API是Java EE的重要组成部分，规定了Servlet和其容器间的接口。Java是开放的，JCP只是规定Servlet API的规范，无论是Servlet还是容器，只要符合API，就可以有各自不同的实现。

Servlet API在1997年发布1.0，到目前已有最新的3.1版本，本人当时做服务端开发接触较多的还是2.5。

Oracle最新的Servlet API文档是和整个Java EE 7 放在一起的，可参看：

[***https://docs.oracle.com/javaee/7/api/***](https://docs.oracle.com/javaee/7/api/)

此外，Tomcat也单独提供了Servlet API部分的文档，版本3.1：

[***http://tomcat.apache.org/tomcat-8.0-doc/servletapi/***](http://tomcat.apache.org/tomcat-8.0-doc/servletapi/)

这里面有4个Java package，而实际上我们主要关注的是其中两个包：

* javax.servlet
* javax.servlet.http

其中前者定义了Servlet相关的很多通用类，后者则是基于http应用层面的实现。这两个包里面定义了这样几个主要概念：

* Servlet，一个为请求服务，并返回响应的处理具体业务逻辑单元。
* Filter，顾名思义，在Servlet前后进行过滤处理操作的单元，多个Filter依次序形成一个过滤器链，请求资源完成前依次执行其doFilter()前的逻辑，请求后反序执行过滤器链上doFilter()之后的逻辑。
* 各类Event及Listener。不考虑版本3之后增加的异步支持外，则围绕着Request、Servlet Context以及HttpSession提供了6类Event和8类Listener，这些API对于基于Servlet进行的开发设计和实现都有着很重要的意义。

这几项通常需要在web.xml中配置好，一起在Servlet容器中生效。

在Servlet API3.0以后，增加了servlet处理过程中的异步支持、Servlet配置注解化等新特性，更详细的可参考IBM developer上的这篇技术日志：

[***Servlet 3.0 新特性详解***](http://www.ibm.com/developerworks/cn/java/j-lo-servlet30/)

**4. JSP**

最后再介绍下JSP。Servlet虽然很好的替代了传统的CGI，解决了服务端的需求，但由于Web超文本内容大多都是遵循HTML的标签格式。这样，Servlet中大部分都是字符串输出处理，而且在代码中大部分出现HTML标签常量，开发起来并不是很方便。同时，微软也在自家的IIS里加入了ASP技术。

在这样一种情况下，Java也考虑将HTML标签常量等“静态”地放置于页面里，而将需要动态处理的代码部分也以特殊的形式置入其中。并在页面中，内置一些常用对象，方便页面开发中使用。

这就是JSP。Sun在1999年发布了JSP技术标准第一版。

然而，现如今JSP由于将页面内容和业务逻辑缠绕在一起，在很多场景下并不是最优选择。一些页面的合成的工作有被Freemarker和Velocity等取代的趋势。

关于Java Servlet和JSP相关的内容，就先整理到这。