IIS 内部运行机制

2017-09-13 ※森林小居※ [DotNET攻城狮](https://mp.weixin.qq.com/s?__biz=MzI1ODAwMjA1OA==&mid=2247483798&idx=1&sn=251d38b2a6d33bfe75aed5f28d75153a&chksm=ea0f9c73dd7815653aff67aaf1daadb8a0801ae819cd4e5d263a7c985e28b4ac6417592bb1ec&mpshare=1&scene=1&srcid=1111KLivuMaNVImMuQNZyVT9&key=cf67e8f56e0f00748c77c95bb0afe81bbff634f8b37a4ccc6b1c8e5021206f0843ebbbfa850423c4d2cb3976bceba93888725a921edbc4964149940d4cf0c711c2fef3b64c1854cf2baca5eca1f39953&ascene=1&uin=Mjg1Mzg5MzU%3D&devicetype=Windows+7&version=62060028&pass_ticket=nPnGh5Pqw267BuzPN%2FJfJv8nOXExtWwPTjqGXgvW0DQ%3D&winzoom=1" \l "#)

　　ASP.NET是一个非常强大的构建Web应用的平台，它提供了极大的灵活性和能力以致于可以用它来构建所有类型的Web应用。

　　绝大多数的人只熟悉高层的框架如： WebForms 和 WebServices — 这些都在ASP.NET层次结构的最高层。

　　这篇文章的资料收集整理自各种微软公开的文档，通过比较 IIS5、IIS6、IIS7 这三代 IIS 对请求的处理过程， 让我们熟悉 ASP.NET的底层机制并对请求(request)是怎么从Web服务器传送到ASP.NET运行时有所了解。通过对底层机制的了解，可以让我们对 ASP.net 有更深的理解。

**IIS 5 的 ASP.net 请求处理过程**

　　对图的解释：

　　IIS 5.x 一个显著的特征就是 Web Server 和真正的 ASP.NET Application 的分离。作为 Web Server 的IIS运行在一个名为 InetInfo.exe 的进程上，InetInfo.exe 是一个Native Executive，并不是一个托管的程序，而我们真正的 ASP.NET Application 则是运行在一个叫做 aspnet\_wp 的 Worker Process 上面，在该进程初始化的时候会加载CLR，所以这是一个托管的环境。

　　ISAPI： 指能够处理各种后缀名的应用程序。 ISAPI 是下面单词的简写 ：Internet Server Application Programe Interface，互联网服务器应用程序接口。

　　IIS 5 模式的特点：

　　1、首先，同一台主机上在同一时间只能运行一个 aspnet\_wp 进程，每个基于虚拟目录的 ASP.NET Application 对应一个 Application Domain ，也就是说每个 Application 都运行在同一个 Worker Process 中，Application之间的隔离是基于 Application Domain 的，而不是基于Process的。

　　2、其次，ASP.NET  ISAPI 不但负责创建 aspnet\_wp Worker Process，而且负责监控该进程，如果检测到 aspnet\_wp 的 Performance 降低到某个设定的下限，ASP.NET  ISAPI 会负责结束掉该进程。当 aspnet\_wp 结束掉之后，后续的 Request 会导致 ASP.NET ISAPI 重新创建新的 aspnet\_wp Worker Process。

　　3、最后，由于 IIS 和 Application 运行在他们各自的进程中，他们之间的通信必须采用特定的通信机制。本质上 IIS 所在的 InetInfo 进程和 Worker Process 之间的通信是同一台机器不同进程的通信（local interprocess communications），处于 Performance 的考虑，他们之间采用基于 Named pipe 的通信机制。ASP.NET ISAPI 和 Worker Process 之间的通信通过他们之间的一组 Pipe 实现。同样处于 Performance 的原因，ASP.NET ISAPI 通过异步的方式将 Request 传到 Worker Process 并获得 Response，但是 Worker Process 则是通过同步的方式向 ASP.NET ISAPI 获得一些基于 Server 的变量。

**IIS6 的 ASP.net 请求处理过程**

　　对图的解释：

　　IIS 5.x 是通过 InetInfo.exe 监听 Request 并把 Request 分发到Work Process。换句话说，在 IIS 5.x 中对 Request 的监听和分发是在 User Mode 中进行，在IIS 6中，这种工作被移植到 Kernel Mode中 进行，所有的这一切都是通过一个新的组件 — http.sys 来负责。

　　注：为了避免用户应用程序访问或者修改关键的操作系统数据，Windows 提供了两种处理器访问模式：用户模式（User Mode）和内核模式（Kernel Mode）。一般地，用户程序运行在 User mode 下，而操作系统代码运行在 Kernel Mode 下。Kernel Mode 的代码允许访问所有系统内存和所有CPU指令。

　　在 User Mode 下，http.sys 接收到一个基于 aspx 的 http request，然后它会根据 IIS 中的 Metabase 查看基于该 Request 的 Application 属于哪个 Application Pool， 如果该 Application Pool 不存在，则创建之。否则直接将 request 发到对应 Application Pool 的 Queue中。

　　每个 Application Pool 对应着一个 Worker Process — w3wp.exe，毫无疑问他是运行在 User Mode 下的。在 IIS Metabase 中维护着 Application Pool 和 Worker Process 的Mapping。WAS（Web Administrative Service）根据这样一个 mapping，将存在于某个 Application Pool Queue 的 request 传递到对应的 Worker Process (如果没有，就创建这样一个进程)。在 Worker Process 初始化的时候，加载 ASP.NET ISAPI，ASP.NET ISAPI 进而加载 CLR。最后的流程就和 IIS 5.x 一样了：通过 AppManagerAppDomainFactory 的 Create 方法为 Application 创建一个 Application Domain；通过 ISAPIRuntime 的  ProcessRequest 处理 Request，进而将流程进入到 ASP.NET Http Runtime Pipeline。

**IIS 7  的 ASP.net 请求处理过程**

　　IIS7 站点启动并处理请求的步骤如下图：

　　步骤 1 到 6 ，是处理应用启动，启动好后，以后就不需要再走这个步骤了。

　　上图的8个步骤分别如下：

　　1、当客户端浏览器开始 HTTP 请求一个WEB 服务器的资源时，HTTP.sys 拦截到这个请求。

　　2、HTTP.sys 联系 WAS 获取配置信息。

　　3、WAS 向配置存储中心(applicationHost.config)请求配置信息。

　　4、WWW 服务接收到配置信息，配置信息指类似应用程序池配置信息，站点配置信息等等。

　　5、WWW 服务使用配置信息去配置 HTTP.sys 处理策略。

　　6、WAS starts a worker process for the application pool to which the request was made.

　　7、The worker process processes the request and returns a response to HTTP.sys.

　　8、客户端接受到处理结果信息。

　　W3WP.exe 进程中又是如果处理的呢？ IIS 7 的应用程序池的托管管道模式分两种： 经典和集成。这两种模式下处理策略各不相同。

**IIS 6 以及 IIS7 经典模式的托管管道的架构**

　　在 IIS7 之前，ASP.NET 是以 IIS ISAPI extension 的方式外加到 IIS，其实包括 ASP 以及 PHP，也都以相同的方式配置（PHP 在 IIS 采用了两种配置方式，除了 IIS ISAPI extension 的方式，也包括了 CGI 的方式，系统管理者能选择 PHP 程序的执行方式），因此客户端对 IIS 的 HTTP 请求会先经由 IIS 处理，然后 IIS 根据要求的内容类型，如果是 HTML 静态网页就由 IIS 自行处理，如果不是，就根据要求的内容类型，分派给各自的 IIS ISAPI extension；如果要求的内容类型是 ASP.NET，就分派给负责处理 ASP.NET 的 IIS ISAPI extension，也就是 aspnet\_isapi.dll。下图是这个架构的示意图。

　　IIS 7 应用程序池的托管管道模式“经典”模式也是这样的工作原理。这种模式是兼容 IIS 6 的方式， 以减少升级的成本。

　　IIS6 的执行架构图，以及 IIS7  应用程序池配置成经典模式的执行架构图

**IIS  7 应用程序池的托管管道模式 — 集成模式**

　　而 IIS 7 完全整合 .NET 之后，架构的处理顺序有了很大的不同（如下图），最主要的原因就是 ASP.NET 从 IIS 插件（ISAPI extension）的角色，进入了 IIS 核心，而且也能以 ASP.NET 模块负责处理 IIS 7 的诸多类型要求。这些 ASP.NET 模块不只能处理 ASP.NET 网页程序，也能处理其他如 ASP 程序、PHP 程序或静态 HTML 网页，也因为 ASP.NET 的诸多功能已经成为 IIS 7 的一部份，因此 ASP 程序、PHP 程序或静态 HTML 网页等类型的要求，也能使用像是 Forms 认证（Forms Authentication）或输出缓存（Output Cache）等 ASP.NET 2.0 的功能（但须修改 IIS 7 的设定值）。也因为 IIS 7 允许自行以 ASP.NET API 开发并加入模块，因此 ASP.NET 网页开发人员将更容易扩充 IIS 7 和网站应用程序的功能，甚至能自行以 .NET 编写管理 IIS 7 的程序（例如以程序控制 IIS 7 建置网站或虚拟目录）。

IIS 7 的执行架构图（集成托管信道模式下的架构）

**小结**

　　IIS5 到 IIS6 的改进，主要是 HTTP.sys 的改进。

　　IIS6 到 IIS7 的改进，主要是 ISAPI 的改进。

**参考资料：**

　　ASP.NET Process Model之一：IIS 和 ASP.NET ISAPI

　　http://www.cnblogs.com/artech/archive/2007/09/09/887528.html

　　ASP.NET Internals – IIS and the Process Model

　　http://dotnetslackers.com/articles/iis/ASPNETInternalsIISAndTheProcessModel.aspx

　　模组化的IIS 7 与.NET 能力整合

　　http://www.microsoft.com/taiwan/technet/columns/profwin/33-iis7-componentization-integration.mspx

　　Introduction to IIS 7.0 Architecture

　　http://learn.iis.net/page.aspx/101/introduction-to-iis7-architecture/