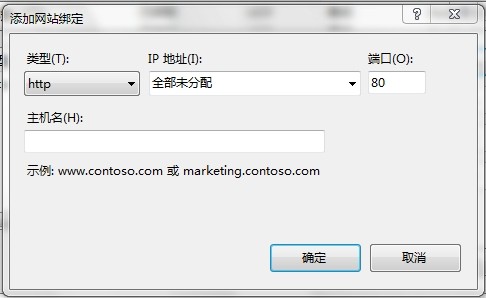
[**IIS与Asp.net**](http://www.cnblogs.com/kissdodog/p/3567228.html)

**一、IIS**

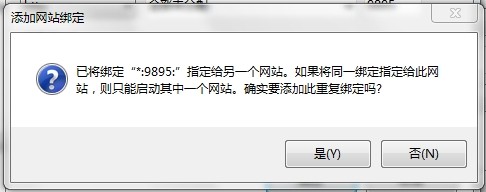
**1、绑定**

　　为了将特定的请求映射到相应的网站，IIS允许我们配置“绑定”。所谓“绑定”就是将一个特定的地址、端口号和HTTP主机名对应到特定的网站。

　　IIS7添加绑定的代码如下图所示：



　　在IIS7中，如果新设置的绑定与网站中已经存在的绑定冲突，那么将会弹出一个提示框，提示我们绑定冲突。



　　对于每个网站来说，存在一个虚拟的网站文件系统，这个网站文件系统是相对于磁盘文件系统来说的。磁盘文件系统是存在于磁盘上的操作系统管理的文件系统，而网站文件系统是独立于磁盘文件系统的另外一种文件系统。这种文件系统更加类似于Unix中的文件系统。我们通常称为虚拟目录。

　　对于每一个网站文件系统，存在一个根目录，根目录的名字永远使用(/)表示。

　　当用户访问网站目录的时候，IIS将会把网站文件系统的访问映射到磁盘文件系统中，**网站根目录映射到磁盘文件系统目录是我们在IIS中指定的。**在IIS7中，当创建新的网站的时候，我们就需要指定根目录映射的物理文件目录。



　　默认情况下，网站的根目录映射为C:\inetpub\wwwroot，也就是说，在网站中，根目录/将会被映射到文件系统的C:\inetpub\wwwroot，当我们访问网站中的/index.html的时候，实际上，IIS将会到C:\inetpub\wwwroot目录中访问index.html这个文件。在Asp.net程序中，这个映射通过HttpServerUtility方法的MapPath来完成，我们传入一个当前网站的虚拟目录，返回的结果就是映射到文件系统的文件目录。当然，这个文件或者目录可能并不存在。

**2、网站应用程序**

　　在IIS中，每一个网站都至少包含一个网站应用程序，即根网站应用程序。根据需要，一个网站中还可以包含更多的网站应用程序。

　　网站应用程序是监听网络连接并进行处理的基本单位，每个网站应用程序有一个唯一的标识，对于IIS7中的默认网站Default Web Site来说，其ID是1。在IIS中，逻辑上，所有的网站都被包含在一个同样的根目录/W3SVC之下，所以，默认的网站在IIS内部表示为/W3SVC/1,我们通常称为/网站应用程序。

　　如果我们再默认网站中创建了一个网站应用程序demo，那么这个网站应用程序在IIS内部将被表示为/W3SVC/1/Root/demo，我们通常称之为/demo网站应用程序。

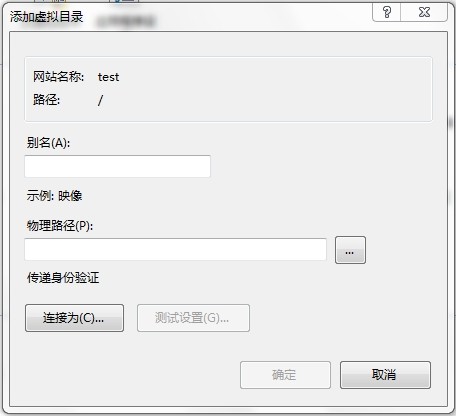
　　对于每一个网站应用程序来说，需要坚挺某个网络端口以处理客户端的请求，例如，处理来自80端口的HTTP请求就是网站的一个基本工作。

　　对于Asp.net来说，一个网站应用程序就是一个独立的应用程序单位，对应一个网站应用程序，Asp.net将会创建一个AppDomain来进行处理。所以，在Asp.net的世界里，网站应用程序被映射到一个AppDomain，应用程序域提供了在一个进程中的托管代码的隔离，更重要的是，对于在部分新人环境下的应用程序提供了网站寄宿能力。

　　在IIS7环境下，位于%windir%\system32\inetsrv\config\目录下的applicationhost.config文件定义了IIS中的网站及其应用程序。

**3、虚拟目录**

　　对于网站应用程序来说，内容的来源不仅可以来自其相应的物理文件目录，还可以通过虚拟目录来提供。这非常类似于UNIX中符号链接的概念，在IIS中，这种映射称为虚拟目录。在IIS7中，创建虚拟目录的界面如下图：



　　一个网站应用程序至少包含一个虚拟目录，这个也可以在applicationhost.config里看到。

　　网站应用程序借助于虚拟目录来获取网站中的资源。开发中所使用的虚拟路径就是基于虚拟目录的路径。

　　网站应用程序和虚拟目录可以互相嵌套，也就是说，网站应用程序中可以包含虚拟目录，虚拟目录中也可以包含网站应用程序，它们可以任意嵌套组合。

　　对于Asp.net网站来说，父网站应用程序将会影响子网站中的应用程序，父网站应用程序中的配置参数将会被子网站应用程序所继承，所引用的程序集也会被子网站所引用。

**二、通过ISAPI扩展IIS**

　　IIS是一个完整的Web服务器软件，本身就可以完成基本的Web服务任务，如果我们需要扩展IIS的话，可以通过多种方式来实现，例如CGI或者ISAPI，对于IIS服务器来说，其中最重要的就是ISAPI。

　　ISAPI是微软提供的一套面向Internet服务的基于C语言的API接口，它能实现CGI提供的全部功能，并在此基础上进行了扩展，如提供了过滤器应用程序接口等。

　　ISAPI提供了扩展Web服务器的简单而有效的方法。开发人员可以设计ISAPI服务器动态链接库(ISAPI Server Extension DLL)，它可以被HTTP服务器调用。例如，客户端用户填写了一个表单，按下“提交”按钮后，输入的数据将被传送至HTTP服务器，激活相应的ISAPI扩展应用程序，该应用程序可以处理用户的输入信息，进行相应的操作，或者按照用户的要求访问数据库，读取用户指定的数据，动态生成HTML文件，再传回客户端。

　　考虑到性能原因，ISAPI和调用它的IIS是在同一个进程中的，这使得他们彼此间可以共享某些数据，从而大大提高了相互访问的速度，但这种直接访问使得ISPAI本身出现错误时很可能也影响到IIS，导致IIS服务器瘫痪。ECB正式基于这一考虑而出现，它是一个非托管资源包，具有对ISAPI接口完整的访问能力，当IIS接收到一个Web请求后，首先调用自身的ProcessRequest函数进行处理，这个函数通过传递一个ECB的指针来间接地操作ISAPI。

　　ISAPI分为两种：ISAPI扩展和ISAPI过滤器。

**1、ISAPI扩展**

　　对于ISAPI扩展来说，ISAPI向IIS提供了三个方法的接口。

* BOOL WINAPI GetExtensionVersion(HSE\_VERSION\_INFO\*pVer);
* DWORD WINAPI HttpExtensionProc(LPEXTENSION\_CONTROL\_BLOCK lpECB);
* BOOL WINAPI TerminateExtension(DWORD dwFlags);

　　对于IIS来说，一旦装载了一个ISAPI扩展的DLL，一个用于管理ISAPI扩展的线程便启动了，在调用入口函数DLLMain之后，将会通过调用GetExtensionVersion函数来注册版本信息。如果函数返回为真，表示成功。

　　在此之后，当需要通过ISAPI扩展处理请求的时候，IIS服务器通过之前提到的ECB指针调用ISAPI扩展中的一个名为HttpExtensionProc的函数，该函数的任务是将处理过后的数据协会到客户端。

　　返回值是下列之一：

* HSE\_STATUS\_SUCCESS,扩展完成了处理，服务器应该与客户端断开连接，释放分配的资源。
* HSE\_STATUS\_SUCCESS\_AND\_KEEP\_CONN，扩展完成了处理，在客户端支持Keep-Alive的情况下，服务器等待客户端的下一个请求进行处理。
* HSE\_STATUS\_ERROR，处理出现错误。

　　ISAPI扩展的第三个操作，同时也是ISAPI执行的最后一步，就是在ISAPI扩展从内存中卸载之前执行将会TerminateExtension操作。ISAPI使用的所有的资源应该在这个函数中释放。

　　ISAPI扩展类似于Asp.net中的处理程序，效率很高，但是实现比较复杂。

**2、ISAPI过滤器**

　　ISAPI过滤器是另外一种ISAPI，在IIS内部处理请求的过程中，将处理过程分为多个处理阶段，在每个阶段将会发出一个处理的消息供我们处理，ISAPI过滤器可以响应这些消息，在相应的消息发出后，过滤器将会被调用。在过滤器中，可以监视并修改处理过程中的数据，不管是客户端发送到服务器的数据还是服务器准备返回到客户端的数据。可以使用ISAPI过滤器提供增强的HTTP请求记录、自定义加密、自定义压缩或其他身份验证方法。

　　IIS中预定义的消息按照下面的顺序依次发出：

1. SF\_NOTIFY\_READ\_RAW\_DATA，当IIS要从用户读入数据时发生。过滤器可以在IIS处理他们之前检查甚至修改用户输入的原始数据。
2. SF\_NOTIFY\_PREPROC\_HEADERS,IIS预处理HTTP请求包头后发生。过滤器可以检查修改增加包头。
3. SF\_NOTIFY\_URL\_MAP,IIS试图将URL解释为物理文件时发生。过滤器可以将请求重定向到其他的文件。
4. SF\_NOTIFY\_AUTHENTICATION,IIS试图验证用户身份时发生。过滤器可以实现自己的验证方案。
5. SF\_NOTIFY\_AUTH\_COMPLETE,当身份验证完成时发生。
6. SF\_NOTIFY\_AUTH\_RESPONSE,在IIS处理请求之后，但是任何Header发送回客户端之前发生。
7. SF\_NOTIFY\_SEND\_RAW\_DATA，当其他程序处理完，IIS准备将数据发回给用户时发生。可能多次发出。
8. SF\_NOTIFY\_END\_OF\_REQUEST，当一个HTTP请求结束时发生。
9. SF\_NOTIFY\_LOG,当IIS写记录到LOG文件时发生。过滤器可以搜集更多的信息写入记录文件中。
10. SF\_NOTIFY\_END\_OF\_NET\_SESSION,连接结束时发生。注意如果浏览器支持keep-alive，一次连接可能包含几个HTTP请求。过滤器可以用它来释放一些用户的资源。

　　对于ISAPI过滤器来说，ISAPI向IIS提供了三个方法的接口：

BOOL WINAPI GetFilterVersion(PHTTP\_FILTER\_VERSION pVer);

DWORD WINAPI HttpFilterProc(PHTTP\_FILTER\_CONTEXT pfc,DWORD notificationType,LPVOID pvNotification);

BOOL WINAPI TerminateFilter(DWORD dwFlags);

　　GetFilterVersion方法是IIS调用过滤器的第一个方法，在这个方法中，需要注册过滤器需要需要监听的事件。如果过滤器正常工作的话，必须返回真，否则，IIS不会向过滤器发送任何消息。

　　对于事件的处理在HttpFilterProc中进行，对于不同的事件，HttpFilterProc的第三个参数将会得到不同类型的数据。处理的结果为以下之一。

* SF\_STATUS\_REQ\_FINISHED,过滤器处理了请求，服务器应该断开会话。
* SF\_STATUS\_REQ\_FINISHED\_KEEP\_CONN,过滤器不支持。
* SF\_STATUS\_REQ\_NEXT\_NOTIFICATION,应该调用下一个过滤器。
* SF\_STATUS\_REQ\_HANDLED\_NOTIFICATION,过滤器处理了请求，不需要在调用其他的过滤器了。
* SF\_STATUS\_REQ\_ERROR,告诉IIS处理过程中出现了错误。
* SF\_STATUS\_REQ\_READ\_NEXT,仅用于原始数据读取通知。

　　可以看到，ISAPI非常类似于HttpModule，不过ISAPI是使用C语言或者C++完成的，虽然效率很高，但是实现起来难度较大。

**3、CLR是一个COM组件**

　　通过ISAPI可以处理HTTP请求，但是，使用C语言来创建一个网站显然是代价高昂的。然而，我们可以通过.Net来更加容易地开发网站应用程序，然后，通过COM技术，我们可以将需要处理的参数传递给.Net应用程序，最后，在非托管程序中取回托管程序处理的结果。

**4、ASP.NET中的ISAPI扩展和过滤器**

　　在Asp.net中，同样使用ISAPI来扩展IIS，在位于%windir%\Microsoft.NET\Framework\v4.0.30319的ASP.NET的系统文件夹中，包含两个特殊的文件，分别用于ISAPI扩展和ISAPI过滤器。

* aspnet\_isapi.dll
* aspnet\_filter.dll

　　过滤器用于支持无Cookie的状态管理，将迁入在URL路径中的票据去除，使得请求路径转化为正常的请求地址，将票据信息转化为请求头参数，以便于在ASP.NET中进行处理。

　　对于嵌入在URL中的标识信息，不同的前缀表示这是不同的标识。

1. S-Cookieless ticket for session state
2. A-Cookieless ticket for anonymous identification
3. F-Cookieless ticket for forms authentication

　　在经过ISAPI处理之后，嵌入在URL中的标识信息称为了请求头中参数的一部分。例如，在Cookieless状态下，嵌入在地址中的SessionID被转化为了名为AspFilterSessionId的请求头，而即使是通过HttpRequest对象的RawUrl也只能取得转换后的地址了。

过滤器的第二个作用是用来保护ASP.NET特殊的、受保护文件夹，例如App\_Code,App\_Data,App\_Browsers等。

　　ISAPI扩展器是ASP.NET的关键所在，它是接收和分派对各种ASP.NET资源请求的控制中心。该模块存在于IIS进程中，在具有管理员权限的SYSTEM账户下运行。

　　aspnet\_isapi.dll是一个非托管的动态连接库，它通过命令管道将请求转发到ASP.NET所在的辅助进程，进程中的应用程序与将对请求进行最终的处理。ISAPI和进程的通讯是使用一组命名管道进行的。命名管道是一种Win32通讯机制，用于跨进程边界传输数据。对于本地进程间通讯，管道是Windows中的最有效、最灵活的工具。为了确保获得最优性能，aspnet\_isapi使用异步命名管道来将请求转发给辅助进程并获得响应。另一方面，辅助进程在需要查询有关IIS环境的信息(服务器变量)时又使用同步管道。aspnet\_isapi模块创建固定数量的命名管道，并使用重叠的操作以通过小的线程池处理同一时间进行的连接。当通过管道进行的数据交换操作结束后，将断开客户端，并重新使用管道实例为新的客户端服务。线程池和重叠操作均可以保证使ASP.NET ISAPI的性能达到令人满意的水平。