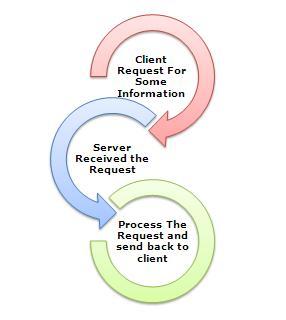
**Beginner’s Guide: How IIS Process ASP.NET Request**

### **Introduction**

When request come from client to the server a lot of operation is performed before sending response to the client. This is all about how IIS Process the request.  Here I am not going to describe the Page Life Cycle and there events, this article is all about the operation of IIS Level.  Before we start with the actual details, let’s start from the beginning so that each and everyone understand it’s details easily.  Please provide your valuable feedback and suggestion to improve this article.

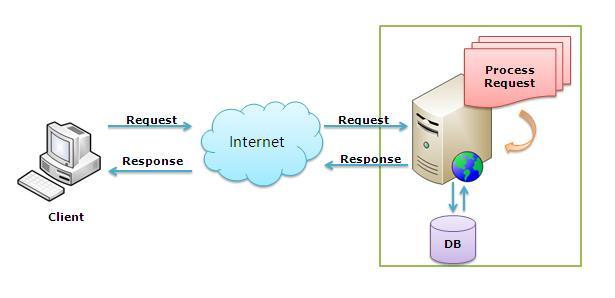
### **What is Web Server ?**

When we run our ASP.NET Web Application from visual studio IDE, VS Integrated ASP.NET Engine is responsible for executing all kind of asp.net requests and responses.  The process name is “**WebDev.WebServer.Exe**” which takes care of all request and response of a web application which is running from Visual Studio IDE.  
Now, the name “Web Server” comes into picture when we want to host the application on a centralized location and wanted to access from many places. Web server is responsible for handle all the requests that are coming from clients, process them and provide the responses.



### **What is IIS ?**

IIS (Internet Information Services) is one of the most powerful web servers from Microsoft that is used to host your ASP.NET Web application. IIS has its own ASP.NET Process Engine to handle the ASP.NET request. So, when a request comes from client to server, IIS takes that request and process it and send the response back to clients.



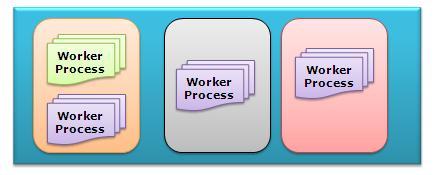
### **Request Processing :**

Hope, till now it’s clear to you that what is the Web server and IIS is and what is the use of them. Now let’s have a look how they do things internally. Before we move ahead, you have to know about two main concepts

1.    Worker Process  
2.   Application Pool

**Worker Process:** Worker Process (w3wp.exe) runs the ASP.Net application in IIS. This process is responsible for managing all the request and response that are coming from the client system.  All the ASP.Net functionality runs under the scope of the worker process.  When a request comes to the server from a client worker process is responsible for generating the request and response. In a single word, we can say worker process is the heart of ASP.NET Web Application which runs on IIS.

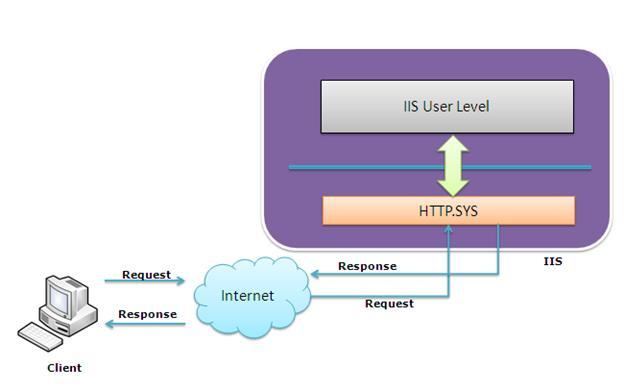
**Application Pool:** Application pool is the container of the worker process.  Application pools are used to separate sets of IIS worker processes that share the same configuration.  Application pools enable a better security, reliability, and availability for any web application.  The worker process serves as the process boundary that separates each application pool so that when one worker process or application is having an issue or recycles, other applications or worker processes are not affected. This makes sure that a particular web application doesn’t impact other web application as they are configured into different application pools.



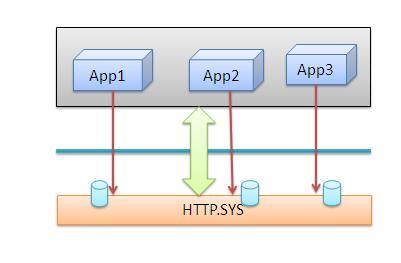
Application Pool with multiple worker processes is called “Web Garden.”  
Now, I have covered all the basic stuff like the Web server, Application Pool, Worker process. Now let’s have a look how IIS process the request when a new request comes up from a client.  
If we look into the IIS 6.0 Architecture, we can divide them into Two Layer

1.    Kernel Mode  
2.    User Mode

Now, **Kernel mode** is introduced with IIS 6.0, which contains the **HTTP.SYS**.  So whenever a request comes from Client to Server, it will hit HTTP.SYS First.

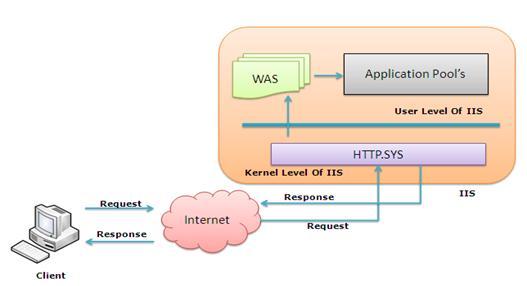


Now, HTTP.SYS is Responsible for pass the request to the particular Application pool. Now here is one question, How HTTP.SYS does come to know where to send the request?  This is not a random pickup. Whenever we create a new Application Pool, the ID of the Application Pool is being generated, and it’s registered with the HTTP.SYS. So whenever HTTP.SYS Received the request from any web application, it checked for the Application Pool and based on the application pool it sends the request.

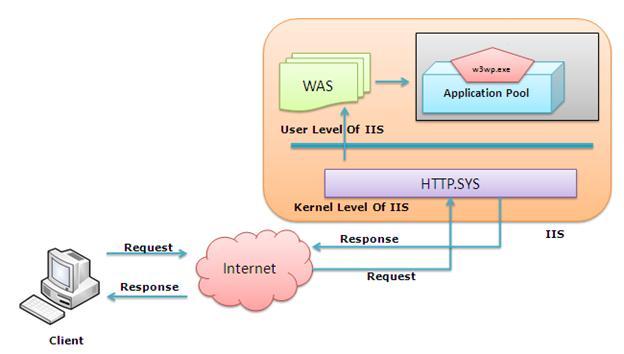


So, this was the first steps of IIS Request Processing.  
Till now, Client Requested for some information and request came to the Kernel level of IIS means at HTTP.SYS. HTTP.SYS has been identified the name of the application pool where to send. Now, let’s see how this request moves from HTTP.SYS to Application Pool.

In **User Level**of IIS, we have Web Admin Services (WAS) which takes the request from HTTP.SYS and pass it to the respective application pool.



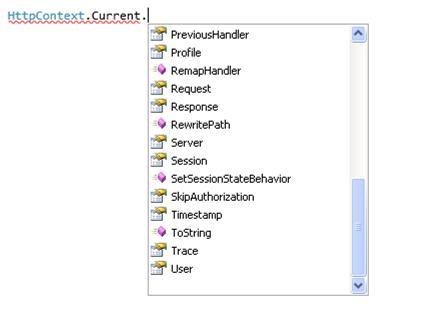
When Application pool receives the request, it just passes the request to worker process (w3wp.exe). The worker process “w3wp.exe” looks up the URL of the request to load the correct ISAPI extension. ISAPI extensions are the IIS way to handle requests for different resources. Once ASP.NET is installed, it installs its own ISAPI extension (aspnet\_isapi.dll) and adds the mapping into IIS.  
Note: Sometimes if we install IIS after installing asp.net, we need to register the extension with IIS using an aspnet\_regiis command.



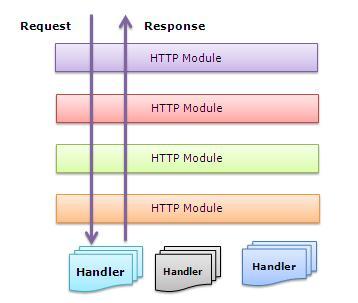
When Worker process loads the aspnet\_isapi.dll, it starts an **HTTPRuntime**, which is the entry point of an application. HTTPRuntime is a class which calls the ProcessRequest method to start Processing.



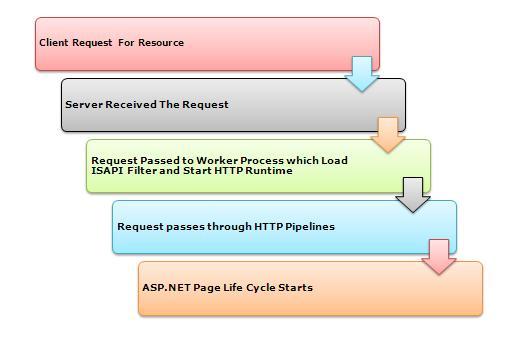
When this method called, a new instance of **HTTPContext** is created.  Which is accessible using **HTTPContext.Current**  Properties. This object remains alive during the life time of object request.  Using HttpContext.Current we can access some other objects like Request, Response, Session, etc.



After that HttpRuntime load, an HttpApplication object with the help of  HttpApplicationFactory class. Every request should pass through the corresponding HTTPModule to reach to HTTPHandler, this list of a module is configured by the HTTPApplication.  
Now, the concept comes called “**HTTPPipeline**.” It is called a pipeline because it contains a set of HttpModules ( For Both Web.config and Machine.config level) that intercept the request on its way to the HttpHandler. HTTPModules are classes that have access to the incoming request. We can also create our HTTPModule if we need to handle anything during upcoming request and response.



HTTP Handlers are the endpoints in the HTTP pipeline. All request that is passing through the HTTPModule should reach to HTTPHandler.  The  HTTP Handler generates the output for the requested resource. So, when we were requesting for any aspx web pages,   it returns the corresponding HTML output.  
All the request now passes from httpModule to respective HTTPHandler then the method and the ASP.NET Page life cycle starts.  This ends the IIS Request processing and starts the ASP.NET Page Lifecycle.



### **Conclusion**

When the client request for some information from a web server, request first reaches to HTTP.SYS of IIS. HTTP.SYS then send the request to particular  Application Pool. Application Pool then forwards the request to worker process to load the ISAPI Extension which will create an HTTPRuntime Object to Process the request via HTTPModule and HTTP handler. After that, the ASP.NET Page LifeCycle events start.  
This was a just overview of IIS Request Processing to let Beginner’s know how the request gets processed in the backend.  If you want to learn about details, please check the link for Reference and Further Study section.