Step 1: The user sends a request to IIS. IIS first checks which ISAPI extension can serve this request. Depending on file extension the request is processed. For instance, if the page is an '.ASPX page', then it will be passed to '*aspnet_isapi.dll*' for processing.

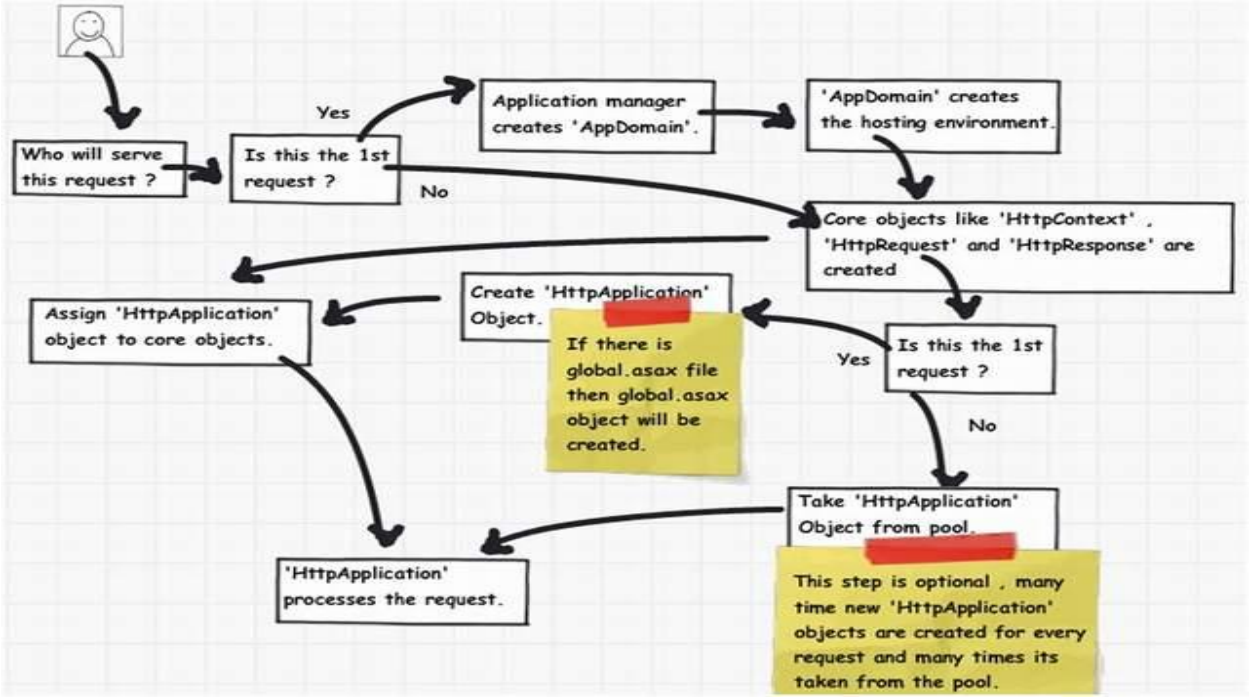
Step 2: If this is the first request to the website, then a class called as '`ApplicationManager`' creates an application domain where the website can run. As we all know, the application domain creates isolation between two web applications hosted on the same IIS. So in case there is an issue in one app domain, it does not affect the other app domain.

Step 3: The newly created application domain creates hosting environment, i.e. the '`HttpRuntime`' object. Once the hosting environment is created, the necessary core ASP.NET objects like '`HttpContext`' , '`HttpRequest`' and '`HttpResponse`' objects are created.

Step 4: Once all the core ASP.NET objects are created, '`HttpApplication`' object is created to serve the request. In case you have a '*global.asax*' file in your system, then the object of the '*global.asax*' file will be created. Please note *global.asax* file inherits from '`HttpApplication`' class.
Note: The first time an ASP.NET page is attached to an application, a new instance of '`HttpApplication`' is created. Said and done to maximize performance, `HttpApplication` instances might be reused for multiple requests.

Step 5: The `HttpApplication` object is then assigned to the core ASP.NET objects to process the page.

Step 6: `HttpApplication` then starts processing the request by HTTP module events, handlers and page events. It fires the MHPM event for request processing.



In What Event Should We Do What?

The million dollar question is in which events should we do what? Below is the table which shows in which event what kind of logic or code can go.

Section	Event	Description
`HttpModule`	`BeginRequest`	This event signals a new request; it is guaranteed to be raised on each request.
`HttpModule`	`AuthenticateRequest`	This event signals that ASP.NET runtime is ready to authenticate the user. Any authentication code can be injected here.
`HttpModule`	`AuthorizeRequest`	This event signals that ASP.NET runtime is ready to authorize the user. Any authorization code can be injected here.
`HttpModule`	`ResolveRequestCache`	In ASP.NET, we normally use outputcache directive to do caching. In this event, ASP.NET runtime determines if the page can be served from the cache rather than loading the patch from scratch. Any caching specific activity can be injected here.
`HttpModule`	`AcquireRequestState`	This event signals that ASP.NET runtime is ready to acquire session variables. Any processing you would like to do on session variables.
`HttpModule`	`PreRequestHandlerExecute`	This event is raised just prior to handling control to the `HttpHandler`. Before you want the control to be handed over to the handler any pre-processing you would like to do.
`HttpHandler`	`ProcessRequest`	`Httphandler` logic is executed. In this section, we will write logic which needs to be executed as per page extensions.
`Page`	`Init`	This event happens in the ASP.NET page and can be used for: • Creating controls dynamically, in case you have controls to be created on runtime. • Any setting initialization. • Master pages and the settings. In this section, we do not have access to viewstate, postedvalues and neither the controls are initialized.
`Page`	`Load`	In this section, the ASP.NET controls are fully loaded and you write UI manipulation logic or any other logic over here.

Section	Event	Description
Page	Validate	If you have valuators on your page, you would like to check the same here.
	Render	It's now time to send the output to the browser. If you would like to make some changes to the final HTML which is going out to the browser, you can enter your HTML logic here.
Page	Unload	Page object is unloaded from the memory.
HttpModule	PostRequestHandlerExecute	Any logic you would like to inject after the handlers are executed.
HttpModule	ReleaserequestState	If you would like to save update some state variables like session variables.
HttpModule	UpdateRequestCache	Before you end, if you want to update your cache.
HttpModule	EndRequest	This is the last stage before your output is sent to the client browser.