ASP .Net Part 1

- Subset of .Net Framework
- Successor of classic ASP
- Needs IIS – Internet Information Services – web server for ASP .Net
- Uses HTTP protocol – stateless protocol

IIS

- When a request comes to the server from the client, IIS takes the request, processes it and sends the request back to the client.
- The worker process of IIS is w3wp.exe

Request Processing

- A request arrives at HTTP.sys.
- HTTP.sys determines if the request is valid. If the request is not valid, it sends a code for an invalid request back to the client.
- If the request is valid, HTTP.sys checks to see if the request is for static content (HTML) because static content can be served immediately.
- If the request is for dynamic content, HTTP.sys checks to see if the response is located in its kernel-mode cache.
- If the response is in the cache, HTTP.sys returns the response immediately.
- If the response is not cached, HTTP.sys determines the correct request queue, and places the request in that queue.
- If the queue has no worker processes assigned to it, HTTP.sys signals the WWW service to start one.
- The worker process pulls the request from the queue and processes the request, evaluating the URL to determine the type of request.
- The worker process sends the response back to HTTP.sys.
- HTTP.sys sends the response back to the client and logs the request, if configured to do so.

ASP.Net Application Life Cycle

- **The stages of the ASP.NET application life cycle are:**

1. User requests an application resource from the Web server.
    1. When a Web server receives a request, it examines the file name extension of the requested file, determines which ISAPI (Internet Server Application Program Interface) extension should handle the request, and then passes the request to the appropriate ISAPI extension.
    2. ASP.NET handles file name extensions that have been mapped to it, such as .aspx, .ascx, .ashx, and .asmx.

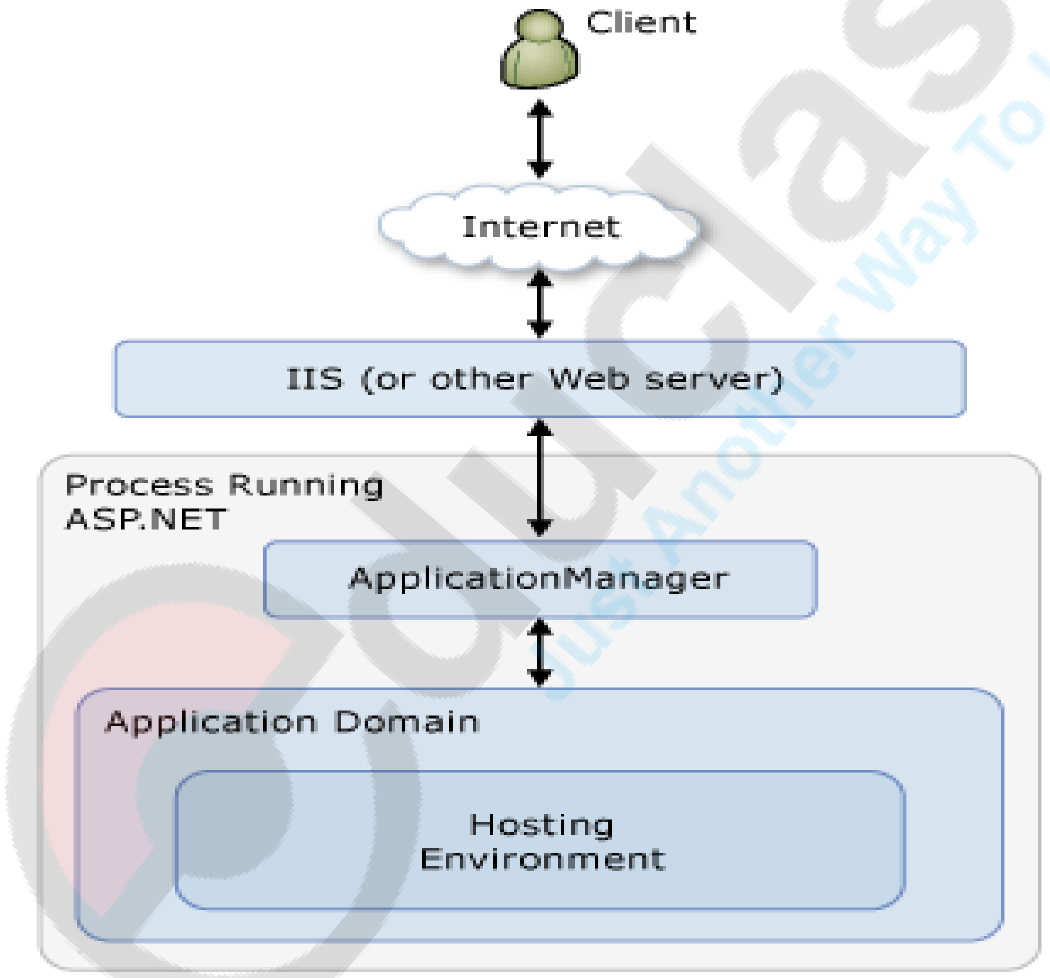
    – [ISAPI: ISAPI filters are DLL files that can be used to modify and enhance the functionality provided by IIS. ISAPI filters always run on an IIS server, filtering every request until they find one they need to process.

ASP.Net Application Life Cycle

- **The stages of the ASP.NET application life cycle are:**
- ASP.NET receives the first request for the application

1. When ASP.NET receives the first request for any resource in an application, a class named ApplicationManager creates an application domain.
2. Application domains provide isolation between applications for global variables and allow each application to be unloaded separately.
3. Within an application domain, an instance of the class named HostingEnvironment is created, which provides access to information about the application such as the name of the folder where the application is stored.
4. The following diagram illustrates this relationship:

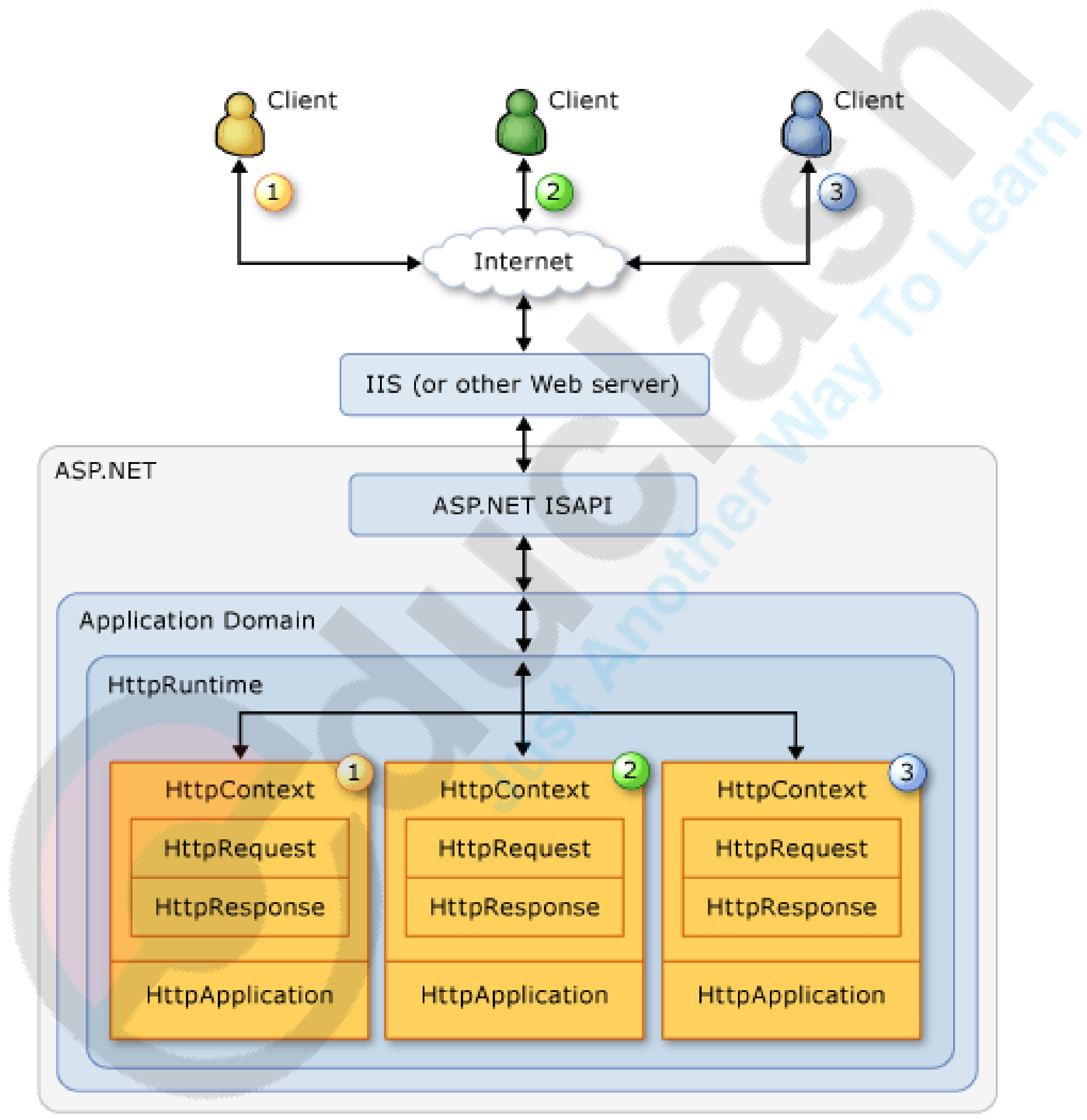
ASP.Net Application Life Cycle



ASP.Net Application Life Cycle

- **The stages of the ASP.NET application life cycle are:**
- ASP.NET core objects are created for each request.
  1. After the application domain has been created and the **HostingEnvironment** object instantiated, ASP.NET creates and initializes core objects such as HttpContext, HttpRequest, and HttpResponse.
  2. The **HttpContext** class contains objects that are specific to the current application request, such as the **HttpRequest** and **HttpResponse** objects.
  3. The **HttpRequest** object contains information about the current request, including cookies and browser information.
  4. The **HttpResponse** object contains the response that is sent to the client, including all rendered output and cookies.

ASP.Net Application Life Cycle



General Page Life Cycle Stages

1) Page request

When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled (therefore beginning the life of a page), or whether a cached version of the page can be sent in response without running the page.

2) Start

In the start stage, page properties such as Request and Response are set. At this stage, the page also **determines whether the request is a postback or a new request and sets the IsPostBack property.**

3) Initialization

During page initialization, controls on the page are available and **each control's UniqueID property is set.** A **master page and themes are also applied** to the page if applicable. If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state.

4) Load

During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state.

General Page Life Cycle Stages

5) Postback event handling

If the request is a postback, control event handlers are called. After that, the Validate method of all validator controls is called, which sets the IsValid property of individual validator controls and of the page.

6) Rendering

Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the Render method

7) Unload

The Unload event is raised after the page has been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties such as Response and Request are unloaded and cleanup is performed.

Page Lifecycle Events

PreInit:

•Raised after the start stage is complete and before the initialization stage begins.

•Use this event for the following:

1. Check the IsPostBack property to determine whether this is the first time the page is being processed.
2. The IsCallback and IsCrossPagePostBack properties have also been set at this time.
3. Create or re-create dynamic controls.
4. Set a master page dynamically.
5. Set the Theme property dynamically.
6. Read or set profile property values.

Page Lifecycle Events

Init:

• Raised after all controls have been initialized and any skin settings have been applied.

• The Init event of individual controls occurs before the Init event of the page.

• Use this event to read or initialize control properties.

Page Lifecycle Events

InitComplete:

- Raised at the end of the page's initialization stage.
- Only one operation takes place between the Init and InitComplete events: tracking of view state changes is turned on.
- View state tracking enables controls to persist any values that are programmatically added to the ViewState collection.
- Until view state tracking is turned on, any values added to view state are lost across postbacks.
- Controls typically turn on view state tracking immediately after they raise their Init event.
- Use this event to make changes to view state that you want to make sure are persisted after the next postback.

Page Lifecycle Events

PreLoad:

• Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.

Load:

• The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded.

• The Load event of individual controls occurs after the Load event of the page.

• Use the OnLoad event method to set properties in controls and to establish database connections.

Page Lifecycle Events

Control events

- Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.

LoadComplete

- Raised at the end of the event-handling stage.
- Use this event for tasks that require that all other controls on the page be loaded.

Page Lifecycle Events

Prerender:

• Raised after the Page object has created all controls that are required in order to render the page.

• The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control.

• The PreRender event of individual controls occurs after the PreRender event of the page.

• Use the event to make final changes to the contents of the page or its controls before the rendering stage begins.

Page Lifecycle Events

PreRenderComplete:

•Raised after each data bound control whose DataSourceID property is set calls its DataBind method.

SaveStateComplete:

• Raised after view state and control state have been saved for the page and for all controls.

• Any changes to the page or controls at this point affect rendering, but the changes will not be retrieved on the next postback.

Page Lifecycle Events

Render:

- This is not an event; instead, at this stage of processing, the Page object calls this method on each control.
- All ASP.NET Web server controls have a Render method that writes out the control's markup to send to the browser.

Unload:

- Raised for each control and then for the page.
- In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.
- For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks.

PostBack and CrossPage posting

- **PostBack** is the name given to the process of submitting an **ASP**.**NET** page to the server for processing.
- **IsPostBack** is a property of the **Asp**.**Net** page that tells whether or not the page is on its initial load or if a user has perform a button click on your web page that has caused the page to **post back** to itself.
- The value of the Page.IsPostBack property will be set to true when the page is executing after a postback, and false otherwise.

PostBack and CrossPage posting

- When we want to post values from one page to another page it is known as CrossPage Post.
- By default when a button is clicked it postbacks to its own page.
- For CrossPage Posting, we have to assign the property PostBackURL of the button to the second page.