The Two Step Process

* ASP.NET creates an environment which can process the request. In other words, it creates the application object, request, response and context objects to process the request.
* Once the environment is created, the request is processed through a series of events which is processed by using modules, handlers and page objects. To keep it short, let's name this step as MHPM (Module, handler, page and Module event),

**What is the Global.asax**

The Global.asax file, sometimes called the ASP.NET application file, provides a way to respond to application or module level events in one central location. You can use this file to implement application security, as well as other tasks.

The Global.asax file, which is derived from the HttpApplication class, maintains a pool of HttpApplication objects, and assigns them to applications as needed.

**In a web application, events can occur at 3 levels. what are they?**  
**1.** At the Application Level(Example: Application Start)  
**2.** At the Page Level(Example: Page Load)  
**3.** At the Control Level (Example: Button Click)

#### Available Events

An **HttpApplication** class provides a number of events with which modules can synchronize. The following events are available for modules to synchronize with on each request. These events are listed in sequential order:

* **BeginRequest**: Request has been started. If you need to do something at the beginning of a request (for example, display advertisement banners at the top of each page), synchronize this event.
* **AuthenticateRequest**: If you want to plug in your own custom authentication scheme (for example, look up a user against a database to validate the password), build a module that synchronizes this event and authenticates the user how you want to.
* **AuthorizeRequest**: This event is used internally to implement authorization mechanisms (for example, to store your access control lists (ACLs) in a database rather than in the file system). Although you can override this event, there are not many good reasons to do so.
* **ResolveRequestCache**: This event determines if a page can be served from the Output cache. If you want to write your own caching module (for example, build a file-based cache rather than a memory cache), synchronize this event to determine whether to serve the page from the cache.
* **AcquireRequestState**: Session state is retrieved from the state store. If you want to build your own state management module, synchronize this event to grab the Session state from your state store.
* **PreRequestHandlerExecute**: This event occurs before the HTTP handler is executed.
* **PostRequestHandlerExecute**: This event occurs after the HTTP handler is executed.
* **ReleaseRequestState**: Session state is stored back in the state store. If you are building a custom session state module, you must store your state back in your state store.
* **UpdateRequestCache**: This event writes output back to the Output cache. If you are building a custom cache module, you write the output back to your cache.
* **EndRequest**: Request has been completed. You may want to build a debugging module that gathers information throughout the request and then writes the information to the page.

### The Two Interceptors: HttpModule and HttpHandler

 ASP.NET provides two ways of injecting logic in the request pipeline HttpHandlers and HttpModules.

System.Web

HttpHandler help us to inject pre-processing logic based on the extension of the file name **EXTENSION** requested.

HttpModule is an event based methodology to inject pre-processing logic before any resource is requested. When any client sends a request for a resource, the request pipeline emits a lot of events as shown in the figure below and we can hook into them So when the request pipe line executes depending on the event registered to, the logic from the modules is processed.

**Modules are called before and after the handler executes. Modules enable developers to intercept, participate in, or modify each individual request**

An HTTP module is an assembly that is called on every request that is made to your application (for pre-processing logic implementation).  
HTTP modules are called as part of the ASP.NET request pipeline and have access to life-cycle events throughout the request. HTTP modules let you examine incoming and outgoing requests and take action based on the request.

**Handlers are used to process individual endpoint requests. Handlers enable the ASP.NET Framework to process individual HTTP URLs or groups of URL extensions within an application.**

An ASP.NET HTTP handler is the process (frequently referred to as the "endpoint") that runs in response to a request made to an ASP.NET Web application. The most common handler is an ASP.NET page handler that processes .aspx files. When users request an .aspx file, the request is processed by the page through the page handler. You can create your own HTTP handlers that render custom output to the browser.

**Extra Questions**

**In general, Application events** are used to initialize data that needs to be available to all the current sessions of the application. Where as **Session events** are used to initialize data that needs to be available only for a given individual session, but not between multiple sessions.

**What is a Session, in a web application?**  
A session is a unique instance of the browser. A single user can have multiple sessions, by visiting your application, with multiple instances of the browser running with a different session-id on his machine.

To use cookie-less sessions set the cookieless attribute to true in web.config as shown below.

how can we force cookieless sessioins?

**<sessionState mode="InProc" cookieless="false"></sessionState>**

**What are the differences between view,session and application state?**

**So, in short, the differences are as follows**  
**ViewState:**  
**1.** ViewState of a webform is available only with in that webform  
**2.** ViewState is stored on the page in a hidden field called \_ViewState. Because of this, the ViewState, will be lost, if you navigate awaya from the page, or if the broswer is closed.  
**3.** ViewState is used by all asp.net controls to retain their state across postback  
  
**Session State:**  
**1.** Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data.  
**2.** Session state variables are stored on the web server.  
**3.** SessionState variables are cleared, when the user session times out. The default is 20 minutes. This is configurable in web.config  
  
**Application State:**  
**1.** Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data.  
**2.** Application State variables are stored on the web server.  
**3.** Application State variables are cleared, when the process hosting the application is restarted.