Q1.

When an ASP.NET page runs, the page goes through a life cycle in which it performs a series of processing steps. These include initialization, instantiating controls, restoring and maintaining state, running event handler code, and rendering. In general terms, the page goes through the stages outlined in the following table. In addition to the page life-cycle stages, there are application stages that occur before and after a request but are not specific to a page. Within each stage of the life cycle of a page, the page raises events that you can handle to run your own code. For control events, you bind the event handler to the event, either declaratively using attributes such as onclick, or in code.

Individual ASP.NET server controls have their own life cycle that is similar to the page life cycle. For example, a control's Init and Load events occur during the corresponding page events.

Although both Init and Load recursively occur on each control, they happen in reverse order. The Init event (and also the Unload event) for each child control occur before the corresponding event is raised for its container (bottom-up). However the Load event for a container occurs before the Load events for its child controls (top-down). Master pages behave like child controls on a page: the master page Init event occurs before the page Init and Load events, and the master page Load event occurs after the page Init and Load events.

Q3.

ASP objects: application, request, response, server. We can see a summary of the objects in ASP in the Microsoft ASP objects page.

1. **Application**: Global.ASA
2. **Request**: allows access to both header and body of the HTTP request.
    - Query String: used in the past with method GET to pass parameters.
      Syntax: Request.QueryString ("parameter")
    - Form: used today with method POST to provide access to the pair - fieldname = value.
      Syntax: Request.Form ("fieldname")
    - Server Variables: displays the content of the server environment variables, including **REQUEST_METHOD** which allows the testing of method (GET

    or POST).
    Syntax: Request.ServerVariables ("environment-varname")
  - Cookies: allows cookie processing. Syntax: Request.Cookies ( )
  - ClientCertificate: used in secure transactions. Syntax: Request.ClientCertificate ( )

3. **Response**: allows control over the HTTP response to the client.
   - Properties need to be inserted after <%@ LANGUAGE="VBSCRIPT"%> directive, but before the first HTML tag.
     - Buffer: controls if the response is sent as the page and script are created, or if are kept in a buffer and sent when processing is completed.
       Syntax: Response.Buffer = True (False is the default)
     - CacheControl: allows caching by a Proxy server, but the Proxy server needs to be setup to cache pages.
       Syntax: Response.CacheControl = "setting" , where setting is Public or Private(default)
     - ContentType: allows setting the page content type. The default id text/html. Syntax: Response.ContentType = "type" . The following examples set the ContentType property to other common values:

       <% Response.ContentType = "text/HTML" %>

       <% Response.ContentType = "image/GIF" %>

       <% Response.ContentType = "image/JPEG" %>

       <% Response.ContentType = "text/plain" %>

       <% Response.ContentType = "image/JPEG" %>
       Character set: allows setting the character set to be used. The default on the Web is ISO-LATIN-1.
       Syntax: Response.Charset ("charset")
     - Expires: specifies the length of time (in minutes) that a client will cache the ASP page. The default is 24 hours. **Note**: if you want to specify Options Explicit include Response.Expires = 0, otherwise you will receive an error message.
       Syntax: Response.Expires = intMinutes
     - others: ExpiresAbsolute, IsClientConnected,PICS, Status.
   - Methods affect the behavior of the response. Lets see some of them:

        - Clear: empties the content of the buffer, not sending information to the client browser.
          Syntax: Response.Clear
        - End: sends all data currently in the buffer to the client and all other code after End is not processed.
          Syntax: Response.End
        - Flush: sends all data currently in the buffer to the client, but allows code after it to be processed and later sent to the client.
          Syntax: Response.Flush
        - Redirect: redirects the client's request to another URL. It can be a local or remote page URL.
          Syntax: Response.Redirect "strURL"
        - Write: writes information to the HTTP response body. To write special characters use the VB Script function Chr ( ). To add variables just use their names without quotes and & to include it in the message.
          Syntax: Response.Write("text")

4. **Server**: provides additional methods and properties associated with IIS or PWS. We will see its only property and three methods:
    - ScriptTimeout (property): total time allowed for the script to run in seconds. The default is 90 seconds.
      Syntax: Server.ScriptTimeout = intSecs
    - CreateObject (method): allows to instantiate an object implemented using DLLs, regularly installed in the machine.
      Syntax: Set objName = Server.CreateObject ("progam ID name") -- the program ID name is the name of the DLL in the Windows Registry.
    - Execute (method): allows to call and execute an ASP script from another ASP script. Modular programming can be implemented this way. Please note that this is different of using the #INCLUDE directive, which actually inserts the code in your current script.
      Syntax: Server.Execute ("aspscript.asp")
    - Transfer (method): allows to redirect the execution of one script to another. It differs from Execute in that the called script does not return control to the original script.
      Syntax: Server.Transfer("aspscript.asp")
5. **Session**: allows the management of a user connection with the server. We will not discuss it here, like we did not discuss cookies. See the Microsoft ASP objects page on session.