ASP.NET question

1. Explain the differences between Server-side and Client-side code? Server side scripting means that all the script will be executed by the server and interpreted as needed. ASP doesn’t have some of thefunctionality like sockets, uploading, etc. For these you have to make a custom components usually inVB or VC++. Client side scripting means that the script will be executed immediately in the browser such as form field validation, clock, email validation, etc. Client side scripting is usually done in VBScript or JavaScript. Download time, browser compatibility, and visible code - since JavaScript and VBScript code is included in the HTML page, then anyone can see the code by viewing the page source. Also a possible

security hazards for the client computer.

2. What type of code (server or client) is found in a Code-Behind class? C#

3. Should validation (did the user enter a real date) occur server-side or client-side? Why?

Client-side validation because there is no need to request a server side date when you could obtain a date from the client machine.

4. What does the "EnableViewState" property do? Why would I want it on or off? Enable ViewState turns on the automatic state management feature that enables server controls to re-populate their values on a round trip without requiring you to write any code. This feature is not free however, since the state of a control is passed to and from the server in a hidden form field. You should be aware of when ViewState is helping you and when it is not. For example, if you are binding a control to data on every round trip (as in the datagrid example in tip #4), then you do not need the control to maintain it’s view state, since you will wipe out any re-populated data in any case. ViewState is enabled for all server controls by default. To disable it, set the EnableViewState property of the control to false.

5. What is the difference between Server.Transfer and Response.Redirect? Why would I choose one over the other?

Server.Transfer() : client is shown as it is on the requesting page only, but the all the content is of the requested page. Data can be persist accros the pages using Context.Item collection, which is one of the best way to transfer data from one page to another keeping the page state alive. Response.Dedirect() :client know the physical loation (page name and query string as well).Context.Items loses the persisitance when nevigate to destination page. In earlier versions of IIS, if we wanted to send a user to a new Web page, the only option we had was Response.Redirect. While this method does accomplish our goal, it has several important drawbacks. The biggest problem is that this method causes each page to be treated as a separate transaction. Besides making it difficult to maintain your transactional integrity, Response.Redirect introduces some additional headaches. First, it prevents good encapsulation of code. Second, you lose access to all of the properties in the Request object. Sure, there are workarounds, but they’re difficult. Finally, Response.Redirect necessitates a round trip to the client, which, on high-volume sites, causes scalability problems. As you might suspect, Server.Transfer fixes all of these problems. It does this by performing the transfer on the server without requiring a roundtrip to the client.

6. Can you give an example of when it would be appropriate to use a web service as opposed to a non serviced .NET component? When to Use Web Services:

\* Communicating through a Firewall When building a distributed application with 100s/1000s of users spread over multiple locations, there is always the problem of communicating between client and server because of firewalls and proxy servers. Exposing your middle tier components as Web Services and invoking the directly from a Windows UI is a very valid option.

\* Application Integration When integrating applications written in various languages and running on disparate systems. Or even applications running on the same platform that have been written by separate vendors.

\* Business-to-Business Integration This is an enabler for B2B intergtation which allows one to expose vital business processes to authorized supplier and customers. An example would be exposing electronic ordering and invoicing, allowing customers to send you purchase orders and suppliers to send you invoices electronically.

\* Software Reuse This takes place at multiple levels. Code Reuse at the Source code level or binary componet-based resuse. The limiting factor here is that you can reuse the code but not the data behind it. Webservice overcome this limitation. A scenario could be when you are building an app that aggregates the functionality of serveral other Applicatons. Each of these functions could be performed by individual apps, but there is value in perhaps combining the the multiple apps to present a unifiend view in a Portal or Intranet.

\* When not to use Web Services: Single machine Applicatons When the apps are running on the same machine and need to communicate with each other use a native API. You also have the options of using component technologies such as COM or .NET Componets as there is very little overhead.

\* Homogeneous Applications on a LAN If you have Win32 or Winforms apps that want to communicate to their server counterpart. It is much more efficient to use DCOM in the case of Win32 apps and .NET Remoting in the case of .NET Apps.

7. Let’s say I have an existing application written using Visual Studio (VBInterDevand this application utilizes WindowsCOM+ transaction services. How would you approach migrating this application to .NET?

8. Can you explain the difference between an ADO.NET Dataset and an ADO Recordset?

In ADO, the in-memory representation of data is the recordset. In ADO.NET, it is the dataset. There are important differences between them.

\* A recordset looks like a single table. If a recordset is to contain data from multiple database tables, it must use a JOIN query, which assembles the data from the various database tables into a single result table. In contrast, a dataset is a collection of one or more tables. The tables within a dataset are called data tables; specifically, they are DataTable objects. If a dataset contains data from multiple database tables, it will typically contain multiple DataTable objects. That is, each DataTable object typically corresponds to a single database table or view. In this way, a dataset can mimic the structure of the underlying database. A dataset usually also contains relationships. A relationship within a dataset is analogous to a foreign-key relationship in a database —that is, it associates rows of the tables with each other. For example, if a dataset contains a table about investors and another table about each investor’s stock purchases, it could also contain a relationship connecting each row of the investor table with the corresponding rows of the purchase table. Because the dataset can hold multiple, separate tables and maintain information about relationships between them, it can hold much richer data structures than a recordset, including self-relating tables and tables with many-to-many relationships.

\* In ADO you scan sequentially through the rows of the recordset using the ADO MoveNext method. In ADO.NET, rows are represented as collections, so you can loop through a table as you would through any collection, or access particular rows via ordinal or primary key index. DataRelation objects maintain information about master and detail records and provide a method that allows you to get records related to the one you are working with. For example, starting from the row of the Investor table for "Nate Sun," you can navigate to the set of rows of the Purchase table describing his purchases.

A cursor is a database element that controls record navigation, the ability to update data, and the visibility of changes made to the database by other users. ADO.NET does not have an inherent cursor object, but instead includes data classes that provide the functionality of a traditional cursor. For example, the functionality of a forward-only, read-only cursor is available in the ADO.NET DataReader object. For more information about cursor functionality, see Data Access Technologies.

\* Minimized Open Connections: In ADO.NET you open connections only long enough to perform a database operation, such as a Select or Update. You can read rows into a dataset and then work with them without staying connected to the data source. In ADO the recordset can provide disconnected access, but ADO is designed primarily for connected access. There is one significant difference between disconnected processing in ADO and ADO.NET. In ADO you communicate with the database by making calls to an OLE DB provider. In ADO.NET you communicate with the database through a data adapter (an

OleDbDataAdapter, SqlDataAdapter, OdbcDataAdapter, or OracleDataAdapter object), which makes calls to an OLE DB provider or the APIs provided by the underlying data source. The important difference is that in ADO.NET the data adapter allows you to control how the changes to the dataset are transmitted to the database — by optimizing for performance, performing data validation checks, or adding any other extra processing. Data adapters, data connections, data commands, and data readers are the components that make up a .NET Framework data provider. Microsoft and third-party providers can make available other .NET Framework data providers that can be integrated into Visual Studio.

\* Sharing Data Between Applications. Transmitting an ADO.NET dataset between applications is much easier than transmitting an ADO disconnected recordset. To transmit an ADO disconnected recordset from one component to another, you use COM marshalling. To transmit data in ADO.NET, you use a dataset, which can transmit an XML stream.

\* Richer data types.COM marshalling provides a limited set of data types — those defined by the COM standard. Because the transmission of datasets in ADO.NET is based on an XML format, there is no restriction on data types. Thus, the components sharing the dataset can use whatever rich set of data types they would ordinarily use.

\* Performance. Transmitting a large ADO recordset or a large ADO.NET dataset can consume network resources; as the amount of data grows, the stress placed on the network also rises. Both ADO and ADO.NET let you minimize which data is transmitted. But ADO.NET offers another performance advantage, in that ADO.NET does not require data-type conversions. ADO, which requires COM marshalling to transmit records sets among components, does require that ADO data types be converted to COM data types.

\* Penetrating Firewalls.A firewall can interfere with two components trying to transmit disconnected ADO recordsets. Remember, firewalls are typically configured to allow HTML text to pass, but to prevent system-level requests (such as COM marshalling) from passing.

9. Can you give an example of what might be best suited to place in the Application\_Start and Session\_Start subroutines?

The Application\_Start event is guaranteed to occur only once throughout the lifetime of the application. It’s a good place to initialize global variables. For example, you might want to retrieve a list of products from a database table and place the list in application state or the Cache object. SessionStateModule exposes both Session\_Start and Session\_End events.

10. If I’m developing an application that must accomodate multiple security levels though secure login and my ASP.NET web appplication is spanned across three web-servers (using round-robbin load balancing) what would be the best approach to maintain login-in state for the users?

11. What are ASP.NET Web Forms? How is this technology different than what is available though ASP?

Web Forms are the heart and soul of ASP.NET. Web Forms are the User Interface (UI) elements that give your Web applications their look and feel. Web Forms are similar to Windows Forms in that they provide properties, methods, and events for the controls that are placed onto them. However, these UI elements render themselves in the appropriate markup language required by the request, e.g. HTML. If you use Microsoft Visual Studio .NET, you will also get the familiar drag-and-drop interface used to create your UI for your Web application.

12. How does VB.NET/C# achieve polymorphism? By using Abstract classes/functions.

13. Can you explain what inheritance is and an example of when you might use it? Inheritance is a fundamental feature of an object oriented system and it is simply the ability to inherit data and functionality from a parent object. Rather than developing new objects from scratch, new code can be based on the work of other programmers, adding only new features that are needed.

14. How would you implement inheritance using VB.NET/C#? When we set out to implement a class using inheritance, we must first start with an existing class from which we will derive our new subclass. This existing class, or base class, may be part of the .NET system class library framework, it may be part of some other application or .NET assembly, or we may create it as part of our existing application. Once we have a base class, we can then implement one or more subclasses based on that base class. Each of our subclasses will automatically have all of the methods, properties, and events of that base class ? including the implementation behind each method, property, and event. Our subclass can add new methods, properties, and events of its own - extending the original interface with new functionality. Additionally, a subclass can replace the methods and properties of the base class with its own new implementation - effectively overriding the original behavior and replacing it with new behaviors. Essentially inheritance is a way of merging functionality from an existing class into our new subclass. Inheritance also defines rules for how these methods, properties, and events can be merged.