**-What would be the best way to validate the contents of a postal code field in Web Forms?**

RequiredFieldValidator

**-Which of the following control(s) will initiate a Postback by default when changed?** TextBox

**-Which of the following is a key limitation of HTTP that ASP.NET is designed to address?**

Stateless Protocol

**-ASP**.**NET** provides a property called **IsPostBack** that is TRUE when the page is being loaded as a result of a **post back**, and is FALSE otherwise.

**-Which is the LEAST flexible way to change the appearance of controls and text in an ASP.NET site?**

Directly change the attributes individually using the Properties Window of the Designer

**-When will a Page object be created for the current page?** Only the first time the application starts

**-By default when a TextBox control is edited, the edit event is:**

Queued and handled on the next user action that initiates a PostBack

**-Which of the following statements about the ASP.NET ViewState is FALSE?**

Is the best way to store control information between multiple pages

**-Standard Controls:** render standard form elements such as buttons, input fields, and labels

**-Validation Controls:** validate form data before you submit the data to the server.

**-Rich Controls:**  to render things such as calendars, file upload buttons,

**-Data Controls:**  to interact/ display with data such as database data. .

**-Navigation Controls:** to display standard navigation elements such as menus, .

**-Login Controls:** to display login, change password, and registration forms.

**-HTML Controls: enable us** to convert any HTML tag into a server-side control.

-PreInit -> Init -> InitComplete -> PreLoad -> Load -> LoadComplete -> PreRender -> PreRenderComplete -> SaveStateComplete -> Unload

-Submitting Form DATA : Button, LinkButton, ImageButton.

-In order for a (submit) button to trigger the validation process, its CausesValidation property must be set to true (by default)

-To validate on the server side make sure to check to Page.IsValid property in event handlers

-All validation controls are held in the Page.Validators collection

• RequiredFieldValidator— Enables you to require a user to enter a value in a form field.

• RangeValidator— Enables you to check whether a value falls between a certain minimum and maximum value.

• CompareValidator— Enables you to compare a value against another value or perform a data type check.

• RegularExpressionValidator— Enables you to compare a value against a regular expression.

• CustomValidator— Enables you to perform custom validation.

• ValidationSummary— Enables you to display a summary of all validation errors in a page.

-public IEnumerable<Review> Get([FromUri] ReviewQuery query)

DataClasses1DataContext db = new DataClasses1DataContext();

IQueryable<Review> queryable = db.Reviews.AsQueryable();

if (query.BookId != 0)

queryable = queryable.Where(review => review.bookId == query.BookId).AsQueryable();

List<Review> reviews = queryable.ToList();

db.books.InsertOnSubmit(book); db.Books.DeleteOnSubmit(book); db.SubmitChanges();

Book book = db.Books.Where(b => b.Id == id).SingleOrDefault();

HttpResponseMessage reponse = client.GetAsync(url).Result;

if (reponse.IsSuccessStatusCode) {

var data = reponse.Content.ReadAsStringAsync().Result;

List<Book> books = JsonConvert.DeserializeObject<List<Book>>(data);

}

string json = JsonConvert.SerializeObject(new {

isbn = tbBookIsbn.Text

});

StringContent content = new StringContent(json, Encoding.UTF8, "application/json");

string url = "http://mega-book.azurewebsites.net/api/book";

HttpResponseMessage reponse = client.PostAsync(url, content).Result;

if (reponse.IsSuccessStatusCode)

Save: HttpCookie cookie = new HttpCookie(“my”); cookie.Expires = DateTime.Now.AddDatys(1d);

Response.Cookies.Add(cookie);

Load: HttpCookie cookie = Request.Cookies[“my”];

If (cookie != null) var value = cookie[“some”];

<asp:SqlDataSource ID=”SqlDataSource1” runat=”server” ConnetionString=”<%$ ConnectionStrings:Some %>”

SelectCommand=”SELECT \* FROM [users]”></asp:SqlDataSource>

SqlDataReader: very efficient. Limited connected, forward only, read only

DataSet: can be updated; read/ writed. Can store data and work offline (disconnected from database)

LINQ: var result = from s in Collection where s.Contains(“something”) select s;

Var result = from s in List where s.Age > 12 s.Age < 20 select s;