ASP.NET Data Controls(GridView) Kullanımı

Dört tane Data Control vardır:

1. **GridView** : Büyük tablolar atanır. Çok önemlidir.
2. **DetailsView** : Atanan tablo'da bir süre içinde sadece tek bir kayıt/satır göstermek için kullanılır. Editing işlemleri yapmayı da sağlar.
3. **FormView** : **DetailsView**'den tek farkı templates kullanımına izin verdiği için esnek bir layout sağlar.
4. **ListView** : **GridView**'den tek farkı templates kullanımına izin verdiği için esnek bir layout sağlar.

Data Binding mantığı ile kullanılan **List** control'ler basit işlemleri yapmayı sağlarken bu control'ler birden çok column gösterebildikleri için zengin control'lerdir ve bir çok işlem yapmaya olanak tanırlar.
Bu control'ler **selecting, editing, sorting** de yapabilmektedirler.

GridView

Birden çok sütuna sahip tablo göstermeyi sağlar. Database'deki tablomuzun her satırı **GridView** 'de bir satıra denk gelir. Tablomuzdaki her field'te **GridView** control'de bir sütuna denk gelir. **GridView**'in **DataSource**'unu set ettiğimiz zaman **DataBind()** metodu otomatik olarak çağrılmaktadır. Fakat **ListBox**'ın sağladığı **DataTextField, DataValueField** gibi özellikleri sağlamaz. Çünkü **GridView** otomatik olarak tablomuzdaki her field için bir column üretir. Tabi bunu sağlayan ise **GridView**'in **AutoGenerateColumns** property'sinin **true** olmasıdır.

```
<asp:GridView ID="GridView1" runat="server" />
```

ifadesi sayfanın source code kısmına eklenirse **GridView1** isimli bir **GridView** nesnesi oluşturulmuş olur. Bu sayfanın kod kısmına gelip şu kodları **Page_Load** event'inde yazalım :

```
protected void Page_Load(object sender, EventArgs e)
{
    string connectionString = WebConfigurationManager.ConnectionStrings["DilekceConnectionString"].ConnectionString;
    SqlConnection myConnection = new SqlConnection();
    SqlCommand myCommand = new SqlCommand();
    SqlDataAdapter myAdapter = new SqlDataAdapter();
    DataSet myDataSet = new DataSet();
    myConnection.ConnectionString = connectionString;
    myCommand.Connection = myConnection;
    myCommand.CommandText = "Select * from tblDilekce";
    myAdapter.SelectCommand = myCommand;
    try
    {
        myConnection.Open();
        myAdapter.Fill(myDataSet,"tblDilekce");
        myConnection.Close();
        GridView1.DataSource=myDataSet; //Görüldüğü gibi GridView'e DataSet nesnesi atanıyor.
        GridView1.DataBind();//this.DataBind() değil GridView'in DataBind()'i çağrıldı

    }
    catch (Exception er)
    {
        throw;//Hata meydana geldiğinde Page_Load metodunu çağıran metoda sorun gönderilir.
    }
}
```

Sütunlar

Default olarak **GridView**'in **AutoGenerateColumns** property'sinin true olduğunu ifade etmiştik. Biz bunu **false** yaparak tablomuzun sadece belirli sütünlarının gösterilmesini, sıralamasının değiştirilmesini veya gösterilme şeklinde değişikliklikler sağlayabiliriz.

Sütun Tipleri :

1. **BoundField** : Database'deki tablomuzun sütun değerlerini gösterir.
2. **HyperLinkField** : Database'deki tablomuzun sütun değerlerini **hyperlink** gibi gösterir.
3. **ButtonField** : Button gösterir
4. **CheckBoxField** : CheckBox gösterir
5. **CommandField** : **Selection** ve **Editing** yapmayı sağlayan buton gösterir.
6. **ImageField** : Resim gösterir.
7. **TemplateField** :

CommandField Tip

ShowSelectButton property'i **true** yap. **ButtonType Link** yap. Select button'a tıkladığında **posted back** olur. **SelectedIndexChanged** event **fired** olur. **SelectedIndex** property seçilen row'un index değerini alır.

ButtonField Tip

ButtonField ekleyin. **DataTextField**'e kullanmak istediğimiz field'i yazarız. Yani tablomuzun bir sütun ismini. Bu field **RowCommand** event meydana getirir. Fakat **SelectedIndex** property'e seçilen row'un index'ini manually yazmamız gerekir. Bunu otomatik yapmak için yani **CommandField**'te olduğu gibi otomatik olarak **SelectedIndex** property'sine seçilen row'un index'inin atanması için **ButtonField**'in **CommandName** özelliğine **Select** yazmak yeterli olacaktır.

Editing with The GridView

Nasıl ki bir row'u select moda getirmek için **SelectedIndex** property'e row'un row numarasını yazıyorsak, aynı şekilde bir row'u edit moda getirmek için row'un row numarasını **EditIndex**'e atamak gerekir. Tabi ki bu iki operasyonu otomatik hale getirebiliriz; eğer button tiplerini özelleştirirsek :
SelectMode için **Command** Field column'un **ShowSelectButton** property'sini true yaparak.
EditMode için **Command Field** column'un **ShowEditButton** property'sini **true** yaparak.

Formatting Specific Values

Özel bir satırın veya bir sütunun veya satırların/sütunların özelliklerini değiştirmek için **GridView** bize **RowDataBound** eventini sağlar. Bu event **GridView**'in data ile satır satır

doldurulmasında meydana gelir. Bir satır doldurulduktan hemen sonra meydana gelir. 5 tane satır varsa 5 kez oluşur.

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    //Yaratılan GridViewRow nesnesinin tipi DataRow mu diye kontrol edilmiştir.
    // Eğer edilmeseydi Header  nesnesi de olabilirdi.
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        //DataBinder tüm data türlerini anlar Eval metod ise runtime esnasında DataBinding ifadelerini
        //değerlendirir ve object türünden nesne dönderir. Bu örneğimizde dönen object nesnesi ise
        //oluşan satırın Ad sütunundaki değerdir.
        string str = (string)DataBinder.Eval(e.Row.DataItem, "Ad");
        if (str == "Hasan")
        {
            //e GridViewRowEventArgs nesnesidir ; e.Row ise GridViewRow nesnesidir.
            e.Row.BackColor = System.Drawing.Color.Maroon;
        }
    }
}
```

e.Row.DataItem ifadesindeki **DataItem** property **e.Row**'un bir property'sidir. Bu property, **e.Row**'un data nesnesini dönderir. Eğer bu nesne dönmeseydi, biz **DataBinder.Eval()** metodunun ikinci parametresine database'deki tablomuzun bir sütunu olan **"Ad"** sütununu yazamazdık ; çünkü **GridViewRow** bir **row nesnesidir. Row** nesnesi column ismine ulaşamaz. Ulaşabilecek nesne sadece Bu **GridViewRow** nesnesinin **DataItem** property'siyle oluşan data nesnesidir.

Sıralama(Sorting)

GridView'in **AllowSorting** property'sini **true** yapın. **BoundField**'in **ShortExpression**'una, **BoundField**'in Data Field'inde ne yazıyorsa onu yazın.

Sayfalandırma(Paging)

Arama motorlarında arama sonucu sayfalar halinde verilir. İşte bu işlem **GridView**'in **Paging** özelliğiyle sağlanmaktadır. Otomatik **Paging** yapmak için **GridView**'in **AllowPaging** property'si true yapılır. Eğer özel ayarlar yapmak istiyorsak **GridView**'in Pager Settings property'sinde düzenlemeler yapmalıyız. Bir sayfada kaç tane row göstermek istiyorsak **GridView**'in **PageSize** property'sine bir int değeri gireriz.3, GridView'in **Pager** Style property'siyle farklı stiller kullanabiliriz.

NOT: Her yeni sayfaya tıkladığımızda database ile bağlantı kurulduğundan database'in yükünü artırırız. Bunu önlemek için database'den tüm bilgi çekilip DataSet'e kaydedilir ve **DataSet** içeriği de Web Server'in memory'sinde **cache** edilirse kullanıcı yeni bir sayfaya geçtiğinde data'yı memory'den alır ve tekrar **GridView**'e atarsak database'in yükünü baya bir azaltmış oluruz. Bu teknik daha sonra anlatılacaktır.