ASP.NET State Management

Desktop application ile web application arasındaki en önemli fark **state management'tır.** Application çalıştığı sürece bilgilerin nasıl tutulacağı ile ilgili bilgiler vereceğiz.

The Problem of State

Tipik bir web request'te, kullanıcı bir sayfayı request eder ve server sayfayı html kodlarına dönüştürdükten sonra kullanıcıya yollar ve bağlantıyı sonlandırır. Bu sayede aynı anda binlerce kullanıcı aynı uygulama için request'te bulunabilir. **Fakat** bağlantı kapatıldığında, kullanıcı bilgileri de server'dan silinmiş olur. Diyelim ki bir kullanıcı login yaptı, daha sonra aynı uygulamanın başka sayfasına geçiş yapınca login bilgileri kaybolacaktır. İşte bu tarz durumları önlemek için, onlarca kullanıcının aynı anda login yapabilmelerini sağlamak için daha doğrusu bir kullanıcının bilgilerinin her postback durumunda ve yeni bir sayfaya geçiş durumunda kaybolmaması için, bazı düzenlemeler gereklidir.

View State

En yaygın bilgi kaydetme yollarından biri **view state** kullanmaktır. Web Server'dan yanıt olarak html sayfası gönderildiğinde

```
<input type="hidden" value="....."/>
```

şeklinde olan bu kısmı ASP.NET html sayfasına otomatik olarak ekler.
Hatırlarsak **postback** ve yeni bir sayfaya geçiş durumlarında kullanıcı bilgileri kayboluyordu. Eğer kullanıcı bilgilerinin veya kullanıcının girdiği input'ların veya yaptığı işlemlerin kaybolmasını önlemek istiyorsak, view state kısmına bu bilgileri kaydedebiliriz.
View State'te control'lerin belirli özellik değerleri otomatik olarak kaydedilir; çünkü control'lerin **EnableViewState** özellikleri **default** olarak **true'dur.** Mesela label'in **Text** özelliğine yeni değer atarsak, Label control'ü otomatik olarak Text özelliğindeki değerini view state'e kaydeder ve sayfa **posted back** olduğunda bu yeni text değeri gözükür.

The View State Collection

Bu property **Page** class'ının property'sidir. **System.Web.UI.StateBag** class'ının instance'dır. **StateBag** class'ı bir **dictionary collection'dur.** Yani her item unique bir string name ile ayrı bir slot içerisinde kayıt edilir.
Örnek:

```
this.ViewState["Counter"] = 1; // ifadesiyle ViewState collection'a kayıt işlemi yapılır.
int counter;
counter = (int)this.ViewState["Counter"]; //ifadesiyle de kaydedilen bilgi geri alınır.
```

Making View State Secure

```
<input type="hidden" value="dDw3....="/>
```

Buradaki value değerini bir çok kişi şifrelendiğini düşünür. Oysa ki, **View State** bilgileri basitçe memory'e patch edilmiştir ve **Base64** isimli özel bir string'e dönüştürülerek value attribute'sine atanmıştır. Bir hacker çok kısa bir sürede bu vale değerini çözebilir. Bu nedenle **View State** 'i daha güvenli hale getirmeliyiz.

```
<%@ Page ViewStateEncryptionMode="Always" %> //Web sayfası her zaman şifreli olur.
<%@ Page ViewStateEncryptionMode="Never" %>//Web sayfası hiçbir zaman şifreli olmaz.
<%@ Page ViewStateEncryptionMode="Auto" %> //şifreli olabilmesi için
```

Page.RegisterRequiresViewStateEncryption() metodunu sayfadaki bir control çağırmalıdır. Mesela bir button'un **click** event'inde bu metodu çağırırsak bu button sayfa post back olduğunda **hidden** tipindeki **input** value değeri şifrelenmiş olur. Eğer **Page_Load()** eventi içerisinde çağırırsak tüm sayfadaki control'lerin değerleri şifrelenmiş olur.

Retaining Member Variables

Kullanılacak Event'lar :
1. Page_PreRender()
2. Page_Load()

```
protected void Page_Load(Object sender, EventArgs e)
{
    string contents;
    if (this.IsPostBack) //Sayfa her post back olduğunda if'in içerisine girer.
    {
        // Restore variables.
        contents = (string)ViewState["contents"];
    }
}
protected void Page_PreRender(Object sender, EventArgs e)
{
    // Persist variables.
    ViewState["contents"] = contents;
}

protected void cmdSave_Click(Object sender, EventArgs e)
{
    // Transfer contents of text box to member variable.
    contents = txtValue.Text;
    txtValue.Text = "";
}
protected void cmdLoad_Click(Object sender, EventArgs e)
{
    // Restore contents of member variable to text box.
    txtValue.Text = contents;
}
```

Storing Custom Objects

```
[Serializable]
public class BenimClass
{
    //.....
}
```

Şeklinde tanımlanmış Class'lardan yaratılan nesneler **ViewState'e** eklenebilir. Bu sayede bir nesneyi bile **ViewState'e** kaydedebiliriz. Zaten **default** olarak var olan control'ler yani **button, TextBox** gibi, **ViewState'e** kaydedilmektedirler.

Kullanımı :

```
BenimClass nesnem=new BenimClass();
ViewState["nesneEkle"]=nesnem;

BenimClass seninNesnen=(BenimClass)ViewState["nesneEkle"];
```

Transferring Information Between Pages

ViewState'tin en büyük dezavantajı bir sayfaya bağımlı olmasıdır. Yani bir sayfadaki **ViewState** bilgileri bir başka sayfada kullanılamamaktadır.

Peki bir sayfadaki bilgiyi bir başka sayfada nasıl kullanacağız?

Dört Tane Teknik vardır :

1. Cross-Page Posting
2. Query String
3. Cookies
4. Session State

Cross-Page Posting

Sayfa1.aspx ve Sayfa2.aspx isimlerinde iki sayfamız olsun. Sayfa1.aspx'te bir tane **TextBox** ve **Button** ; Sayfa2.aspx'te de bir tane **Label** olsun. **TextBox'ın** değerini **Label'de** nasıl gösteririz? **Button'a** tıklanınca Sayfa2 açılacak ve Label'de Sayfa1 deki **TextBox'ın Text** property'sindeki değer gösterilecek.

Button,ImageButton,LinkButton control'lerinin **PostBackUrl** isimli bir property'si vardır.

Şu işlemlerin yapılması gerekir :

1. Sayfa2'nin ismini Sayfa1'deki **Button'un PostBackUrl'sine** yazıyoruz.
2. Bir sayfadaki control'ler **private** oldukları için bu control'lerin değerlerini dönderecek bir **public** metod veya **public Property** tanımlamak gerekir. Bu metod'u Sayfa1 de tanımlıyoruz.
3. Sayfa1'den bir nesne yaratarak bu nesne ile Sayfa1 de tanımlamış olduğumuz **public metod'u** çağırıyoruz.
4. Metod'tan dönen değeri Sayfa2 de bulunan **Label'in Text** kısmına kaydediyoruz.

Sayfa1.aspx 'teki Kodlar

```
protected void Page_Load(object sender, EventArgs e)
{
   Sayfa1 prevPage = PreviousPage as Sayfa1;
   Label1.Text = prevPage.TextBoxDegeriniGoster;
}
```

Sayfa2.aspx 'teki Kodlar

```
public partial class Sayfa2 : System.Web.UI.Page
{
     protected void Page_Load(object sender, EventArgs e)
     {

     }
     public string TextBoxDegeriniGoster
     {
          get { return TextBox1.Text; }
     }
}
```

NOT: Bu sayfadaki Button'un PostBackUrl'sine Sayfa1.aspx yazacaksınız.

The Query String

Bir sayfadan bir başka sayfaya bilgi aktarmanın bir diğer yolu query string oluşturmaktır. **Query String'te** bilgi **URL** ile gönderilir. Mesela Google web sitesinde bir search işlemi yaptığımızda yeni bir sayfaya yönlendiririliriz. Diyelim ki "organic gardening" 'i search ediyoruz. http://www.google.com/search?q=organic+gardening Url adresine sahip olan yeni bir sayfaya yönlenmiş oluruz. Aslında yönlendiğimiz yeni sayfa http://www.google.com/search? sayfasıdır . q=organic+gardening ise gönderilen bilgidir. Yani biz arama yaptığımız sayfadan . http://www.google.com/search? sayfasına q=organic+gardening bilgisini göndermiş olmaktayız.

Görüldüğü gibi **Query String ?** işaretinden sonra gelen kısımdır. Buradaki **q** harfi değişken ismidir. organic+gardening ise **q** değişkeninin değeridir.

Query String'in Dezavantajları :

1. Gönderilecek bilgi **string** türünden olmak zorundadır.
2. Bilgi görülebilir olduğu için güvenirlik yoktur. Bir kullanıcı saçma sapan bir değer girerek sayfayı crash ettirebilir.
3. Bir çok browser **1kb-2kb** arasında URL hacmi ister. Bu nedenle çok fazla bilgi gönderilemez.

Kullanıldığı Yerler :

Database uygulamaları için elverişlidir. Mesela search işleminde biz aranacak kelimeleri gireriz . Ara buton'una tıkladığımız zaman organic+gardening bilgisi http://www.google.com /search? sayfasına gönderilir. Gerekli işlemler ile organic+gardening string'i yeni sayfada elde edilir ve database'e bağlanarak gerekli bilgiler bu sayfada gösterilir. Bu tarz işlemler için kullanılır.

Kullanım Şekli :

Search.aspx Sayfası Kodları:

```
public partial class Search : System.Web.UI.Page
```

```
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnSearch_Click(object sender, EventArgs e)
    {
        string url = "SearchResult.aspx?";
        url += "q=" + txtSearch.Text;
        Response.Redirect(url);
        //Eğer iki tane bilgi göndermek istiyorsak
        string url = "SearchResult.aspx?";//Yönlendirilecek sayfanin ismi ve ismin sonuna ? isareti konur.
        url += "q=" + txtSearch.Text+"&";//& işaretine dikkat. q= ifadesine dikkat
        url += "baslik=" + "Bitkiler";//baslik= ifadesine dikkat.
        Response.Redirect(url);
    }
}
```

SearchResult.aspx Sayfası Kodları:

```
public partial class SearchResult : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = Request.QueryString["q"]; //q bilgisi alınır. Yani txtSearch.Text bilgisi.
        Label1.Text = Request.QueryString["baslik"]; //baslik bilgisi alınır. Yani  "Bitkiler"
    }
}
```