WPF Virtualization Kavramı

WPF'in list controlleri tarafından sağlanan en önemli özellik UI virtualization'dır. Bu teknik büyük ölçekli listelerin tamamının memory'e yüklenmesi yerine sadece ekranda gösterilen item sayısı kadar memory'e yüklenmesini sağlar. Örneğin, 50.000 item sayısına sahip bir listeyi memory'e yüklemek performans kayıplarına sebep olacaktır. Bundan dolayı WPF bu tekniği geliştirmiştir. Bu teknikle ekranda gösterilen item sayısı 30 ise, sadece 30 tane item yaratılıp memory'de tutulması sağlanır.

Virtualization olmasaydı 50.000 tane item yaratılıp memory'e yüklenecekti. **ListBox**, **ListView** ve **DataGrid** elementleri, elemanlarını (child elementlerini) yerleştirirken, otomatik olarak **VirtualizationStackPanel** sınıfını kullanır. Bu sınıf virtualizationun implement edildiği sınıftır. Bundan dolayı virtualization kullanmak için bu elementleri kullanırken ekstra bir işlem yapmaya gerek yoktur. Ancak **ComboBox** standart nonvirtualizated **StackPanel** kullandığı için bu özelliği desteklemez. Eğer **ComboBox**'ın bu tekniği kullanmasını istersek aşağıdaki gibi kod yazmalıyız:

```
<ComboBox>
    <ComboBox.ItemsPanel>
        <ItemsPanelTemplate>
            <VirtualizingStackPanel>
            </VirtualizingStackPanel>
        </ItemsPanelTemplate>
    </ComboBox.ItemsPanel>
</ComboBox>
```

Bazı sınıflar virtualization'u desteklemelerine rağmen, varsayılan olarak bu özellik devre dışı bırakılmıştır. Örneğin **TreeView** sınıfı virtualization'u desteklemektedir; ancak bu özellik devre dışıdır. Aktif hale getirmek için `<TreeView VirtualizingStackPanel.IsVirtualizing="True" ... >` şeklinde özellikle belirtmeliyiz.

Not: WPF'in eski sürümlerinde **VirtualizingStackPanel** default olarak hiyerarşik veriyi desteklememektedir. Geriye dönük uyumluluk problemleri çıkmaması için bu özellik **TreeView** sınıfında devre dışı bırakılmıştır.

Not: ScrollViewer içerisine list control'ümüzü koyduğumuzda virtualization gerçekleşmez. Ayrıca item'ları kodsal olarak yaratırsak, virtualization uygulanmayacaktır.

Item Container Recycling

Bir item yaratıldığı zaman onun container nesnesi de yaratılmaktadır. Eğer recycling aktif edilirse, container nesneleri baştan yaratılmaz. Sadece item'lar yaratılır. Bundan dolayı performans artışı olur. Bu özelliği aktif hale getirmek için **VirtualizingStackPanel.VirtualizationMode** property **true** yapılmalıdır:

```
<ListBox VirtualizingStackPanel.VirtualizationMode="Recycling" ... >
```

Bu özellik aktif edildiği zaman, scrolling performansı artarken, memory tüketimi azalır. Çünkü garbage collector item'ları silmez. **DataGrid** hariç default olarak tüm controllerde pasif haldedir. Eğer büyük bir listemiz varsa bu özelliği aktif etmeliyiz.

Cache Uzunluğu

WPF 4.5 ile birlikte gelen iki property vardır: **CacheLength** ve **CacheLengthUnit.** WPF virtualization özelliğinde scrolling kullanılırken gösterilen item'lara ek olarak bir kaç tane daha item yaratılır. **CacheLength** ve **CacheLengthUnit** property'ler tam olarak kaç tane fazladan item yaratılacağını belirtmemizi sağlar.

```
<ListBox VirtualizingStackPanel.CacheLength="1"
         VirtualizingStackPanel.CacheLengthUnit="Page" ... />
```

CacheLengthUnit değeri Page olarak belirlenmiştir. **CacheLength** değeri de 1 olduğu için 1 Page anlamına gelir. Daha açık ifade edecek olursak, ekranda gösterilen item'lara ek olarak, bu item'lardan önce ve sonra 1 Sayfa daha fazla item yaratılacağını ifade eder. **Toplamda 2** sayfa yaratılacaktır. 1 Sayfadaki item sayısı ekranda o an gösterilen item sayısı kadardır. Örneğin ekranda 30 tane item gösteriliyorsa, 1 Sayfa değeri 30 olacaktır.

CacheLengthUnit, **Page, Item ve Pixel** değerlerinden birini alır: **Item** olduğu zaman kaç tane item olacağını direkt belirleyebiliriz:

```
<ListBox VirtualizingStackPanel.CacheLength="100"
         VirtualizingStackPanel.CacheLengthUnit="Item" ... />
```

Bu örnekte ekranda gösterilen item sayısına ek olarak bu item'lardan önce 100 tane ve bu itemlardan sonra 100 tane daha item yaratılır.

Cache özelliği ile scrolling işlemi daha performanslı olur.

Deferred Scrolling

Scrolling performansını artırmak için kullanıcının scollbar'daki butonu bıraktığı zaman yeni item'lar yaratılmasını sağlayabiliriz. Bu özelliğin aktif edilmesini istiyorsak **ScrollViewer.IsDeferredScrollingEnabled** property'sini kullanabiliriz:

```
1    <ListBox ScrollViewer.IsDeferredScrollingEnabled="True" ... />
```

Not: WPF'te scrolling işlemleri item tabanlı olmaktadır. Yani scrollu aşağı kaydırdıkça yeni item'lar yaratılır. Bir item'ın yarısı gösterilmez bütün olarak gösterilir. Eğer yaratılan item'lar büyük ise pixel tabanlı gösterim sağlayabiliriz. Bu sayede scroll'u aşağı doğru kaydırdıkça item'ın bir parçasını görebiliriz:

```
1    <ListBox VirtualizingStackPanel.ScrollUnit="Pixel" ... />
```

Sonuç

Bu makalede WPF Virtualization ile collection'ları daha efektif bir şekilde nasıl listeleneceği hakkında bilgi verilmiştir.