

Windows Phone

12:38

Uygulama Geliştirme Kılavuzu



Microsoft

Student Partners

Türkiye

Editör' den...

Başarmak denilen bitimsiz şevkin alt yapısı; dün de, bugün de ve elbet yarın da, azim ve çalışmadan geçmekte... Buna sağduyu sahibi hiç kimsenin de itiraz etmediği ve edemeyeceği bir gerçek! Değişen ve değişken olan; zaman ve azimle birlikte “neyi? ne zaman? nereye kadar? nasıl?” çalışacağı sorularıyla birlikte, insanın merak ederek araştırmaya yönlenmesidir.

İşte, dün bu sorulara muhatap olan insanoğlu; bugün bu sorular noktasında, istese de istemese de teknoloji ile karşılaşmakta.Teknolojiye sırtını dönmemek, akıllı insanın olmazsa olmazı...

Teknoloji ile barışık olmak, teknolojiye ayak uydurmak, çağı yakalamak isteyen ülkelerin de vazgeçilmezi elbet!

Bir toplumu, millet yapan değerler manzumesini ifade etmek, kültürel mirasa sahip çıkmak bile, ilerleyen dünyada iyi bir yer edinmekle ve ilerleyen dünyaya ayak uydurmakla mümkün. İlerleyen dünyaya ayak uydurmaksa teknolojidten geçmekte...

O halde ne kadar çok teknoloji, o kadar çok başarı; ne kadar başarı, o kadar çok üretme... Her yeni üretim ise, bir diğerini tetiklemekte,bir diğerini teşvik etmekte...

Bugün, bu hızlı ve baş döndürücü çalışmada hüsrana uğramamak için “kalite” denilen beklentilere sadece cevap vermeli değil, beklentiler aşılmalıdır da... Çünkü; insanların da, ülkelerin de beklentileri her karşılandığında, yenisine davetiye çıkarılmış demektir.

Evet, Microsoft 30 yıl önce yani 1983'te işletim sistemini önce sadece bilgisayar için geliştirdi ve kullandı... Sonra cep telefonunda kullanılmak üzere “Windows Phone” işletim sistemini geliştirdi... Çok da tutularak geniş kitlelerce rağbet gördü... Neden? Tabi ki kalitesinden!!! Dinamik bir kavram olan kalitenin tecrübeli ve kendini yenileyen bir kuruluşla ve tabi ki yönetim kalitesine dönüşmesiyle...

Teknoloji dünyasının baş döndürücü bir şekilde geliştirmesi, biz yazılımcıların da kendimizi yenilememizi ve bizlerin de beklentilere cevap vermemizi gerektirmekte. Mobil sistem; hem iyi, hem de pek çok zor yanları beraberinde getirmekte... Sözgelimi, kodlama yaparken, takıldığımız ya da hatırlamadığımız kısımlarda çoğu zaman elimizin altında derli toplu bir kaynağın bulunmasını isteriz...

Bizler de “*Microsoft Student Partner Türkiye*” ekibi olarak sizlerin yararlanabileceği bir kitap hazırlama kararı verdik.

Her konunun başlangıcında, ilgili kısımdaki yazarın, o bölüm hakkında kısa bir bilgilendirme yazısı da mevcut...

Öncelikle; bizleri “Student Partner” programında yalnız bırakmayan, bizlerden bu kitabımızda olduğu gibi, hiç bir konuda yardımlarını esirgemeyen, ışığı ile bizleri aydınlatan, bizlere örnek bir rehber olan, *Microsoft Türkiye Akademik Programlar Yöneticisi Sayın Dr. Mustafa KASAP Hocamız’ a*, tüm Microsoft Student Partner arkadaşlarımız adına sonsuz şükranlarımızla, saygılar sunarız...

Kitabımızın el birliği ile oluşmasında yardımcı olan, aşağıda listelediğimiz Microsoft Student Partner yazar arkadaşlarımıza teşekkürü borç biliriz...

1. Giriş (Mehmet Ali Afşar Meral – Melikşah Üniversitesi)
2. Genel Kavramlar (Tahsin Kasap – Pamukkale Üniversitesi)
3. Araçlar (Gazi Uslu – Ege Üniversitesi)
4. Kontroller, Yatay-Düşey Durum ve Temalar (Aybala Şeyda Sert/İlkay Taşkıran – Erciyes Üniversitesi)
5. Touch Input (Alp Arslan Eren – İzmir Ekonomi Üniversitesi)
6. Application Tile ve Notification Servisler (Burak Gönüldaş – Selçuk Üniversitesi)
7. Sensörler ve Servisler (Doğukan Demir – Doğu Akdeniz Üniversitesi)
8. Tasklar (Can Özdemir – Çankaya Üniversitesi)
9. Application Bar (Merve Özel – Kadir Has Üniversitesi)
10. Veri Bağlama İşlemleri (Doğukan Demir – Doğu Akdeniz Üniversitesi)
11. Animasyon (Can Eyüpoğlu – İstanbul Kültür Üniversitesi)
12. Panoramik Uygulamalar (Göknur Aslan – Çukurova Üniversitesi)
13. Web Bağlantıları (Ülkü Buket Nazlıcan – Boğaziçi Üniversitesi)
14. Veri Saklama ve Senkronize Etme (Nagihan Sema Kudu – Karadeniz Teknik Üniversitesi)
15. Multitasking (Nagihan Sema Kudu - Karadeniz Teknik Üniversitesi)
16. Marketplace
17. Güvenlik (Alp Arslan Eren – İzmir Ekonomi Üniversitesi)
18. Sayfalar Arası geçiş ve Veri Transferi (Tahsin Kasap – Pamukkale Üniversitesi)
19. Cloud ile İşlemler (Mehmet Ali Afşar Meral – Melikşah Üniversitesi)
20. Harita ile İşlemler (Mehmet Ali Afşar Meral – Melikşah Üniversitesi)

Ayrıca, kitabı gözden geçirerek, bazı kısımlarında düzeltme yaparak yardımlarını esirgemeyen Microsoft Student Partner sevgili *Yunus Emre ARAÇ* ve *Onur TIRPAN* arkadaşlarımıza da teşekkür ederiz.

Kitap hakkında geri dönüşlerinizi yardim@msakademik.net adresine gönderebilirsiniz. Kitabımızı beğenerek okumanız ve yararını görmemiz dileğiyle...

Mehmet Ali Afşar MERAL
Melikşah Üniversitesi Bilgisayar Mühendisliği
Microsoft Student Partner

İçindekiler

1.	Giriş.....	9
2.	Genel Kavramlar.....	11
	a. İşletim Sistemi.....	11
	b. Silverlight.....	14
	c. XNA.....	15
	d. Araçlar.....	17
3.	Araçlar.....	19
	a. Windows Phone 7 SDK.....	19
	b. Expression Blend.....	20
	c. Emulator.....	21
	d. "Merhaba Dünya".....	21
4.	Kontroller.....	27
	a. Yatay-Düşey Konum.....	27
	i. Düşey Durum (Portrait).....	27
	ii. Yatay Durum (Landscape).....	29
5.	Touch Input.....	32
	i. Silverlight Uygulamaları.....	32
	ii. XNA.....	32
	El Hareketleri (Gestures).....	34
6.	Application Tile ve Notification Servisler.....	39
	a. Application Tile.....	39
	i. Application Tile Genel Özellikleri.....	40
	ii. Ekleme, Çıkarma ve Güncelleme.....	42
	b. Notification Servisler.....	44
	i. Notification Çeşitleri.....	45
	1. Toast Notifications.....	45
	2.Tile Notifications.....	46
	3.Raw Notifications.....	46
	ii. Örnekler.....	46
	1. Toast Notifications Örneği.....	46
	2. Tile Notifications Örneği.....	51
	3. Raw Notifications Örneği.....	54

7.	Sensörler ve Servisler	60
	a. SensorBase Sınıfı	60
	b. Accelerometer(İvme Ölçer).....	61
	c. Compass (Pusula).....	62
	d. Gyroscope (jiroskop).....	63
	e. Motion (Hareket).....	64
8.	Tasklar.....	68
	CameraCaptureTask.....	71
	Photo Chooser Task.....	72
	Phone Call Task.....	73
	Phone Number Chooser Task.....	75
	Save Phone Number Task.....	76
	SMS Compose Task	77
	E-Mail Adress Chooser Task.....	79
	E-Mail Compose Task.....	80
9.	ApplicationBar	83
10.	Veri Bağlama İşlemleri	89
11.	Animasyon	93
	Frame Tabanlı ve Zaman Tabanlı.....	93
	Tıklama ve Dönme.....	94
	Animasyon Özellikleri.....	96
	XAML-Tabanlı Animasyonlar	97
	Key Frame Animasyonlar.....	97
	Perspektif Dönüşümleri	98
	Animasyonlar ve Özellik Önceliği	99
12.	Panoramik Sayfalar.....	102
	Panoramik Sayfalar Hakkında.....	102
	Panoramik Sayfaların Yapısı	102
	Panoramik Sayfalardaki Blok Yapısı	103
	Panorama	103
	Panoramik Uygulama Oluşturulması.....	104
13.	Web Bağlantıları.....	106
	a. Genel Bilgiler.....	106
	b. Web ve Data Servisleri	106

	c. Rss Uygulaması Geliştirme	107
14.	Veri Saklama ve Senkronize Etme.....	114
	A. Ayarlar	115
	Isolated Storage' yi Kullanmak için Yardımcı Sınıf Tanımlama.....	115
	B. Dosya İşlemleri.....	117
	C. Veri Tabanı İşlemleri	120
	1. Veri Tabanı Model ve Görünümünün Oluşturulması.....	120
	2. Başlangıç Sayfasının	126
	3. Ana Sayfanın Oluşturulması	128
	4. Panorama Sayfasının.....	134
	5.Panorama Sayfası Kontrollerinin Oluşturulması.....	139
	6. Yeni Ürün Oluştur Sayfasının Hazırlanması	145
	7. Yeni Alış-veriş Listesi Ekle Sayfasının Hazırlanması	147
	8. Ürün Bilgisi Düzenleme Sayfasının Oluşturulması.....	148
	9. Mağazalar Sayfası	150
	10. Mağaza Ekle Sayfası.....	152
	11. Sevilen Ürünler Sayfası.....	153
15.	Multitasking.....	157
	Arka Planda Yürütülen Müzikler	157
	Planlanmış Görevler	157
	Arka Plan Dosya Transferleri	157
	Planlanmış Bildirimler	157
	Uygulamalar Arası Hızlı Geçiş	157
16.	Marketplace.....	159
	a..... Uygulamaların YallaApps/Marketplace Üzerinde Yayınlanması	159
	b.MarketPlace Uygunluk testleri.....	161
17.	Windows Phone 7'de Güvenlik	164
	Çember Mimarisi.....	164
	Capabilities	165
	Uygulamalar Arası İletişim ve Multitasking.....	165

	Uygulama Kurulumu.....	165
	Veri Korunumu	166
	Microsoft Exchange ActiveSync (EAS)	167
18.	Sayfalar Arası Geçiş ve Veri Transferi	169
19.	Cloud ile İşlemler	176
20.	Harita İle İşlemler	188



*Bu bölümde Windows Phone 7' programlaya geçmeden önce biraz geriye gideceğiz. Akıllı telefonların öneminden, akıllı telefonlar içinden Windows Phone 7' nin öneminden bahsediyor olacağız. Windows Phone donanımında ne gibi özellikleri barındıyor bunlara göz atıyor olacağız...
Yararlı olması dileğiyle...*

Yazardan...

1. Giriş

Geçmişten günümüze bütün bilgisayar ve bilgisayar sistemlerinin temelinde 1 lerin ve 0 ların olduğunu, taa ki kuantum bilgisayarların çıkmasına kadar süreceğini bilmemiz gerekir. Platformların, işlenen bit sayılarının, işlemci sayılarının değişiyor olması; temelinin 1 ler ve 0 lardan oluştuğu gerçeğini değiştirmez. Hem taşınabilir olup hem de işlevselliğinizi sürdürdüğünüz cihazlar olan mobil cihazlarında mimarisi bunun gibidir.

Telefonların ilk çıktığı zamanlar, insanlara bugün ki teknoloji den bahsetseydik, 2 kişinin yüz yüze olmadan nasıl iletişimde olduğunu kavramaya çalıştıkları sırada, bu olaylara anlam veremeyeceklerdi. İşte tam bu nokta da mobil cihazlar ile artık sadece telefon görüşmesi ya da kısa mesaj göndermenin dışında pek çok özellekle beraber geliyor. GPS ile yönünüzü bulabiliyor, geceleri el fenerimiz olabiliyor, ihtiyacımız olduğunda anı ölümsüzleştirmek için fotoğraf makinamız oluyor, Wireless ya da 3G-4G-LTE ile internete hızlı bir şekilde bağlanıp bilgi edinebiliyor, uzaktan bir bilgisayara bağlanıp onu yönetebiliyor, görselliğiyle zenginleşen oyunlar ile eğlenip vakit geçirebiliyor, sosyal medya ile eş zamanlı bağlı olup haberler alabiliyor ve belki de sayamadığım pek çok özelliği yapabiliyoruz. Bu hususları da en iyi yapmaya çalışan ve bu yolda sağlam adımlarla ilerleyen Windows Phone 7' nin püf noktalarını sizin dilinizden anlatıyor olacağız.

Windows Phone 7 ile uygulama yapacaksınız, öncelikle Windows Phone 7' yi tanımalısınız. Windows Phone 7, Microsoft' un Windows Mobile' dan sonra çıkarttığı, tamamen farklı bir işletim sistemidir. Bu farklılıkların kullanıcı tarafındaki en belirgin özelliği Metro UI ara yüzüdür. Metro UI, ismini metro istasyonlarından almaktadır. Metro istasyonlarında, hava alanlarında, şehirlerarası terminalerde kullanılan semboller kullanılmıştır. Bu semboller bütün insanların ortak kullandıkları ve bildikleri semboller. Anlaşılmayan küçük yazı ve sembollerini kullanmak yerine bu sembollerini kullanmak gayet hoş ve başarılı olmuştur.

Windows Phone 7 işletim sistemi ile belli donanımsal standartlar getirilmiştir. Bu standartların altında kalan cihazlar Windows Phone 7 işletim sistemine sahip olamıyorlar. Bu standartlar ise şu şekilde;

- DirectX 9 destekli en az 1Ghz Qualcomm Snapdragon işlemci
- Ekran çözünürlüğü 800x480(WVGA)
- Kapasitif multitouch bir ekran (En az 4 temas noktası)
- En az 5 fiziksel buton(kamera tuşu, güç tuşu, arama tuşu, start ve geri tuşu)
- En az 256 mb RAM (Mango' da 512mb)
- En az 8GB depolama alanı
- En az 5 mp kamera(flaşlı)
- FM radyo
- WiFi
- Accelerometer(iveme ölçücü)
- AGPS

İsterseniz şimdi, hızlı bir şekilde konularımıza başlayabiliriz...



Kitabımızın bu bölümünde, Windows Phone işletim sisteminin tasarım konseptinden ve kullanıcılara ne gibi yenilikler getirdiğinden bahsediyor olacağız. Geliştiriciler tarafında ise kısaca Silverlight ve XNA' e değinip, Windows Phone için uygulaması geliştirmek için gerekli olan araçlara kısaca değiniyor olacağız.

2. Genel Kavramlar

a. İşletim Sistemi

Microsoft' un mobil alandaki çalışmaları Windows Mobile ile başlamıştır. Windows Mobile işletim sisteminin 2008 yılına kadar çeşitli sürümleri çıkmıştır. Ancak gelişen mobil sektörde Windows Mobile' ın geri kalması, Microsoft' u büyük bir karar alarak yeni bir işletim sistemine odaklanmasını sağlamış ve Windows Phone 7' nin temelleri atılmıştır.

Sıfırdan yazılan işletim sistemi, son kullanıcının bütün ihtiyaçları düşünülerek hazırlandı. Tüketicileri odak noktasına koyan işletim sistemi iki temel unsur üzerine inşa edildi:

- Smart Desing
- Integrated Expreience

Smart Desing; Windows Mobile işletim sisteminden sonra görsel olarak farklı bir arayüz düşünülmüştür. Herkesin rahatlıkla kullanabileceği, telefonu ilk defa eline alan birinin bile daha önce aynı telefonu kullanmış biri kadar rahat etmesi amaçlanmaktadır. Bu düşünceden yola çıkarak insanların ortak kullandığı yerlerdeki işaretler incelenmiştir. Metro ve havaalanlarını düşünün orada daha önce bulunmamış bile olsanız rahatlıkla yönünüzü tayin edebiliyorsunuz. Tüm bunlar baz alınarak ve farklı kullanıcılar üzerine yapılan çalışmalar ile Metro arayüzü oluşturulmuştur.

Integrated Expreience; Windows Phone 7'deki diğer bir konsept ise kullanıcıların favori sitelerini ve uygulamalarını sosyal ağlar ile bir arada tutmaktır. Sosyal ağların gücünü düşündüğümüzde bu tarz bir konseptin ne kadar önemli olduğunu anlayabiliriz.

Telefonda bulunan "Start" tuşuna basıldığında karşımıza gelen ekranı incelediğimizde kullanıcıların kısa yollarını barındıran ve sürekli olarak güncel olan bir yapı karşımıza çıkmaktadır.



Kısa yollardan bir kaçına değinecek olursak;

People; kısmında Live ve Facebook hesaplarınızı ilişkilendirerek sosyal ağınızdaki güncellemeleri hızlı bir şekilde takip edebilirsiniz.



Office ile Office ile Word, Excel, Powerpoint dosyalarınızı düzenleyip paylaşabilirsiniz. Ayrıca Outlook ile e-posta hesaplarınızı rahatlıkla kontrol edebilirsiniz.



Pictures; bu kısımda resimlerinizi ve videolarınızı görüntüleyip sosyal ağlarda paylaşabilirsiniz. Ayrıca bu kısa yol içinden ulaşılabilen uygulamalar geliştirebilirsiniz.



Music – Video; tahmin edeceğimiz üzere medya uygulamalarımızın olduğu bu kısayol da isterseniz online müzik servisi olarak Zune'u kullanılabilirsiniz. Ayrıca bir medya

uygulaması geliştirdiyse, bu uygulama isterseniz start menüde isterseniz bu kısa yolun içinde gözükmektedir.



Marketplace ile geliştireceğiniz uygulamalar dahil olmak üzere telefona yeni uygulamalar ve oyunlar indirebilirsiniz. Ayrıca telefonunuza ve uygulamalara gelen güncellemelerini kontrol edebilirsiniz.



Birazda durumu geliştiriciler tarafında ele alalım; Windows Mobile işletim sisteminde geliştiriciler Windows form tarzında uygulamalar geliştiriyorlardı. Hatta bu uygulamaları emülatör olmaksızın Windows form uygulaması gibi bilgisayarınızda çalıştırabiliyordunuz.

Windows Phone 7' de durum biraz farklı Microsoft'un iki güçlü teknolojisi ile geliştirme yapabiliyoruz. Silverlight ile zengin görsel içeriğe sahip uygulamalar geliştirebilirken, DirectX kütüphanelerini rahat bir şekilde yönetmemizi sağlayan XNA Game Studio ile de oyun geliştirebiliyorsunuz.

Özetleyecek olursak Windows Phone 7 işletim sistemi, kullanıcılara daha iyi bir deneyim yaşatmak için Windows Mobile' dan sonra köklü değişikliklere gitmiş bir işletim sistemidir.

b. Silverlight

Windows Phone için uygulama geliştirmek isteyenlerin tercih edebileceği teknolojilerden biri olan Silverlight 'ı tanımak için biraz geriye Windows Vista'nın çıkışına gidelim. Vista'dan önceki form uygulamaları görsel açıdan pek zengin değildi. Uygulama ara yüzleri GDI(Graphics Device Interface) kullanılarak çizdiriliyordu. Bu kısımda geliştiricinin kendi istediği şekilde bir çizim yaptırması, uygulamadaki bir butona istediği şekli vermesi son derece zor ve karmaşık bir işti.

Windows Vista ile kullanılmaya başlanan WPF(Windows Presentation Foundation) bir devrim niteliği taşımaktadır. Win Form uygulamalarından farklı olarak görsel ara yüzler GDI veya GDI+ API' leri yerine WPF' te DirectX kütüphaneleri kullanılarak çizdirilmektedir. Bu şekilde uygulamaların görsel içerikleri zenginleştirilebilmektedir.

Basit bir örnek vermek gerekir ise bir Windows Form uygulaması için video oynatmak istediğinizde GDI video ve ses işlemlerini desteklemez bunun için ekstra olarak form içinde Windows Media Player referanslarını kullanmalısınız. WPF ise video ve ses işlemlerini direkt olarak desteklemektedir.



Masaüstü uygulamalarda grafik desteği böylesine iyi olan bir teknoloji varken, web uygulamalarında benzer bir teknoloji var mı sorusu aklınıza gelebilir? İşte tam bu noktada Silverlight devreye girmektedir. Silverlight zengin görsel içeriğe sahip web uygulamaları geliştirebileceğiniz bir teknolojidir. WPF ile benzer özelliklere sahiptir.

Windows Mobile' da uygulamaların masaüstü uygulamaları ile benzer yapıda olduğunu söylemiştik. Tam bu noktada farklılık yaratmak isteyen Microsoft, Windows Phone 7 için görselliği ile ilgi çekici uygulamalar geliştirilebilecek olan Silverlight'ı seçmiştir.

Silverlight ile uygulama geliştirmek istediğimizde karşımıza C#, VB.NET gibi programlama dillerinin yanı sıra XAML (Extensible Application Markup Language) çıkmaktadır. XAML Silverlight projelerinde kullanıcı ara yüzünü oluşturmak için kullanılan XML tabanlı bir dildir.

Silverlight projelerinde XAML ve .NET dilleri arasındaki en büyük ilişki kullanıcı ara yüzünü oluştururken XAML ile yazdığımız buton, textbox gibi elementlerin özelliklerine kod tarafından rahatlıkla erişebilirsiniz.

```

13 SupportedOrientations="Portrait" Orientation="Portrait"
14 shell:SystemTray.IsVisible="True"/>
15
16 <!--LayoutRoot is the root grid where all page content is placed-->
17 <Grid x:Name="LayoutRoot" Background="Transparent">
18     <Grid.RowDefinitions>
19         <RowDefinition Height="Auto"/>
20         <RowDefinition Height="*"/>
21     </Grid.RowDefinitions>
22
23     <!--TitlePanel contains the name of the application and page title-->
24     <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
25         <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"/>
26         <TextBlock x:Name="PageTitle" Text="page name" Margin="0,-7,0,0" Style="{StaticResource PhoneTextTitleStyle}"/>
27     </StackPanel>
28
29     <!--ContentPanel - place additional content here-->
30     <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"/></Grid>

```

```

14 namespace PhoneApp3
15 {
16     public partial class MainPage : PhoneApplicationPage
17     {
18         // Constructor
19         public MainPage()
20         {
21             InitializeComponent();
22             page
23         }
24     }
25 }

```

XAML ve .NET dilleri arasındaki bu ilişki bize kod tarafından da görsel alanlara müdahale edebilmemizi sağlamaktadır. Bu geliştiricilerin aklında benim XAML bilmeden uygulama geliştirebilir miyim sorusunu getirebilir. Her ne kadar XAML bilmeden uygulama geliştirebileceğiniz doğru olsa da karşılaşacağınız bazı sorunları XAML tarafında değişiklik yaparak çözmek zorunda kalabilirsiniz. Bu yüzden XAML hakkında bilgi sahibi olmakta Windows Phone uygulamaları için son derece önemlidir.

c. XNA



XNA başlangıçta Windows ve XBOX' a .NET altyapısını kullanarak oyun geliştirmek için hazırlanmış olan kütüphaneler bütünüdür . XNA ile C# ve VB.Net dillerini kullanarak oyun geliştirebilmektedir.

XNA ile oyun geliştirirken XNA'ın bir oyun motoru olmadığını unutmamalıyız. XNA .NET altyapısını

kullanarak DirectX kütüphaneleri rahatça kullanmamızı sağlamaktadır. Bir oyun motorundan ziyade ekran kartına grafikleri çizdirmek için gerekli olan komutları bizim yerimize gönderir. Biz ise oyunumuzda nerde ne olacak ve nasıl çalışacak kısmı ile ilgileniriz.

Windows Phone 7 için Silverlight ile oyun geliştirebileceğiniz gibi XNA'yi kullanarak da oyun geliştirebilirsiniz. Ancak XNA bize Silverlight' tan farklı bir yapı sunmaktadır.

Oyunlarımız aslında gözümüzün algılayamayacağı aralıklarda resimlerin tekrar tekrar çizdirilmesi ile oluşmaktadır. Windows Phone' da bu değer saniyede 30 kare olarak tanımlanmaktadır. İnsan gözü yaklaşık olarak 40 milisaniyenin altındaki değişimleri seçememektedir ki bu yaklaşık olarak saniye de 25 kareye denk gelmektedir. Bu düşünüldüğünde saniyede 30 kare gayet iyi bir değer diyebiliriz.

Windows Phone için oyun projenizi oluşturduğunuzda Visual Studio sizin için bir Game sınıfı oluşturacak ve bu sınıf içindeki metotları kullanarak oyun geliştirmeye başlayabilirsiniz. Bu metotlardan bazılarını kısaca inceleyecek olursak;

- LoadContent,. oyunumuz başladığında bir kez çalışan bu metotta görsel içeriklerimizi değişkenlerimize yükleyebiliyoruz
- Update metodu ise daha oyunuzun FPS değerine bağlı olarak çağırılmaktadır. Bu metot içinde kullanıcının davranışlarını (touch vs.) algılayıp ve oyunu yönlendirebilirsiniz.
- Draw ise yine oyunumuzun FPS değerine bağlı olarak çağırılır ve oyunumuzun görüntülerinin ekranda çizdirilmesini sağlar.

Bu temel metotların yanı sıra XNA bize sağladığı hazır metotlar ile matematiksel olarak hesaplama yapmamız gereken bazı işlemlerden kurtarmaktadır. Örneğin; hazırladığımız oyunda mermiler düşman gemileri ile çarpıştığında bir patlama efekti vereceğimizi düşünelim. Bu çarpışma durumunun şartlarını mantıksal operatörler kullanarak hesaplamamız bizim için büyük bir sorun. Bu tarz durumlarda XNA imdadımıza Intersect metodunu sunmaktadır. Intersect oyun içindeki nesnelimizin kesişim durumunu algılamaktadır..

XNA ile Windows üzerinde çalışan bir oyun geliştirdiniz ve oyunun Windows Phone için de güzel bir oyun olacağını düşünüyorsunuz. Bu noktada Windows phone için oyunu sıfırdan yazmanıza gerek kalmamaktadır. Visual Studio' da Solution Explorer penceresinde hazırlamış olduğunuz projeye sağ tıkladığınızda karşınıza çıkacak olan “Create Copy of Project for Windows Phone ” seçeneği ile oyununuzu Windows Phone 7’de çalışabilecek hale getirebilirsiniz. Bu işlemi yaparken oyunun kontrollerini WP7’ye göre ayarlamayı unutmamalıyız. Çünkü bilgisayarda oyunumuzu klavye ve mouse tan gelen bilgilere göre yönlendirmekteyiz. Oyun kontrollerimizi dokunmatik ekrana göre ayarlamalıyız.

d. Araçlar

Windows Phone 7 uygulamaları geliştirebilmek için kullanabileceğimiz yazılım teknolojilerinden bahsettik. Bu teknolojiler ile uygulama geliştirebilmemiz için gerekenler nelerdir kısaca bir göz atalım.

Bahsedeceğimiz araçların Windows Phone 7 ile ilgili olarak kullanılmak üzere özelleştirilmiş versiyonları SDK ile birlikte yüklenmektedir. create.msdn.com adresinden SDK'nın son sürümüne ulaşabilirsiniz. Araçlardan kısaca bahsetmek gerekirse;

Visual Studio: .NET geliştiricilerinin uygulamalarını geliştirmekte kullandıkları IDE olan Visual Studio temel aracımızdır. Windows Phone 7 uygulaması geliştirirken Visual Studio 2008 ve daha aşağı sürümleri ile uygulamalarımızı gerçekleştiremiyoruz. Ancak bu uygulama geliştirmek için bilgisayarımızda Visual Studio 2010'un yüklü olması gerektiği anlamına gelmiyor. SDK kurulumunu gerçekleştirdiğimizde Visual Studio 2010 Express for Windows Phone sürümü bilgisayarımıza yüklenmektedir.

Visual Studio 2010 ile çalışan geliştiriciler ise SDK'yı kurduklarında proje oluşturma aşamasında Windows Phone 7 ile ilgili olarak proje taslaklarının geldiğini görecektirler. SDK 7.1 kurmak istiyorsanız ve bilgisayarınızda Visual Studio 2010 yüklü ise öncelikle Visual Studio 2010'un service pack 1 güncelleştirmesini yüklemeniz gerekmektedir.

Emülatör: Uygulamalarımızı test etmek amacıyla bilgisayarımıza bir sanal Windows Phone 7 kurulmaktadır. Bu sayede telefona ihtiyaç duymaksızın uygulamalarımızı test edebilmekteyiz. SDK 7.1 ile sensörleri de emülatör üzerinde kullanıp testlerinizi gerçekleştirebilirsiniz.

Expression Blend: Windows Phone 7 uygulamaları Silverlight'tan gelen zengin bir görsel içeriğe sahiptir. Microsoft tarafından tasarımcılar düşünülerek geliştirilmiş olan Blend ile uygulamanızdaki görsel öğeleri ve animasyonları rahatlıkla düzenleyebilirsiniz. Ayrıca Blend içinde gerek XAML kodlarını gerekse arka planda kullandığımız C# ya da VB.NET kodlarını düzenleyebilmekteyiz. Kitabın sonraki bölümlerine ayrıntılı olarak bahsedilecek olan Blend bilgisayarınızda yüklü değil ise SDK ile Windows Phone 7 için özel sürümü gelmektedir.

Windows Phone 7 uygulamaları geliştirmek için bilgisayarların minimum olarak belli özelliklere sahip olması gerekmektedir. Bu özellikleri sıralayacak olursak;

- Windows 7 ve Windows Vista işletim sistemi
- Windows Vista: Service Pack 2 yüklü olmak şartıyla Starter versiyonu hariç tüm versiyonlar
- Windows 7: Starter versiyonu hariç tüm versiyonlar
- SDK kurulumu için 4GB alan
- 3 GB Ram
- DirectX10



Bu bölümde mobil uygulama geliştirmeye başlamadan önce bilgisayara kurulması gereken yazılımlar hakkında bilgi verilmek ve basit bir uygulamayla mobil dünyaya giriş yapmak amaçlanmıştır. Bu kapsamda Visual Studio ortamında Windows Phone 7 uygulaması hazırlayabilmek amacıyla gerekli olan SDK'nın kurulumu, ardından mobil ortamda ilk uygulamanın yazılması daha sonra da uygulamaya animasyon tarzı görselliklerin eklenebilmesi için gerekli ortam olan Expression Blend'in tanıtımı ve son olarak da geliştirilen uygulamanın cihaz üzerinde nasıl çalışacağıнын bir simülasyonunu sağlayan emulatörden bahsedilmiştir.

Yazardan...

3. Araçlar

a. Windows Phone 7 SDK

Bu bölümde mobil uygulama geliştirmeye başlamadan önce bilgisayara kurulması gereken yazılımlar hakkında bilgi verilmek ve basit bir uygulamayla mobil dünyaya giriş yapmak amaçlanmıştır. Bu kapsamda Visual Studio ortamında WP7 uygulaması hazırlayabilmek amacıyla gerekli olan SDK'nın kurulumu, ardından mobil ortamda ilk uygulamanın yazılması daha sonra da uygulamaya animasyon tarzı görselliklerin eklenebilmesi için gerekli ortam olan Expression Blend'in tanıtımı ve son olarak da geliştirilen uygulamanın cihaz üzerinde nasıl çalışacağıнын bir simülasyonunu sağlayan emulatörden bahsedilmiştir.

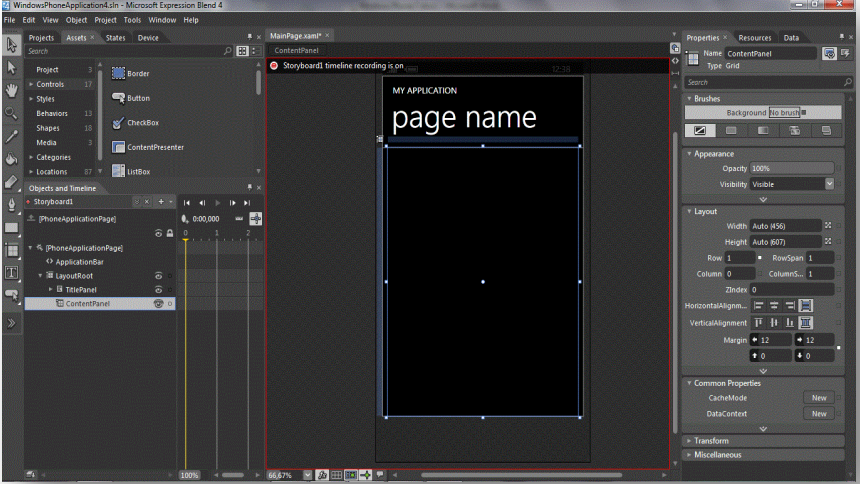


Windows Phone 7 SDK, Microsoft'un mobil işletim sisteminde uygulama ve oyun geliştirmek için gerekli olan tüm bileşenleri içeren yazılım geliştirme aracıdır. Bu SDK ile birlikte gelenler;

- Microsoft Visual Studio 2010 Express for Windows Phone
- Windows Phone Emulator
- Windows Phone SDK 7.1 Assemblies
- Silverlight 4 SDK and DRT
- Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0
- Microsoft Expression Blend SDK for Windows Phone OS 7.1

- WCF Data Services Client for Windows Phone
 - Microsoft Advertising SDK for Microsoft Phone
- Bu bileşenler sayesinde Windows Phone 7 için Visual Studio ortamında çok farklı ve zengin içerikli mobil uygulamalar ve oyunlar rahatça geliştirilebilmektedir.

b. Expression Blend



Expression Blend Microsoft' un tasarım araçlarını bir araya topladığı Expression Studio' nun bir parçasıdır. Blend, Silverlight animasyonları WPF uygulamaları geliştirmek için kullanılan bir tasarım aracıdır. Görsel tasarımcılar için işlevsel, iş akışlarının rahat kontrol edilebildiği bir geliştirme platformudur.

Blend' de Silverlight animasyon yapısı çok kolay ve pratik olarak uygulanabilmektedir. Silverlight daha çok web üzerinde uygulama geliştirmek için kullanılabilecek bir teknolojidir. Bununla birlikte Blend sadece Silverlight' ta animasyon geliştirmek için kullanılan bir araç değildir. Aynı zamanda WPF için de bir tasarım aracıdır. Böylelikle hem web hem de desktop uygulamalar için tasarım yapılabilir. Bunlarla birlikte Sketch Flow ile prototip de geliştirilebilir.

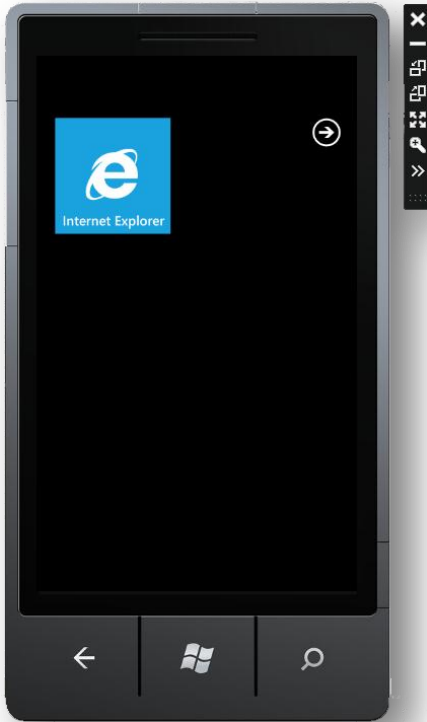
Blend' in en önemli özelliklerinden biri de Visual Studio gibi bir yazılım geliştirme platformu ile beraber çalışmasıdır. Bunun getirdiği en büyük avantaj ise, programcı ve tasarımcı ayrımının yapılabilmesidir. Böylelikle daha hızlı projeler geliştirmek mümkün hale gelir.

Data tarafında ise Blend yine tasarımcılar için göz ardı edilemeyecek kolaylıklar sunmaktadır. Blend sayesinde data konusunda örnek veriler ile tasarım yapma imkanı ve datasource kullanma gibi önemli konular rahatlıkla gerçekleştirilebilmektedir.

Diğer bir önemli nokta da Blend üzerinde tasarlanan arayüze hareket kazandıracak olan behaviorlardır. Bunlar Blend üzerinde çok önemli bir yapıya sahiptirler çünkü en basit bir animasyonu bile ele alacak olursak tasarımcının behaviour kullanması gerekecektir.

Son olarak Blend’ teki şablonlara göz atacak olursak tasarımcılar açısından vazgeçilmez özellikleri ile yeniden kullanılabilirliği bir kat daha artırıyor. En basitinden tasarlanan bir butonu farklı sayfa ve projelerde kullanmanın yanı sıra butunun vektörel özelliği ve state manager yapısıyla hareket kazandırılacağı düşünülürse tasarımcıların işini çok kolaylaştıracağı apaçık ortadadır.

c. Emulator



Emulator, Windows Phone 7 cihazının bilgisayar üzerindeki simülasyonunu sağlayan bir uygulamadır.

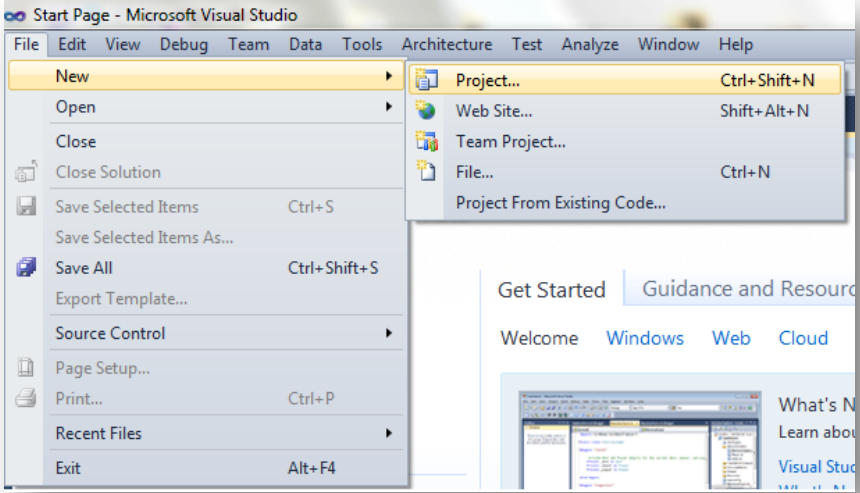
Windows Phone 7 için uygulama geliştirilirken uygulamanın belirli bir aşamasında veya uygulama tamamlandığında test edilmesi gerekir. Her test işlemi için uygulamanın telefona deploy edilmesi zaman alır veya program geliştiriciler Phone 7 cihazına sahip olmayabilir. İşte bu tür durumlarda geliştirilen uygulamanın bilgisayardaki sanal bir Phone 7 cihazı üzerinde test edilmesi sağlanır. İşte bu telefon simülasyonunu sağlayan araca emulator adı verilir.

d. Windows Phone 7 “Merhaba Dünya”

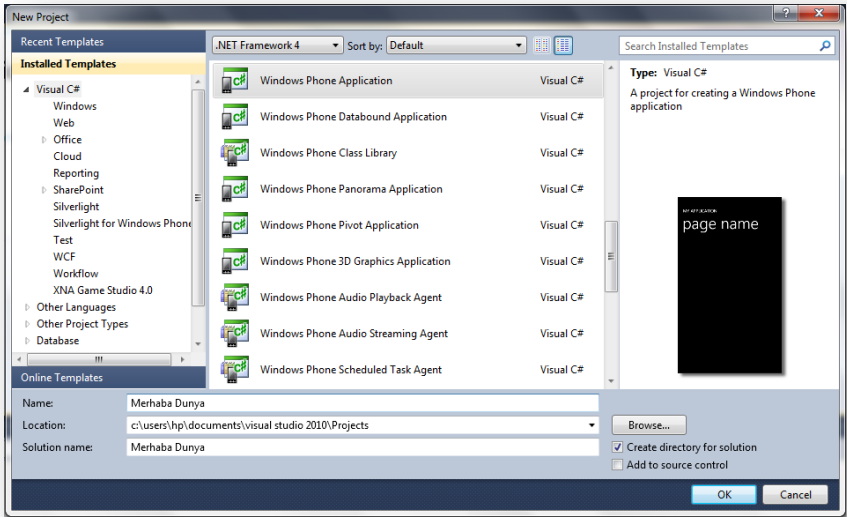
Windows Phone 7 mobil uygulama geliştirme platformunda ilk uygulamamız olan telefonun ekranına “Merhaba Dünya” yazısını yazdıran bir uygulama geliştireceğiz. Bunun için öncelikle bilgisayarımızda Visual Studio 2010 ve Windows Phone 7 SDK’ sının yüklü

olması gerekmektedir. Böylelikle Phone 7 üzerinde mobil uygulama geliştirebilmemiz için gerekli olan Developer Tools eklentileri de kurulmuş olacaktır.

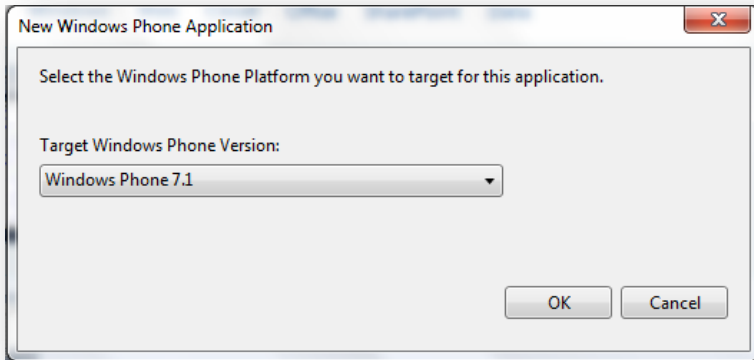
Bu işlemlerin ardından ilk projemizi oluşturmaya başlayabiliriz. Bunun için öncelikle Visual Studio'yu açtıktan sonra sol üst köşedeki File' a tıklayıp ardından Project'e tıklayarak yeni bir proje oluşturuyoruz.

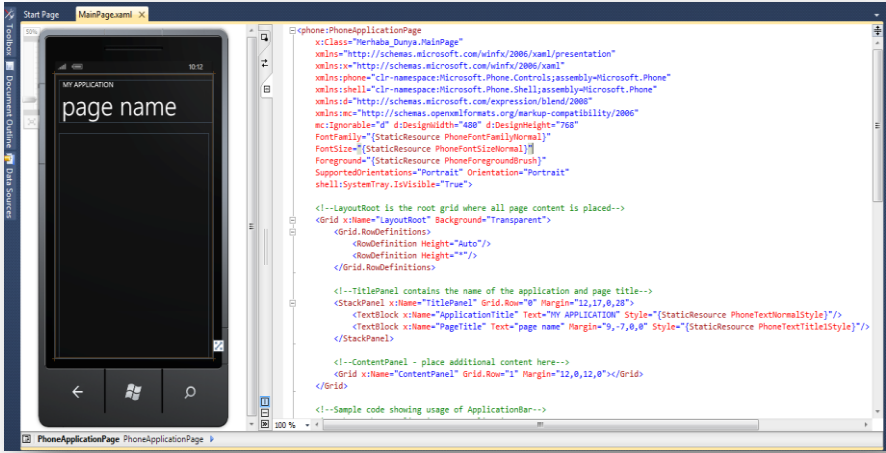


Bundan sonraki aşamada karşımıza proje template' leri gelecektir. Buradan bizim uygulamamız için uygun olan Windows Phone Application' ı seçtikten sonra alttaki Name kısmına projemizin ismini giriyoruz. Bu örnek için “Merhaba Dünya “ yazarak OK diyoruz ve devam ediyoruz.



Projemize başlamadan önce son karşılaşacağımız ekranda ise bizden uygulamayı geliştirecek olduğumuz Windows Phone Platformunun sürümünü seçmemizi isteyecek. Biliyorsunuz ki Windows Phone mobil bir işletim sistemidir bundan dolayı işletim sisteminin versiyonunu seçiyoruz. Buradan da 7.1 sürümünü seçerek projemize geçiyoruz.





Karşımıza gelen yukarıdaki temel ekranda ise yapacak olduğumuz mobil uygulamanın ekran görüntüsünü sağ tarafta görmekteyken sol tarafta da .xaml uzantılı dosyamızı görüyoruz. .xaml kısmından projemizin solda görülen tasarım kısmına kod üzerinden müdahale edebiliyoruz.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    textBlock1.Text = "Merhaba Dünya";
}
```



Gelelim bizim yapacak olduğumuz uygulamaya, öncelikle ekranda “Merhaba Dünya” yazısını yazdırabilmek için ekranımızın sol tarafında bulunan Toolbox’tan bir textblock alıp page name’in altındaki bizim uygulama alanımız olan dikkörtgen bölgeye bırakıyoruz. Page name’i de üzerine tıklayarak sol taraftaki ekrandan rahatça değiştirebileceğiniz gibi sağ taraftaki xaml kodundan da değiştirebilirsiniz. Bu yapacağımız Phone 7’de ilk uygulama olduğu için “ilk uygulama” şeklinde isimlendirdik. Şimdi de bir tane buton alıp bu textblock’ un altına koyuyoruz. Butonu koymadaki amacımız ise şu; kullanıcı butona dokunduğunda yani buton click yapıldığında textblock’ ta “Merhaba Dünya” yazacaktır. Bunun için uygulama alanımıza koymuş olduğumuz butonumuzun üstüne iki kez tıklayıp butonun click olayına gidiyoruz. Gördüğünüz gibi .cs uzantılı bir C# dosyası, burada çok

basit bir şekilde C# kodumuzu yazıyoruz ;

Bu kod bize ekrandaki butona click yapıldığında butonun hemen üzerinde bulunan textblok' un gösterecek olduđu text' i “Merhaba Dünya” olarak ayarla anlamını taşıyor. Bundan sonra ise uygulamamızı emulatorde test etme aşamasına geçiyoruz. Bunun için Visual Studio' nun üst kısmındaki Windows Phone Emulator yazısının solundaki sağa bakan yeşil oka tıklamamız yeterli olacaktır. Karşımıza gelecek olan olan telefon ekranındaki buton' a tıkladığımızda şekilde de görüleceđi üzere butonun hemen üzerinde “Merhaba Dünya” yazısı belirecektir.

4



Bu bölümde Windows Phone 7' de kullanıcıların telefonlarını tutma şekillerine göre, yazdığımız uygulamaların yatay bir şekilde mi yoksa dikey bir şekilde mi ekrana yansıtılacağını anlatıyor olacağız. Yararlı olması dileğiyle...

Yazardan...

Aybala Şeyda SERT
İlkay TAŞKIRAN

4. Kontroller

a. Yatay-Düsey Konum

Sayfa oryantasyonu ve yerleşimi konusunu bu bölümde inceleyeceğiz. Ekranın yatay veya düşey olma durumu, telefonumuza işlevsellik kazandırabilmek açısından büyük bir paya sahiptir.

Oryantasyon değiştiğinde uygulamamızda ne gibi değişiklikler olduğunu aşama aşama göreceğiz. İlk olarak oryantasyon kelimesini biraz daha açalım. Türkçe’ ye oryantasyon olarak çevrilen “orientation” kelimesi ile telefonu sağ veya sola çevirdiğinizde, uygulamanızın telefonun yönüne uyumlu olarak çalışması ifade ediliyor. Oryantasyonu sağladığımızda ekran için farklı kullanım çeşitleri oluşturmuş oluyoruz. Dolayısıyla oryantasyon ifadesi uygulamanın ekran içinde yerleşimi olarak da kullanılabilir.

Windows Phone 7 projelerinde, ekranın yatay ve düşey konumda kullanılmasına imkan sağlanması, uygulama geliştiriciler için büyük bir avantaj. Windows Phone 7 uygulamasında, kodların Silverlight veya XNA ile geliştirilmesi, oryantasyon açısından bir kısıtlamaya neden olmuyor.

Ekranın daha aktif kullanılacağını düşündüğümüz WP7 Silverlight projelerinde, oryantasyonu yatay(landspace) olarak düzenleyip uygulama geliştirmemizde hiçbir sakınca yok. Aynı şekilde WP7 projesinde XNA ile oyun geliştirdiğimizi düşünelim, oyun için de ekranı yatay veya düşey konumlandırabiliriz.

İlerleyen kısımlarda yatay ve düşey konumlandırma için kodlarda, elementlerde yapılması gereken değişiklikleri ayrıntılı olarak inceleyeceğiz.

İlk olarak, Silverlight kodlarıyla geliştirilen Windows Phone7 projesinde ekranın düşey olma durumunu ve emülatörümüzün programa verdiği yanıtları inceleyelim.

i. Düşey Durum (Portrait)

Öncelikle yeni bir WindowsPhoneApplication oluşturalım. Kodlarımızda hiçbir değişiklik yapmadan uygulamamızı çalıştıralım ve emülatörün nasıl çalıştığını inceleyelim.

Oryantasyon, Silverlight kodlarıyla geliştirilen Windows Phone 7 projelerinde başlangıç kodlarıyla düşey konuma ayarlandığından, kodları değiştirmeden elde ettiğimiz görüntü düşey konumda olacaktır.



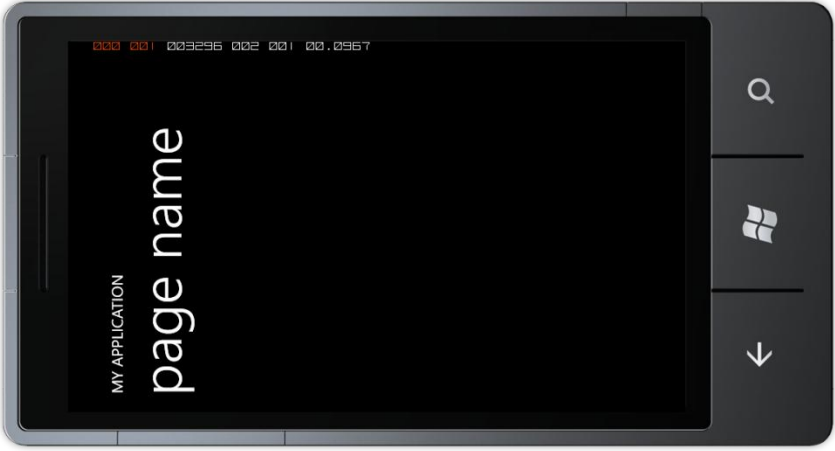
Oryantasyonumuz düşey konuma ayarlandığında telefonun yönünü değiştirmek de görüntüler üzerinde bir değişiklik elde edemeyiz. Peki bu kontrolü nasıl sağlıyoruz?

Uygulamamızın MainPage.xaml sayfasında, aşağıdaki kodlar içinde de gördüğümüz SupportedOrientations özelliğini Portrait belirlemek, ekranı sadece düşey konumda kullanabileceğimizi belirliyor. Bu özellik ile ekranın hangi yönleri destekleyeceğini belirleriz, ekranı sadece yatay, sadece düşey ve hem yatay hem düşey kullanabiliriz.

Aşağıdaki kodlarda sayfanın sadece düşey konumu desteklediğini görebiliyoruz. Orientation="Portrait" ifadesi ile sayfanın ilk hangi konumda açılacağını belirleriz.

```
SupportedOrientations="Portrait" Orientation="Portrait"  
shell:SystemTray.IsVisible="True">
```

Programı çalıştırarak, emülatörü sağ menüsünde bulunan ekran çevirme kontrolü ile yatay çevirdiğimizde neler olduğuna bakalım. Kodlar ekranı sadece düşey konumda kullanacağımızı belirtiyor, bunu SupportedOrientations ile belirlemiştik. Peki şimdi ekranı yatay çevirdiğimizde nasıl bir görüntüyle karşılaşacağız? Hep birlikte görelim;



Emülatör de yan çevrildiğinde görüntünün yer değiştirmediğini görebiliyoruz.

Şimdi de yatay konumlandırmayı inceleyelim ve düşey konumlandırma ile farkını anlamaya çalışalım.

ii. Yatay Durum (Landscape)

Bazı uygulamalarda daha geniş bir kullanım alanı sağlayabilmek ve kullanılabilirliği artırabilmek için ekranı yatay biçimde konumlandırmak ve kullanıcı için uygulamayı bu şekilde geliştirmek daha elverişli olabilir. Bu şekilde kullanım genellikle programın hemen hemen her noktasında kullanıcının daha aktif rol aldığı, işlevlerin programdan çok kullanıcıya bırakıldığı uygulamalarda tercih edilirken, bu durumu belirlemek uygulama geliştiricinin vereceği bir karardır.

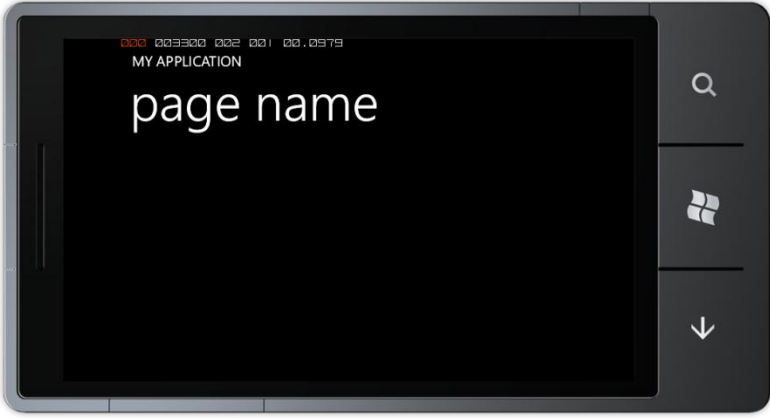
Genellikle görüntü tabanlı uygulamalarda ekran yatay konumlandırma ile kullanılır. Bunun nedeni görüntüye daha geniş bir alan oluşturmak ve kullanıcının görüntüye daha rahat erişmesini sağlamaktır. Bu uygulamalara örnek olarak oyun ve konum belirleyici harita, navigasyon gibi uygulamalarını gösterebiliriz.

Program içerisinde ekranı yatay olarak konumlandırma kontrollerinin nasıl sağlandığını inceleyelim;

```
SupportedOrientations="Landscape" Orientation="Portrait"  
shell:SystemTray.IsVisible="True">
```

Düşey konumlandırmada detaylı olarak bahsettiğimiz SupportedOrientations özelliği burada düşey özellikten farklı olarak Landscape yani yatay olarak değiştiriliyor. Bu değişiklik, ekranı yatay olarak kullanmamıza imkan sağlıyor. Yine Orientation ifadesinin portrait olarak belirlenmesiyle, uygulamanın başlangıçta hangi konumlandırma ile çalışacağını belirtiyoruz. Tabi ki bu ifadeyi portrait seçmek zorunda değiliz, uygulamamıza

göre oryantasyonu yatay, dişey, hatta saęa veya sola yatay gibi seeneklerle dzenleyebiliriz. Buna karar vermek tamamen sizin elinizde.



Düşey konumda beklediğimiz görüntüyü elde edemezken, desteklenen konumu yatay olarak deęiştirdiğimizde uygulama içerięi de yatay olarak dzenleniyor ve kullanıcıya rahat bir kullanım ortamı saęlanıyor.

Eęer uygulamamızın hem yatay hem de düşey konumda kullanılmasını istiyorsak, desteklenen oryantasyonu yani SupportedOrientation özelliğini "PortraitOrLandscape" olarak belirtmemiz yeterlidir.

Bu durum genelde, kullanıcının isteęine bıraktığımızda kullanılır. Örneğin kullanıcının metin gireceęi bir uygulamayı ele alalım. Kullanıcının böyle bir uygulamada telefonunu yatay ve düşey konumda kullanabilme alternatifi saęlamak daha elverişlidir.

Peki kullanıcının telefon yönünü deęiştirdiğini kodlarımız içinde nasıl anlayacağız?

İşte bu durumda OrientedChanged event' ini çağırırız. Bu event ile ekran konumunu kontrol edebiliriz. Ekran konumuna göre uygulamamız için kodlarımızı dzenleriz.





Bu bölümde, akıllı mobil cihazların en önemli özelliklerinden biri olan dokunmatik ekranın marifetlerini Windows Phone 7 işletim sisteminde nasıl kullanacağımızı öğreneceğiz. Hem tek dokunuşu, hem de aynı anda birden çok dokunuşu kontrol etmeyi, ve ayrıca kullanılabilirlik faktörünü kat kat yükselten "el-hareketlerini" Windows Phone 7'nin bize sağladığı hazır sınıflarla kolayca uygulamamıza entegre etmeyi basit örneklerle işleyerek, dokunmatik ekran kullanımının temellerini kavrayacağız.

Silverlight uygulamalarının yanı sıra XNA kullanarak yaratacağınız Windows Phone 7 oyunlarında da dokunmatik ekranın bize sağladığı imkanları kullanarak, hem kullanımı basit uygulamalar, hem de eğlenceli ve yaratıcı oyunlar geliştirmeniz mümkün.

5. Touch Input

Akıllı telefonların vazgeçilmezi olan dokunmatik ekranlar, uygulama yazımı açısından da önceliklere göre daha farklı yaklaşımları beraberinde getirdi. Windows Phone 7 için gerek Silverlight uygulamalarında, gerekse de XNA tabanlı oyunlarda kullanıcıdan girdi almak amacıyla kullanılan bu dokunma olaylarına, multi-touch ve gesture (el hareketi) kullanımını da katarak uygulamamızı bir hayli zenginleştirebilir ve kullanışlı hale getirebiliriz.

Dokunma bilgilerine erişebilmek için kullanacağımız TouchPanel sınıfı ile kullandığımız cihazın dokunma panelinin kullanıma uygun olup olmadığına, desteklenen maksimum aynı anda dokunma sayısına, dokunma panelinin o anlık durumuna, ve gesture bilgilerine erişebiliriz. Şu an için Windows Phone 7 aynı anda 4 taneye kadar ayrı dokunmayı desteklemektedir.

i. Silverlight Uygulamaları

Silverlight uygulamaları, olaya dayalı (event based) uygulamalardır. Uygulamaya kullanıcı ile uygulama arasında etkileşim sağlayan kontroller yerleştirilir ve bir etkileşim olduğu takdirde olay yöneticileri (event handler) aracılığıyla belirli kodlar yürütme ortamı tarafından çalıştırılır. Silverlight uygulamalarında dokunma olayları da bu şekilde işler, yani bir düğme koyduğumuzda yapmamız gereken tek şey düğmeye basıldığında ne yapmasını istediğimiz olay yöneticisini ayarlamaktır. Bu nedenle normal şartlarda dokunmayı kontrol etmemize ya da TouchPanel kullanmamıza gerek yoktur.

Silverlight uygulamamızın içinde el hareketlerini kullanmak istediğimizde bile bu yapı değişmez. Silverlight Toolkit tarafından sağlanan GestureListener sınıfı ile kullanmak istediğimiz el hareketlerine bir olay yöneticisi ayarlayarak amacımıza ulaşabiliriz. El hareketi kullanımına ilerideki kısımlarda daha detaylı olarak değineceğiz.

ii. XNA

XNA uygulamalarının yapısı ise Silverlight uygulamalarından farklıdır. Oyunlar, Update ve Draw metodları arasında bir döngü olarak ilerlediği için, bu tarz işlemleri polling dediğimiz periyodik olarak sorgulamaya dayanan bir yapıda incelemeliyiz. Bu nedenle, uygulamamızın içerisindeki her döngüde dokunma panelinin durumunu kontrol edip uygulamamızın durumunu ona göre değiştirmemiz gerekir.

Bunu sağlamak için, update metodumuz içinde önce dokunma panelinin durumunu almalı, ardından da dokunma ile ilgili bilgileri çekmeliyiz.

TouchCollection sınıfını kullanarak dokunma panelinden o anki tüm dokunma bilgilerini çekip kullanabiliriz.

```
TouchCollection touchCollection = TouchPanel.GetState();
```

Bu kodun ardından, touchCollection objemiz o anki tüm dokunma verisini liste şeklinde tutmaktadır. TouchLocation sınıfını kullanarak bir TouchCollection içindeki verilere erişebiliriz.

Bir TouchLocation objesinde bulunan veriler şunlardır:

- **Id:** Bu değer her dokunma için özeldir ve o dokunma olayını tanımlayan bir kimlik numarasıdır. Ekrana yapılan her yeni dokunuşta yeni bir Id numarası yaratılır. Aynı dokunma, kullanıcı parmağını bırakmadan devam ettiği sürece aynı Id numarasına sahiptir. Aynı anda birden çok dokunma verisi çekildiğinde, hangi dokunmanın hangi parmağa ait olduğunu anlamak için bu değer çok önemlidir.
- **Position:** Bu değer, dokunulan noktanın yerini gösteren 2 boyutlu bir vektördür.
- **State:** Dokunma'nın durumu ile ilgili bilgi sağlayan Touch Location State Enumeration değeridir. Alabileceği 4 değer vardır:
 - **Pressed:** Yeni oluşan dokunmaları belirler.
 - **Moved:** Önceden varolan dokunmaları belirler. Bir dokunmanın State değerinin Moved olması için parmağın fiziksel olarak hareket etmiş olması şart değildir, basılı durması yeterlidir.
 - **Released:** Bir dokunmadan parmağın çekildiğini belirler.
 - **Invalid:** Sadece TryGetPreviousLocation metodu çağırıldığında eğer dokunmanın önceki değeri yok ise State değeri Invalid olur.

TouchLocation sınıfını ve foreach ifadesini kullanarak, TouchCollection içinde içinde dolaşarak o anki tüm dokunmaların bilgilerine ulaşabiliriz. Aşağıda dokunma bilgilerini konsola bastıran örnek bir update metodu bulunmaktadır.

```
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
    ButtonState.Pressed)
        this.Exit();

    TouchCollection touchCollection = TouchPanel.GetState();

    foreach (TouchLocation touchLocation in touchCollection)
    {
        System.Diagnostics.Debug.WriteLine("Id: " +
touchLocation.Id +
        " State: " + touchLocation.State.ToString() +
        " X: " + touchLocation.Position.X +
        " Y: " + touchLocation.Position.Y);
    }

    base.Update(gameTime);
}
```

Bu şekilde hem tek, hem de birden çok dokunma olayını kontrol edebiliriz.

Aynı zamanda, eğer sadece tek bir dokunmayı kontrol etmek istiyorsak, telefonumuzda fare olmamasına rağmen Mouse sınıfını kullanarak daha basit bir yaklaşımda bulunabiliriz. Mouse sınıfı, dokunma paneli üzerindeki ana dokunma noktasını sol tıklama olarak algılayacak şekilde ayarlanmıştır. Aşağıda bu sınıfı kullanılarak yapılmış örnek bir update metodu bulunmaktadır.

```

protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
        this.Exit();

    MouseState mouseState = Mouse.GetState();

    if (mouseState.LeftButton == ButtonState.Pressed)
    {
        System.Diagnostics.Debug.WriteLine("X: " +
mouseState.X + " Y: " + mouseState.Y);
    }

    base.Update(gameTime);
}

```

Update metodumuz her çalıştığında dokunma verileri güncellendiği için, Mouse sınıfını kullanarak sürükle – bırak tarzı hareketleri kolayca yapabiliriz.

a. El Hareketleri (Gestures)

El hareketleri, kullanıcıların dokunmatik ekran üzerinde yaptıkları belirli genel hareketlere denir. Bunlara örnek olarak ekrana çift dokunma, dokunup basılı tutma ya da sürükleme hareketlerini verebiliriz. Bu tarz hareketleri XNA Framework içerisindeki TouchPanel sınıfı sayesinde kendi uygulamalarımıza kolayca ekleyebiliriz.

Uygulamalarımızda kullanabileceğimiz el hareketleri tipleri kısa açıklamaları ile birlikte şu şekildedir:

El Hareketi

Tap

DoubleTap

Hold

FreeDrag

Açıklaması

Parmağın ekrana dokunup fazla hareket etmeden geri çekilmesi.

Ardı ardına yapılan iki Tap hareketi.

Parmağın ekrana dokunup kısa bir süreliğine aynı noktaya dokunmaya devam etmesi.

Parmağın ekrana dokunup herhangi bir yöne sürüklenmesi.



VerticalDrag	Parmağın ekrana dokunup dikey yönde (yukarı / aşağı) sürüklenmesi.
HorizontalDrag	Parmağın ekrana dokunup yatay yönde (sağ / sol) sürüklenmesi.
DragEnd	FreeDrag, VerticalDrag ve HorizontalDrag hareketlerinin sonlandığını gösterir.
Flick	Parmağın ekran boyunca sürüklenip parmağı durdurmadan geri çekilmesi.
Pinch	İki parmağın dokunup hareket etmesi.
PinchEnd	Pinch hareketinin sonlandığını gösterir.

Bu el hareketlerini kullanabilmemiz için öncelikle kullanmak istediklerimizi uygulamamızda aktifleştirmeliyiz. İstersek tüm hareketleri aktifleştirebiliriz, ancak bu performans açısından gereksiz bir yük olacağı için sadece kullanacağımız hareketleri aktifleştirmemiz tercih edilmelidir. El hareketleri, aşağıdaki şekilde aktifleştirilir:

```
TouchPanel.EnabledGestures = GestureType.Tap |
GestureType.DoubleTap |
GestureType.FreeDrag;
```

Bu kod ile Tap, DoubleTap ve FreeDrag hareketlerini aktif hale getirdik. Bu şekilde Gesture Type Enumeration dahilindeki istediğimiz hareketleri aktifleştirmemiz mümkün.

Hareketler aktifleştirildikten sonra, TouchPanel sınıfının ReadGesture metodu ile okunarak uygulama içerisinde kullanılabilir. Ancak dikkat edilmesi gereken bir nokta vardır. Her el hareketi fark edildiğinde, bir liste içine atılır, ve ReadGesture metodu o listeden sıradaki hareketi getirir. Eğer her Update metodumuz içerisinde bütün hareketleri okumazsak, bu ileride hareketleri gecikerek okumamıza ya da aynı anda gerçekleşen hareketleri kaçırmamıza neden olarak uygulamamızda sorun çıkarabilir. Bu nedenle, aşağıdaki gibi bir while döngüsünü ve TouchPanel.IsGestureAvailable değerini kullanarak o an gerçekleşmiş bütün hareketleri okuyabiliriz:

```
while (TouchPanel.IsGestureAvailable)
{
    GestureSample gesture = TouchPanel.ReadGesture();

    switch (gesture.GestureType)
    {
        case GestureType.Tap: {
System.Diagnostics.Debug.WriteLine("Tap.");
break; };

        case GestureType.DoubleTap: {
System.Diagnostics.Debug.WriteLine("Double Tap."); break; };
        case GestureType.FreeDrag: {
System.Diagnostics.Debug.WriteLine("Free Drag."); break; };
    }
}
```

Not: Bir DoubleTap hareketinden önce her zaman için bir Tap hareketi gerçekleşir.

İstedığımız el hareketlerine uygulamamız içinde ulaştıktan sonra, geriye tek kalan şey bu hareketler ile ilgili verilere ulaşmak. GestureSample sınıfının kullanabileceğimiz GestureType dışında 5 adet niteliği vardır: Timestamp, Position, Position2, Delta ve Delta2. Timestamp dışındaki nitelikler hareketin yerini ve yönünü belirlememizi sağlayan iki boyutlu vektörlerdir. Timestamp değeri ise, o hareketin yapıldığı zaman ile ilgili bilgi almamızı sağlayan bir TimeSpan objesidir. Timestamp değeri, XNA içerisindeki GameTime'dan bağımsızdır, ve hareketlerin kendi aralarındaki bağlantıyı belirlememiz için kullanılır. Örneğin, önceden olmuş bir Tap hareketinin Timestamp değerini, sonradan olmuş bir Flick hareketinin Timestamp değerinden çıkararak aralarında kaç saniye olduğunu bulabiliriz. Her el hareketi, GestureSample sınıfının her değerini kullanmaz. Kullanılmayan değerler default olarak (0,0) döndürür. Her el hareketinin kullandığı değerler aşağıdaki gibidir:

<u>El Hareketi</u>	<u>Kullandığı Değerler</u>
Tap	Position
DoubleTap	Position
Hold	Position
FreeDrag	Position, Delta
VerticalDrag	Position, Delta
HorizontalDrag	Position, Delta
DragComplete	Hiçbiri
Flick	Delta
Pinch	Position, Position2, Delta, Delta2
PinchComplete	Hiçbiri

Çoğu değer in içeriği adından da anlaşılacağı gibidir, Position bulunulan yeri ve Delta da yer değişimini gösterir. Ancak dikkat edilmesi gereken noktalar vardır:

- Flick hareketinde Delta değeri hareketin hızını gösterir. Bu hız, piksel/saniye cinsinden verilir. Bu değeri kullanarak kullanıcının oyun içerisindeki objeleri etrafa fırlatması gibi işlevleri sağlayabiliriz.
- VerticalDrag ve HorizontalDrag hareketlerinde Position değeri parmağın ekranda bulunan o anki yerini gösterse de, Delta değeri kullandıkları eksen ile sınırlandırılmıştır. Bunun anlamı, bir VerticalDrag hareketinin Delta değerinin X'i ve bir HorizontalDrag hareketinin Delta değerinin Y'si her zaman için 0 olur. Bu durum, menü gibi kaydırmak istediğimiz bölmelerde eksen değerlerini düzeltme ile uğraşmamamızı sağlayarak bize kolaylık sağlar.
- Pinch hareketinde Position ve Delta değerleri birinci parmağın, Position2 ve Delta2 değerleri de ikinci parmağın değerlerini tutar. Bu değerler ile hareketin bir önceki durumuna kolayca erişilerek, gerek bir objenin büyütülüp küçültülmesi, gerekse de kameranın döndürülmesi gibi hareketler rahatça gerçekleştirilebilir.

Eğer istersek bir Silverlight uygulamasında da el hareketlerini kullanmamız mümkün. Bunun için öncelikle uygulamamızda referanslar kısmına Microsoft.Phone.Controls.Toolkit derlemesini eklemeliyiz. Ardından da, bu namespace'i XAML içerisinde aşağıdaki şekilde tanımlamalıyız:

```
xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Co
ntrols.Toolkit"
```

Bütün bunları yaptıktan sonra, hareketleri kullanmak istediğimiz objeye bir GestureListener yerleştirmeliyiz. Aşağıda koddan örnek olarak bir image kontrolüne DoubleTap ve Hold hareketleri için GestureListener eklenmiştir:

```
<Image x:Name="Resim"
        Source="Merhaba.png">
    <toolkit:GestureService.GestureListener>
        <toolkit:GestureListener DoubleTap="OnDoubleTap"
                                Hold="OnHold" />
    </toolkit:GestureService.GestureListener>
</Image>
```

Bu noktadan sonra geriye kalan tek şey yukarıda tanımladığımız GestureListener için gerekli olay yöneticilerini ayarlamak.

```
private void OnDoubleTap(object sender,
DoubleTapGestureEventArgs e)
{
    System.Diagnostics.Debug.WriteLine("Double Tap.");
}
```

GestureListener kullanarak Silverlight uygulamamıza yerleştirebileceğimiz olaylar şunlardır:

- **PinchStarted, PinchDelta, PinchCompleted:** Pinch hareketlerini başlangıcından bitişine kadar kontrol etmemizi sağlayan olaylardır.
- **Tap, DoubleTap:** Tap ve DoubleTap hareketlerini kontrol etmemizi sağlayan olaylardır.
- **DragStarted, DragDelta, DragCompleted:** VerticalDrag, HorizontalDrag ve FreeDrag hareketlerini kontrol etmemizi sağlayan olaylardır.
- **Flick:** Flick hareketini kontrol etmemizi sağlayan olaydır.
- **Hold:** Hold hareketini kontrol etmemizi sağlayan olaydır.
- **GestureBegin, GestureCompleted:** Herhangi bir hareketin başlangıç ve bitişini kontrol etmemizi sağlayan olaylardır.



Bu bölümde Windows Phone 7' de canlı uygulamalar yapabileceğimiz Notification Servislere değineceğiz. Öncelikle Application Tile' lardan bahsedip ardından Tile Notification, Toast Notification ve Row Notification' ı anlatıyor olacağız. Örnek ile pekiştireceğiz..

Yazardan...

6. Application Tile ve Notification Servisler

a. Application Tile

Application Tile, Windows Phone 7’de ön plana çıkan gelişmiş ve çok kullanışlı kısayollardır. Her Windows Phone uygulamasında bulunan ve Start Screen(başlangıç ekranı) içerisinde bir resim karesi olarak algılayabileceğimiz; fakat özünde gelişmiş bir simge olan Application Tile; genelde kullanıcıların uygulamaya girmesini gerektirmeden, bazı bilgileri yansıtabilmek için de kullanılır.

Uygulamalar ilk yüklendiğinde *pin*lenmemiştir ve 62 x 62 px boyutunda küçük bir ikon şeklinde menüde bekler.Kullanıcı bu uygulamayı *pin*lediğinde ikonumuz artık 173 x 173 px boyuta sahip yukarıda bahsedilmiş olan Application Tile halini alarak Start Screen’e yerleşir. Bir Windows Phone 7 projesi açıldığında “Solution Explorer” içerisinde **ApplicationIcon.png** ve **Background.png** isminde iki adet resim dosyası mevcuttur. Bu dosyalar başlangıçta standart olarak da gelmektedir. Dikkatli bakıldığında bu dosyaların az önce açıklanan resimler olduğu açıkça görülebilmektedir. Pinleme işleminin ardından standart **ApplicationIcon.png** yerini **Background.png** dosyasına bırakacaktır.

Application Tile ikiye ayrılır:

1. Normal Application Tile
2. Secondary Application Tile

Normal Application Tile: Kullanıcılar tarafından oluşturulan kısayollardır (pinleme).

Secondary Application Tile: Bu kısayollar doğrudan uygulamayı açabilecek şekilde oluşturulabileceği gibi, program içerisinde belirli bir görevi de olabilir. Örneğin uygulamanızda bulunan kişi listesinden herhangi birinin pinlenmesi ve bu kısayola dokunulduğunda doğrudan program içerisinde o kişi ile ilgili işlemlerin yapılabilmesi gibi.

Tile kullanımına geçmeden önce Visual Studio içerisinde Tile Image’lerin nasıl ayarlanacağına bakalım. Tile’ların ön ve arka olmak üzere iki görüntü içerdiğini unutmayalım.

Oluşturmuş olduğunuz Windows Phone Application içerisinde, **Solution Explorer**’a gelin ve ardından uygulamanıza sağ tıklayarak **Properties**’i açın.

Icon:

ApplicationIcon.png

Tile options

Title:

ApplicationTileSampleApp

Background image:

Background.png

Şekilde görüldüğü gibi **ApplicationIcon.png** ve **Background.png** başka bir resim dosyası ile değiştirilebilmektedir; fakat dikkat edilmesi gereken bazı noktalar bulunmaktadır.

- Resimlerin boyutları yukarıda bahsedildiği gibi olmalıdır yani ikon için 62 x 62 px, pinlenmiş görüntü için ise 173 x 173 px olmalıdır. Ayrıca her iki resim de 250 dpi olmalıdır.
- Eklediğiniz resim dosyasının uzantısı .jpg veya .png olmalıdır.
- Eklenen resimlerin **Build Action** özelliği **Content** olarak ayarlanmalıdır. Bu özelliği resme tıkladıktan sonra Properties panelinden değiştirebilirsiniz.
- Eklemiş olduğunuz resimler kök dizinde olmalıdır.

Bunların dışında tasarımsal açıdan da bazı konulara dikkat etmemiz gerekmektedir. Kaçınılması gereken noktalar kitabın ilgili bölümünde anlatılmıştır.

i. Application Tile Genel Özellikleri

Application Tile ile ilgili teorik bilgilerin ardından biraz da teknik ayrıntılarına inelim. Tile'ların ön ve arka olmak üzere iki adet görüntüye sahip olduğunu az önce belirtmiştik. Bu görüntüler arasında geçiş yapmak için döndürme işlemi yapılır. Eğer sadece bir görüntü gösterim için ayarlanmışsa bu özellik kullanılamayacaktır.



Bu özellikler Application Tile ve Secondary Tile'in ön yüzü için tamamen aynıdır. Bu özellikleri açacak olursak:

Title: Uygulamanın ana başlığıdır. 12px – 15px arasında bir yazı yazılması önerilmektedir. Aksi durumda metin kaymakta ve Tile dışına çıkmaktadır.

BackgroundImage: Application Tile'in arka plan resmidir.

Count (Badge): 1 ile 99 arasında sayı atandığında bir çember içerisinde değeri gösterir. Hiçbir atama yapılmadığında veya "0" değeri atandığında gösterim yapılmaz. 99 değerini geçtiğinizde sayı 99 olarak değerini koruyacaktır.

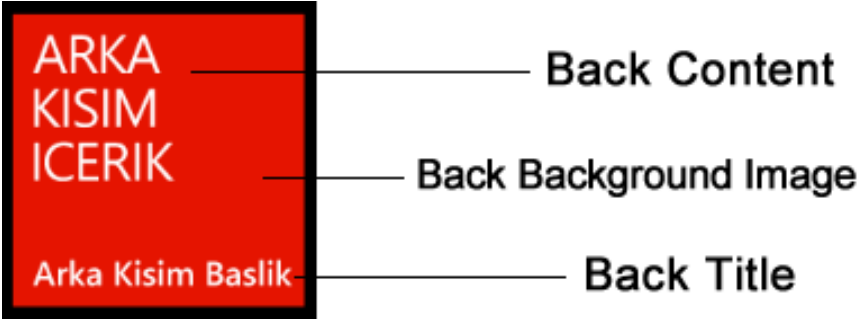
Uygulamanızdaki bu değerleri **Solution Explorer** içerisinde proje isminin altında görebileceğiniz **Properties** altındaki **WMAppManifest.xml** içerisinde bulabilirsiniz.

Yukarıda anlatılan özellikler Tile'in ön (front) tarafındaki görüntüye ve üzerindeki nesnelere aittir. Şimdi ise arka (back) tarafındaki nesnelere bakalım.

BackContent: Arka taraftaki içeriğin başlığıdır. String tipinde olan bu özelliği kullanırken bilinmesi gereken nokta sondan kırılabilceğidir. Dolayısıyla tasarımın etkilenmemesi için buraya verilecek olan metine dikkat edilmelidir.

BackBackgroundImage: Arka tarafın arka plan görüntüsüdür.

BackTitle: Arka tarafın en altında bulunan string tipindeki değişkendir. Yaklaşık 15 karakter uzunluğunda olması tavsiye edilmektedir.



BackgroundImage ve **BackBackgroundImage** hakkında dikkat edilmesi gereken bazı noktalar bulunuyor. Konunun başlarında resimler ile ilgili yapılmış olan uyarılar yine geçerli olmakla birlikte, ek olarak;

- Ağ iletişimi ve performans açısından bu dosyaların Isolated Storage veya XAP içerisinde bulunması yerinde olacaktır. Eğer URI içerisine vermiş olduğunuz resim dosyası Isolated Storage içerisindeyse bu dosyalar Shared\ShellContent klasörü içerisinde olmalıdır.

Ağ üzerinden gelecek olan resimlerde;

- **https** desteğinin olmadığı unutulmamalıdır.
- Boyutu **80kb**'ı geçen resimler indirilmeden iptal edilecektir.
- Resim indirilmeye başladığından itibaren, 30 saniyelik süre içerisinde halen indirilemediyse işlem iptal edilecektir.
- Resimlerin biri veya ikisi içinde iptal işlemi yaşandığında, iki tarafta da diğer özellikler için update işlemi iptal olacaktır.

ii. Application Tile Ekleme, Çıkarma ve Güncelleme

Bu kadar teorik bilgiden sonra artık kod yazmaya başlayabiliriz. Tüm işlemleri **ShellTile API** kullanarak yapacağız. ShellTile API için aşağıdaki direktifi eklemeniz gerekmektedir.

ApplicationTileSampleApp Project

`using Microsoft.Phone.Shell;`

İlk olarak Main Page içerisinde hazırlamış olduğumuz test arayüzünü kullanarak Application Tile üzerinde değişiklikler yapalım, ardından yeni bir Tile ekleyelim.

Yapılan işlemlerin ardından “DEĞİŞTİR” butonuna basıldığında Application Tile’ın belirtilen özellikleri aşağıdaki örnek kod ile değiştirilecektir.

```
private void BtnChangeApplicationTile_Click(object sender,
RoutedEventArgs e)
{
    //Aktif Tile nesnesi alınıyor.
    ShellTile AppTile = ShellTile.ActiveTiles.First();
```

```

if (AppTile != null)
{
    //Yeni bir Tile oluşturuluyor.
    StandardTileData NewTile = new StandardTileData();

    //Yeni Tile'in özellikleri değiştiriliyor.
    NewTile.Title = TxtTitle.Text;
    if (!string.IsNullOrEmpty(TxtCount.Text))
        NewTile.Count = Convert.ToInt32(TxtCount.Text);
    else NewTile.Count = 0;
    NewTile.BackgroundImage = UriBackgroundImage;

    //AppTile Arka kısım için atamalar yapılıyor.
    NewTile.BackTitle = TxtBackTitle.Text;
    NewTile.BackContent = TxtBackContent.Text;
    NewTile.BackBackgroundImage = UriBackBackgroundImage;

    //Son olarak Application Tile Update ediliyor.
    AppTile.Update(NewTile);
}
}

```

Değişiklikleri görmek isterseniz, emülatörde bulunan windows tuşuna basıp, pinlemiş olduğunuz uygulama üzerinde görüntüleyebilirsiniz. Bu basit işlemin ardından bir de Secondary Tile ekleme işlemini gerçekleştirelim. Bu işlem için uygulamadaki “YENİ” butonu kullanılacaktır.

Secondary Tile ile uygulamanızda bulunan bazı hazır makrolara, sayfalara veya nesnelere ulaşabileceğinizi vurgulamıştık. Yapacağımız işlemlerde bunu göz önüne almak, konuyu anlamamız için faydalı olacaktır.

Önce bir Tile oluşturalım.

```

private void BtnNewSecondaryTile_Click(object sender,
RoutedEventArgs e)
{
    //Yeni bir Tile oluşturuluyor.
    StandardTileData NewTile = new StandardTileData();

    //Yeni Tile'in özellikleri değiştiriliyor.
    NewTile.Title = TxtTitle.Text;
    if (!string.IsNullOrEmpty(TxtCount.Text))
        NewTile.Count = Convert.ToInt32(TxtCount.Text);
    else NewTile.Count = 0;
    NewTile.BackgroundImage = new UriBackgroundImage;

    //AppTile Arka kısım için atamalar yapılıyor.
    NewTile.BackTitle = TxtBackTitle.Text;
}

```

```
NewTile.BackContent = TxtBackContent.Text;  
NewTile.BackBackgroundImage = UriBackBackgroundImage;
```

```
//Son olarak Secondary Tile'ı oluşturup, bu Tile için bir yol  
belirliyoruz.  
ShellTile.Create(new Uri("/SecondaryTilePage.xaml?Person=Burak",  
UriKind.Relative), NewTile);  
}
```

SecondaryTile oluşturduk ve gideceği yolu verdik. Uri verirken farkedebileceğiniz gibi bir query gönderdik. Bu query'ler sayesinde uygulamaya farklı değerler göndererek farklı kişilerin profillerine (senaryoya göre) ulaşabiliriz.

Update işlemi ApplicationTile yaratma ile tamamen aynı mantıkta çalışmaktadır bu nedenle tekrar ele almıyorum. Sadece bir noktaya dikkat çekmek istiyorum. ApplicationTile'ı çekerken ilk kaydı alıyorduk ve bu instance üzerinde işlem yapıyorduk; fakat SecondaryTile kullanıldığında bu işlemi farklı bir yol kullanarak yapmalıyız.

Örneğin;

```
ShellTile STile = ShellTile.ActiveTiles.FirstOrDefault(  
x => x.NavigationUri.ToString().Contains("Kisi=Burak"));
```

Silme işlemini uygularken yine yukarıda gösterilmiş olan ifade ile hazırda bekleyen SecondaryTile bulunarak yeni bir instance alır. Instance aldıktan sonra ise bu nesne üzerinde Tile ile ilgili tüm işlemleri gerçekleştirebilirsiniz.

Örneğin *STile.Delete()*; yaparak secondary Tile'ı silebilirsiniz.

b. Notification Servisler

Microsoft Push Notification Service Windows Phone'un biz yazılımcılara sunduğu; uygulamalarımıza bir web servisi ile veri alış-verişini sağlayan sağlam, esnek ve devamlı işleyen bir kanaldır. Bu kanal sayesinde kullanıcılara; uygulamanın yapmış olduğu işlemleri, değişiklikleri veya çeşitli servis güncellemeleri sonucunda gelen bildirimleri yansıtabilirsiniz.

Notificationları anlatmaya geçmeden önce Notification'ların temelinde kullanılan **XML Schema** hakkında bilgi vermemiz gerekiyor. Windows Phone'a gönderecek olduğumuz Notification'ların hepsi özünde birer **XML** şeklindedir. Windows Phone bu XML verilerini işleyerek gerekli işlemleri uygular. Biz de vereceğimiz tüm değerleri bu XML içerisindeki gerekli *tag*'lar içerisine yazarız. Bu *tag*'lar her notification için farklı bir isimdedir. Şu an bu kavram soyut kalsa da örnek uygulamalar bölümünde bu konuyu çok daha iyi anlayabileceksiniz.

Toplamda 3 tip notification servisi vardır. Bunlar; *Toast*, *Tile* ve *Raw* Notifications'lardır.

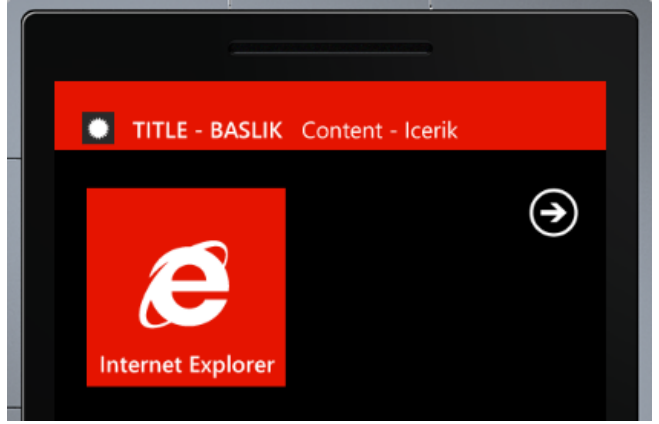
i. Notification Çeşitleri

1. Toast Notifications

Toast Notifications uygulamadaki bir olaya bağlı olarak oluşturulan notification çeşididir. Örneğin RSS'ten veri çeken bir uygulama düşünüldüğünde yeni gelen bir kayıt için kullanıcıya haber vermek isteyebilirsiniz veya uygulamanızda mesajlaşma gibi özellikler varsa yeni mesajın geldiğini bildirebilirsiniz. Bunların dışında haberler, hava durumu gibi birçok örnek verilebilir.

Şekilde de görebileceğiniz gibi bir Toast Notification; uygulama ikonu, başlık ve içerikten oluşmaktadır.

Ekranın en üst bölümünde ortaya çıkan Toast Notification, standart olarak yaklaşık 10 saniye bekler. Kullanıcı ekrandan paneli çekerse notification gösterimi de biter. fakat kullanıcı notification'a dokunursa, gelen notification hangi uygulamaya aitse o uygulama başlatılır.



Toast Notification yapısında açıklamamız gereken 4 özellik vardır.

İkon: Uygulamanızın ikonudur ve notification'a özel olarak değiştirilemez. Uygulamanızın ikonunu doğrudan alır.

Title: Notification başlığıdır ve uygulama ikonunun hemen yanında yer alır. XML Schema içerisinde "Text1" değeridir.

Content: Title'nin hemen ardından gelir. XML Schema içerisinde "Text2" değeridir.

Parameter: Notifications içerisinde görünme de en önemli bilgidir. Kullanıcı eğer notification'a dokunursa uygulamaya bu parametre ile geçiş yapılacaktır. Siz de hangi sayfaya geçileceği gibi bilgileri burada aktarabilirsiniz. Dolayısıyla uygulamaya aktaracağınız değere dikkat etmelisiniz. XML Schema içerisinde "Param" değeridir. Parametreler konusunda dikkat etmemiz gereken unsur, bu işlevin Windows Phone 7.1 ve sonraki sürümleri için geçerli olduğudur. Eğer daha düşük bir sistemde çalışıyorsanız PushErrorTypePayloadFormatInvalid hatası oluşacak ve kanal kapatılacaktır.

Toast Notifications için dikkat edilmesi gereken başka bir unsur ise bilgilerin tamamının bildirim içerisinde görünüp görünemeyeceğidir. Bu sorun ile karşılaşmamak için Toast Notifications özelliklerine iyice dikkat edilmesi gerekir. Bu noktaların ilki **Title**'in kalın harflerle, **Content**'in ise ince harflerle yazılıyor olmasıdır. Eğer sadece Title yazacaksanız ve kesinti olmasını istemiyorsanız 40 karakter uzunluğunu geçmemeniz önerilmektedir. Eğer sadece Content yazacaksanız yaklaşık 47 karakter yazabilmektesiniz. İkinci bir nokta ise gösterilen yazıların (mesajların) notification paneline sığdırılmayıp doğrudan kesilmesidir. Yani karakter uzunluğunu geçtiğinizde yazının devamı kaybolacaktır. Mesaj gösterimi ile ilgili prensiplerinizi baştan düşünmeniz faydanıza olacaktır.

2. Tile Notifications

Tile Notifications ise tahmin edebileceğiniz gibi bir önceki konuda bahsetmiş olduğumuz Application Tile'ı güncelleme işlemini üstlenir.

3. Raw Notifications

Raw Notification'lar uygulamanıza bilgi göndermeniz için kullanabileceğiniz bildirim çeşididir. Bu bildirim kullanabilmeniz için uygulamanızın açık olması gerekmektedir. Aksi halde mesaj, telefona ulaşacak; fakat işlem yapılmayacaktır.

ii. Notifications Kullanımı İle İlgili Örnekler

Örnekler bölümünde bir Windows Phone client'ına notification'ların nasıl gönderildiğini inceleyip bununla ilgili örnek uygulamalar yapacağız. Örnek uygulamaları yaparken ASP.NET de kullanmamız gerekiyor, bu nedenle ASP.NET hakkında da ufak bir bilgiye ihtiyaç duyabilirsiniz. Ayrıca **Server** ve **Client** hakkında yeterli temel bilgiye sahip olduğunuzu varsayıyorum. Windows Phone uygulamasını yaparken her notification örneği için ayrıca bir uygulama yazmayacağız. Çünkü tüm örneklerde sadece belirli bir satırı değiştirerek ilerleyeceğiz. Sırasıyla önce Toast Notification, daha sonra Tile Notification ve son olarak da Raw Notification gönderimi yapacağız. Web uygulamasında ise her notification için farklı bir sayfa oluşturacağız.

1. Toast Notifications Örneği

Notification gönderimi yapmadan önce Windows Phone client'ımızı yaratmamız gerekmektedir.

Öncelikle Visual Studio'yu açıyoruz ardından **Silverlight For Windows Phone** kategorisine gelerek **Windows Phone Application** oluşturuyoruz. İsmi **NotificationsPhoneClientApp** olarak değiştirelim.

Projemize Sayfa1.xaml adında bir sayfa ekleyelim.) Kullanıcı Toast Notification' a tıkladığında bu ikinci sayfaya yönlendirilsin ve göndermiş olduğumuz bilgiyi bize yansıtın. Yanlış anlaşılmayı önlemek adına şunu hatırlatmakta fayda var; Toast Notification'ın yönlendireceği sayfaya siz de uygulama içinden link verebilirsiniz. Yani Toast Notification ile ilgili bu konuda herhangi bir kısıtlama bulunmamaktadır. Zaten amacımız da kullanıcıyı yormamak adına Toast Notification ile doğrudan gerekli sayfaya yönlendirmek olacaktır.

ToastNotificationClientApp - MainPage.xaml.cs

```
using Microsoft.Phone.Notification;  
using System.Text;
```

Yukarıdaki direktifleri ekledikten sonra MainPage() içerisine gerekli kodları yazmaya başlıyoruz.

Öncelikle belirtmiş olduğumuz isimde bir kanal var mı, onu kontrol ediyoruz.

```
BildirimKanali = HttpNotificationChannel.Find("Kanal");
```

Eğer kanal yoksa BildirimKanali *null* değere sahip olacaktır. Dolayısıyla bu değeri kontrol ederek kanalın varolma durumuna göre işlem yapacağız. Eğer böyle bir kanal yoksa aşağıdaki işlemleri uygulayacağız.

```
BildirimKanali = new HttpNotificationChannel("Kanal");
```

Öncelikle BildirimKanali nesnesini set ediyoruz. Ardından ChannelUriUpdated event'ini oluşturuyoruz. *ChannelUri* notification için bir adres tutar ve biz de *ChannelUri* güncellemelerinden haberdar olmak için bu event'i kullanıyoruz. Web uygulaması ile notification gönderdiğimizde daha iyi anlaşılacaktır.

```
BildirimKanali.ChannelUriUpdated += new  
EventHandler<NotificationChannelUriEventArgs>(BildirimKanali_Channel  
UriUpdated);
```

Daha sonra ShellToastNotificationReceived event'ini aktif hale getiriyoruz. Burada aklımıza şöyle bir soru gelebilir: "Toast Notification uygulama kapalıyken çalışıyorsa bu event'e neden gerek duyuyoruz?" Şöyle ki; uygulamanız kapalıyken Windows Phone, gelen notification'a dokunulduğunda uygulamanızı açarak doğrudan "wp:Param" içerisindeki değere yönlendirir. Eğer uygulama açık ise siz bu durumun böyle olmasını istemeyebilirsiniz. Dolayısıyla gelen notification'ları önceden değerlendirme sürecine bağlı tutarsınız.

```
BildirimKanali.ShellToastNotificationReceived += new  
EventHandler<NotificationEventArgs>(BildirimKanali_ShellToastNotific  
ationReceived);
```

Ardından Notification açılması ve telefona bind edilmesi için gerekli fonksiyonlar yazılıyor.

```
BildirimKanali.Open();  
BildirimKanali.BindToShellToast();
```

Eğer kanal varsa sadece event'ları bu kanala yönlendirerek işlemlerimize devam ediyoruz. ShellToastNotificationReceived fonksiyonu ile gelen bilgiyi doğrudan işleyebiliyoruz. Bu fonksiyonun sadece uygulama açık iken çalıştığını tekrar hatırlatmakta fayda var.

```
void BildirimKanali_ShellToastNotificationReceived(object sender,
    NotificationEventArgs e);
```

Bu bilgiyi işlemek için *NotificationEventArgs* ile elde ettiğimiz *Collection.Keys* içerisinde dolaşarak bilgileri tek tek alabiliyoruz. *Text1*, *Text2* ve *Param*'dan gelen değeri gelenBilgi içerisine yazıyoruz ve eğer gelen bilgi *Param*'a aitse kullanıcıya bir yönlendirme varolduğunu bildiriyor ve gitmek isteyip istemediğini soruyoruz. Bu işlem *void* Yonlendir(*string* value); fonksiyonu içerisinde yapılmaktadır.

```
foreach (string key in e.Collection.Keys)
{
    gelenBilgi.AppendFormat("{0}: {1} \n", key, e.Collection[key]);
    if(key == "wp:Param") Dispatcher.BeginInvoke(() =>
Yonlendir(e.Collection[key]));
}
```

Kalan diğer fonksiyonları veya diğer işlem ayrıntılarını kitap ile birlikte edindiğiniz içerikten **NotificationsPhoneClientApp** projesini bularak çalıştırıp test edebilirsiniz.

Bir client oluşturduk artık sıra ASP.NET uygulamasında. Bu uygulama ile kullanıcıya Toast bildirimini göndereceğiz. Burada konumuz ASP.NET olmadığı için sadece çok önemli olan bölümleri sizlere aktaracağım.

Yeni bir solution başlatacağız ve web projemizi bu solution içinde saklayacağız. **File -> Close Solution** yapabilir veya yeni bir Visual Studio başlatabilirsiniz. Ardından **File -> New -> Project** (Ctrl + Shift + N) diyerek yeni proje penceresini açalım. **C#** kategorisindeki **Web** bölümüne geçelim. Ardından **ASP.NET Web Application**'ı seçerek **OK** diyelim. İsim olarak **NotificationSenderWebApp** verebilirsiniz.

Ardından Default.aspx dosyasını silelim. **Solution Explorer**'dan Projemize sağ tıklayıp **Add -> New Item** diyoruz ve **ToastGonder.aspx** isimli bir Web Form oluşturuyoruz. Şimdi ToastGonder.aspx'in içerisini doldurmaya başlayalım.

Bir Toast Notification'ın 3 adet özelliği vardır demiştik. Bunlar Title, Subtitle ve Parameter şeklindeydi. Dolayısıyla form içerisine bu üç özelliği aktarabileceğimiz TextBox'larımızı, URI girebileceğimiz bir TextBox'ı, gönderi sonucunu alabileceğimiz bir adet TextBox'ı ve son olarak gönder diyebileceğimiz bir Button nesnesini sayfamıza ekliyoruz.

ToastGonder.aspx.cs

Arayüzü hazırladıktan sonra kod yazmaya başlayabiliriz.

```
using System.Net;
using System.IO;
using System.Text;
```


Yukarıdaki direktifleri sayfamıza ekleyerek BtnSendToast_Click event'ine kodlarımız yazıyoruz. Bu event'i oluşturmak için sayfanın design kısmında "GÖNDER" butonuna çift tıklayabilir ve event'in otomatik yaratılmasını sağlayabilirsiniz.

```
string ChannelUri = TxtChannelUri.Text;
```

Öncelikle cihazdan almış olduğumuz URI'yi web uygulamamıza alıyoruz. Ardından Push Notification Service'e Toast Notification gönderebilmek için bir HttpRequest oluşturuyoruz.

```
HttpRequest toastNotificationIstegi =  
(HttpRequest)WebRequest.Create(ChannelUri);
```

Bir veriyi gönderirken GET veya POST yöntemlerini seçebilirsiniz. Biz Notification'ların çalışma prensibinden dolayı gönderim metodunu POST olarak ayarlıyoruz.

```
toastNotificationIstegi.Method = "POST";
```

Bu işlemin ardından XMLSchema'yı uygun bir şekilde oluşturuyoruz. Ardından bu mesajı bir akış olarak yazacağımız için bu mesajı byte dizisine çeviriyoruz ve gönderiyoruz. Daha sonra gelen cevabı ise aşağıdaki şekilde alıyoruz.

```
string mesaj = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +  
              "<wp:Notification xmlns:wp=\"WPNotification\">" +  
              "<wp:Toast>" +  
              "<wp:Text1>" + TxtTitle.Text + "</wp:Text1>"  
+  
              "<wp:Text2>" + TxtContent.Text +  
"</wp:Text2>" +  
              "<wp:Param>" + TxtParameter.Text +  
"</wp:Param>" +  
              "</wp:Toast>" +  
              "</wp:Notification>";
```

```
byte[] notificationMesaji = Encoding.Default.GetBytes(mesaj);  
toastNotificationIstegi.ContentLength =  
notificationMesaji.Length;  
toastNotificationIstegi.ContentType = "text/xml";  
toastNotificationIstegi.Headers.Add("X-WindowsPhone-Target",  
"toast");  
toastNotificationIstegi.Headers.Add("X-NotificationClass", "2");  
using (Stream requestStream =  
toastNotificationIstegi.GetRequestStream())
```

```
{  
    requestStream.Write(notificationMesaji, 0,  
notificationMesaji.Length);  
}
```

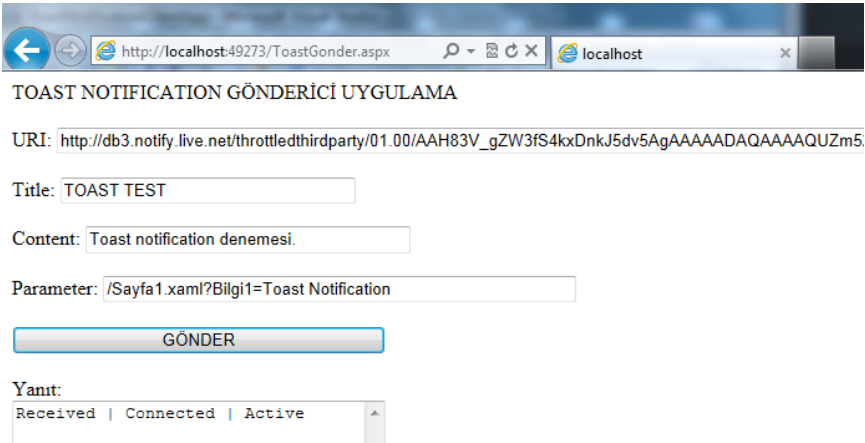
```
HttpWebResponse gonderiSonucu =  
(HttpWebResponse)toastNotificationIstegi.GetResponse();
```

gonderiSonucu nesnesi işinize yarayacak birçok bilgi içerecektir. Gönderinin durumunu, karşı cihazın durumunu ve bağlantı durumu gibi birçok bilgiyi size iletacaktır. Burada gelen mesajları ve bu mesajların anlamlarını aşağıdaki adresten bulabilirsiniz.

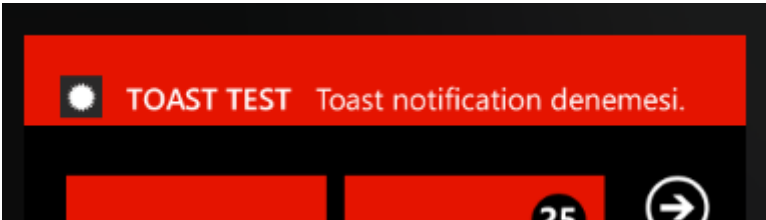
<http://msdn.microsoft.com/en-us/library/ff941100%28v=vs.92%29.aspx>

```
string notificationDurumu = gonderiSonucu.Headers["X-NotificationStatus"];
string notificationKanalDurumu = gonderiSonucu.Headers["X-SubscriptionStatus"];
string cihazBaglantiBilgisi = gonderiSonucu.Headers["X-DeviceConnectionStatus"];
string sonuc = notificationDurumu + " | " + cihazBaglantiBilgisi + " | "
+ notificationKanalDurumu;
TxtCevap.Text = sonuc;
```

Anlatılan kodların devamını ve yukarıdaki bağlantı linkini de kod içerisinde bulabilirsiniz. Kodu da yazdığımıza göre artık Toast Notification'ımızı gönderebiliriz.



The screenshot shows a web browser window with the address bar displaying <http://localhost:49273/ToastGonder.aspx>. The page title is "TOAST NOTIFICATION GÖNDERİCİ UYGULAMA". Below the title, the URI is shown as http://db3.notify.live.net/throttledthirdparty/01.00/AAH83V_gZW3fS4kxDnkJ5dv5AgAAAAADAQAAAAQUZm5. The form includes a "Title" field with the value "TOAST TEST", a "Content" field with the value "Toast notification denemesi.", and a "Parameter" field with the value "/Sayfa1.xaml?Bilgi1=Toast Notification". A "GÖNDER" button is located below the fields. Underneath the button, the "Yanıt:" (Response) section shows a dropdown menu with the selected option "Received | Connected | Active".



2. Tile Notifications Örneği

Şimdi aynı uygulamayı emülatör içerisinde pinleyelim ve bir Tile Notification gönderimi için gerekli hazırlıklara başlayalım. Application Tile konusunda yapmış olduğumuz örnekteki arkaplan için kullandığımız resimleri bu uygulamamıza aktaralım. Resimlerin Build Action özellikleri “Resource” olarak kalsın.

Bu kısımda yapılacak olan işlemlerin aynısını Application Tile konusunda ayrıntılı bir şekilde yapmıştık. Şimdi bu işlemleri Tile Notification kullanarak uzaktan nasıl yapacağımızı öğreneceğiz.

Önceki örnekteki yapılan işlemleri hiç değiştirmeyelim. Uygulama içerisinde Toast Notification için açmış olduğumuz kanalı kullanarak Tile Notification da göndereyim.

ToastNotificationClientApp - MainPage.xaml.cs

MainPage() içerisinde BildirimKanali.BindToShellToast(); yazan satırın hemen altına aşağıdaki kodu eklememiz yeterli olacaktır.

```
BildirimKanali.BindToShellTile();
```

Gördüğünüz gibi Tile Notification için ek bir satır eklememize gerek kalmadı. Bunun nedeni Tile Notification’dan gelen bilginin doğrudan telefon tarafından işleme alınmasıdır. Yani sizin ayrıca bir işlem yapmanıza gerek kalmıyor. Bu işlem sadece Tile için değil tüm Push Notification’lar için geçerlidir.

Şimdi web projemizi tekrar açarak ona yeni bir sayfa ekliyoruz. İsmi TileGonder.aspx olarak değiştirdim.

Tile Notification için gerekli olan nesnelere düşünelim; aklımıza Application Tile konusunda öğrenmiş olduğumuz özellikler geliyor. Bu özellikler; Title, BackgroundImage, Count (Badge), BackContent, BackBackgroundImage ve BackTitle’dır. Daha sonra nereye göndereceğimizi forma aktarabilmek için gerekli olan bir alan ve bu gönderim sonucunu alabileceğimiz yeni bir alana ihtiyaç duyuyoruz. Dolayısıyla TileGonder.aspx sayfamın arayüzünü bu işlemlere yetecek şekilde ayarlıyorum. Bu işlemlerin ardından kod yazımına geçebiliriz.

Daha önce yapmış olduğumuz sayfadan (ToastGonder.aspx.cs) tek farkı sizin de tahmin edebileceğiniz gibi XML yapısı. Tile Notification için XML yapısını aşağıdaki gibi oluşturuyoruz.

```
string mesaj = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
               "<wp:Notification xmlns:wp=\"WPNotification\">" +
               "<wp:Tile>" +
```

```

        "<wp:BackgroundImage>" +
TxtBackgroundImage.Text + "</wp:BackgroundImage>" +
        "<wp:Count>" + TxtCount.Text + "</wp:Count>" +
        "<wp:Title>" + TxtTitle.Text + "</wp:Title>" +
        "<wp:BackBackgroundImage>" +
TxtBackBackgroundImage.Text + "</wp:BackBackgroundImage>" +
        "<wp:BackTitle>" + TxtBackTitle.Text +
"</wp:BackTitle>" +
        "<wp:BackContent>" + TxtBackContent.Text +
"</wp:BackContent>" +
        "</wp:Tile>" +
"</wp:Notification>";

```

Şimdi sayfamızı oluşturmaya başlayabiliriz.

TileGonder.aspx

```

<div>
TILE NOTIFICATION GÖNDERİCİ UYGULAMA<br />
URI:<br />
<asp:TextBox ID="TxtUri" runat="server"
Width="666px"></asp:TextBox>
<br /><br />
FrontTitle:<br />
<asp:TextBox ID="TxtTitle" runat="server"></asp:TextBox>
<br />
<br />
FrontBackgroundImage:<br />
<asp:TextBox ID="TxtBackgroundImage"
runat="server"></asp:TextBox>
<br />
<br />
Count:<br />
<asp:TextBox ID="TxtCount" runat="server"></asp:TextBox>
<br />
BackTitle:<br />
<asp:TextBox ID="TxtBackTitle" runat="server"></asp:TextBox>
<br />
BackBackgroundImage:<br />
<asp:TextBox ID="TxtBackBackgroundImage"
runat="server"></asp:TextBox>
BackContent:<br />
<asp:TextBox ID="TxtBackContent" runat="server"></asp:TextBox>
<asp:Button ID="BtnGonder" runat="server"
Text="GÖNDER" Width="291px" onclick="BtnGonder_Click" />
Yanıt:<br />
<asp:TextBox ID="TxtYanıt" runat="server" Height="84px"
Width="270px"
TextMode="MultiLine"></asp:TextBox>
</div>

```

Arayüzü hazırladıktan sonra şimdi arka plan kodlarını oluşturalım.

```
protected void BtnGonder_Click(object sender, EventArgs e)
{
    try
    {
        string ChannelUri = TxtUri.Text;
        HttpRequest tileNotificationIstegi =
(HttpWebRequest)WebRequest.Create(ChannelUri);

        tileNotificationIstegi.Method = "POST";

        string mesaj = "<?xml version=\"1.0\" encoding=\"utf-8\"?>"
+
        "<wp:Notification xmlns:wp=\"WPNotification\">" +
        "<wp:Tile>" +
        "<wp:BackgroundImage>" + TxtBackgroundImage.Text +
"</wp:BackgroundImage>" +
        "<wp:Count>" + TxtCount.Text + "</wp:Count>" +
        "<wp:Title>" + TxtTitle.Text + "</wp:Title>" +
        "<wp:BackBackgroundImage>" +
TxtBackBackgroundImage.Text + "</wp:BackBackgroundImage>" +
        "<wp:BackTitle>" + TxtBackTitle.Text +
"</wp:BackTitle>" +
        "<wp:BackContent>" + TxtBackContent.Text +
"</wp:BackContent>" +
        "</wp:Tile>" +
        "</wp:Notification>";

        byte[] notificationMessage =
Encoding.Default.GetBytes(mesaj);
        tileNotificationIstegi.ContentLength =
notificationMessage.Length;
        tileNotificationIstegi.ContentType = "text/xml";
        tileNotificationIstegi.Headers.Add("X-WindowsPhone-Target",
"token");
        tileNotificationIstegi.Headers.Add("X-NotificationClass",
"1");

        using (Stream requestStream =
tileNotificationIstegi.GetRequestStream())
        {
            requestStream.Write(notificationMessage, 0,
notificationMessage.Length);
        }

        HttpResponseMessage gonderiSonucu =
(HttpWebResponse)tileNotificationIstegi.GetResponse();
    }
}
```

```

        string notificationDurumu = gonderiSonucu.Headers["X-
NotificationStatus"];
        string notificationKanalDurumu = gonderiSonucu.Headers["X-
SubscriptionStatus"];
        string cihazBaglantiBilgisi = gonderiSonucu.Headers["X-
DeviceConnectionStatus"];
        TxtYanıt.Text = notificationDurumu + " | " +
        cihazBaglantiBilgisi + " | "
+ notificationKanalDurumu;
    }
    catch (Exception ex)
    {
        TxtYanıt.Text = "HATA: " + ex.Message;
    }
}

```

Uygulamamızı çalıştırdıktan sonraki ekran görüntülerine baktığımızda Application Tile'ın hemen değiştiğini görebilirsiniz.



3. Raw Notifications Örneği

Raw Notification'da diğer örneklerdekinden farklı olarak Shell Bind yapmadan doğrudan bu işleme bir event açacağız. Kanalin HttpNotificationReceived event'ini kullanarak bu işlemi gerçekleştireceğiz.

MainPage.xaml.cs

```

public MainPage()
{
    InitializeComponent();
    BildirimKanali = HttpNotificationChannel.Find("Kanal");

    if (BildirimKanali == null)
    {
        BildirimKanali = new HttpNotificationChannel("Kanal");
        BildirimKanali.ChannelUriUpdated += new
        EventHandler<NotificationChannelUriEventArgs>(BildirimKanali_Channel
        UriUpdated);
        BildirimKanali.ShellToastNotificationReceived += new
        EventHandler<NotificationEventArgs>(BildirimKanali_ShellToastNotific
        ationReceived);
        BildirimKanali.Open();
        BildirimKanali.BindToShellToast();
        BildirimKanali.BindToShellTile();
    }

    else
    {
        BildirimKanali.ChannelUriUpdated += new
        EventHandler<NotificationChannelUriEventArgs>(BildirimKanali_Channel
        UriUpdated);
        BildirimKanali.ShellToastNotificationReceived += new
        EventHandler<NotificationEventArgs>(BildirimKanali_ShellToastNotific
        ationReceived);
        //Raw Notification için event yaratılıyor.
        BildirimKanali.HttpNotificationReceived += new
        EventHandler<HttpNotificationEventArgs>(BildirimKanali_HttpNotificat
        ionReceived);

        System.Diagnostics.Debug.WriteLine(BildirimKanali.ChannelUri.ToStrin
        g());
    }
}

```

Açmış olduğumuz event'in içeriğini aşağıda görebilirsiniz. *HttpNotificationEventArgs* kullanılarak gelen bilgiyi yakalıyoruz ve *MessageBox* içerisinde gösteriyoruz.

```

void BildirimKanali_HttpNotificationReceived
(object sender, HttpNotificationEventArgs e)
{
    string gelenMesaj;

    using (System.IO.StreamReader oku = new
    System.IO.StreamReader(e.Notification.Body))
    {
        gelenMesaj = oku.ReadToEnd();
    }
}

```

```

    }

    Dispatcher.BeginInvoke(() =>
        MessageBox.Show(String.Format("Yeni RAW Notification
- {0}:\n{1}",
            DateTime.Now.ToShortTimeString(), gelenMesaj));
    }

```

Client uygulamamıza gerekli ekleri yaptıktan sonra web projesine geçiyoruz. RawGonder.aspx sayfasını ve arka plan kodlarını oluşturalım.

Raw Notification'larda Value'ların olduğundan bahsetmiştik. Dolayısıyla formumuzu buna göre hazırlıyoruz.

RawGonder.aspx

```

<div>
    RAW NOTIFICATION GÖNDERİCİ UYGULAMA<br />
    <br />
    URI:
    <asp:TextBox ID="TxtUri" runat="server"
Width="666px"></asp:TextBox>
    Değer 1:
    <asp:TextBox ID="TxtDeger1" runat="server"></asp:TextBox>
    <br />
    Değer 2:
    <asp:TextBox ID="TxtDeger2" runat="server"></asp:TextBox>
    <br />
    <asp:Button ID="BtnSendRaw" runat="server"
onclick="BTnSendRaw_Click"
    Text="GÖNDER" Width="247px" />
    <br />
    Yanıt:<br />
    <asp:TextBox ID="TxtYanit" runat="server" Height="105px"
Width="247px"
    TextMode="MultiLine"></asp:TextBox>
</div>

```

RawGonder.aspx.cs

```

protected void BTnSendRaw_Click(object sender, EventArgs e)
{
    try
    {
        string ChannelUri = TxtUri.Text;
        HttpRequest rawNotificationIstegi =
(HttpWebRequest)WebRequest.Create(ChannelUri);
        rawNotificationIstegi.Method = "POST";

        // Raw notification mesajı oluşturuluyor.
        string mesaj = "<?xml version=\"1.0\" encoding=\"utf-8\"?>"

```

+


```

        "<root>" +
            "<Value1>" + TxtDeger1.Text + "<Value1>" +
            "<Value2>" + TxtDeger2.Text + "<Value2>" +
        "</root>";
        byte[] notificationMesaji =
        Encoding.Default.GetBytes(mesaj);
        rawNotificationIstegi.ContentLength =
        notificationMesaji.Length;
        rawNotificationIstegi.ContentType = "text/xml";
        rawNotificationIstegi.Headers.Add("X-NotificationClass",
        "3");

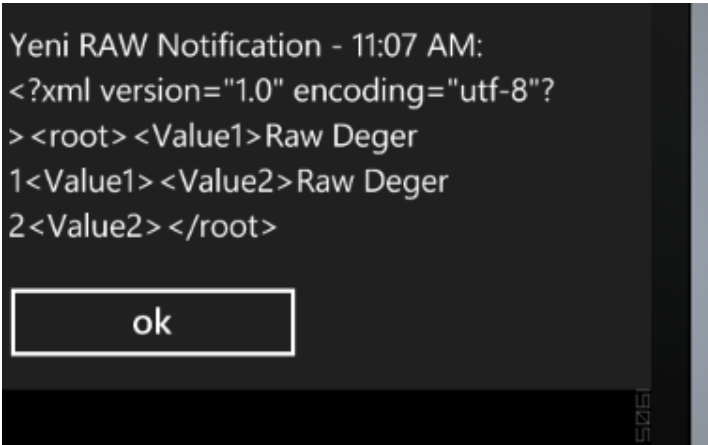
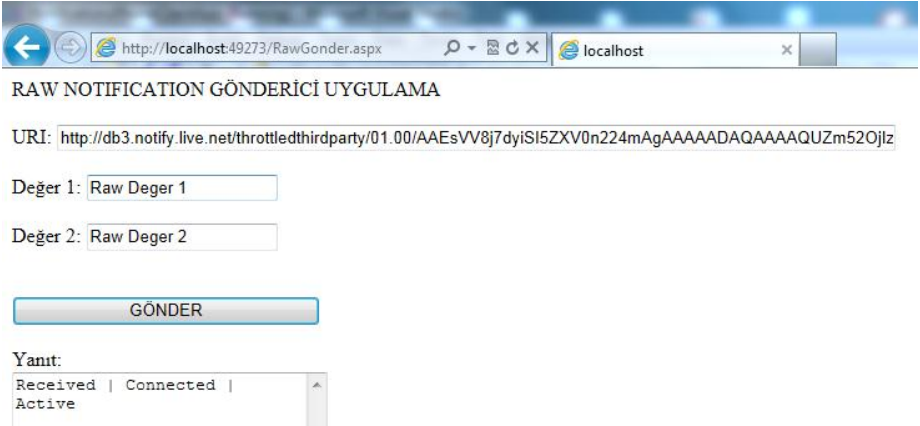
        using (Stream requestStream =
        rawNotificationIstegi.GetRequestStream())
        {
            requestStream.Write(notificationMesaji, 0,
        notificationMesaji.Length);
        }

        HttpResponseMessage gonderiSonucu =
        (HttpResponseMessage)rawNotificationIstegi.GetResponse();

        string notificationDurumu = gonderiSonucu.Headers["X-
        NotificationStatus"];
        string notificationKanalDurumu = gonderiSonucu.Headers["X-
        SubscriptionStatus"];
        string cihazBaglantiBilgisi = gonderiSonucu.Headers["X-
        DeviceConnectionStatus"];
        TxtYanit.Text = notificationDurumu + " | " +
        cihazBaglantiBilgisi + " | "
        + notificationKanalDurumu;
    }
    catch (Exception ex)
    {
        TxtYanit.Text = "HATA: " + ex.Message;
    }
}

```

Web uygulamamızı da hazırladıktan sonra artık Raw Notification testine başlayabiliriz. Hem Windows Phone hem de web uygulamamızı açalım. Ardından verileri girerek gönder butonuna basalım.



Böylelikle son notification örneğimizi yapmış olduk. Örnek uygulamaları kitap ile birlikte dosya halinde bulabilirsiniz. Örnek uygulamalar, ASP.NET 4.0 ve Windows Phone 7.1 SDK ile gerçekleştirilmiştir.



Windows Phone üzerinde bir çok fiziksel sensör bulunmaktadır ve bu sensörler sayesinde uygulamalarınızın daha fonksiyonel olmasını sağlayabiliyoruz. Windows Phone biz geliştiricilere bu sensörlerden gelen verileri okuyabilmek için API'ler sunmaktadır. Bu API'lerin neler olduğunu ve nasıl kullanılacağını bu bölümde inceleyeceğiz.

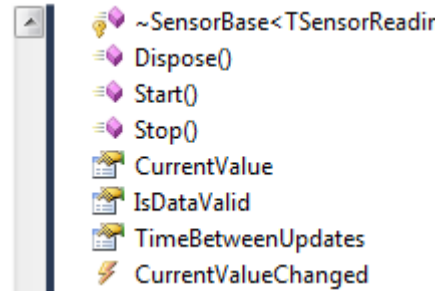
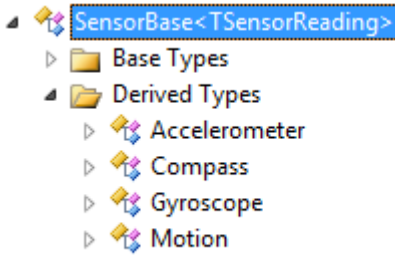
Yazardan...

7. Sensörler ve Servisler

Windows Phone üzerinde bir çok fiziksel sensör bulunmaktadır ve bu sensörler sayesinde uygulamalarınızın daha fonksiyonel olmasını sağlayabiliyoruz. Windows Phone biz geliştiricilere bu sensörlerden gelen verileri okuyabilmek için API'ler sunmaktadır. Kullandığımız bütün sensörler `SensorBase<TSensorReading>`^[1] sınıfından türemektedir. Bu sensör API'lerinin neler olduğunu, `SensorBase` sınıfımızın elemanlarını örneklerle birlikte irdelemeye başlayalım.

- `SensorBase` sınıfı
- Accelerometer (ivme ölçer)
- Compass (pusula)
- Gyroscope (jiroskop)
- Motion (hareket)

a. `SensorBase` Sınıfı



`SensorBase` sınıfının özelliklerine baktığımızda `Start`, `Stop` metodları, `CurrentValue`, `IsValid`, `TimeBetweenUpdates` özellikleri ve `CurrentValueChanged` olayı bulunmaktadır. `SensorBase` sınıfının türetildiği sınıflara baktığımızda da `Accelerometer`, `Compass`, `Gyroscope` ve `Motion` sınıflarını görüyoruz. Bunları incelemeye başlayalım.

`Start` metodu türetildiği sınıfın fiziksel sensöründen gelen verilerin okunmaya başlamasını sağlamaktadır. `Stop` metodu da tam aksine verilerin gelmesini durdurmaktadır. `CurrentValue` özelliği sensörün o anki değerini `ISensorReading` tarafından implemente edilmiş şekilde döndürmektedir. `IsValid` özelliği gelen verinin doğruluğunu kontrol etmektedir. `TimeBetweenUpdates` özelliği ise `CurrentValueChanged` olayı tarafından tetiklenen olaylar arasındaki süreyi `TimeSpan` olarak döndürmektedir. `CurrentValueChanged` olayımız da sensörümüze yeni bir veri geldiği anda tetiklenmektedir.

b. Accelerometer (İvme Ölçer)



Accelerometer sensörü biz geliştiricilere cihazın X, Y ve Z eksenindeki durumunu vermektedir. Cihazımızın tam ortası başlangıç noktası (0,0,0) olmaktadır. Soldaki resmi incelediğimizde hangi yönün hangi eksen olduğunu daha rahat anlayabilirsiniz.

Accelerometer'ın CurrentValueChanged olayını incelediğimizde argüman olarak dönen değerimizin 3 boyutlu bir vektör olduğunu görüyoruz. Ayrıca bu Vector3 sınıfının XNA içerisinde bulunduğunu farketmişsinizdir. XNA namespace'ini ekledikten

sonra Vector3 sınıfını kullanabiliriz. Vector3 sınıfının içeriğine baktığımızda X, Y ve Z gibi özellikleri olduğunu görüyoruz. Bu özellikler Start metodu çalıştıktan sonra cihazın X, Y ve Z eksenlerindeki değerlerini göstermektedir.

Bunu bir örnekle pekiştirelim.

```
Accelerometer accelerometer;  
Vector3 accVector;  
  
void MainPage_Loaded(object sender, RoutedEventArgs e)  
{  
    if (!Accelerometer.IsSupported)  
    {  
        txtAccelerometer.Text = "Accelerometer desteklenmiyor!";  
    }  
  
    accelerometer = new Accelerometer();  
    accVector = new Vector3();  
  
    try { accelerometer.Start(); }  
    catch { MessageBox.Show("Accelerometer başlatılmadı!"); }  
    accelerometer.CurrentValueChanged += new  
    EventHandler<SensorReadingEventArgs<AccelerometerReading>>(accelerom  
    eter_CurrentValueChanged);  
}
```

```

void accelerometer_CurrentValueChanged(object sender,
SensorReadingEventArgs<AccelerometerReading> e)
{
    Dispatcher.BeginInvoke(() =>
AccelerometerVerisiGuncelle(e.SensorReading));
}

private void AccelerometerVerisiGuncelle(AccelerometerReading
accelerometerReading)
{
    accVector = accelerometerReading.Acceleration;
    txtAccelerometer.Text = String.Format("X : " + accVector.X +
"{0}Y : " + accVector.Y + "{0}Z : " + accVector.Z,
Environment.NewLine);
}

```

Gördüğümüz gibi ilk olarak Accelerometer'ın desteklenip desteklenmediğini kontrol ettik. Desteklenmediği durumda kullanıcıyı bilgilendirdik. Eğer destekleniyorsa işlemler devam edecek ve accelerometer'ımız başlatılacaktır. Başlatılmadığı durumda yine kullanıcıyı bilgilendirdik. Eğer başlatılırken bir sorun yaşanmadıysa CurrentValueChanged olayımızın çağırıldığında yapılacak işlemleri tanımlamaya başladık. Burada Dispatcher ile AccelerometerVerisiGuncelle metodunu farklı UI thread üzerinde işlem yaparken sorun yaşamamak için kullandık ve arka planda çalışan başka bir thread içerisinde metod işlemleri yapılmaktadır. AccelerometerVerisiGuncelle metodumuzda gelen AccelerometerReading tipindeki parametreyi accVector değişkenimize atadık ve accVector değişkeninin X, Y ve Z özelliklerini txtAccelerometer'a yazdırdık.

c. Compass (Pusula)



Hepimiz çocukluğumuzda mutlaka pusulalarla oynamışızdır diye düşünüyorum. Tabii ki yanda gördüğümüz pusulalardan bahsetmiyorum :) Bilmeyenler için pusulanın yön bulmak amaçlı kullanıldığını söyleyebiliriz.

Windows Phone içerisinde manyetik pusulalardan bulunmaktadır ve bu pusulanın değerini alabilmek için yine SensorBase tarafından türetilmiş olan Compass sınıfını kullanacağız. Örneğimize bakacak olursak.

```

Compass compass;
Vector3 comVector;

void MainPage_Loaded(object sender, RoutedEventArgs e)
{

```

```

if (!Compass.IsSupported)
{
    txtCompass.Text = "Compass desteklenmiyor!";
}

compass = new Compass();
comVector = new Vector3();

try { compass.Start(); }
catch { MessageBox.Show("Compass başlatılamadı!"); }

compass.CurrentValueChanged += new
EventHandler<SensorReadingEventArgs<CompassReading>>(compass_Current
ValueChanged);
}

void compass_CurrentValueChanged(object sender,
SensorReadingEventArgs<CompassReading> e)
{
    Dispatcher.BeginInvoke(() =>
CompassVerisiGuncelle(e.SensorReading));
}

private void CompassVerisiGuncelle(CompassReading compassReading)
{
    comVector = compassReading.MagnetometerReading;
    txtCompass.Text = String.Format("HeadingAccuracy : " +
compassReading.HeadingAccuracy + "{0}" +
    "MagneticHeading : " + compassReading.MagneticHeading +
"{0}" +
    "TrueHeading : " + compassReading.TrueHeading + "{0}" +
    "X : " + comVector.X + "{0}Y : " + comVector.Y + "{0}Z : " +
comVector.Z, Environment.NewLine);}

```

d. Gyroscope (jiroskop)

Jiroskop, (İngilizce: Gyroscope, Gyro) veya Yalpalık, Cayroskop, Cayro, yön ölçümü veya ayarlamasında kullanılan, açısal dengenin korunması ilkesiyle çalışan bir alet. Jiroskopik hareketin temeli fizik kurallarına ve açısal momentumun korunumu ilkesine dayalıdır. ^[2]

Gyroscope verisini nasıl alabileceğimize göz atalım;

```

Gyroscope gyroscope;
Vector3 gyVector;

void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    if (!Gyroscope.IsSupported)

```

```

{
    txtGyroscope.Text = "Gyroscope desteklenmiyor!";
}

gyroscope = new Gyroscope();
gyVector = new Vector3();

try { gyroscope.Start(); }
catch { MessageBox.Show("Gyroscope başlatılamadı!"); }

gyroscope.CurrentValueChanged += new
EventHandler<SensorReadingEventArgs<GyroscopeReading>>(gyroscope_Cur
rentValueChanged);
}

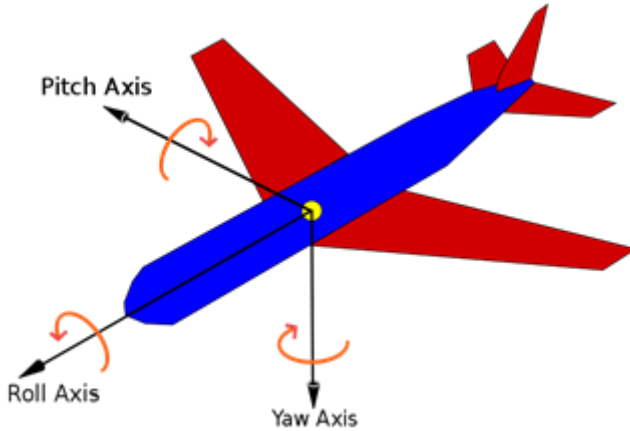
void gyroscope_CurrentValueChanged(object sender,
SensorReadingEventArgs<GyroscopeReading> e)
{
    Dispatcher.BeginInvoke(() =>
GyroscopeVerisiGuncelle(e.SensorReading));
}

private void GyroscopeVerisiGuncelle(GyroscopeReading
gyroscopeReading)
{
    gyVector = gyroscopeReading.RotationRate;
    txtGyroscope.Text = String.Format("X : " + gyVector.X + "{0}Y :
" + gyVector.Y + "{0}Z : " + gyVector.Z, Environment.NewLine);
}

```

e. Motion (Hareket)

Motion sınıfını cihaz hareketlerine göre gelen değişkenlerin son durumlarını almak için kullanırız. Aldığımız değişkenlere göre de UI üzerindeki nesnelerin pozisyonlarını değiştirebilir veya uygulamanıza göre farklı işlemler yaptırabiliriz. Örnek vermek gerekirse augmented reality ile ilgili bir uygulama yaptığınızda kameradaki ilk pozisyona göre ekrandaki nesnelerin yeni pozisyonlarını veya farklı bir açıda görünmelerini sağlamak gibi ihtiyaçlarınız olabilir. Bu tarz ihtiyaçlarımızı karşılamak için Motion sınıfından yardım alabiliriz. Nesnelimizin pozisyonlarını değiştirirken Pitch, Yaw ve Roll değişkenlerinden gelen değerleri kullanacağız. Bu değişkenlerin Türkçe karşılıklarına bakacak olursak; Pitch: Yunuslama, Yaw: Sapma, Roll: Yuvarlanma. Bunlar uçuş terimleridir fakat programlama içerisinde de aynı anlamlarla kullanılmaktadır. Aşağıdaki resimde hangi terimin hangi eksene denk geldiğini görebilirsiniz.



Diğer örneklerimizdeki gibi bu örneğimizde de benzer kodlar kullanacağız.

```
Motion motion;

void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    if (!Motion.IsSupported)
    {
        txtMotion.Text = "Motion desteklenmiyor!";
    }

    motion = new Motion();

    try { motion.Start(); }
    catch { MessageBox.Show("Motion başlatılamadı!"); }

    motion.CurrentValueChanged += new
    EventHandler<SensorReadingEventArgs<MotionReading>>(motion_CurrentVa
    lueChanged);
}

void motion_CurrentValueChanged(object sender,
SensorReadingEventArgs<MotionReading> e)
{
    Dispatcher.BeginInvoke(() =>
MotionVerisiGuncelle(e.SensorReading));
}

private void MotionVerisiGuncelle(MotionReading motionReading)
{
    AttitudeReading attitude = motionReading.Attitude;
```

```
txtMotion.Text = String.Format("Pitch : " + attitude.Pitch +  
"{0}Yaw : " + attitude.Yaw + "{0}Roll : " + attitude.Roll,  
Environment.NewLine);  
}
```

Örneklere farkettiğiniz üzere sadece verilerin nasıl alınacağı üzerine bilgiler verildi. Bu verileri kullanan uygulamanızın fonksiyonallitesi sizin yaratıcılığınızla doğru orantılı olarak artacaktır.



Merhaba,

Ben Faruk Can Özdemir. Çankaya Üniversitesi Microsoft Student Partner görevini yürütürken, siz değerli kullanıcı ve geliştiricilere böyle bir kılavuz hazırlama gereksinimi duyduk. Microsoft Türkiye ve değerli MSP dostlarımız ile hazırladığımız bu kılavuzda umarız ki ihtiyacınız olan başlangıç noktalarını bulabileceksiniz.

Bu bölümde, Windows Phone 7 Mobil İşletim Sistemi üzerinde, mobil hayatınıza entegre olarak çalışan ve günlük hayatınızda sıklıkla kullandığınız fotoğraf çekme, mesaj yazma, arama yapma, rehberle kişi kaydetme gibi temel ancak hayati uygulamaların tabanlarını, Windows Phone 7 uygulamalarınıza nasıl entegre edeceğinizi anlatacağız. Bu anlatımı yaparken en basit örneklerden ve modellerden yola çıktık. Bu örneklemeleri irdeleyerek ve kendi uygulamalarınıza göre şekillendirerek çok daha fazla "Task" ortaya çıkarabilir ve bunları kullanıcılarınıza yararlı bir şekilde sunabilirsiniz. Windows Phone 7 teknolojisi ile beraber gelen hazır işlemler ile kullanıcıların uygulamalar olsun standart kullanım olsun yoğun bir şekilde kullandıkları işlemleri, sizler de geliştireceğiniz uygulamalara entegre ederek kullanıcıların, uygulamanızı daha efektif ve verimli kullanmasını sağlayabilirsiniz.

"Task"ları kullandıkça, geliştirmelerinize hız ve modülerlik katacaksınız. Windows Phone 7 Mobil İşletim Sisteminin hazır gelen bu muhteşem dünyasında siz de çok hızlı ilerleyeceksiniz.

8. Tasklar

Telefon sistemlerinde “Task” denilen ve Türkçe’ ye “Ana İşlemler” olarak çevirebileceğimiz, Windows Phone 7’ nin içinde belli mini işlemleri çalıştırabilmek için kullanılan işlemler bütünlerini bu bölümde inceleyeceğiz. Bu sistemler kameranız ile fotoğraf çekmekten tutun da e-mail atarken ekranda çıkan ikonlara kadar birçok işlemi kontrol eden bir aile. Öyle büyük bir aile ki sizin işlemlerinizi yapmak için geri planda telefonun tüm kapasitesini en güzel şekilde kullanarak bir yerden veri çekip bir yere aktarıyorlar, bunları harmanlayıp ekrana ve bizlere tek tuş veya tek işlem haline getiriyorlar. Bu bölümde bu küçük iş bitiricileri daha detaylı şekilde inceleyeceğiz. Aslında küçük gibi görünen ancak telefonları telefon yapan bu işlemleri daha yakından tanıdığımızda Windows Phone 7’ yi de daha verimli ve etkili kullanabileceğiz.

Peki bu sistemleri daha iyi anlamak bize ne gibi avantajlar sağlar? Özellikle geliştirici yönünde nasıl ön plana geçer? Konunun bu kısmında kendinizi Windows Phone uygulamaları geliştiren biri gibi düşünmeniz önemli çünkü Windows’ un telefon sistemlerini diğerlerinden ayıran en önemli özelliği de belki bu, geliştirilmesinin de heyecan vermesi.

Bir geliştirici olarak belli bir konuda uygulama geliştireceksiniz. Bu uygulama hakkındaki fikirlerinizi ve öngörülerinizi gerçekleştirmek için ne gibi öğelere ihtiyaç duyacağınızı sıraladınız, bu öğelerin nasıl yapılacağı hakkında da fikirleriniz var ve belki de bu fikirler ve çalışmalarınız hazır. İşte bir uygulamayı hayata geçirirken ihtiyaç duyacağınız ve kodlarınızı, işlemlerinizi daha hızlı ve verimli çalıştıracak olan da bu “Task” lar. Uygulamanız ne ile alakalı olursa olsun, nasıl veri toplayacak, nasıl verileri kullanıcıya sunacak ve bunları hangi görünüm, tema, renk gibi görsel öğelerle sunacak olsun “Task” lar size bu işlemleri yapmak için kestirme yollar sunar.

Örneğin bir fotoğraf uygulamasında, bir geliştirici olarak sizler uygulamanın temel kodlarını oluşturursunuz ve “Task” lar sizin kodlarınızı Windows altyapısı ile harmanlayarak daha hızlı bir şekilde, ek ve uğraştırıcı diğer kodlara gerek kalmadan birbirine bağlar. Veritabanları, görsel öğeler, kamera öğeleri gibi işlemler birbiri ardına Windows altyapısı ile birleştirilerek uygulamanız hazır hale getirilir. “Task” lar olmadan bu işlemleri, örneğin kamerayı açma, fotoğraf çekme vb. etkiler için tek tek kod blokları yazmanız gerek. “Task” lar sizi bu yüklerden kurtaran ve size kısa yollar sunarak daha seri uygulama geliştirmenizi sağlayan minik yardımcımlar aslında.

“Task” ları incelerken dikkat edeceğimiz nokta ise her işlemi kendi içerisinde değerlendirmek olacaktır. Görsel ve interaktif öğeler dışındaki kısmı ayrı, bu kısımları ayrı olarak inceleyeceğiz ve en son aralarındaki bağlantıları, ortaklıkları ve bilinmesi gerekenleri sıralayacağız. Bu şekilde her işlemi daha ayrıntılı ve self olarak inceleme fırsatı bulacağız.



Microsoft Windows Phone 7’ in “Task” ları, “Microsoft.Phone.Tasks Namespace” adlı bir kod kütüphanesinde bulunur ve buradan işleme koyulur. Burada, çeşitli uygulamalar için geliştirilmiş olan ve kodlamalarda “Library” olarak tanımlanan kütüphaneler mevcuttur. Task’ ları gerçekleştirmek için kullanacağımız tüm kütüphaneler burada barındırılır. Kodlarımızı buradaki kütüphanelerce belirlenmiş değerleri ekleyerek Windows Phone üzerinde işlemleri kolaylıkla yapabiliriz.

```
Microsoft.Phone.Tasks Namespace
├── AddressChooserTask Class
├── AddressResult Class
├── BingMapsDirectionsType Class
├── BingMapsTask Class
├── CameraCaptureTask Class
├── ChooserBase(TTaskEventArgs) Class
├── ConnectionSettingsTask Class
├── ConnectionSettingsType Enumeration
├── EmailAddressChooserTask Class
├── EmailComposeTask Class
├── EmailResult Class
├── GameInviteTask Class
├── LabeledMapLocation Class
├── MarketplaceContentType Enumeration
├── MarketplaceDetailTask Class
├── MarketplaceHubTask Class
├── MarketplaceReviewTask Class
├── MarketplaceSearchTask Class
```

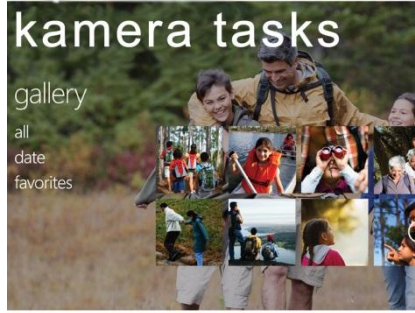
```
MediaLocationType Enumeration
MediaPlaybackControls Enumeration
MediaPlayerLauncher Class
MediaPlayerOrientation Enumeration
PhoneCallTask Class
PhoneNumberChooserTask Class
PhoneNumberResult Class
PhotoChooserTask Class
PhotoResult Class
SaveContactResult Class
SaveContactTask Class
SaveEmailAddressTask Class
SavePhoneNumberTask Class
SaveRingtoneTask Class
SearchTask Class
```

```
ShareLinkTask Class
ShareStatusTask Class
ShareTaskBase Class
SmsComposeTask Class
TaskEventArgs Class
TaskResult Enumeration
WebBrowserTask Class
```

Bu görevleri aşağıdaki başlıklar ve konular dahilinde tek tek, örneklerle ve renkli görsellerle inceleyeceğiz;

- Kamera ve Fotoğraf Görevleri
- Arama Görevleri
- SMS Görevleri
- E-Mail Görevleri
- Diğer Görevler (Search, Marketplace, Maps vb.)

Yukarıdaki “Task” listesine siz de online olarak göz atmak ve incelemek isterseniz, [http://msdn.microsoft.com/en-us/library/ff428753\(v=VS.92\).aspx](http://msdn.microsoft.com/en-us/library/ff428753(v=VS.92).aspx) adresinden ulaşabilirsiniz. Ayrıca konumuz ile alakalı sorularınızı da Microsoft MSDN Forumlarında da tartışabilirsiniz.



Kamera ve Fotoğraflar



Güncel hayatta belki de bir Windows Phone' un kullanılabileceği en etkili yönlerinden biri de multimedia özelliklerinin başı olarak sıralayabileceğimiz kamera ve fotoğraflardır. Kullanıcıların yoğun olarak kullandığı ve uygulama geliştiricilerinin de etkin olarak üzerine uygulama yaptıkları önemli bölümlerden biri. Kamera ve fotoğraf bölümlerinin ve bunlarla alakalı işlemlerin, bunlara uygun uygulama geliştirmenin, bu öğeleri en azından dijital olarak düzenlemenin, geliştirmenin, efektlemenin ve etkin biçimde interaktif işlemlerle internet ortamlarına bağlamanın ne kadar önemli olduğunu ve bir geliştirici için ne kadar önem taşıdığını söylememize gerek yok sanırım.

Kamera ve fotoğraf işlemlerini kontrol eden görevler, yani Task' lar hem kod altyapısına hem de grafik altyapısına ve etkisine sahip olduğundan önem verilmesi gereken bir konu. Uygulamalar içerisinde kamera eylemlerini en optimum başarımda yapmak size ve uygulamalarınıza büyük avantajlar sağlayacaktır. İnteraktif uygulamalarda ve özellikle fotoğraf slide durumlarında bu taskları yoğun olarak kullanacaksınız.

Kamera Task' larından gösterebileceğimiz en önemli task, "CameraCaptureTask". Bu görev sayesinde kamera uygulamasını açıp bir görüntü elde edebiliyorsunuz. Kamera donanımını kullanarak size fotoğraf sağlayan bir task da denilebilir.

CameraCaptureTask



Bu küçük uygulama kamera donanım altyapısını kullanarak tek bir tuş ile fotoğraf çekmenizi sağlar. İşte “Task” lar size bu gibi işlemlerde kolaylık sağlayarak uygulamalarınızı daha seri üretmenize olanak sağlayarak minik yardımcılardır. CameraCaptureTask olmadan tek tek kamera donanımına ulaşmak için değişik kodlar yazmanız gerekecekti ancak şimdi sadece Task’ ımızı tanımlayıp anında uygulamamız içerisinde kullanabilmekteyiz.

Fotoğraf Task’ larına geldiğimizde, karşımıza iki önemli task çıkıyor, biri fotoğrafları seçmede kullanılan Photo Chooser Task , diğeri ise Photo Result Task. Photo Result Task bir lokal tanımlayıcıdır. Bu sebepten bu Task’ ın açılımına girmeyeceğiz.

Önce kamera uygulamamız için uygulamalarımız içerisinde kullanabileceğimiz bir değişikken tanımlıyoruz ;

```
CameraCaptureTask Kameragorev = new CameraCaptureTask(); // Task’ımızı tanımladık
```

Bu değişikkeni kullanım yöntemi ise şöyle ;

```
Kameragorev.Show();  
Kameragorev.Completed += Eventhandler < PhotoResult >  
(Kameragorev.Completed);  
  
public void Kameragorev_Completed(object sender, PhotoResult a )  
{  
    Debug.WriteLine (“Dosya ismi: “ + a.Originalfilename);  
    Debug.WriteLine (“Boyutu: “ + a.Chosenphoto.Length + “ Bytes” );  
}
```

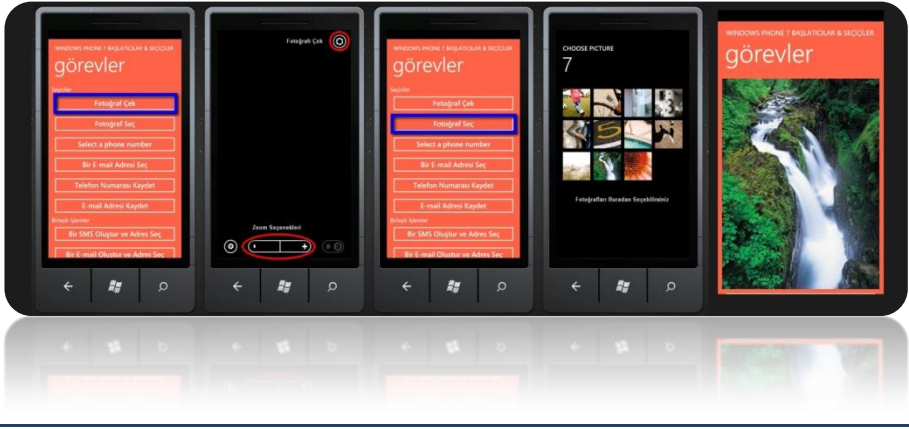


Photo Chooser Task

Bu görev ile bir uygulama içerisinde veya dışarısında işlem halindeyken galerinizden fotoğraf çekebilir ve bunları paylaşabilirsiniz.

Önce kamera uygulamamız için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz ;



```
PhotoChooserTask Fotosecmegorev = new PhotoChooserTask(); // Task'ımızı tanımladık
```

Bu değişkeni kullanım yöntemi ise şöyle ;

```
Fotosecmegorev.Tamam += EventHandler < PhotoResult >  
(Fotosecmegorev_Completed);  
Fotosecmegorev.Show();  
  
public void Kameragorev_Completed(object sender, PhotoResult a )  
{  
    Debug.WriteLine (“Boyutu: “ + a.Chosenphoto.Length + “ Bytes” );  
    Debug.WriteLine (“Dosya ismi: “ + a.Originalfilename);  
}
```




Kullanıcıların mobil uygulamalar ve telefonlar üzerinde istediği büyük niteliklerden olan arama yapma ve metin mesajları gönderme işlemleri yapılan uygulamaların bazılarının kilit noktası. SMS' lerinizi kontrol eden ve düzenleyen, aramalarınızı şekillendiren, adres defterinizden numaralar seçen ve bunları düzenleyebilen uygulamalar bazı kullanıcıların göz bebeği. Bu durum da uygulamalarınıza bu tarz özellikler katarak daha fazla kullanıcıya hitap edebileceğiniz anlamına geliyor. Uygulamalar içerisinde konumlayacağınız arama ve SMS kullanma özellikleri sayesinde, internet kullanımı dışında ekstra bir interaktivite elde edeceğinizi söylememize de gerek yok sanırım. Windows Phone 7' de uygulamalar üzerinden aramalar yapmak ve metin mesajları göndermek artık daha da kolay çünkü yine bu işlemlerimiz için bizlere yardımcı olacak minik görev bütünlüklerine sahibiz.

Bu bölümümüzde yukarıda bahsettiğimiz işlemleri gerçekleştiren dört önemli "Task" ı inceleyeceğiz. Bunlar; arama yapmanızı, rehberinize telefon kaydetmenizi, bu telefonları seçmenizi ve metin mesajları yaratıp gönderebilmenizi sağlayan minik uygulamalar.

Phone Call Task

Bu görev ile arama ekranına geçmeden önce sizden bir onay istenecek, onay verdiğiniz anda, standart Windows Phone 7 yazılımına uygun olarak arama ekranı görüntülenecek ve buradan, kullanıcı, istediği gibi ister aramasına devam edecek, ister arama seslerini yükseltip alçaltabilecek, ister ek numaralar girebilecek ve dahası. Bu küçük "Task" ile, uygulamanız içerisinden, örneğe dayanarak, Microsoft Türkiye' yi interaktif olarak aratabileceksiniz.



Önce arama uygulamamız için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz ;

```
PhoneCallTask arama_gorev = new PhoneCallTask(); // Task'ımızı tanımladık
```

Bu değişkeni kullanım yöntemi ise şöyle ;

```
Arama_gorev.DisplayName = "Microsoft Türkiye";  
Arama_gorev.PhoneNumber = "0212xxxxxxx"; // Aranacak olan numara bilgileri  
Arama_gorev.Show();
```



Phone Number Chooser Task

Bu “Task” ile uygulamanız içerisinde bir numaraya ihtiyaç duyduğunuzda, bir numara değerini belli bir yere işlemeniz gereken durumlarda, uygulamanız içerisinde bir form doldurmada numaranıza, numaralara ihtiyaç duyduğunuzda veya uygulamanızı arkadaşlarınızla paylaşmanız için göndereceğiniz davetiyeler için telefon numaralarına ihtiyaç duyduğunuzdaki gibi birçok telefon numarası seçme işleminde hızlı bir şekilde kullanabilirsiniz.

Önce numara seçme görevi için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz;



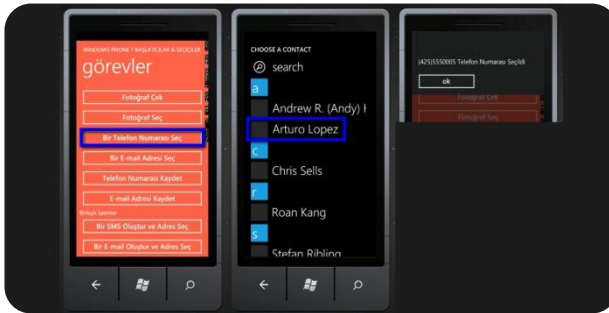
```
PhoneNumberChooserTask numsec_gorev = new  
PhoneNumberChooserTask();
```

// **Task**'ımızı tanımladık

Bu değişkeni kullanım yöntemi ise şöyle ;

```
numsec_gorev.Completed += EventHandler < PhoneNumberResult >  
(numsec_gorev_Completed);  
numsec_gorev.Show();  
  
public void numsec_gorev_Completed(object sender, PhoneNumberResult number1 )  
{  
    Debug.Writeline (number1.PhoneNumber );  
}
```

// **Uygulama çalıştırıldığında seçtiğiniz ismin numarasını programınıza döndürerek kullanmanıza olanak verecektir.**



Save Phone Number Task

Önce numara seçme görevi için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz ;

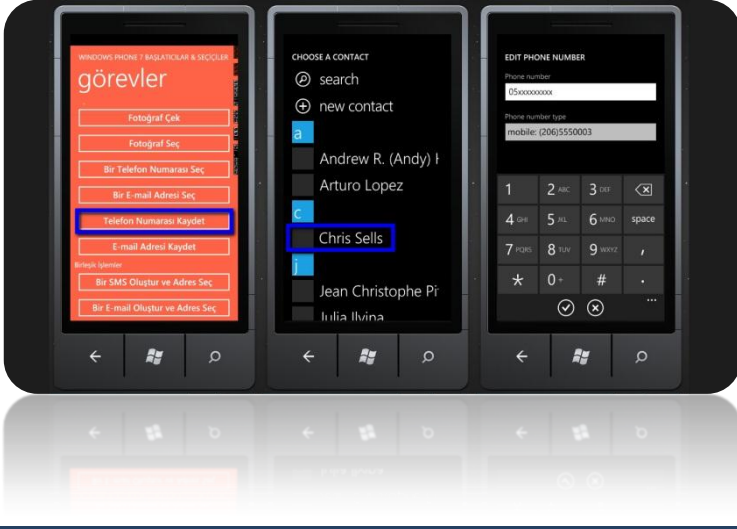
```
SavePhoneNumberTask numkaydet_gorev = new SavePhoneNumberTask();
```

// **Task'ımızı tanımladık**

Bu değişkeni kullanım yöntemi ise şöyle ;

```
numkaydet_gorev.PhoneNumber = "xxxxxxxxxxx"; // Kaydetmek istediğimiz numara değerleri  
numkaydet_gorev.Show();
```

Bu "Task", uygulamanız içerisinde bir numarayı kaydetme ihtiyacı duyarsanız kullanabileceğiniz minik görevlerden biri. Uygulamanız üzerinden interaktif şekilde telefon rehberinize numara ekleyebilmeniz, hem kullanıcılarınız için hem de sizin kodlarınızı ve programınızı daha iyi idare edebilmemiz için, özellikle rehber ve bunun gibi bölümlerde, avantaj sağlayacaktır.



SMS Compose Task

Önce numara seçme görevi için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz ;

```
SmsComposeTask mesajgonder_gorev = new SmsComposeTask(); // Task'ımızı tanımladık
```

Bu değişkeni kullanım yöntemi ise şöyle ;

```
mesajgonder_gorev.Show(); // Bu kodu eklemenizle artık kullanıcı SMS gönderebilecek
```

Bu kolay ve küçük “Task” ile uygulamalarınız artık metin mesajları gönderme altyapısına da sahip olabilecek. Bu kadar küçük bir kodun bu işlemi yapabilmesinin ardında, önceden de bahsettiğimiz, “Task” ların Windows Phone 7 altyapısı ile olan muazzam bağlantısı yatıyor. Bu görev ile kullanıcılar, uygulamanız üzerinden metin mesajları gönderebilecekler, peki bu ne demek? Uygulamadan bağımsız olarak bir metin mesajı gönderilseydi uygulamanızın geri plana alınması gerekirdi ve bu da uygulamanızın önemini ve kullanılabilirliğini düşürürdü ancak bu görevler ile artık kullanıcılar daha interaktif şekilde uygulamanızı kullanabilecekler ve bir geri plana dönme mevzusu söz konusu olmayacak. Windows Phone 7 altyapısının inceliklerinden biri de bu tarz küçük bağlantılar ile uygulamalarınıza çok daha fazla değer katabiliyor olmanız belki de.





Windows Phone 7'in etkin ve kullanışlı özelliklerinden biri de elbette zaman içinde birçok değişikliğe uğrayan ve mobil hayatın vazgeçilmezlerinden olan "E-mail". Her gün binlerce, milyonlarca insan birbirleri ile mail yoluyla iletişim kurduğu, şirketlerin, kurumsal bazı iletişimlerin, anlaşmaların, duyuruların, tanıtımların, faturaların artık mail yoluyla işlediği günümüz dünyasında, mobil olarak da maili etkin şekilde kullanabilmek çok önemli bir özellik olarak her telefonda karşımıza çıkıyor. Windows Phone 7' in etkin E-Mail yönetim sistemleriyle kullanıcının maillerine ulaşması, düzenlemesi, değiştirmesi ve göndermesi o kadar kolay bir hale gelmiştir ki, birkaç dokunmayla yüzlerce kilometre yolu kat edebilir, birkaç dokunma ile mailinize Picasso'yu, Salvador Dalıyi, Van Gogh'u davet edebilir veya birkaç dokunma ile sevdiklerinize ulaşabilir, yüzlerinde minik bir gülümseme yaratabilirsiniz.

İşte tüm bu özellikleri en verimli şekilde kullanabilmeniz için geliştirilmiş E-mail altyapısı da yine Windows Phone 7'in yeniliklerine ve teknolojisine uygun şekilde tasarlanmış minik yardımcıları ile beraber geliyor. Bu bölümümüzde bu yardımcıları önemli olanları inceleyeceğiz. Bunların kimisi maillerinizi göndermenizi, kimisi yazmanızı sağlayan küçük yardımcıları. Ama belki de yaptıkları önemli işlemler, bir developer olarak sizlere, kullanıcılarınıza sevdiğinize, işlerine ve e-mail ile yapabildikleri her şeye bağlanmalarını sağlamanın hazzını veriyor. Bir uygulama içerisinde mail özelliklerini kullanıp, bu özellikler dahilinde uygulamanın özelliklerini, renklerini ortaya serabilmek sizin de en iyi yardımcısı olacaktır.

E-Mail Address Chooser Task



Bu görevi, uygulama içinde olsun dışında olsun, bir kişinin e-mail adresini seçmek istediğimizde kullanıyoruz. Uygulamalar içinde kimi zaman bir e-mail girdisine ya da kişinin e-mail adresine ihtiyacınız olabilir. İşte bu durumların tamamında bu küçük yardımcıyı kullanıp uygulamanızın bu değerleri almasını sağlayabilirsiniz.

Önce e-mail adres seçme uygulamamız için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz;



```
EmailAddressChooserTask esg = new EmailAddressChooserTask();
```

```
// Task'ımızı tanımladık
```

Bu değişkeni kullanım yöntemi ise şöyle ;

```
esg.Completed += new EventHandler<EmailResult>(esg_Completed);  
esg.Show();  
  
public void esg_Completed(object sender, EmailResult email)  
{  
    if (email.TaskResult == TaskResult.OK)  
    {  
        Debug.WriteLine(email.Email);  
    }  
  
    else  
    {  
    }  
}
```



E-Mail Compose Task

Windows Phone 7 geliştirirken kullanacağımız bir diğer yardımcı görev ise mail sistemlerinin de temelini oluşturan mail yazmayı kolaylaştıran ve bunu uygulamalarımıza içerisine gömen “E-Mail Compose Task”. Bu görevin de kullanımı yine oldukça basit ve bizlere uygulamalarımızın içerisinde e-mail yazım kolaylığı yaşatabilmesi de ayrı bir artı olarak uygulamanıza puan katıyor.

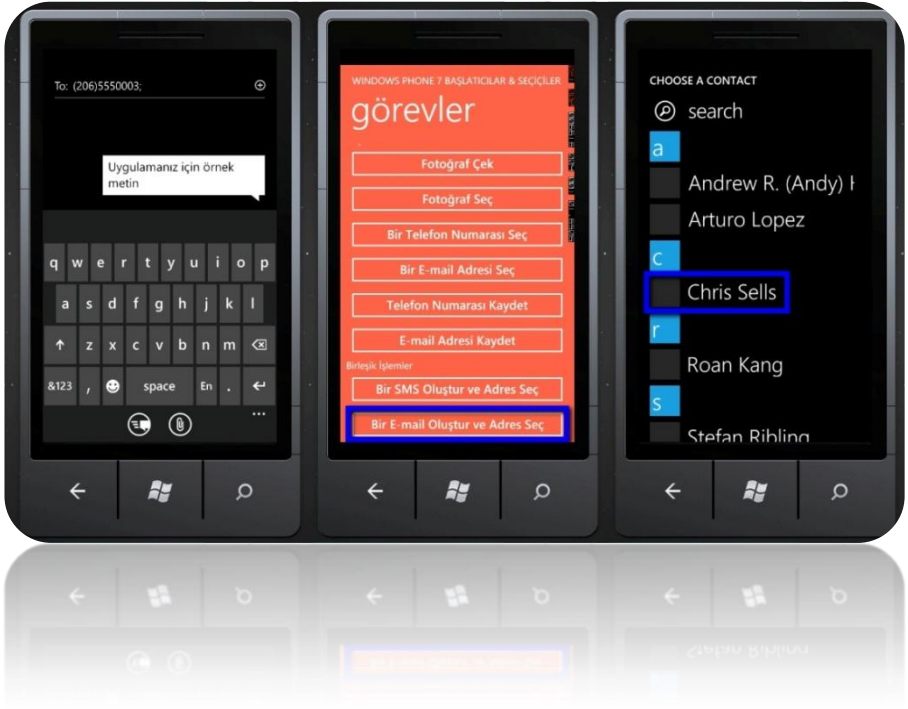
Önce e-mail adres seçme uygulamamız için uygulamalarımız içerisinde kullanabileceğimiz bir değişken tanımlıyoruz;

```
EmailComposeTask ey = new EmailComposeTask (); //
Task'imizi tanımladık
```



Bu değişkeni kullanım yöntemi ise şöyle;

```
task.To = “microsoft-turkiye@microsoft.com”;
task.Cc = “microsoft-turkiye-kopya@microsoft.com”;
task.Subject = “Merhaba Microsoft Türkiye...”;
task.Body = “Deneme E-mail’i”;
task.Show();
```



Yazardan...

Merhaba

Ben Merve Özel. Kadir Has Üniversitesi Bilgisayar Mühendisliği bölümünde okumaktayım ve artık mezun olmama çok az kaldı. Meslek hayatıma atılmadan önce herkes de bulunması gerektiğine inandığım, en önemli eğitim ve tecrübeye sahip olabileceğiniz programlardan biri olan Microsoft Student Partner programının son döneminde bulunmaktayım.

Bu program boyunca öğrendiklerimizin yanında MSP arkadaşlarımızla birlikte sizlere bu kitabı hazırlamak istedik. Bende, benim gibi bu alanda ilerlemek isteyen arkadaşlarıma faydalı olmak adına Windows Phone 7 uygulama klavuzumuzda Application Bar Menü ve İkonları kısmını yazmış bulunmaktayım. Umarım yazımda sizlere yardımcı olabilmişimdir.

Bizlere çalışmalarımızda yardımcı olan Microsoft Türkiye 'ye ve tecrübelerinden yararlanırken göstermiş olduğu ilgi, hoşgörü ve sabrından dolayı değerli hocamız Mustafa Kasap' a teşekkürlerimi sunarım.

9. ApplicationBar

Windows Mobile Phone 7' nin Application Bar (uygulama çubuğu) kısmına baktığımızda aslında Application Bar;

1)ApplicationBar İkonları (ApplicationBar Icons)

2)ApplicationBar menü (Application menü)

olmak üzere iki kısımdan oluşuyor.

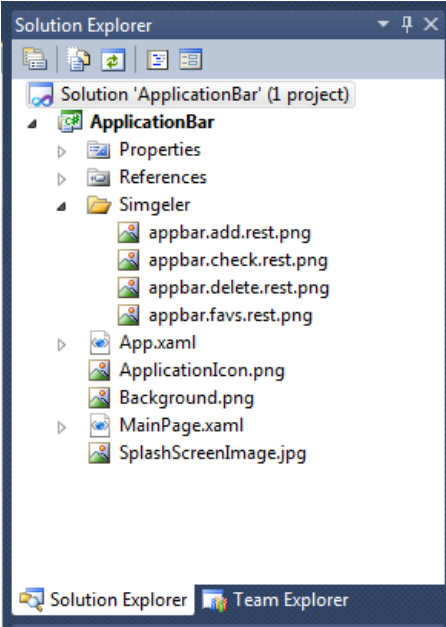
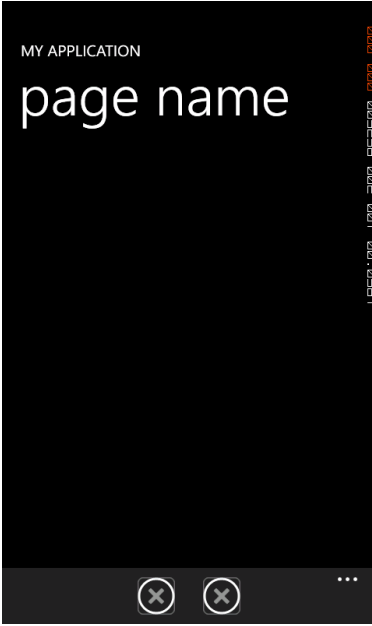
ApplicationBar düğmeleri bizim istediğimiz uygulama için kullandığımız ikonlardır/simgelerdir. Bu ikonlar ekranda sürekli olarak gözüken simgelerdir. Windows Phone 7 de ekrana en fazla 4 temel ikon eklenebilmektedir. Bu ikonlar ekranda bize kısa yollar sunan simgelerdir ve kullanıcılara karmaşıklık yaratmadan kullandıkları öğelere daha hızlı ulaşmalarını sağlamak amacıyla düşünülererek yapılmıştır.

ApplicationBar menü kısmı ise bu applicationbar ikonlarının yanında gördüğümüz üç noktayı (...) tıkladığımızda karşımıza çıkmaktadır. ApplicationBar menü tarafında da en fazla 5 menü ekleyebiliyoruz. Application menüde öğeleri ne kadar sıklıkta kullandığımızı bağlı olarak listeleyebiliyoruz. ApplicationBar menü eklentileri ApplicationBar İkonları gibi simgeleri kullanmamıza izin vermez ,bu eklentiler sadece metin olmalıdır.

Application Bar bölümü Windows Mobile Phone 7 uygulamamızda hazır olarak gelmektedir. Açtığımız projenin MainPage.xaml sayfasının alt kısmında Yorum halinde bulunan şekilde de gördüğü gibi ApplicationBar kod kısmı karşımıza çıkmaktadır. Öncelikle bu Yorum halindeki kodu aktif hale getirmemiz gerekmektedir.

```
<!--Sample code showing usage of ApplicationBar-->
<!--<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
    <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png" Text="Button 1"/>
    <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png" Text="Button 2"/>
    <shell:ApplicationBar.MenuItems>
      <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
      <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
    </shell:ApplicationBar.MenuItems>
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar-->
```

ApplicationBar kodunu aktifleştirdikten sonra uygulamamızı çalıştırdığımızda karşımıza şekil 1 deki gibi hazır olarak gelen ikon ve menüler çıkmaktadır.



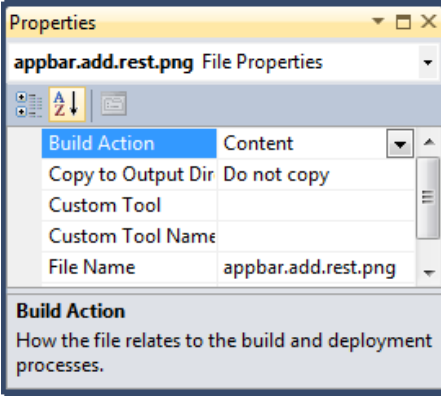
Ayrıca kendimiz de Application Bar İkonları için simgeler ekleyebiliriz. Fakat eklediğimiz simgeler 48x48 boyutunda olmalıdır. Ekleyeceğimiz simgelere Program Dosyaları/ Microsoft SDKs /Windows Phone / v7.1/ Icons dosyasından da ulaşabiliriz. Bu dosyada bulunan simgeler/ikonlar örnek olup, isterseniz 48x48 boyutunda farklı simgeler de ekleyebilirsiniz.

Şimdi uygulamamız için açtığımız projeye gelerek, eklemek istediğimiz ikonlar /simgeler için projenin altına yeni bir klasör ekliyoruz.(Bu klasörde ikonlarımız/simgelerimiz olacağı için ben Simgeler adını verdim sizde istediğiniz bir ad ile projenize yeni bir klasör ekleyebilirsiniz.)

Eklemiş olduğumuz klasöre seçtiğimiz

simgeleri/ikonları kopyalayıp yapıştırarak,

ikonları/simgeleri bir klasör altında toplamış oluyoruz.



Aşağıdaki şekilde de gördüğümüz gibi eklemek istediğimiz ikonları Simgeler adlı klasörün içinde topladık.

Simgelerimizi seçtikten sonra ApplicationBar ikonların/simgelerin görünebilmesi için her bir simgenin üzerine tıklayıp özellikler bölümünden şekildeki gibi Build Action kısmını Content olarak seçmemiz gerekmektedir.

Seçtiğimiz her bir ikon/simg için de kod kısmına aşağıdaki örnek gibi yazarak ekliyoruz. .

```

<!--Sample code showing usage of ApplicationBar-->
<phone:PhoneApplicationPage.ApplicationBar>
    <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
        <shell:ApplicationBarIconButton IconUri="/Simgeler/appbar.add.rest.png" Text="ADD"/>
        <shell:ApplicationBarIconButton IconUri="/Simgeler/appbar.check.rest.png" Text="CHECK"/>
        <shell:ApplicationBarIconButton IconUri="/Simgeler/appbar.delete.rest.png" Text="DELETE"/>
        <shell:ApplicationBarIconButton IconUri="/Simgeler/appbar.favs.rest.png" Text="FAVS."/>
        <shell:ApplicationBar.MenuItems>
            <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
            <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
        </shell:ApplicationBar.MenuItems>
    </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>

```

Her biri için Build Action 'ı Content seçtikten sonra artık kod kısmında da seçtiğimiz simgeleri tanımladıktan sonra, çalıştırdığımızda karşımıza şekildeki gibi bir ekran çıkmaktadır.

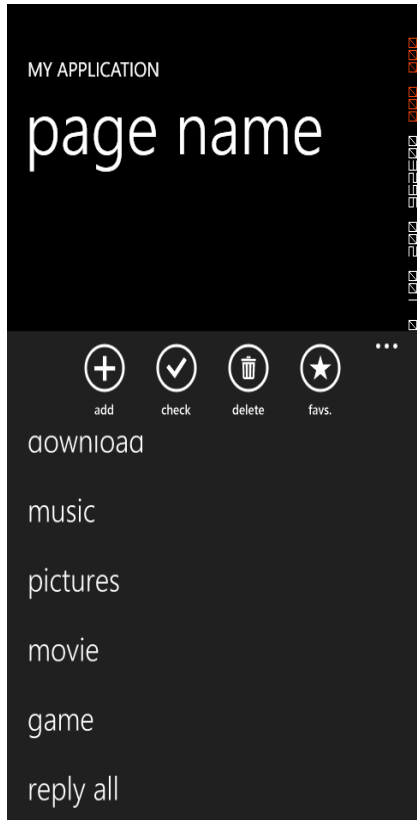
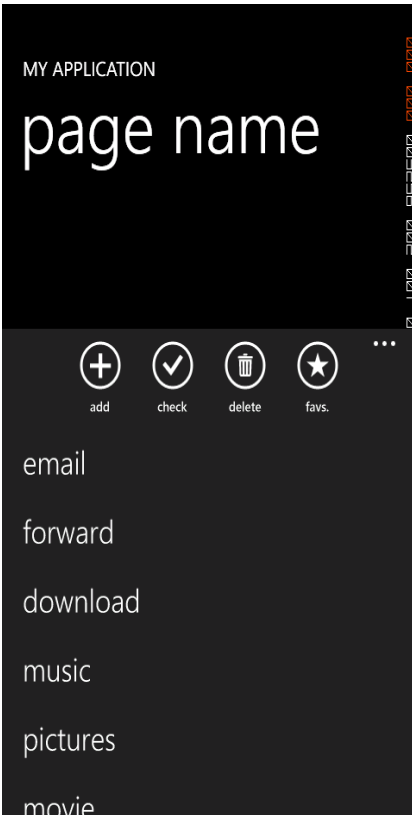


Şekildeki kodu incelersek ApplicationBar Menü bölümüne 5 menüden fazla ekleme yaptığımız için karşımıza aşağıdaki gibi bir ekran çıkmaktadır.

```

<!--Sample code showing usage of ApplicationBar-->
<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
    <shell:ApplicationBarIconButton IconUri="/Singeler/appbar.add.rest.png" Text="ADD"/>
    <shell:ApplicationBarIconButton IconUri="/Singeler/appbar.check.rest.png" Text="CHECK"/>
    <shell:ApplicationBarIconButton IconUri="/Singeler/appbar.delete.rest.png" Text="DELETE"/>
    <shell:ApplicationBarIconButton IconUri="/Singeler/appbar.favs.rest.png" Text="FAVS."/>
    <shell:ApplicationBar.MenuItems>
      <shell:ApplicationBarMenuItem Text="EMAIL"/>
      <shell:ApplicationBarMenuItem Text="FORWARD"/>
      <shell:ApplicationBarMenuItem Text="DOWNLOAD"/>
      <shell:ApplicationBarMenuItem Text="MUSIC"/>
      <shell:ApplicationBarMenuItem Text="PICTURES"/>
      <shell:ApplicationBarMenuItem Text="MOVIE"/>
      <shell:ApplicationBarMenuItem Text="GAME"/>
      <shell:ApplicationBarMenuItem Text="REPLY ALL"/>
    </shell:ApplicationBar.MenuItems>
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>

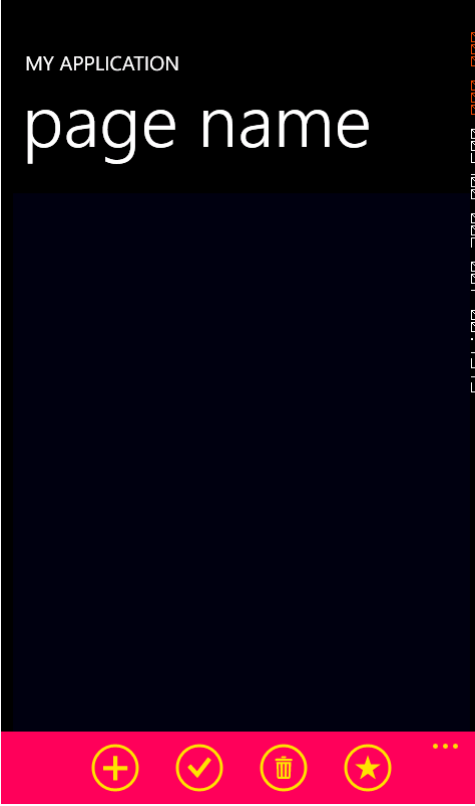
```



Menüleri arttırabiliriz fakat ekranda gözükten en fazla 5 menü olur. Diğerlerini ise ilerletme çubuğu ile ekrana getirmiş oluyoruz.

Ayrıca ekranda gözüken ikon ve menüleri Microsoft Expression Blend 4 ile daha renkli ve göz alıcı hale getirebiliriz.

Örnek :



Yandaki örnek gibi en basit haliyle renklendirerek kendi isteğinize göre biçimlendirip daha güzel ve çekici hale getirebilirsiniz. Microsoft Expression Blend 4 ile simgelerinizi/ikonlarınızı tasarlayabilirsiniz .



Veri bağlama işlemi, temelde elimizdeki verinin bir kontrole bağlanıp kullanıcıya gösterilmesi işlemidir. Bu bölümümüzde verinin kontrole bağlanması, kontrolden gelen verinin başka bir kontrole bağlanması gibi senaryoları nasıl uygulayacağımızı inceleyeceğiz.

Yazardan...

10. Veri Bağlama İşlemleri

Veri bağlama işlemi bizlere bir nesnenin özelliğini başka bir nesnenin özelliğine bağlamamızı sağlıyor. Bahsettiğimiz bu özellik resim, yazı veya nesne olabilir. Geliştirici tarafına baktığımızda Windows Phone 7 üzerinde veri bağlama işleminin Silverlight ve WPF tabanlı uygulamalardan pek farklı olmadığını görüyoruz. UI elemanına bir CLR nesnesi, nesne topluluğu veya başka bir UI elemanı bağlanabilir. Verimiz bağlandıktan sonra kaynağımız hedef üzerinde değişiklik yapabileceği gibi hedefimiz de kaynak üzerinde değişiklik yapabilmektedir. Bu yöntemleri ve kullanıcının veriyle etkileşime geçebilmesi için ihtiyacımız olan bilgileri bu bölümde işleyeceğiz.

Bir UI elemanın başka bir UI elemanına bağlanmasını bir örnekle inceleyelim.

```
<StackPanel>
    <TextBox x:Name="textbox1" Width="450" />
    <TextBlock x:Name="textblock1" Width="400" Foreground="Red" Font
Size="28"
        Text="{Binding ElementName=textbox1, Path=Text}" />
</StackPanel>
```

MainPage.xaml tarafında tasarımımızı bu şekilde düzenleyelim. TextBlock elemanımızı incelediğimizde Text özelliğine textbox1 elemanının Text özelliğini bağladığımızı görüyoruz. Uygulamamızı çalıştırdığımızda TextBox içerisine yazdığımız her karakter aynı anda TextBlock üzerinde de gösterilecektir.

Farklı bir söz dizimlerinin bir arada kullanıldığı farklı bir örneğimizle devam edelim;

```
<Canvas>
    <Slider Height="84" Name="slider1" Width="200"
Value="{Binding ElementName=slider2, Path=Value,
Mode=TwoWay}"
        Canvas.Left="6" Canvas.Top="6" />

    <Slider Height="200" Name="slider2" Width="84"
Orientation="Vertical" Margin="-100,0,0,0"
        Canvas.Left="109" Canvas.Top="96">
        <Slider.Value>
            <Binding ElementName="slider1" Path="Value" />
        </Slider.Value>
    </Slider>

    <TextBlock Height="30" x:Name="tblock1" Text="Değer : "
        Canvas.Left="107" Canvas.Top="96" />
    <TextBlock Height="30" x:Name="tblockDeğer"
        Text="{Binding Value, ElementName=slider1}"
        Canvas.Left="181" Canvas.Top="96" />
</Canvas>
```

Bu örneğimizde slider1'e slider2 değerini, slider2'ye de slider1'in değerini bağlayıp slider1 üzerindeki değeri tblockDeger üzerinde gösteriyoruz. Veri bağlama yöntemindeki farklı söz diziminin farkına varmışsınızdır. slider1'e slider2'yi bağlarken Value özelliğine direk olarak bağliyoruz fakat slider2'ye slider1'i bağlarken Slider.Value içerisinde bulunan Binding'in özelliklerini değiştirerek ilerliyoruz. Bir önceki örneğimizde yaptığımız gibi ElementName olarak bağlamak istediğimiz nesneyi, Path içerisine de hangi özelliğini bağlamak istediğimizi yazıyoruz. slider1'in Mode kısmına dikkatinizi çekmek istiyorum. Özelliğimizi TwoWay olarak tanımladık ve bu bize iki nesnenin de karşılıklı olarak birbirini etkileyebileceği anlamını vermektedir. Örneğimizi çalıştırdığımızda slider1 üzerinde yaptığımız değişikliğin slider2'yi, slider2 üzerinde yaptığımız bir değişikliğin de slider1'i etkilediğini göreceksiniz.

Temel veri bağlama işlemlerini gördüğümüze göre bir class içerisindeki verinin bir nesneye nasıl bağlanacağını incelemeye başlayabiliriz. Yapacağımız örnekte listbox içerisine Urun sınıfımızı bağlayıp dönen değerleri ekranda göstereceğiz. Öncelikle yapmamız gereken projemize sağ tıklayıp Add->Class penceresinden bir class eklemek. Ben bu örneğimizde Urun isimli bir sınıf kullanacağım. Sınıfımızın elemanlarını Listbox içerisine doldururken sizlere resource'ların kullanımını da örneklendirmiş olacağım.

İlk olarak resource kullanmadan verilerimizi listbox'ın DataTemplate'inde nasıl göstereceğimizi öğrenelim. Urun class'ımızı yazarak başlayalım.

```
public class Urun
{
    public int ID { get; set; }
    public string Adi { get; set; }
    public double Fiyat { get; set; }

    public static List<Urun> UrunleriGetir()
    {
        return new List<Urun>() {
            new Urun() { ID=1, Adi="LG Optimus 7 E900", Fiyat=972.54
        },
            new Urun() { ID=2, Adi="HTC HD7", Fiyat=752.91 },
            new Urun() { ID=3, Adi="Nokia Lumia 800", Fiyat=997.35 }
        };
    }
}
```

Urun class'ımızda ID, Adi ve Fiyat şeklinde özelliklerimiz bulunmaktadır. UrunleriGetir metodumuza baktığımızda List<T> türünde bir generic list döndürmektedir. Örneğimizde dönen liste içerisinde 3 tane ürünümüz bulunmaktadır. Bu senaryoyu veritabanından istediğimiz verileri çekip gerekli business işlemleri yapıp döndürdüğümüz bir liste şeklinde de düşünebiliriz.

```
<ListBox HorizontalAlignment="Left" x:Name="listBox1"
VerticalAlignment="Top" Width="456" Height="607">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel>
                <TextBlock Text="{Binding ID}"></TextBlock>
                <TextBlock Text="{Binding Adi}"></TextBlock>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

```

        <TextBlock Text="{Binding Fiyat}"></TextBlock>
    </StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>

```

Bu kod satırıyla ListBox'ımızın her bir elemanının ne şekilde gözükeceğini tanımladık. Bütün elemanlar bir StackPanel içerisinde 3 tane TextBlock olacak şekilde yerleşecektir. Her şey tamam olduğuna göre artık ListBox'ımızın ItemsSource özelliğine sınıfımızdan dönen listemizi atayabiliriz. Bunu da MainPage'in Loaded olayı içerisindeyken aşağıdaki kodu yazarak sağlayabiliriz.

```
listBox1.ItemsSource = Urun.UrunleriGetir();
```

Şimdi de bu DataTemplate'i resource olarak tanımlayıp listbox'ımızda bu şablonumuzu nasıl göstereceğimizi inceleyelim. İhtiyacımız olan şey sadece DataTemplate tag'iyle başlayıp biten alanları kopyalamak olacak. Kopyaladığımız kodları aşağıdaki formatta yazdığımızda artık DataTemplate'imiz resource olarak tanımlanmış olacaktır.

```

<phone:PhoneApplicationPage.Resources>
    <DataTemplate x:Name="UrunTemplate">
        <StackPanel>
            <TextBlock Text="{Binding ID}"></TextBlock>
            <TextBlock Text="{Binding Adi}"></TextBlock>
            <TextBlock Text="{Binding Fiyat}"></TextBlock>
        </StackPanel>
    </DataTemplate>
</phone:PhoneApplicationPage.Resources>

```

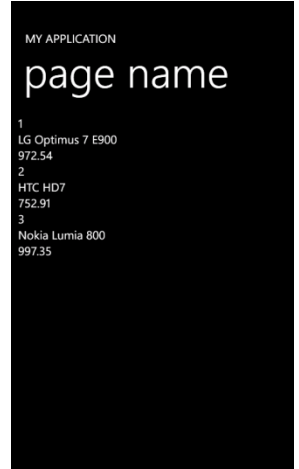
Gördüğümüz gibi yaptığımız tek şey <phone:PhoneApplicationPage.Resources> taglerinin arasına DataTemplate kodumuzu aynen aktarmak ve DataTemplate'imize bir isim vermek. Şimdi ListBox'ımızın ItemTemplate özelliğine UrunTemplate'i atayarak resource içerisindeki şablonu ListBox'ımıza uygulayabiliriz.

```

<ListBox Height="607"
HorizontalAlignment="Left" Name="listBox1"
VerticalAlignment="Top" Width="456"
ItemTemplate="{StaticResource UrunTemplate}"
/>

```

Buradaki önemli kısım ItemTemplate içerisine yazdığımız alanda parantezler içerisinde StaticResource kelimesini kullanmamız. Bu komut bize gelecek template değerinin resource içerisinde olduğunu ve UrunTemplate adında olduğunu bildiriyor. Projemizi çalıştırdığımızda alacağımız ekran görüntüsü yandaki gibidir.





Çalışmamda Windows Phone 7 üzerinde animasyon oluşturulması, hangi tip animasyonların mevcut olduğu, animasyonların hangi şekilde başlatılabileceği, oluşturulan animasyonların özelliklerinin nasıl ayarlandığı gibi konular üzerinde durdum. Bildiğiniz üzere geliştirilen mobil uygulama sayısı günden güne artıyor. Bu uygulamalarda yetersiz olan kısım ise animasyon eksikliği. Animasyonların uygulamada ki önemi tartışılmaz bir gerçek. Kullanılan animasyonlar, uygulamaların çekiciliğini arttırmakta ve beğenilmesini sağlamaktadır. Bu nedenle uygulama geliştiricilerin bu konu üzerinde durmasını öneririm. Çalışmalarımız sırasında uygulama geliştirirken bizlere yardımcı olan Ekin Özçiçekçiler hocama ve bize bu imkanları sunan Microsoft Türkiye Akademik Programlar Yöneticisi Mustafa Kasap'a teşekkürü bir borç bilirim. Yazımın Windows Phone 7 üzerinde uygulama geliştirmek isteyen herkese faydalı olması dileğiyle.

11. Animasyon

Windows Phone 7 ile animasyon yaparken Microsoft tarafından geliştirilen WPF ve Silverlight teknolojilerinden yararlanır. WPF (Windows Presentation Foundation) .NET Framework 3.0'ın grafik altyapısını oluşturan XAML (Extensible Application Markup Language) tabanlı bir sistemdir. Microsoft'un Windows Vista ve Windows 7 işletim sistemlerinde yüklenmiş olarak gelmektedir. Ayrıca, Windows XP SP2'de çalışabilmektedir. Microsoft bu teknolojiyi kullanıcılara yüksek derecede görsel arayüz sağlamak için oluşturmuştur. Bu nedenle XAML günümüzün ve geleceğin programlama teknolojisidir.

Silverlight animasyon, vektör, 3D grafik ve görüntü oynatma imkanları sağlayan internet uygulamalarının geliştirilmesi için kullanılan bir platformdur. Silverlight WTF görsel programlama tekniği ile grafik, animasyon ve interaktif uygulamaların tek eklenti üzerinden yürütülmesini sağlar. Bu nedenle, Windows Phone 7 ile animasyon içeren uygulamalar geliştirirken, Silverlight'in built-in animasyon desteğinden yararlanır.

Silverlight "System.Windows.Media.Animation" isim uzayı içerisinde 50'den fazla sınıf, yapı ve liste barındırmaktadır. Silverlight'in animasyon kütüphanesi alternatiflerine göre daha basittir, çünkü animasyonlar XAML kullanılarak oluşturulabilir. Bu animasyonların çoğu "CompositionTarget.Rendering" özelliğini kullanır. Bunun nedeni, animasyonların telefon üzerinde bulunan GPU (Graphics Processing Unit)'dan faydalanyor olmasıdır.

Frame-Based vs Time-Based (Frame Tabanlı ve Zaman Tabanlı)

Örneğin bir yazının dönmesini sağlayan animasyon yapmak istendiğinde "CompositionTarget.Rendering" özelliği kullanılır. Bu animasyon videonun sürekli yenilenmesiyle yada zamanlama kullanılarak oluşturulur. Video görüntüsünün her yenilenmesine frame denir ve bu yöntemle oluşturulan animasyonlara frame-based animasyon adı verilir. Aşağıdaki kod parçacığında frame-based ve time-based karşılaştırmasını görebilirsiniz.

```
public partial class MainPage : PhoneApplicationPage
{
    DateTime startTime;

    public MainPage()
    {
        InitializeComponent();

        startTime = DateTime.Now;
        CompositionTarget.Rendering += OnCompositionTargetRendering;
    }

    void OnCompositionTargetRendering(object sender, EventArgs args)
    {
        // Frame-based
    }
}
```

```

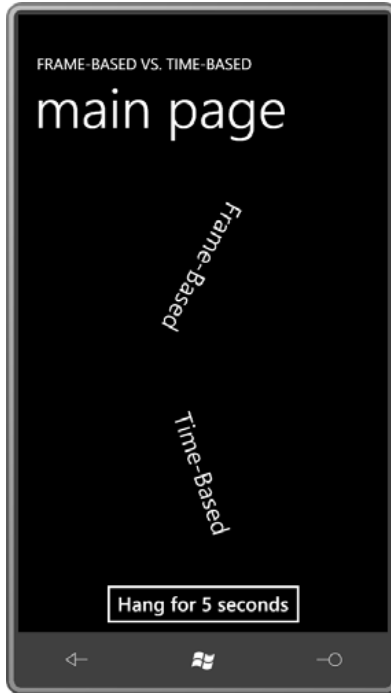
rotate1.Angle = (rotate1.Angle + 0.2) % 360;

// Time-based
TimeSpan elapsedTime = DateTime.Now - startTime;
rotate2.Angle = (elapsedTime.TotalMinutes * 360) % 360;
}

void OnButtonClick(object sender, RoutedEventArgs args)
{
    Thread.Sleep(5000);
}
}

```

Bu 2 kod bloğu neredeyse aynı çalışmaktadır. Ama frame-based’de time-based’e göre biraz gecikme olmaktadır. Bunun nedeni, frame-based’in kullanılan telefonun donanım özelliğine bağlı olmasıdır.



Click and Span (Tıklama ve Dönme)

Windows Phone 7 uygulamalarında kullanıcıların ilgisini çekmek istersek butonlara animasyon ekleyebiliriz. Örneğin aşağıdaki örnekte butona tıklandığında kendi etrafında

dönmektedir. Animasyonun başlaması için butonun click event'ine aşağıdaki kod parçasını yazıyoruz.

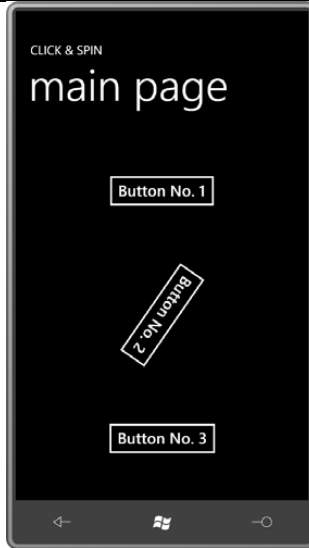
```
void OnButtonClick(object sender, RoutedEventArgs args)
{
    Button btn = sender as Button;
    RotateTransform rotateTransform = btn.RenderTransform as RotateTransform;

    // Animasyonun yaratılması ve tanımlanması
    DoubleAnimation animasyon = new DoubleAnimation();

    animasyon.From = 0;
    animasyon.To = 360;
    animasyon.Duration = new Duration(TimeSpan.FromSeconds(0.5));

    // Özelliklerin ayarlanması
    Storyboard.SetTarget(animasyon, rotateTransform);
    Storyboard.SetTargetProperty
    (animasyon, new PropertyPath(RotateTransform.AngleProperty));

    // Storyboard'ın yaratılması, animasyonun eklenmesi ve çalıştırılması
    Storyboard storyboard = new Storyboard();
    storyboard.Children.Add(animasyon);
    storyboard.Begin();
}
```



Animasyon Özellikleri

- Animasyonların target özelliğini 2 şekilde ayarlayabiliriz. Bunlar özellik ismi veya string şeklindedir.

```
Storyboard.SetTargetProperty(animasyon, new  
PropertyPath(RotateTransform.AngleProperty));
```

```
Storyboard.SetTargetProperty(animasyon, new PropertyPath("Angle"));
```

- Animasyonların devam süresi 2 şekilde ayarlanabilir.

```
animasyon.Duration = new Duration(TimeSpan.FromSeconds(0.5));
```

```
storyboard.Duration = new Duration(TimeSpan.FromSeconds(0.25));
```

- Animasyonların başlama zamanı şu şekillerde ayarlanabilir.

```
animasyon.BeginTime = TimeSpan.FromSeconds(1);
```

```
animasyon.AutoReverse = true;
```

- Animasyonun kaç kez tekrar edeceği şu şekilde ayarlanabilir.

```
animasyon.RepeatBehavior = new RepeatBehavior(2);
```

- Ayrıca animasyonun kaç kez tekrar edeceğini saniye cinsinde belirleyebiliriz.

```
animasyon.RepeatBehavior = new RepeatBehavior(TimeSpan.FromSeconds(0.75));
```

- Animasyonun kaç dereceden başlayıp kaç derece döneceğini şu 2 özelliği ayarlayarak belirleyebiliriz.

```
animasyon.From = 0;
```

```
animasyon.To = 360;
```

- Animasyonlarda herbir tıklama yapıldığında butonun belirli bir derecede dönmesi istenirse by özelliği kullanılır.

```
animasyon.By = 90;
```


XAML-Tabanlı Animasyonlar

Click and Spin örneğini XAML storyboard ile şu şekilde yapabiliriz.

```
// MainPage.xaml

<phone:PhoneApplicationPage.Resources>
    <Storyboard x:Name="storyboard1">
        <DoubleAnimation Storyboard.TargetName="rotate1
Storyboard.TargetProperty="Angle" From="0" To="360" Duration="0:0:0.5" />
    </Storyboard>
</phone:PhoneApplicationPage.Resources>

// MainPage.xaml.cs

void OnButtonClick(object sender, RoutedEventArgs args)
{
    storyboard1.Begin();
}
```

Key Frame Animasyonlar

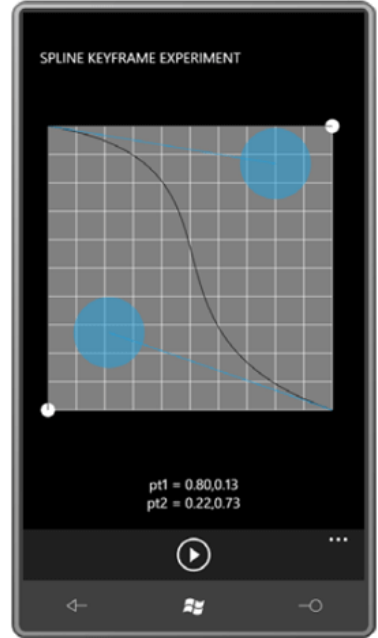
Anahtar framer kullanıcılara daha akıcı ve düzgün animasyonlar sağlamak için kullanılırlar. Bunun için jiggleStoryboard kullanılır.

```
// MainPage.xaml

<phone:PhoneApplicationPage.Resources>
    <Storyboard x:Name="jiggleStoryboard">
    </Storyboard>
</phone:PhoneApplicationPage.Resources>

// MainPage.xaml.cs

void OnButtonClick(object sender, RoutedEventArgs
args)
{
    jiggleStoryboard.Begin();
}
```



Perspektif Dönüşümleri

Bu dönüşümleri yapabilmek için X,Y ve Z yönünde 3 yön tanımlanması gerekir. Aşağıdaki örnekte herbir yön için bir buton kullanılmıştır.

// MainPage.xaml

```
<phone:PhoneApplicationPage.Resources>

    <Storyboard x:Name="rotateX">
        <DoubleAnimation Storyboard.TargetName="planeProjection"
Storyboard.TargetProperty="RotationX" From="0" To="360" Duration="0:0:5" />
    </Storyboard>

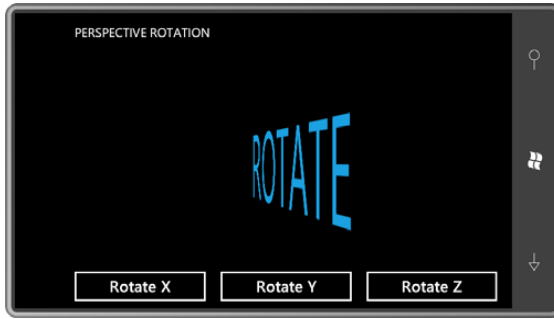
    <Storyboard x:Name="rotateY">
        <DoubleAnimation Storyboard.TargetName="planeProjection"
Storyboard.TargetProperty="RotationY" From="0" To="360" Duration="0:0:5" />
    </Storyboard>

    <Storyboard x:Name="rotateZ">
        <DoubleAnimation Storyboard.TargetName="planeProjection"
Storyboard.TargetProperty="RotationZ" From="0" To="360" Duration="0:0:5" />
    </Storyboard>

</phone:PhoneApplicationPage.Resources>
```

// MainPage.xaml.cs

```
void RotateXClick(object sender, RoutedEventArgs args)
{
    rotateX.Begin();
}
void RotateYClick(object sender, RoutedEventArgs args)
{
    rotateY.Begin();
}
void RotateZClick(object sender, RoutedEventArgs args)
{
    rotateZ.Begin();
}
```



Animasyonlar ve Özellik Önceliği

Animasyonlarda özellik önceliği şöyle sıralanabilir:

1. Local (yerel) ayarlar
2. Stil ayarları
3. Tema stili
4. Özellik kalıtımı
5. Varsayılan değerler

Windows Phone 7 ile kaydırıcı (slider) örneği şöyle yapılabilir.

```
// MainPage.xaml
```

```
<phone:PhoneApplicationPage.Resources>
  <Storyboard x:Name="storyboard">
    <DoubleAnimation Storyboard.TargetName="slider"
Storyboard.TargetProperty="Value" To="50" Duration="0:0:5" />
  </Storyboard>
</phone:PhoneApplicationPage.Resources>
```

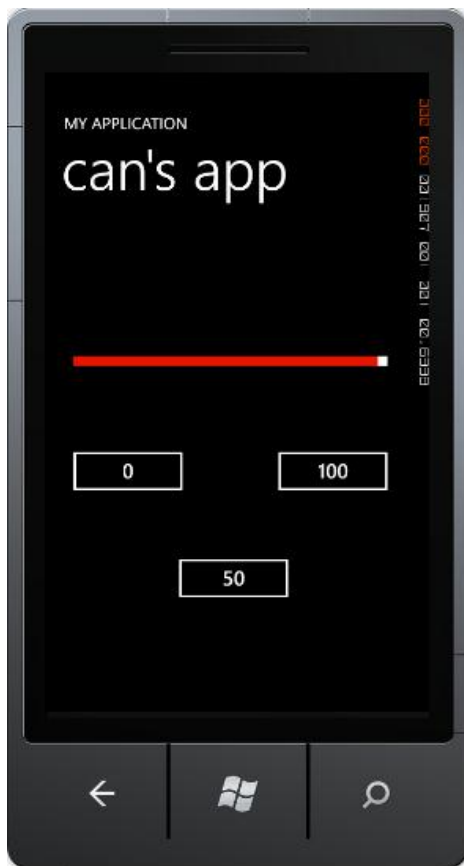
```
// MainPage.xaml.cs
```

```
public partial class MainPage : PhoneApplicationPage
{
    public MainPage()
    {
        InitializeComponent();
    }

    void ZeroClick(object sender, RoutedEventArgs args)
    {
        slider.Value = 0;
    }
}
```

```
void OneHundredClick(object sender, RoutedEventArgs args)
{
    slider.Value = 100;
}

void FiftyClick(object sender, RoutedEventArgs args)
{
    storyboard.Begin();
}
}
```





Bu bölümde Windows Phone 7 de sayfalar arasındaki panoramik geçişleri, özelliklerini, kullanım şekillerini anlatacağız. Sayfalar arasında geçişlerin kolaylıklarını anlatmak, uygulamalarda oluşabilecek hataları önlemeniz üzerine yardımcı olabileceğini umuyoruz. Panoramik kolaylıklar sağlamak dileklerimizle...

Yazardan...

12. Panoramik Sayfalar

Panoramik Sayfalar Hakkında

Windows Phone' un kullanıcılarına sunmuş olduğu harika bir özellik olan panoramik sayfalar mobil dünyaya farklı bir yüz kazandırmış olup kullanıcılarda da oldukça iyi bir izlenim bırakmıştır. Ayrıca standart mobil uygulamalarda alışlagelmiş yapıdan kurtulmamızı sağlayan ve uygulamaya birçok konuda özgürlük kazandıran bir özellik haline gelmiştir. Panoramik sayfalardaki esnekliğin bize kazandırdığı en büyük şey ise; sizin de tahmin edebileceğiniz gibi kullanıcıya istediği içeriği tam anlamıyla rahat bir şekilde bütün olarak sunabilecek olmamızdır. Bu bölümde panoramik bir uygulamanın nasıl oluşturabildiğini öğrenirken içerisine kontroller ekleyerek sayfa kullanımını da anlamış olacağız. Hatırlatmak gerekir ki bir uygulamanın sadece panoramik olma veya olmama gibi bir durumu yoktur.

Panoramik Sayfaların Yapısı

Panoramik resimleri hepimiz duymuşuzdur. Bu resimlerin özelliği birçok açıyı tek bir resimde toplanıp gösterilmesidir. Bu resim içerisinde bölüm bölüm dolaştığımızda baktığımız resim tek olsa da görmüş olduğunuz açı farklı olacaktır. İşte Windows Phone ile birlikte gelen panoramik sayfalar da aynen bu mantık ile çalışmaktadır. Farklı tipteki bilgileri bütün olarak tek bir dikkörtgende topladığımızı düşünelim. Bu dikkörtgenin tamamının ekrana sığdırılmaya çalışılması çok anlamsız olacaktır. Dolayısıyla biz de bu resmin tamamını yerleştirip içerisinde dolaşma işini kullanıcıya bırakıyoruz. Dolaşırken gerçekleşen animasyonlar, sayfaların kayması vs. gibi özellikler (gesture) işletim sistemi ile birlikte dahili olarak bize sunulmaktadır. Yani bu işlemler için bizim ek olarak herhangi bir kod yazmamıza veya olay kurgulamamıza gerek kalmamaktadır. Sunulan işlemler;

- Horizontal Pan: Dikkörtgeni sağa veya sola çekme işlemi.
- Horizontal Flick: Ani ve hızlı bir şekilde dikkörtgeni sağa veya sola çekme işlemi işlemi.
- Navigating Hosted Controls: Her bloğun içerisinde bağımsız olması durumu.

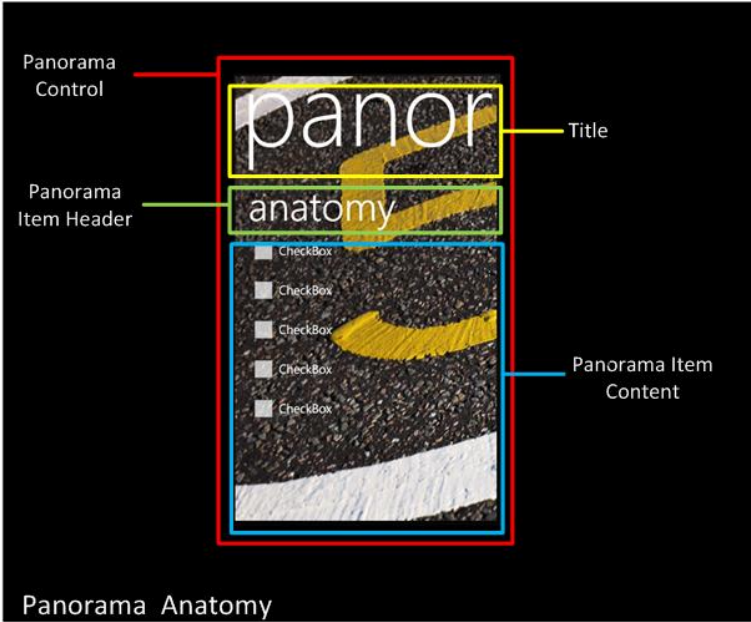


Resimden de görülebileceği gibi sağ taraftaki blok için bir ön izleme bulunmaktadır. 24 piksel genişliğinde olan bu ön izleme kullanıcıya diğer blokta başka verilerin de bulunduğunu anımsatmak için düşünülmüştür.

Ayrıca resim kullanıcılara daha yoğun bir veri akışının nasıl gösterilebileceği konusunda da fikir vermektedir.

Panoramik Sayfalardaki Blok Yapısı

Panoramik yapının en temel bileşeni **PanoramaItem**' dir ve bahsi geçen blokların kendisidir. Panoramik resmin genişliğini yeni PanoramaItem' lar ekleyerek genişletebiliriz bu. Bu kontrolleri ekleyerek sayfamıza yeni özellikler, yeni açılar eklemiş olacağız. Örneğin birinde linkleri gösterirken, diğerinde harita ve başka birinde resimler gibi.



Resimde de gördüğünüz gibi bir **Panorama** kontrolü içerisinde bulunan bir **PanoramaItem** bulunmaktadır. PanoramaItem içerisinde ise tüm resimde ortak olarak hareket eden ve işletim sistemi tarafından otomatik bir şekilde ayarlanan **Title** bulunmaktadır. Bu kısım tüm PanoramaItem' lar için ortaktır.

Panorama

Panorama en temel bileşendir ve her şey bu bileşen içerisinde bulunur. Üç katmandan oluşmaktadır, bunlar;

1. **Background (PanningBackgroundLayer)**

Arkaplan görüntüsünün gösterimini ve animasyonunu sağlar. Panorama bileşenin üzerine tıklayarak Properties paneli içerisinden (veya doğrudan XAML içerisinden) background' a resim verebilirsiniz. Bunun yanında tek renk veya gradient renkler de kullanılabilir. Kontrolün arka planı kesinlikle NULL bir değer olmalıdır, mutlaka set etmelisiniz. Varsayılan olarak Transparent olarak gelmektedir. Ekleme isteyeceğiniz resmin **Build Action** özelliğinin **Resource** olarak işaretlenmesi gerekmektedir.

2. Title (PanningTitleLayer)

Başlığın gösterimini ve animasyonunu sağlar. Panorama bileşenin Title özelliğinden değiştirebilirsiniz. Title' ın dikey yüksekliği değişmeyeceğinden dolayı Content içerisinde daha fazla alan isteyebilirsiniz, bu durumda Title' ı boş bırakarak kendinize alan açabilirsiniz; ancak Windows Phone uygulamalarının genel prensibi olarak Title' ın olması uygun görülmektedir.

3. Items (PanningLayer)

PanoramaItem bileşenlerinin gösterimini sağlar. Kullanıcı etkileşimi sadece bu alanda olur. Kullanıcı bloklar arasında geçişi de yine bu bölgeyi kaydırarak yapar.

Panoramik Uygulama Oluşturulması

Panoramik sayfaları kullanmak için birçok yol mevcuttur.

1. Eğer mevcut bir projeniz varsa ve bu proje içerisinde panoramik sayfa kullanmak isterseniz projeye sayfa olarak ekleyebilirsiniz.
 - Bunun için **Windows Phone projemize** sağ tıklayıp **Add** sekmesine gelip ardından **New Item** ' a tıklıyoruz. Ardından karşımıza gelen pencereden **Windows Phone Panorama Page** ' i seçip bir isim veriyoruz ve **Add** diyerek projemize ekliyoruz. Böylelikle boş bir panoramik sayfayı projemize ekleyebiliyoruz.
2. Sıfırdan bir proje oluşturacaksanız doğrudan Panoramik Sayfa şablonunu kullanarak projeyi başlatabilirsiniz.
 - Bunun için projeyi yaratırken **Windows Phone Panorama Application** şablonun seçerek projemizi yaratıyoruz. Görebileceğiniz gibi şablonu kullanarak içerisinde bize fikir veren birkaç kontrol ile birlikte yeni bir proje açmış olduk.

Eğer panoramik sayfalar hakkında hiçbir fikriniz yoksa irdelemek için bu şablonu kullanarak bir proje açıp test edebilirsiniz.



Web bağlantıları telefon uygulamalarında en çok kullanılan alanlardan biridir. İlk Windows Phone7 uygulamam olan ve Marketplace' te bulabileceğiniz CinePhone7' de web bağlantılarını kullandım. Bu basit uygulama yeni vizyona giren filmleri görüntülüyor ve RSS ile filmler hakkında verilen bilgileri çekerek kullanıcıya sunuyor. RSS ve benzeri yöntemlerle bu tip faydalı olabilecek çok çeşitli uygulamalar yazılacağını fark ettim. Bu yüzden mobil uygulamalar konusunda web bağlantılarının önemli bir yer taşıdığını düşünüyorum. Web bağlantıları bölümünde ilk olarak ağ bilgileri, web servisleri ve güvenlik hakkında genel bilgiler bulunmaktadır. Web ve data servisleri için hangi sınıfların kullanılması gerektiği ve bu sınıfların tanımları anlatılmıştır. Genel bilgileri içeren bu girişin ardından RSS uygulaması geliştirmek için gerekli olan sınıflar ve gerekli kütüphaneleri nasıl ekleyeceğinizi anlatılmıştır. Örnek RSS kodu verilmiş ve kod parçalarının ne amaçla kullanıldığı yorum olarak ifade edilmiştir. RSS uygulaması geliştirmenin temel mantığı öğrenildiğinde RSS desteği veren bütün sitelerden kolayca veri çekebilir, haberler veya blog yazıları gibi verileri kullanarak gazete, dergi gibi uygulamalar yapabilirsiniz. Veya film, müzik bilgilerini çekerek eğlence kategorisinde değişik uygulamalar oluşturabilirsiniz. Çok çeşitli uygulamalar oluşturmanıza yardımcı olacağından dolayı RSS yöntemini öğrenmenizi tavsiye ederim.

13. Web Bağlantıları

Windows Phone Platformu web bağlantılı uygulama geliştirmek için birçok özellik sağlar. Ağ bağlantıları ve web servisleri için oluşturulmuş birçok sınıf sayesinde çeşitli web uygulamaları geliştirilmesine olanak tanır.

a. Genel Bilgiler

Ağ bilgileri, soketler, web ve data servisleri, güvenlik konuları ve kısıtlamalar hakkında genel bilgiler, konunun detaylarına inmeden önce faydalı olacaktır. [Microsoft.Phone.Net.NetworkInformation](#) isim alanının sınıfları aletin ağ bağlantılarının kalitesi ve ulaşılabilirliği hakkında bilgi verir. Anlık mesajlaşma gibi çift taraflı iletişimin gerçekleştiği uygulamalar için soket uygulamaları bulunmaktadır. Bu uygulamalarda System.Net.Sockets API kullanılır.

Web servisleri sayesinde internetteki bir çok bilgiye erişim sağlanır. Data servisi ise HTTP temelli web servsidir. Open Data Protocol (OData) ve Uniform Resource Identifiers (URI) ile verinin kaynak olarak kullanılmasını ve bulunmasını sağlar. Bu servislerde XML-temelli diller kullanılır.

Her Windows Phone7 uygulaması aynı anda maksimum 6 bağlantı çıkışı ile çalışabilir. Windows Phone' da ağ desteği Silverlight üzerinden sağlanmaktadır.

b. Web ve Data Servisleri

Windows Phone' dan web servislerine ulaşmak için HttpWebRequest ve WebClient sınıfları kullanılır. Visual Studio' da bulunan Servis Referansı Ekleme (Add Service Reference) seçeneği ile web servisin sıralama, istek ve cevap kodlarının kolayca eklenmesi sağlanır.

Data servisleri sayesinde HttpWebRequest ve WebClient sınıfları kullanılarak internet üzerindeki bilgilere ulaşılabilir ve bilgiler değiştirilebilir. Bir proxy sınıfı, WSDL veya CSDL dosyalarına dayanan(sırasıyla) web servisi veya data servisini temsil eden sınıftır.

Aşağıdaki listedeki sınıfları kullanarak direkt olarak veya uygulama aracılığı ile web isteğinde bulunabilirsiniz:

- **WebClient Sınıfı:** URI-temelli bir kaynaktan data almak veya oraya data göndermek için kullanılması gereken metotları içerir.
- **HttpWebRequest Sınıfı:** WebRequest sınıfının HTTP-uygulamasının yapılması için kullanılır.
- **Silverlight Service Model Proxy Oluşturma Aracı(SLsvcUtil.exe):** Bir web servisi WSDL dosyasına bağlı Proxy sınıflarını oluşturur.
- **Visual Studio Add Service Reference Özelliği:** Bir web servisi WSDL dosyasına ya da data servisi CSDL dosyasına dayanan Proxy sınıflarını oluşturur.
- **WCF Data Service Client Utility (DataSvcUtil.exe):** Bir data servisi CSDL dosyasına dayanan Proxy sınıflarını oluşturur.

c. Rss Uygulaması Geliştirme

Windows Phone7’de web bağlantılı program yazmak için en çok RSS uygulamaları kullanılır. Haberler, sinema filmleri, blog içerikleri gibi birçok bilgiye kolay ulaşımı sağlayan uygulamalar bu şekilde geliştirilir. Aşağıdaki RSS uygulaması örneği, açıklamalarıyla birlikte kod geliştirme konusunda yardımcı olacaktır.

İlk olarak Visual Studio’da Windows Phone Application seçeneği ile yeni bir Phone7 uygulaması oluşturun. İşletim sistemi olarak da Windows Phone OS 7.1 seçin.

- Uygulamada RSS beslemesi kullanabilmek için SyndicationFeed sınıfı kullanılır. Bu sınıf System.ServiceModel.Syndication isim alanı altında bulunur.SyndicationFeed sınıfını kullanabilmek için projeye DLL (dinamik link kütüphanesi) referansı eklenmelidir.

Visual Studio’ da Project menüsünden Add Reference seçeneğini seçiniz, Browse sekmesine tıklayınız. Program Files dizinin altındaki Microsoft SDKs/Silverlight/v4.0/Libraries/Client/ klasörüne geliniz. System.ServiceModel.Syndication.dll ‘I seçiniz ve OK’ a tıklayınız. Böylece SyndicationFeed sınıfını kullanabilmek için gerekli olan kütüphaneyi projeye eklemiş olunuz.

- Rss beslemelerinin metninde çoğunlukla HTML karakterleri ve uygulamada görünmesini istemeyeceğiniz bilgiler bulunur. Bunları ayırt edip eleyebilmek için RssTextTrimmer sınıfını kullanmak faydalı olur. Proje ismine sağ tıklayarak, Add->New Item seçiniz. Installed Templates->Visual C# bölümünden Class seçeneğini seçiniz. Dosyayı RssTextTrimmer.cs olarak isimlendiriniz ve Add butonuna tıklayınız.

RssTextTrimmer.cs dosyasına şu isim alanlarını ekleyiniz:

```
using System.Windows.Data;
using System.Globalization;
using System.Text.RegularExpressions;
```

Sınıf tanımını şu şekilde değiştiriniz:

```
public class RssTextTrimmer : IValueConverter
```

Aşağıdaki kodu sınıf dosyasına ekleyiniz:

```
// Her SyndicationItem da bulunan metin alanlarını temizlemek için.
public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
{
    if (value == null) return null;
```

```

int maxLength = 200;
int strLength = 0;
string fixedString = "";

// HTML etiketlerini ve newline karakterlerini kaldırarak, HTML
kodlarını
//çözme işlemi:

// HTML etiketlerini kaldır
fixedString = Regex.Replace(value.ToString(), "<[^>]+>", string.
Empty);

// Yeni satıra geçme karakterlerini kaldır
fixedString = fixedString.Replace("\r", "").Replace("\n", "");

// Kodlanmış HTML karakterlerini kaldır
fixedString = HttpUtility.HtmlDecode(fixedString);

strLength = fixedString.ToString().Length;

// Eğer geriye metinde hiçbir şey kalmadıysa null değerini döndü
r
if (strLength == 0)
{
    return null;
}

// Eğer metin çok uzunsa kes.
else if (strLength >= maxLength)
{
    fixedString = fixedString.Substring(0, maxLength);

    // Kesme işleminin kelime ortasına denk gelmemesi için, LastI
ndexOf kullanarak son boşluk karakterini bul ve kesme işlemini gerçe
kleştir.
    fixedString = fixedString.Substring(0, fixedString.LastIndex
Of(" "));
}

fixedString += "...";

return fixedString;
}

// Bu kodda TwoWay binding kullanılmadığı için, ConvertBack fonksiyo
nunu tamamlamak gerekli değil.
public object ConvertBack(object value, Type targetType, object para
meter, CultureInfo culture)
{
    throw new NotImplementedException();
}

```

Bu sınıfı dönüştürücü (converter) olarak ayarlamak için App.xaml dosyasında <Application.Resources> etikeni içine şu satırı ekleyiniz:

```
<converter:RssTextTrimmer xmlns:converter="clr-namespace:sdkRSSReaderCS" x:Key="RssTextTrimmer" />
```

(sdkRSSReaderCS : projenizin isim alanının adı)

- XAML kodunu değiştirmek için MainPage.xaml dosyasını açın.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
saturunu aŝađıdaki kod parçası ile deđiŝtirin.
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="0,0,12,0">
  <Button Content="Load Feed" Height="72"
  HorizontalAlignment="Left" Margin="9,6,0,0" Name="loadFeedButton"
  VerticalAlignment="Top" Width="273" Click="loadFeedButton_Click" />

  <ListBox Name="feedListBox" Height="468"
  HorizontalAlignment="Left" Margin="20,100,0,0"
  VerticalAlignment="Top" Width="444"
  ScrollViewer.VerticalScrollBarVisibility="Auto"
  SelectionChanged="feedListBox_SelectionChanged">
    <ListBox.ItemTemplate>
      <DataTemplate>
        <StackPanel VerticalAlignment="Top">
          <TextBlock TextDecorations="Underline"
          FontSize="24" Name="feedTitle" TextWrapping="Wrap" Margin="12,0,0,0"
          HorizontalAlignment="Left" Foreground="{StaticResource
          PhoneAccentBrush}" Text="{Binding Title.Text,
          Converter={StaticResource RssTextTrimmer}}" />
          <TextBlock Name="feedSummary"
          TextWrapping="Wrap" Margin="12,0,0,0" Text="{Binding Summary.Text,
          Converter={StaticResource RssTextTrimmer}}" />
          <TextBlock Name="feedPubDate"
          Foreground="{StaticResource PhoneSubtleBrush}" Margin="12,0,0,10"
          Text="{Binding PublishDate.DateTime}" />
        </StackPanel>
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
  <Border BorderBrush="{StaticResource PhoneSubtleBrush}"
  BorderThickness="1" Height="2" HorizontalAlignment="Left"
  Margin="20,88,0,0" Name="border1" VerticalAlignment="Top"
  Width="438" />
</Grid>
```

- Arka plandaki kodu değiştirmek için sağ tıklayarak View Code seçeneğini seçin. Aşağıdaki isim alanlarını koda ekleyin:

```
using System.IO;
using System.ServiceModel.Syndication;
using System.Xml;
using Microsoft.Phone.Tasks;
```

Şu kodu ise MainPage'de "constructor"dan sonraki kısma yerleştirin. Kodun yazılış şekli ve açıklaması yorum kısımlarında verilmiştir:

```
// 'load Feed' ya da 'Refresh Feed' butonu tıklandığında çalışan Click işleyici
private void loadFeedButton_Click(object sender, System.Windows.RoutedEventArgs e)
{
    // Bu kodda WebClient kullanılmıştır, çünkü uygulaması daha kolaydır ve bu örnek için HttpWebRequest'in sağladığı ileri işlevselliğe gerek yoktur
    WebClient webClient = new WebClient();

    // Rss beslemesini indirmeden önce, DownloadStringCompleted olayını oluştur.
    webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(webClient_DownloadStringCompleted);

    // RSS beslemesini indir. DownloadStringAsync kullanılmıştır çünkü bilgi akışını açık bırakmaya veya kanalı kapamaya gerek yoktur. Öyle bir durumda ise OpenStreamAsync kullanılmalıdır.
    webClient.DownloadStringAsync(new System.Uri("http://windowsteam.blog.com/windows_phone/b/windowsphone/rss.aspx"));
}

// Besleme tamamen indirildikten sonra çalışan Event işleyicisi
private void webClient_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    if (e.Error != null)
    {
        Deployment.Current.Dispatcher.BeginInvoke(() =>
        {
            MessageBox.Show(e.Error.Message);
        });
    }
    else
    {
        // Uygulamada istenmeyen bir durum oluşmasına karşın, beslemeyi State özelliğine kaydet:
        this.State["feed"] = e.Result;
    }
}
```

```

        UpdateFeedList(e.Result);
    }
}

// Bu metod uygulama çöktükten sonra kullanıcının uygulamaya gidip gitmediğini belirler.
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
    // Beslemenin sayfa durumunda kayıtlı olup olmadığını kontrol et
    :
    if (this.State.ContainsKey("feed"))
    {
        // Eğer uygulama çöktüyse, ListBox boş olmalıdır. Bunu kontrol ederek beslemeyi tekrar yükle. (Bu metod sayfalar arasında geçiş yaparken de kullanıldığı için bu kontrole gerek duyulur.)

        if (feedListBox.Items.Count == 0)
        {
            UpdateFeedList(State["feed"] as string);
        }
    }
}

// Bu metod beslemeyi oluşturarak ListBox 'a bağlar.
private void UpdateFeedList(string feedXML)
{
    // Beslemeyi bir SyndicationFeed nesnesine yükle.
    StringReader stringReader = new StringReader(feedXML);
    XmlReader xmlReader = XmlReader.Create(stringReader);
    SyndicationFeed feed = SyndicationFeed.Load(xmlReader);

    // Windows Phone OS 7.1 'de, WebClient olayları üzerinde çağrılmaları thread'ler ile aynı tipte olur.

    //UI yi değiştirmek için Dispatcher kullanmak en pratik olanıdır, böylece UI thread'i yoğun işlemden bağımsız tutulur.
    Deployment.Current.Dispatcher.BeginInvoke(() =>
    {
        feedListBox.ItemsSource = feed.Items;

        loadFeedButton.Content = "Refresh Feed";
    });
}

// Besleme öğeleri için The SelectionChanged işleyicisi
private void feedListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    ListBox listBox = sender as ListBox;
}

```

```
if (listBox != null && listBox.SelectedItem != null)
{
    SyndicationItem sItem = (SyndicationItem)listBox.SelectedItem;

    if (sItem.Links.Count > 0)
    {
        // Besleme ögesi ile ilişkili olan URI yi çağır.
        Uri uri = sItem.Links.FirstOrDefault().Uri;

        // Besleme ögesini dolaştırmak için yeni bir WebBrowserTask yarat.

        WebBrowserTask webBrowserTask = new WebBrowserTask();
        webBrowserTask.Uri = uri;
        webBrowserTask.Show();
    }
}
```




Bu bölümde, Windows Phone üzerinde verilerimizi nasıl tutup, işleyebileceğimizi öğreneceğiz. Bir veri tabanına ihtiyacınız olduğunda ya da herhangi bir veriyi saklamak istediğinizde nerler yapmanız gerektiğini ve dosya işlemlerinin nasıl gerçekleştiğini göreceğiz.

Yazardan...

14. Veri Saklama ve Senkronize Etme

Phone 7.1 SDK ile uygulamalarımızın datalarını ve ayarları için gerekli olan verileri Isolated Storage olarak adlandırılan data alanında tutabiliriz. Bu alana verilerimizi yazabilir, ihtiyaç duyduğumuzda okuyabiliriz. Yalıtılmış/izole edilmiş depolama alanı anlamına gelen Isolated Storage adından da anlaşılacağı gibi sadece bir uygulamanın aynı anda erişebileceği bir ortamdır. Uygulamamızın datalarını diğer uygulamalar göremez, diğer Phone 7 uygulamalarının datalarını da kendi uygulamamız göremez. Data her uygulama başına izole edilmiş durumdadır. Isolated Storage' yi kullanan bu mobil mimari Silverlight-Based Windows uygulamaları ile benzerdir. Tüm I/O işlemleri Isolated Storage ile sınırlıdır ve işletim sistemi dosya sistemine direk erişime izin verilmemektedir, bu izinsiz data erişimlerini engelleyerek güvenliği sağlar.

Tabii ki veri depolama alanları telefonlarda sınırlıdır. İndirilen uygulamalar, media dosyaları, telefon için senkronizasyon dataları, Isolated Storage kullanımını maksimuma getirir. Windows Phone boş kalan bellek miktarı, tüm bellek miktarının %10 nuna ulaştığı zaman kullanıcıya bazı dosyalarını silmesi için bir uyarı çıkarır, bu nedenle Windows Phone için geliştirdiğimiz uygulamalarda gereksiz bilgileri tutmamalı, bellek alanını oldukça verimli bir şekilde kullanmalıyız. Depolama alanını daha verimli şekilde kullanabilmek için aşağıdaki alternatifleri değerlendirebiliriz.

Geçici Veri ve Dosyalar Oluşturma: Kullanıcının geçici olarak kullanabileceğini düşündüğümüz verileri belli bir süre tutup, daha sonra belirli aralıklarla bu alanı temizleyerek boş bellek alanı açabiliriz. Bu işlev Internet Explorer' ın Temporary Files' ı kullanmasına benzetilebilir, kullanıcının sayfaları daha hızlı yüklemesi için bu cache kullanılırken, uzun süre kullanılmayan alanlar belirli aralıklarla temizlenerek bellek alanı geri kazandırılabilir.

Kullanıcı Üretimli Datalar: Uygulamalarımızda kullanıcılara veri girişi sağladığımız gibi uygun yerlerde bu verileri silme imkanı da tanımalıyız.

Uygulama Dataları: Bir liste verisinde yada bir sözlük uygulamasında sadece ilgili verileri kullanıcıya sunmayı sadece gerekli verileri işlemeyi sağlamak da performans ve veri alanı sağlar.

Windows Phone uygulamaları local datalarını Isolated Storage' de tutabilir ve genelde üç şekilde olur:

Ayarlar: Uygulama ayar bilgileri (key value) yalıtılmış alanda tutulabilir örneğin uygulamamızda ses seviyesinin şuan da kullanıcı tarafından hangi seviyeye alındığını bir değer ile tutabiliriz.

Klasörler ve Dosyalar: Yalıtılmış depolama alanında klasörler oluşturabilir, dosyalar ekleyebiliriz, bu klasör ve dosyalara veri yazabilir yazılan verileri okuyabiliriz.

İlişkisel Data: Yerel veri tabanı oluşturma işlemidir, telefonun kendi hafızasında tutulan bir veri tabanı oluşturabiliriz veri tabanında tablolarımız ve tablolarla ilgili sütunlar ve sütunlar arası ilişkiler kullanılabilir. Veri tabanı işlemleri gerçekleşirken LINQ to SQL kullanılır.

A. Ayarlar

Isolated Storage' yi Kullanmak için Yardımcı Sınıf Tanımlama

Yaptığımız uygulamalarda bazı uygulama durum bilgilerini telefon hafızasında tutmak isteyebiliriz, örneğin uygulamamızda Server ya da Port bilgisi ayarlarını kullanıcıdan bir defa alarak her uygulama açıldığında bu değerleri tekrar tekrar girilmesine gerek kalmadan Isolated storage' de tuttuğumuz bilgiler ile sağlayabiliriz. Isolated Storage' de değerleri tutabilmek gerektiğinde bu verilere yeni değer atamak, var olan değerleri Isolated Storage' den alabilmek için bir sınıf tanımlamamız gerekir. IsolatedSettingsHelper adındaki sınıfı projeye sağ tıklayıp Add-> New Item -> Class 'ı seçerek oluşturalım.

IsolatedStorageSettings 'i kullanarak bir ayar hücresi oluşturalım.

```
private static System.IO.IsolatedStorage.IsolatedStorageSettings
Settings=System.IO.IsolatedStorage.IsolatedStorageSettings.Application
Settings;
```

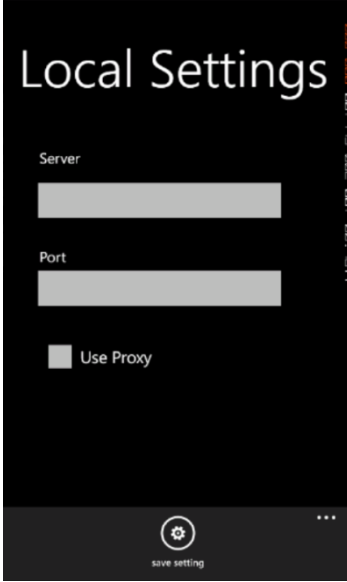
Sınıfın Değer getirme metodu(GetValue) aşağıdaki gibi oluşturulabilir.

```
public static T GetValue<T>(string Key)
{
    T returnValue;
    if (Settings.TryGetValue<T>(Key, out returnValue)
{
        return returnValue;
    }
    else
    {
        return default(T);
    }}
```

T değeri her şey olabilir generic, string, integer vb. yani template türündendir. İkinci olarak tanımlayacağımız metod tanımlanan değerlere değer atayabilen bir metod olacaktır.

```
public static void SetValue<T>(string Key, T Value)
{
    if (Settings.Contains(Key))
    {
        Settings[Key] = Value;
    }
    else
    {
        Settings.Add(Key, Value);
    }
}
```

Bu metod ilgili ayar deęerini bulup g¼nceller, yoksa Isolated Storage' ye ekler. Bu sınıfı oluřturduktan sonra kodlama yapacaęımız sayfanın ara y¼z¼n¼ ařaęıdaki gibi oluřturup, iki adet textbox ve bir checkbox oluřturup, yaptığımız deęiřiklikleri kaydetmek i¼in ApplicationBarIcon butonu kullanalım.



Uygulamanın Loaded metodunda Isolated Storage' de daha ¼nceden kullanıcı tarafından atanmış bir deęer varsa GetValue metodu ile alınıp kullanıcıya g¼sterilir. Eęer atanmış bir deęer yoksa string.Empty yani boř bir string TextBoxlar' da g¼sterilir.

```
private void PhoneApplicationPage_Loaded(object sender,
RoutedEventArgs e)
{
    textBox1.Text =
(IsolatedSettingsHelper.GetValue<string>("Server") ?? string.Empty);
    textBox2.Text =
(IsolatedSettingsHelper.GetValue<string>("Port") ?? string.Empty);
    checkBox1.IsChecked
=IsolatedSettingsHelper.GetValue<bool>("UseProxy");}
```

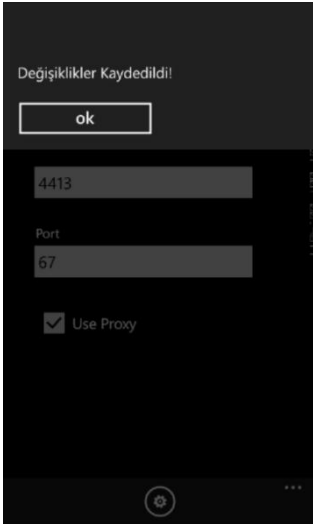
Kullanıcı TextBox' lara yeni veriler girmiř ve daha ¼nce veri girilmemiř ise boř alanlar SetValue metodu yardımı ile yeni verilerle doldurulur, eęer kullanıcı bu alanlara daha ¼nceden bir deęer atmış ise eski veriler TextBox alanlarındaki yeni girilen verilerle g¼ncellenir. Bu iřlem ayarlar simgesi g¼rseline sahip ApplicationBar butona tıklanđında yapılmalıdır dolayısı ile bu butonun click metoduna ařaęıdaki kodlar yazılır.

```

private void ApplicationBarIconButton_Click(object sender, EventArgs e)
{
    IsolatedSettingsHelper.SetValue<string>("Server",
textBox1.Text);
    IsolatedSettingsHelper.SetValue<string>("Port",
textBox2.Text);

    IsolatedSettingsHelper.SetValue<bool>("UseProxy", checkBox1.IsChecked.
Value);
    MessageBox.Show("Değişiklikler Kaydedildi!");
}

```



B. Dosya İşlemleri

Windows Phone 7 mobil işletim sistemi bize Isolated Storage' ı kullanarak klasör ve dosya oluşturma bu oluşumlara veri yazabilme ve yazdığımız dosyalardan veri okuyabilme imkanı sunar. Örnek bir uygulama ile bu konuyu detaylıca inceleyelim.

Visual Studio' yu açarak Silverlight for Windows Phone' u seçerek yeni bir Windows Phone Application oluşturalım. Proje oluşturduktan sonra dosya işlemlerini yapabilmek için aşağıdaki using tanımlamasını yapmamız gerekir.

```

using System.IO.IsolatedStorage;

```

Bu ifadeden sonra bir IsolatedStorage dosyası oluşturacağımız için aşağıdaki tanımlamayı yapalım:

```
IsolatedStorageFile phoneStorage;
```

Ve sayfanın yapılandırıcısında phoneStorage' yi oluşturalım.

```
IsolatedStorageFile phoneStorage;  
public Page_LocalDataFileSystem()  
{  
    InitializeComponent();  
    phoneStorage =  
IsolatedStorageFile.GetUserStoreForApplication();  
}
```

Artık dosya oluşturup yazma yapabiliriz ve bu dosyadan verilerimizi okuyabiliriz. Bunun için sayfanın ara yüzüne iki adet buton yerleştirelim:



Dosyaya Yaz butonunun click eventine ulaşarak, klasör ve dosya oluşturma ve oluşturduktan sonra yazma işlemleri için aşağıdaki kodlamaları yapalım:

Eğer klasör yoksa oluşturmak için:

```
if (!phoneStorage.DirectoryExists("klasorum1"))  
{  
    phoneStorage.CreateDirectory("klasorum1");  
}
```



```
//yazma işlemleri
private void button2_Click(object sender, RoutedEventArgs e)
{
    if (!phoneStorage.DirectoryExists("klasorum1"))
    {
        phoneStorage.CreateDirectory("klasorum1");
    }
    using (IsolatedStorageFileStream filestream = phoneStorage.OpenFile("klasorum1\\ornek.txt",
        System.IO.FileMode.OpenOrCreate))
    {
        using (System.IO.StreamWriter writer = new System.IO.StreamWriter(filestream))
        {
            writer.WriteLine("Microsoft Phone 7");
        }
    }
}
```

Dosyaya yazdığımız verileri StreamReader yardımı ile okuyup, kullanıcıya messagebox vasıtası ile sunalım, bunun için dosyadan oku adlı butonun click 'ine ulaşip aşağıdaki işlemleri yapalım:

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    if(phoneStorage.FileExists("klasorum1\\ornek.txt"))
    {
        using (IsolatedStorageFileStream filestream =
phoneStorage.OpenFile("klasorum1 \\ornek.txt",
System.IO.FileMode.OpenOrCreate))
        {
            using (System.IO.StreamReader reader = new
System.IO.StreamReader(filestream))
            {
                string filecontent = reader.ReadToEnd();
                MessageBox.Show(filecontent);
            }
        }
    }
}
```

Sonuç olarak aşağıdaki şekilde bir uyarı ile dosyaya yazdığımız veriyi ekranda görmeliyiz.

C. Veri Tabanı İşlemleri

1. Veri Tabanı Model ve Görünümünün Oluşturulması

Bu bölümde veri saklama konusunda önemli bir yer tutan ve Windows Phone 7 nin Isolated Storage 'ini kullanarak veri tabanı oluşturma ve veri tabanı işlemlerini öğreneceğiz. Kullanıcın istediği sayıda alış-veriş listesi tutabileceği, bu listelere ürünler ekleyebileceği, fiyatları girilen ürünlerin fiyatlarını toplayarak o anki listedeki alışveriş maliyetini gösterebilecek, kullanıcın sevdiği ürünleri saklayabileceği gibi bir çok özelliği olan bir uygulama yaparak Windows Phone 7 de veri tabanı işlemlerine giriş yapalım.

Bu uygulamada en önemli kısım alışveriş listeleri sayısının kullanıcıya bağlı olarak oluşturulması yani dinamik olarak oluşturulmasıdır tabii ki bu durum ek bazı işlemler gerektirir. Veri tabanı oluşumunda gerçekleştirdiğimiz sorgular LINQ To SQL sorgularındır ve veri tabanı oluşumunda DBDataContext.cs sınıfı ile database modelini belirlerken yani databasedeki tablolar, tabloların sahip olduğu sütunlar, yabancı anahtar vb. tanımlamaları yer alırken, ikinci olarak oluşturacağımız DBView.cs sınıfında da LINQ to SQL sorguları ve bu sorgularla oluşturacağımız ObservableCollection oluşumları, bu koleksiyonları kullanarak add, delete vb. metotları yer alacaktır. Bu veri tabanı metotları kullanılarak örneğin AddNewProduct vb. ile ileride oluşturacağımız sayfalarda ürün, liste, mağaza gibi elemanları veri tabanına ekleyip silebileceğiz.

Yeni bir *Silverlight for Windows Phone* projesi oluşturalım ve MyShopping olarak adlandıralım, proje yaratıldıktan sonra Solution Explorer' da projeye sağ tıklayarak Add Reference ile bazı referansları projeye ekleyelim:

Add Reference->Browse ile karşımıza çıkan tab menüden *Microsoft.Phone.Controls.Toolkit.dll*' yi ekleyelim artık Reference klasöründe Microsoft.Phone.Controls.Toolkit.dll assembly' si gözükecektir. İkinci referansımız ise Add Reference ->.NET tabı dan *System.Data.Linq* 'i seçmek olacaktır. Aynı bölümden pivot sayfaları kullanabilmek için *Microsoft.Phone.Controls* referansı da seçilerek projeye eklenir.

Bu işlemlerden sonra artık database modelimizi oluşturabiliriz bunun için DBDataContext.cs sınıfını oluşturalım yine projemize sağ tıklayıp Add-> New Item diyerek bir class oluşturup ismini bu şekilde belirleyelim. Sınıfı oluşturduktan sonra ilk olarak veri tabanı Connection String tanımlaması yapmalıyız. Daha sonra tablo tanımlamalarımızı yapıp tablo sütunlarını oluşturabiliriz.

```
public class DBDataContext : DataContext
{
    // Pass the connection string to the base class.
    public DBDataContext(string connectionString)
        : base(connectionString)
    { }

    //Table for shopping products
    public Table<TProduct> Products;

    // List of shopping products'
    public Table<TList> PLists;

    // Table for shopping products' store
    public Table<TStore> Stores;}

```


Bu tanımlamadan sonra sıra tablo sütunlarının sütun özelliklerini ve yabancı anahtarları belirlemeye gelir.

Ürünler tablosu için örnek sütun aşağıdaki şekilde yapılır ve bu sütuna Primary Key, Not Null Identity gibi özellikler de atanmıştır.

```
public class TProduct : INotifyPropertyChanged, INotifyPropertyChanging
{
    private int _ProductId;

    [Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT
NULL Identity", CanBeNull = false, AutoSync = AutoSync.OnInsert)]
    public int ProductId
    {
        get { return _ProductId; }
        set
        {
            if (_ProductId != value)
            {
                NotifyPropertyChanging("ProductId");
                _ProductId = value;
                NotifyPropertyChanged("ProductId");
            }
        }
    }
}
```

Tablonun diğer sütunları ve özellikleri benzer şekilde tanımlanır. Ürünler tablosunda yer alan diğer sütunlar:

```
[Table] TProduct
string ProductName
string ProductStore
decimal Price
bool IsLiked
int _PListId
```

benzer mantıkla Listeler tablosu ve Mağazalar tablosu da oluşturulur. Listeler tablosunda:

```
[Table] TList
int Id
string Name
bool Checked
```

kolonları vardır Mağazalar tablosunda mağaza Id ve ismi kolonları yer alır:

```
[Table] TStore
int SID
string StoreName
```

Listeler tablosu ile Ürünler tablosu arasında yabancı anahtar(foreign key) ilişkisi vardır. Bu ilişki şu şekilde sağlanır.

İlk olarak TProduct tablosunda aşağıdaki ilişkilendirme yapılır:

```
[Column]
    internal int _PListId;

    private EntityRef<TList> _tlists;

    [Association(Storage = "_tlists", ThisKey = "_PListId", OtherKey =
"Id", IsForeignKey = true)]
    public TList TListx
    {
        get { return _tlists.Entity; }
        set
        {
            NotifyPropertyChanging("TListx");
            _tlists.Entity = value;

            if (value != null)
            {
                _PListId = value.Id;
            }

            NotifyPropertyChanging("TListx");
        }
    }
}
```

Bu tanımlamada TProduct tablosundaki `int _PListId` ile TList tablosundaki `Id` arasında bir yabancı anahtar ilişkisi olduğu belirtilir benzer şekilde TLists tablosunda aşağıdaki ilişkilendirme yazılır.

```
private EntitySet<TProduct> _plists;

    [Association(Storage = "_plists", OtherKey = "_PListId", ThisKey =
"Id")]
    public EntitySet<TProduct> Mshop
    {
        get { return this._plists; }
        set { this._plists.Assign(value); }
    }
}
```

TList tablosunda buna ek olarak `add` ve `remove` işlemlerinde tablodaki verilerin senkronize olması için aşağıdaki kodlar gereklidir.

```

    // Assign handlers for the add and remove operations,
    respectively.
    public TList()
    {
        _plists = new EntitySet<TProduct>(
            new Action<TProduct>(this.attach_Mshop),
            new Action<TProduct>(this.detach_Mshop)
        );
    }

```

```

    // Called during an add operation
    private void attach_Mshop(TProduct Mshopx)
    {
        NotifyPropertyChanging("TProduct");
        Mshopx.TListx = this;
    }

    // Called during a remove operation
    private void detach_Mshop(TProduct Mshopx)
    {
        NotifyPropertyChanging("TProducts");
        Mshopx.TListx = null;
    }

```

Veri tabanını modelini bu sınıfla oluşturduğumuza göre veri tabanı görünümün için yeni bir sınıf oluşturalım ve DBView.cs olarak adlandıralım. Bu sınıf artık oluşturduğumuz model sınıfını kullanır, ileriki bölümlerde oluşturacağımız tüm sayfalarda Model sınıfını kullanır bunun için sınıfımıza `using MyShopping.Model;` ifadesini ekleriz.

DBView de data context tanımlaması ve datacontext nesnesinin üretilmesi için DBView yapılandırıcısında aşağıdaki kodlar yer alır.

```

// LINQ to SQL data context for the local database.
public DBDataContext DBShop;

// Class constructor, create the data context object.
public DBView(string DBConnectionString)
{
    DBShop = new DBDataContext(DBConnectionString);
}

```

ObservableCollection' lar oluşturulur. Örnek bir koleksiyon tanımlaması aşağıdaki gibidir.

```

private ObservableCollection<TProduct> _AllProductItems;
public ObservableCollection<TProduct> AllProductItems
{
    get { return _AllProductItems; }
    set
    {
        _AllProductItems = value;
        NotifyPropertyChanged("AllProductItems");
    }
}

```

Bu şekilde gerekli olan koleksiyonlar tanımlanarak t-sql sorguları yardımı ile koleksiyonlar oluşturulur. Örneğin AllProductItems koleksiyonu aşağıdaki şekilde bir Linq to SQL sorgusu ile ilişkilendirip oluşturabiliriz.

```

var ProdcusInDB = from TProduct product in DBShop.Products join
TList listr in DBShop.PLists on product._PListId equals listr.Id
select product;
AllProductItems = new
ObservableCollection<TProduct>(ProdcusInDB);

```

Ürünler koleksiyonu oluşturulurken yabancı anahtar ilişkisi unutulmamalıdır dolayısı ile t-sql sorgusunda da buna önem verilmelidir. Diğer Linq to SQL sorgularında yabancı anahtar ilişkisi olmamakta ve daha sade bir sorgulama yer almaktadır.

```

var ListsInDB = from TList list in DBShop.PLists
                select list;
var StoresInDB = from TStore store in DBShop.Stores
                 select store;
ProductLists = new ObservableCollection<TList>(ListsInDB);

                StoreLists = new ObservableCollection<TStore>(StoresInDB);

```

Bu koleksiyonları kullanarak Add Product, List, Store ya da Delete Product, List, Store metotları yazabiliriz.

```

public void AddList(TList newList)
{
    // Add new shop list to the data context.
    DBShop.PLists.InsertOnSubmit(newList);

    // Save changes to the database.
    DBShop.SubmitChanges();

    // Add new shop list to observable collection.
    ProductLists.Add(newList);}

```

```

public void DeleteList(TList ListDelete)
{
    // Remove shop list from the observable collection.
    ProductLists.Remove(ListDelete);

    // Remove shop list from the data context.
    DBShop.PLists.DeleteOnSubmit(ListDelete);

    // Save changes to the database.
    DBShop.SubmitChanges();
}

```

DB Model ve DB View sınıfları oluşturduktan sonra veri tabanının yaratılması için App.xaml.cs sayfasının using alanına ve yapılandırıcısına aşağıdakileri ekleriz.

```

using MyShopping.Model;
using MyShopping.View;

```

```

    // Specify the local database connection string.
    string DBConnectionString = "Data
Source=isostore:/MyShopping.sdf";

    // Create the database if it does not exist.
    using (DBDataContext db = new
DBDataContext(DBConnectionString))
    {
        if (db.DatabaseExists() == false)
        {
            // Create the local database.
            db.CreateDatabase();

            // Save changes to the database.
            db.SubmitChanges();
        }
    }

    // Create the ViewModel object.
    view = new DBView(DBConnectionString);

    // Query the local database and load observable
collections.
    view.LoadCollectionsFromDatabase();

```

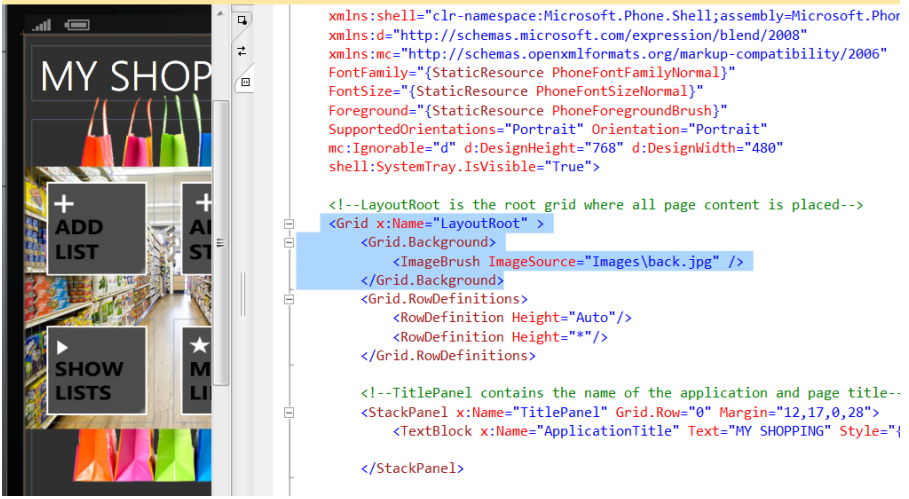
Veri tabanını ve kullanacağımız metotları oluşturduktan sonra bunlar yardımı ile diğer sayfaları (kullanıcı ara yüzlerini) oluşturmaya başlayabiliriz.

2. Bařlangıç Sayfasının Oluřturulması

Start Page, kullanıcının kolayca menülere ulařıp iřlem yapmasını saęlayan dört adet butondan oluřmaktır. Bunun için Projemize saę tıklayıp Add-> New Item deyip yeni bir Portrait Page oluřturalım ve StartPage olarak adlandıralım.



StartPage' in arka plandaki resmini oluřturmak için ařaęıdaki brush iřlemlerini yapalım dięer sayfaların arka planının resimle doldurulması da aynı řekildedir.



Sayfa Toolbox alanından dört adet buton ekleyerek, butonların görsel bir içerik ifade etmesi için her bir butona aşağıdaki background (arka plan) kodlamasını yerleştirelim.

```

<Button Content="" Height="175" HorizontalAlignment="Left"
Margin="228,90,0,0" Name="button2" VerticalAlignment="Top"
Width="193" Click="button2_Click">
  <Button.Background>
    <ImageBrush x:Name="storeImage"
ImageSource="Images\store2.jpg" Stretch="Fill"/>
  </Button.Background>
</Button>

```

Böylece Visual Studio içinde oluşturduğumuz Images klasörüne istediğimiz resmi ekleyip yolunu belirterek buton backgroundu yapabiliriz. TilePanel ve Application Tile başlığımızı da içeriğe uygun bir şekilde değiştirerek sayfamızın görsel kısmını tamamlamış oluyoruz.

StartPage.xaml.cs de ise sadece butonlara tıklandığında hangi sayfalara navigate (hareket) edeceğimizi butonların click eventlerine yazalım.

```

private void button1_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
}

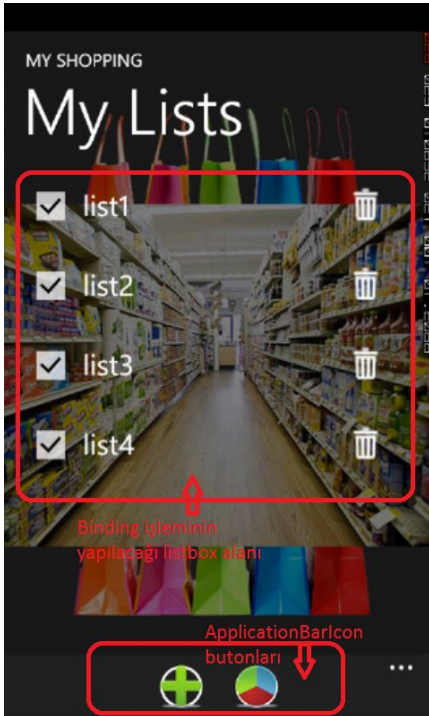
private void button2_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/StorePage.xaml",
UriKind.Relative));
}

private void button3_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/PanoramaLists.xaml",
UriKind.Relative));
}

private void button4_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/ChanceProductPage.xaml",
UriKind.Relative));
}

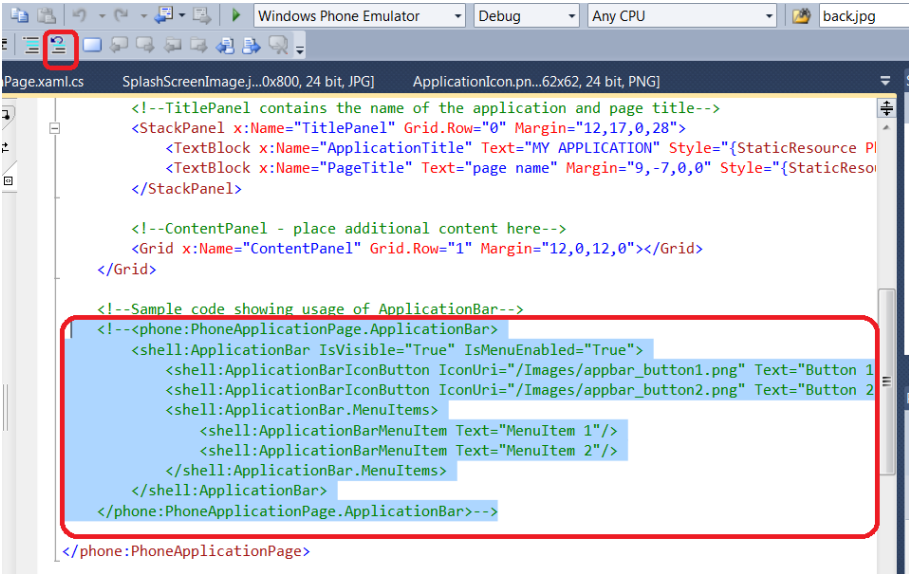
```

3. Ana Sayfanın Oluşturulması



MainPage.xaml sayfası bizim kullanıcıya alışveriş listesi ekleme, silme ve eklenen alışveriş listeleri listeleme imkanı sunabildiğimiz bir sayfadır.

Bu sayfayı oluşturmak için ApplicationBarIcon butonlarından da faydalanacağız bu butonları kullanmak için yeni sayfa oluşturduğumuzda default olarak yorum halinde olan ve aşağıda gösterilen alanı, yorum olmaktan çıkararak erişebiliriz.



```
<!--TitlePanel contains the name of the application and page title-->
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
  <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION" Style="{StaticResource Pl
  <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0" Style="{StaticReso
</StackPanel>

<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
</Grid>

<!--Sample code showing usage of ApplicationBar-->
<!--<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
    <shell:ApplicationBarIconButton IconUri="/Images/appbar_button1.png" Text="Button 1
    <shell:ApplicationBarIconButton IconUri="/Images/appbar_button2.png" Text="Button 2
    <shell:ApplicationBar.MenuItems>
      <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
      <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
    </shell:ApplicationBar.MenuItems>
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar-->
</phone:PhoneApplicationPage>
```

Öncelikle yukarıda belirtilen kodları seçip kırmızı kutu içine alınan kısmı “Uncomment the selected lines” butonuna tıklayarak bu alanı kullanabilir hale geliriz. Bu işlemi tamamladıktan sonra aşağıdaki şekilde kodlamamızı yaparak Add new list ve Showlists uygulama butonlarını oluşturalım yine bu buton resimlerini yolunu vererek kendi seçtiğimiz resimlerden oluşturabileceğimiz gibi

C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons\dark ya da

C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons\light yolu altında bulunan varolan ikon resimlerini kullanabiliriz.

```
<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
    <shell:ApplicationBarIconButton
      IconUri="Icons/add2.png"
      Text="add"
      x:Name="newListAddBarButton"
      Click="newListAddBarButton_Click"/>
    <shell:ApplicationBarIconButton
      IconUri="Icons/show2.png"
      Text="show"
      x:Name="ShowMyListBarButton"
      Click="ShowMyListBarButton_Click"/>
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

ApplicationBarIcon butonlarımızı oluşturduğumuza göre veri tabanı ile ilişkilendirip veri tabanında yer alan bilgileri listeleyebilmemiz için bir ListBox oluşturalım ve ItemsSource' ini ProductLists ile ilişkilendirelim.

```
ItemsSource="{Binding ProductLists}"
```

ListBox' ı oluşturmadan önce veri tabanından gelecek verilerin hangi şablonda olacağını belirleyelim. Bunun için aşağıdaki şekilde bir DataTemplate oluşturalım:

```
<phone:PhoneApplicationPage.Resources>
  <DataTemplate x:Key="ListItemTemplate">

    <Grid HorizontalAlignment="Stretch" Width="420">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="100" />
      </Grid.ColumnDefinitions>

      <CheckBox
        IsChecked="{Binding Checked, Mode=TwoWay}"
        Grid.Column="0" VerticalAlignment="Top"/>
```

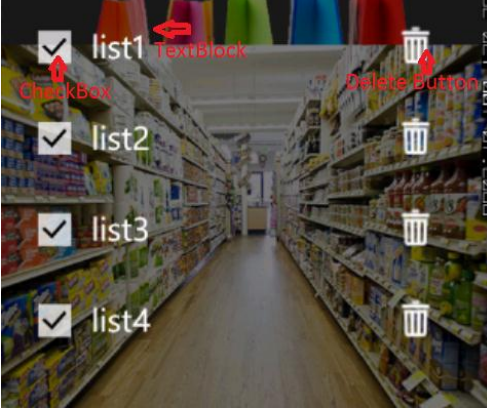
```
      <TextBlock
        Text="{Binding Name}"
        FontSize="{StaticResource PhoneFontSizeLarge}"
        Grid.Column="1" Grid.ColumnSpan="2"
        VerticalAlignment="Top" Margin="-36, 12, 0, 0"/>

      <Button
        Grid.Column="3"
        x:Name="BtnDeleteList"
        BorderThickness="0"
        Margin="0, -18, 0, 0"
        Click="BtnDeleteList_Click">

        <Image
          Source="Icons/appbar.delete.rest1.png"
          Height="75"
          Width="75"/>

      </Button>
    </Grid>
  </DataTemplate>
</phone:PhoneApplicationPage.Resources>
```

Bundan sonraki sayfalarda yapacağımız Binding işlemleri de benzer şekilde olacaktır .görüldüğü gibi verilerimizin önce bir checkbox , daha sonra listenin ismini gösteren (<TextBlock Text="{Binding Name}" ile gösterebilen) bir textblock, bir delete butonu ile devam etmesini sağladık.



Data templateyi oluşturduktan sonra ListBox kodlaması yapalım:

```
<ListBox
    x:Name="alllistItemsListBox"
    ItemsSource="{Binding ProductLists}"
    Margin="12,0,4,0" Width="440"
    ItemTemplate="{StaticResource
    ListItemTemplate}" Grid.ColumnSpan="2" />
```

Böylelikle veri tabanımız ile Kontrol elemanları ListBox vb.' ini ilişkilendirerek(Binding işlemi ile) verilerimizi sunabiliriz. Bu işlemi çoğu sayfada kullanacağız ve hepsinde de ListBox kontrolünü tercih edeceğiz çünkü alışveriş listesi bir listelendirme gerektirmekte, ürünler olsun alış-veriş listesi olsun hepsi liste halinde kullanıcıya sunulmaktadır.

MainPage.xaml.cs kısmında ise aşağıdaki işlemleri yapmalıyız:

İlk olarak kullanıcının yeni liste ekle ApplicationBar butonuna tıkladığında NewListPage ye gitmesi için aşağıdaki kodlamayı yapalım. Bu arada ApplicationBarIcon butonların Click eventine ulaşmak için şunları yaparız:

Xaml tarafında butonla ilgili kısma Click yazdığımızda otomatik olarak bize New Event Handler oluşacaktır.

Alışveriş listelerini silmek için kullanacağımız DeleteList butonu veri tabanında yarattığımız DeleteList metodunu kullanarak seçilen listeyi siler bu listeyi silerken listede bulunan ürünlerde silineceğinden silmeden önce kullanıcıya bir uyarı sunulur.

```
private void BtnDeleteList_Click(object sender, RoutedEventArgs e)
{
    var button = sender as Button;
    if (MessageBox.Show(string.Format("All products of this
shop list will delete.Are you sure ?"), "Confirm Delete",
MessageBoxButton.OKCancel) == MessageBoxResult.OK)
    {
        if (button != null)
        {
            TList ListForDelete = button.DataContext as TList;

            //Eğer liste silinirse listeye ait tüm ürünleri de
            sil.
            foreach (TProduct deleteProduct in
App.View.DBShop.Products)
            {
                if (ListForDelete.Id == deleteProduct._PListId)
                    App.View.DeleteProduct(deleteProduct);
            }

            App.View.DeleteList(ListForDelete);
        }
    }
    this.Focus();
}
```

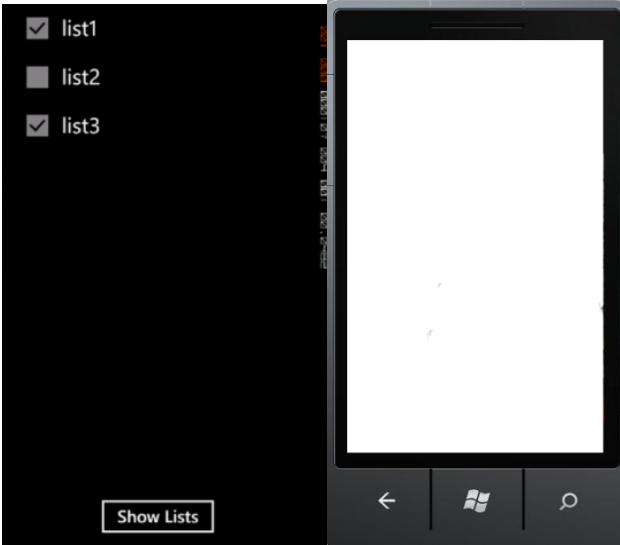
Sender parametresi ile alışveriş listelerinin bulunduğu listede hangi listenin delete butonuna tıklanmış ise o liste seçilip silinebilir.

```
protected override void
OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
    // Save changes to the database.
    App.View.SaveChangesToDB();
}
```

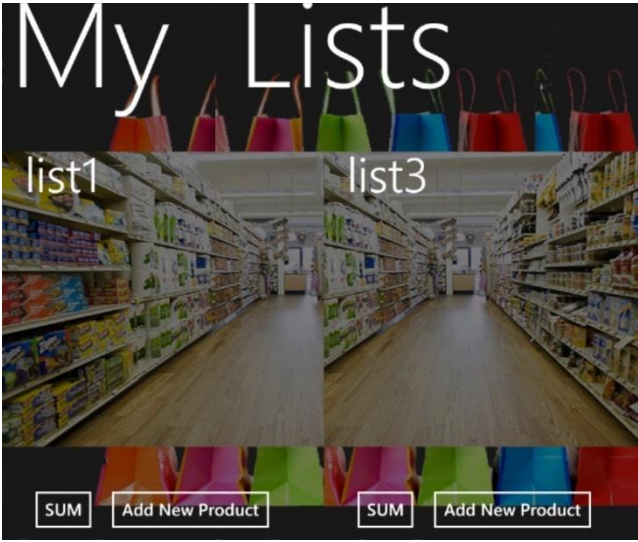
Son olarak On NavigatedFrom() metodu ile yaptığımız değişiklikleri (veri ekleme, silme) veri tabanına yansıtmış oluruz.

4. Panorama Sayfasının Oluřturulması

PanoramaLists sayfası bir *Panorama Page*' dir fakat uygulamada dinamiklięi saęlamak için panorama kontrolleri showlist butonuna basıldıęında gözükür yani panorama sayfasının iç kontrolleri (sayfayı ilerlettikçe kayarak açılan kontroller) en başta gözükmez sayfa tek bir *Portrait Page* görünümündedir dinamik olup kullanıcının seçimine baęlı olarak seçilen sayfalar ShowList butonuna basıldıęında ilgili alanlar açılır ve sayfa tam bir panoramik görünüme ulaşır. Kullanıcı beş listeden üçünü görmek isteyebilir bu durumda PanoramaLists adındaki Panorama sayfasına sadece üç kontrol sayfası eklenir ayrıca Isolated Storegenin kapasitesini aşmayacak şekilde istenilen kadar liste eklenip, kullanıcı belirli bir liste ve her bir liste için sayfa oluşturma sayısı ile sınırlanmaz.



Yukarıdaki görünüm detaylarının gözükmesi istenilen alışveriş listelerinin seçilmesini saęlayan PanoramaLists sayfası görünümüdür şuan için Portrait Page gibidir. Yukarıdaki gibi sadece 1 ve 3 listeler seçilip Show Lists butonuna tıklandıęında ařaęıdaki görünüm oluşur.



Panoramik sayfa mantığını film şeridi gibi düşünebiliriz list1 list3 vb. uzadıkça arka plandaki şerit uzarken telefon içerisinde kayarak ilerler.

İlk resimde gözüken alışveriş listelerini listelemek için yine bir DataTemplate ve bir ListBox kullanalım ve tabii ki veri tabanı ile binding işlemini gerçekleyeceğiz. DataTemplatemiz aşağıdaki gibidir.

```

<phone:PhoneApplicationPage.Resources>
    <DataTemplate x:Key="ListItemTemplate1">

        <Grid HorizontalAlignment="Stretch" Width="420">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="100" />
            </Grid.ColumnDefinitions>

            <CheckBox
                IsChecked="{Binding Checked, Mode=TwoWay}"
                Grid.Column="0" VerticalAlignment="Top"
                Background="Gray"/>

            <TextBlock
                Text="{Binding Name}"
                FontSize="{StaticResource PhoneFontSizeLarge}"
                Grid.Column="1" Grid.ColumnSpan="2"
                VerticalAlignment="Top" Margin="-36, 12, 0, 0"
            />

        </Grid>
    </DataTemplate>
</phone:PhoneApplicationPage.Resources>

```


ListBox için:

```
<ListBox
    x:Name="alllistItemsListBox1"
    ItemsSource="{Binding ProductLists}"
    Margin="12, 0, 12, 0" Width="440"
    ItemTemplate="{StaticResource
ListItemTemplate1}"/>
    <Button Content="Show Lists" Height="72"
HorizontalAlignment="Left" Margin="123,722,0,0"
Name="showlistpanoramaview" VerticalAlignment="Top" Width="189"
Click="showlistpanoramaview_Click" Foreground="White"/>
```

Henüz gözükmeyen panorama kontrolleri ise şu şekildedir:

```
<controls:Panorama Title="My Lists" Name="panorama"
Visibility="Collapsed" Margin="0,0,0,0" >
    <controls:Panorama.Background>
        <ImageBrush ImageSource="Images\back2.png" />
    </controls:Panorama.Background>
</controls:Panorama>
```

PanoramaLists.xaml.cs kısmında ise sadece ShowLists butonunun click eventinde yer alan kodlar mevcuttur bu butona tıklandığında panorama kontrolleri seçilip yüklenmeye başlanır.

```

private void showlistpanoramaview_Click(object sender,
RoutedEventArgs e)
    {
        panorama.Visibility =
System.Windows.Visibility.Visible;
        allListItemsListBox1.Visibility =
System.Windows.Visibility.Collapsed;
        showlistpanoramaview.Visibility =
System.Windows.Visibility.Collapsed;

        var selectedLists = (allListItemsListBox1.ItemsSource
as ObservableCollection<TList>).Where(chain => chain.Checked ==
true).ToList();
        for (var index = 0; index <
selectedLists.Count; index++)
            {
                PanoramaPages chan = new
PanoramaPages(selectedLists[index], index);
                chain.panoNewsItem1.Header =
selectedLists[index].Name;
                panorama.Items.Add(chan);
            }
    }

```

Görüldüğü gibi bu butona tıklandığında artık PanoramaLists adındaki panorama sayfasının kontrolleri oluşturulur. Bunun için panorama değişkeni:

```

<controls:Panorama Title="My Lists" Name="panorama"
Visibility="Collapsed"

```

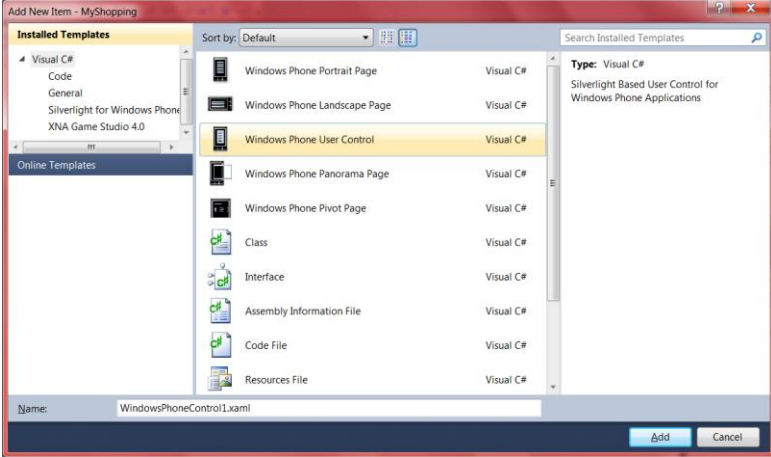
Collapsed' den Visibility hale atanır ve gözükmesi sağlanır. Alış-veriş listelerini listeleyen datatemplate ve binding ile oluşturduğumuz listbox (allListItemsListBox1) ve Show List butonun görünürlüğü ise yok edilir (collapsed).

Alışveriş listelerinin bulunduğu listede kullanıcı tarafından check edilenler(seçilenler) lambda operatörü ile bir metotta test edilir chain(zincir) parametresi ile panorama kontrolü olarak oluşturulup zince eklenir, bu panorama kontrollerinin başlıkları ise selectedLists adlı seçilen alışveriş listelerini içeren diziden alınarak elde edilmiş olup, bu isimler panorama page de tab' lere karşılık gelir.

5. Panorama Sayfası Kontrollerinin Oluşturulması

PanoramPages isimli oluşturacağımız sayfa, daha önce oluşturduğumuz PanoramaLists.xaml adındaki alışveriş sayfalarının listesini içeren Show List butonuna basıldığında ise Panoramik sayfa görünümünü alarak alışveriş listesi başlığı ve altında ürün detaylarını gösterebilen panoramik sayfa ile ilişkilidir. PanoramaPages adındaki yapı bir usercontrol' dür bir sayfa yapısında değildir.

Projemize sağ tıklayarak Add->New Item dediğimizde Windows Phone UserControl' u seçerek oluşturabileceğimiz bir elemandır.



Bu Usercontrol' ün xaml tarafında alış-veriş listesine ait ürünler listelenir ürünlerin listelenmesini yine bir Listbox oluşturup data binding yaparak elde edeceğiz. Öncelikle oluşturduğumuz data templatemiz şu şekildedir.

```

<UserControl.Resources>
  <DataTemplate x:Key="ProductItemTemplate">

    <Grid HorizontalAlignment="Stretch" Width="570">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="50" />
        <ColumnDefinition Width="75" />
        <ColumnDefinition Width="90" />
        <ColumnDefinition Width="90" />
        <ColumnDefinition Width="90" />
      </Grid.ColumnDefinitions>

      <CheckBox
        IsChecked="{Binding IsBuy, Mode=TwoWay}"
        Grid.Column="0" VerticalAlignment="Top"
Background="gray"/>

      <TextBlock
        Text="{Binding ProductName}"
        FontSize="{StaticResource
PhoneFontSizeLarge}"
        Foreground="White"
        Grid.Column="1" Grid.ColumnSpan="2"

        VerticalAlignment="Top" Margin="0, 12, 0, 0"
/>

```

```

<Button
    Grid.Column="3"
    x:Name="BtnEditProduct"
    BorderThickness="0"
    VerticalAlignment="Top"
    HorizontalAlignment="Left"
    Margin="-10, 10, 0, 0"
    Click="BtnEditProduct_Click" Foreground="Gray">
    <Image
        Source="Images/red1.png"
        Height="30"
        Width="30"/>

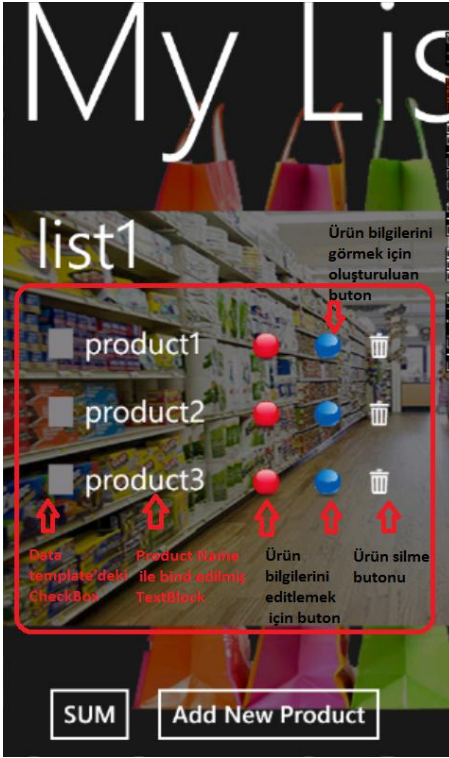
</Button>

<Button
    Grid.Column="4"
    x:Name="BtnShowProduct"
    BorderThickness="0"
    VerticalAlignment="Top"
    HorizontalAlignment="Left"
    Margin="-35, 10, 0, 0"
    Click="BtnShowProduct_Click" Foreground="Gray">

    <Image
        Source="Images/blue1.png"
        Height="30"
        Width="30" />
</Button>

<Button
    Grid.Column="5"
    x:Name="BtnDeleteProduct"
    BorderThickness="0"
    VerticalAlignment="Top"
    HorizontalAlignment="Left"
    Margin="20, 8, 0, 0"
    Click="BtnDeleteProduct_Click">
    <Image
Source="Images/appbar.delete.rest.png"
        Height="30"
        Width="30"/>
    </Button>
</Grid>
</DataTemplate>
</UserControl.Resources>

```



Yukarıdaki resimde data template' de sırası ile oluşturulan tüm kontrollerin örnek bir üzerinde ne amaçlı kullanıldığı gösterilmiştir.

Data template' yi oluşturduktan sonra ListBox oluşumu ve yukarıdaki resimde de görülen SUM ve Add New Product butonlarının oluşumu aşağıdaki gibidir.

```

<ListBox
    x:Name="ProductPanoItemsListBox"
    Margin="12, 0, 12, 0" Width="440"
    ItemTemplate="{StaticResource
ProductItemTemplate}" />
    <Button Content="SUM" Height="78"
HorizontalAlignment="Left" Margin="14,390,0,0" Name="button1"
VerticalAlignment="Top" Width="107" Click="button1_Click"/>
    <Button Content="Add New Product" Height="78"
HorizontalAlignment="Left" Margin="128,390,0,0" Name="button2"
VerticalAlignment="Top" Width="258" Click="button2_Click" />

```

PanoramaPages.xaml.cs tarafında her butunun click eventinde bir işlem yapılmalıdır, ilk olarak PageLoad' a bakacak olursak sayfa yüklendiğinde, oluşturduğumuz veri tabanı ile ilişkili olan yabancı anahtar(foreign key) kısıtlayıcısı ile bir LinqToSql sorgusu gerçekleştirilerek ilgili alışveriş listelerine ait ürünler sıralanır.

```
void PanoramaPages_Loaded(object sender, RoutedEventArgs e)
{
    foreach (TList ShopLists in App.View.DBShop.PLists)
    {
        if (ShopLists.Name == panoNewsItem1.Header.ToString())
            kaynak = ShopLists;
    }
    var query = from TProduct product in
App.View.DBShop.Products join TList listr in App.View.DBShop.PLists on
product._PListId equals listr.Id where product._PListId == kaynak.Id
select product;
    ProductPanoItemsListBox.ItemsSource = query;
}
```

Seçilen ürünün silinmesini sağlayan DeleteProduct butonu object sender parametresi ile ilgili ürünle ilişkilendirilir ve veri tabanında oluşturduğumuz DeleteProduct metodu yardımı ile ürün silinir.

```
private void BtnDeleteProduct_Click(object sender,
RoutedEventArgs e)
{
    var button = sender as Button;
    if (MessageBox.Show(string.Format("This product will
delete.Are you sure ?"), "Confirm Delete", MessageBoxButton.OKCancel)
== MessageBoxResult.OK)
    {
        if (button != null)
        {
            TProduct ProductForDelete =
button.DataContext as TProduct;

            App.View.DeleteProduct(ProductForDelete);
        }
    }
    this.Focus();
}
```

EditProduct butonu seçilen ürün hakkında fiyat mağaza vb. bilgileri girebileceğimiz bir sayfaya bizi yönlendirir, ve bu sayfaya QueryString ile ilgili ürünün Product ID si iletilir.

```
private void BtnEditProduct_Click(object sender, RoutedEventArgs e)
{
    var button = sender as Button;

    TProduct ProductIdForSend = button.DataContext as
TProduct;
    var idvalue = ProductIdForSend.ProductId.ToString();

    var frame = App.Current.RootVisual as
PhoneApplicationFrame;
    frame.Navigate(new
Uri(string.Format("/OrderPage.xaml?id={0}", idvalue),
UriKind.Relative));
}
```

```
private void BtnShowProduct_Click(object sender, RoutedEventArgs e)
{
    var button = sender as Button;
    TProduct ProductIdForSend = button.DataContext as TProduct;
    var id1value = ProductIdForSend.ProductId.ToString();

    var frame1 = App.Current.RootVisual as
PhoneApplicationFrame;
    frame1.Navigate(new
Uri(string.Format("/ProductShowPages.xaml?id1={0}", id1value),
UriKind.Relative));
}
```

ShowProduct butonu ise ürünle ilgili girilen bilgileri tek bir ekranda kullanıcıya sunan ProductShow Page' e yönlendirir ve yine ilgili ürünün Id' si QueryString ile gidilecek olan sayfaya işlemlerde kullanılması için iletilir.

Geriye kalan en altta bulunan iki buton SUM ve Add New Product butonlarının clickleri ve işlevleri aşağıdaki gibidir.

SUM panorama kontrolünün başlığında veri tabanında bulunan listeleri araştırarak aynı isme sahip olan listeleri kaynak olarak alır ve listede bulunan ürünlerin fiyatlarını toplayarak toplam alış veriş maliyetini kullanıcıya MessageBox vasıtası ile gösterir.


```

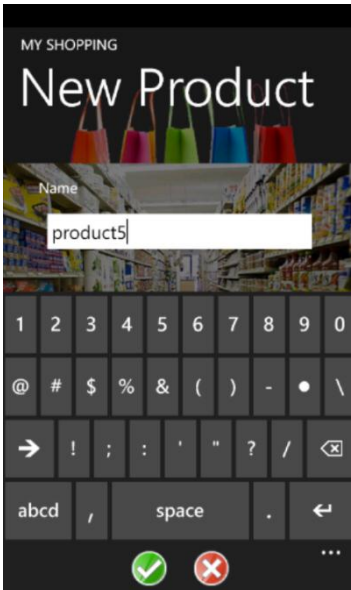
private void button1_Click(object sender, RoutedEventArgs e)
{
    decimal sum = 0;
    foreach (TList ShopLists in App.View.DBShop.PLists)
    {
        if (ShopLists.Name ==
panoNewsItem1.Header.ToString())
            kaynak = ShopLists;

    }
    var query = from TProduct product in
App.View.DBShop.Products join TList listr in App.View.DBShop.PLists
on product._PListId equals listr.Id where product._PListId ==
kaynak.Id select product;
    foreach (TProduct proce in query)
    {
        sum += proce.Price;
    }

    MessageBox.Show(string.Format("Your shopping costs:{0}",
sum));
}

```

Benzer şekilde Add New Product butonu panorama sayfasındaki kontrolün(iç elemanın) sayfa başlığı ile veri tabanındaki listeleri karşılaştırarak aynı isimde olan listeyi bulur ve kaynak olarak atar, bu kaynağın ID sini QueryString parametresi olarak NewProductPage ye ileterek ilgili listeye yeni bir ürün eklenmesini sağlar.



6. Yeni Ürün Oluştur Sayfasının Hazırlanması

NewProductPage, veri tabanına yeni ürün eklememizi sağlar. Veri tabanında Product tablosunda belirlediğimiz ve foreign key olarak kullandığımız ürünün _PListId kolonuna değer atamak ve ürünün hangi liste ile ilişkili olduğunu belirlemek için bir bu sayfaya parametre aktarılır işte bu parametre NewProductPage' nin PageLoad metodunda işlenerek alınır ve gerekli yerlerde kullanılır.

```

void AddNewProductPage_Loaded(object sender, RoutedEventArgs e)
{
    int foo;
    int.TryParse(NavigationContext.QueryString["itemID"],
out foo);
    ListID = foo;
}

```

Yeni ürün oluşturma sayfası bir ürün isminin girilmesi için TextBox ve onay ve iptal için iki adet ApplicationBarItem butonlarından oluşur.

İlgili ürün ismi yazıldıktan sonra onay tuşuna basıldığında ürün veri tabanına eklenir.

```

private void appBarOkButton_Click(object sender, EventArgs e)
{
    if (addnewproductTextBox.Text.Length > 0)
    {
        Foreign key ataması
        // Create a new shop product.
        TProduct newProductItem = new TProduct
        {
            ProductName = addnewproductTextBox.Text,
            _PListId = ListID,
        };

        Yeni ürün oluşumu
        // Add the shop product to the view.
        App.View.AddProduct(newProductItem);

        // Return to the main page.
        if (NavigationService.CanGoBack)
        {
            NavigationService.GoBack();
        }
    }
}

```

Ürün kaydedilmek istenmez ve iptal butonuna basılırsa:

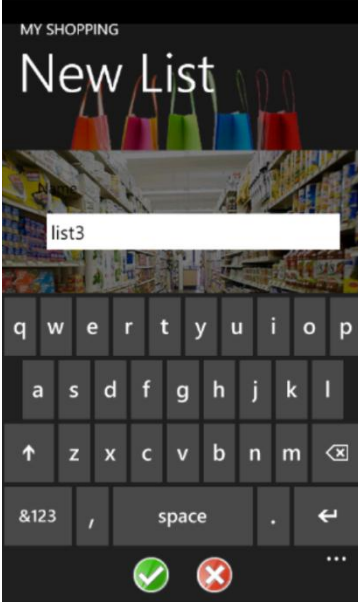
```

private void appBarCancelButton_Click(object sender, EventArgs
e)
{
    // Return to the main page.
    if (NavigationService.CanGoBack)
    {
        NavigationService.GoBack();
    }
}

```

7. Yeni Alış-veriş Listesi Ekle Sayfasının Hazırlanması

NewListPage, kullanıcıya yeni alışveriş listeleri ekleme imkanı sunar. Liste eklemek için sadece liste ismine ihtiyaç vardır. ID si otomatik olarak veri tabanı tarafından Identity özelliği ile üretilir, dolayısı ile bu sayfaya parametre iletmek zorunda değiliz. Sayfamızın tasarımında bir TextBox ve iki adet buton onay ve iptal butonları yer alır.



```
private void appBarOkButton_Click(object sender, EventArgs e)
{
    if (ListNameTextBox.Text.Length > 0)
    {
        // Create a new shoplist
        TList newListItem = new TList
        {
            Name = ListNameTextBox.Text,
        };
    }
}
```

Listenin ismi yazılıp onay butonuna tıkladığında :

```

        // Add the new shoplist to the View.
        App.View.AddList(newListItem);

        // Return to the main page.
        if (NavigationService.CanGoBack)
        {
            NavigationService.GoBack();
        }
    }
}

```

Eğer liste eklenmekten vazgeçilir iptal butonuna tıklanırsa:

```

private void appBarCancelButton_Click(object sender,
EventArgs e)
{
    // Return to the main page.
    if (NavigationService.CanGoBack)
    {
        NavigationService.GoBack();
    }
}

```

8. Ürün Bilgisi Düzenleme Sayfasının Oluşturulması



EditProduct Page, alışveriş listesine eklenen ürünlerin fiyat, alındığı mağaza, sevilen bir ürün kategorisinde olduğu gibi bilgilerin kullanıcı tarafından girilmesini sağlayan sayfadır.

Bu sayfa için oluşturacağımız kontroller sırası ile fiyat girdisi için TextBox, store girdisi için TextBox, sevilen ürün kategorisinde olup olmadığını anlamak için bir CheckBox ve son olarak onay ve iptal için ApplicationBarItem butonlarından ibarettir.

EditProduct Page, New Product sayfasındaki gibi parametre alır ve işler bu parametre ilgili ürünün kendi ID' si yani ProductID' sidir.

```

private void PhoneApplicationPage_Loaded(object sender,
RoutedEventArgs e)
{
    int foo;
    int.TryParse( NavigationContext.QueryString["id"],out
foo);
    idvalue = foo;
}

```

Onay tuşuna basıldığında ise ürün ile ilgili kullanıcı tarafından girilen bilgiler veri tabanına işlenir.

```

private void appBarOkButton_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length > 0)
    {
        TProduct proc=new TProduct();
        foreach (TProduct procxx in
App.View.DBShop.Products)
        {
            if (procxx.ProductId == idvalue)
                proc = procxx;
        }
    }
}

```

```

proc.Price = Convert.ToDecimal(textBox1.Text);

proc.ProductStore = textBox2.Text;

if(checkBox1.IsChecked==true)
proc.IsChance = true;
}

App.View.DBShop.SubmitChanges();
// Return to the main page.
if (NavigationService.CanGoBack)
{
    NavigationService.GoBack();
}
}

```

Alınan product ID ye baęlı olarak ürünler taranarak ilgili ürüne veriler girilir. Eęer ürün bilgileri girilmesinden vazgeçilmiş ise daha önceki sayfalarda ki gibi iptal butonuna basıldığında NavigationService.GoBack(); ile önceki sayfaya geri dönülür.

9. Maęazalar Sayfasının Hazırlanması

Store sayfası yeni maęazaların eklenebileceęi ve kullanıcının ekledięi maęazaları liste halinde görebileceęi bir sayfadır.



Store page için oluşturacaęımız data template yukarıdaki resimden de anlaşılacaęı gibi maęaza ismini tutan bir TextBlock ve bir Delete butondan oluşacaktır. Ayrıca maęaza eklemek için bir ApplicationBarIcon buton vardır.

```

<phone:PhoneApplicationPage.Resources>
  <DataTemplate x:Key="StoreItemTemplate">

    <Grid HorizontalAlignment="Stretch" Width="420">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="100" />
      </Grid.ColumnDefinitions>

      <TextBlock
        Text="{Binding StoreName}"
        FontSize="{StaticResource PhoneFontSizeLarge}"
        Grid.Column="1" Grid.ColumnSpan="2"
        VerticalAlignment="Top" Margin="-36, 12, 0, 0" />

      <Button
        Grid.Column="3"
        x:Name="BtnDeleteStore"
        BorderThickness="0"
        Margin="0, -18, 0, 0"
        Click="BtnDeleteStore_Click">

        <Image
          Source="Icons/appbar.delete.rest1.png"
          Height="75"
          Width="75" />

      </Button>
    </Grid>
  </DataTemplate>
</phone:PhoneApplicationPage.Resources>

```

```

<ListBox
  x:Name="allStoreItemsListBox"
  ItemsSource="{Binding StoreLists}"
  Margin="12,0,4,0" Width="440"
  ItemTemplate="{StaticResource
StoreItemTemplate}" Grid.ColumnSpan="2" />

```

Data Template' ini oluşturduğumuz ve veri tabanında Store tablosu ile Bind edeceğimiz ListBox aşağıdaki gibidir:

Store.xaml.cs tarafında aşağıdaki kodlamaları yapmalıyız:

Yeni mağaza ekle butonuna tıklandığında AddNewStore sayfasına yönlendirme ve DeleteStore butonlarından birine tıklandığında ise ilgili mağazayı veri tabanından silmeliyiz.

```
private void newListAddBarButton_Click(object sender, EventArgs e)
{
    NavigationService.Navigate(new Uri("/AddStorePage.xaml",
    UriKind.Relative));
}

private void BtnDeleteStore_Click(object sender, RoutedEventArgs e)
{
    var button = sender as Button;

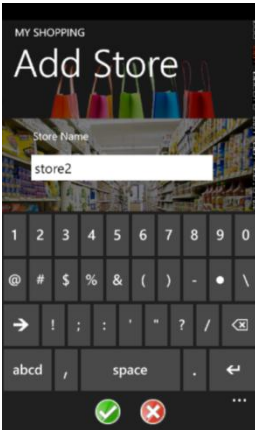
    if (button != null)
    {
        TStore StoreForDelete = button.DataContext as TStore;

        App.View.DeleteStore(StoreForDelete);
    }

    this.Focus();
}
```

10. Mağaza Ekle Sayfasının Oluşturulması

Add Store sayfası veri tabanına kullanıcı tarafından yeni mağazalar ekleme imkanı sunan bir ara yüzdür.



Bu sayfa için bir TextBox(mağaza ismi girdisi için), girilen mağaza ismini onay ya da iptal için iki adet ApplicationBar butonları gerekir.


```

private void newListAddBarButton_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length > 0)
    {
        // Create a new store
        TStore newStoreItem = new TStore
        {
            StoreName = textBox1.Text,
        };

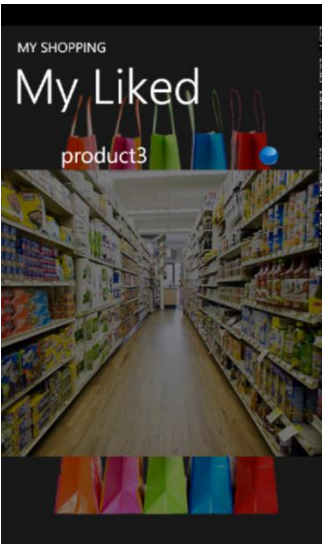
        App.View.AddStore(newStoreItem);

        // Return to the main page.
        if (NavigationService.CanGoBack)
        {
            NavigationService.GoBack();
        }
    }
}

```

Girilen mağaza isminin iptali için `NavigationService.GoBack();` ile önceki sayfaya geçilir.

11. Sevilen Ürünler Sayfasının Hazırlanması



MyLiked sayfası kullanıcıya avantajlı bulduğu (fiyatı uygun ve kaliteli ürünler yada hoşça giden ürünler vb.) ürünleri listeleyebileceği bir sayfadır, projenin sonraki aşamalarında bu ürünleri eğer kullanıcı isterse sosyal ağlarda arkadaş çevresi ile paylaşabileceği bir eklenti yapılabilir.

Bu sayfada da bir listeleme söz konusu olduğu için `ListBox` kullanılır ve data template vardır. Resimde görülen mavi butona basıldığında ürünle ilgili detayların sunulduğu `ShowProduct` sayfasına navigatörlü edilir.

```

<phone:PhoneApplicationPage.Resources>
    <DataTemplate x:Key="LikedItemTemplate">

        <Grid HorizontalAlignment="Stretch" Width="420">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="100" />
            </Grid.ColumnDefinitions>

            <TextBlock
                FontSize="{StaticResource PhoneFontSizeLarge}"
                Text="{Binding ProductName}"
                Grid.Column="1" Grid.ColumnSpan="2"
                VerticalAlignment="Top" Margin="-36, 12, 0,
0"/>

```

Data Template ve ListBox:

```

        <Button
            Grid.Column="3"
            x:Name="BtnGoProduct"
            BorderThickness="0"
            Margin="0, 8, 0, 0"
            Click="BtnGoProduct_Click">

            <Image
                Source="Images/blue1.png"
                Height="30"
                Width="30"/>

        </Button>

    </Grid>
</DataTemplate>
</phone:PhoneApplicationPage.Resources>

```

```
<ListBox
                                x:Name="ChanceItemsListBox"
                                Margin="12,0,4,0" Width="440"
                                Grid.ColumnSpan="2"
                                SelectionChanged="ChanceItemsListBox_SelectionChanged"
                                ItemTemplate="{StaticResource LikedItemTemplate}"/>
```

MyLiked.xaml.cs tarafında:

PageLoad metodunda ilgili ürünlerin sayfa yüklendiğinde kullanıcıya sunulması için bir t-sql sorgusu çalışarak ilgili ürünler bulunur.

```
private void PhoneApplicationPage_Loaded(object sender,
RoutedEventArgs e)
{
    var query1 = from TProduct product in
App.View.DBShop.Products where product.IsChance == true select
product;
    LikedItemsListBox.ItemsSource = query1;
}
```

İlgili ürün bilgilerine ulaşmayı sağlayan butonun click eventinde ise aşağıdaki kodlamaları yapmalıyız.

```
private void BtnGoProduct_Click(object sender, RoutedEventArgs e)
{
    var button = sender as Button;
    TProduct ProductIdForSend = button.DataContext as
TProduct;
    var id1value = ProductIdForSend.ProductId.ToString();

    var frame1 = App.Current.RootVisual as
PhoneApplicationFrame;
    frame1.Navigate(new
Uri(string.Format("/ProductShowPages.xaml?id1={0}", id1value),
UriKind.Relative));
}
```



Bu bölümde, Windows Phone 7.5 ile gelen multitasking mantığını ve nasıl gerçekleştiğini öğrenmeye çalışacağız.

Yazardan...

15. Multitasking

Hızlı ve cevap verebilirliği yüksek bir kullanıcı deneyimi için Windows Phone 7 ön planda sadece bir uygulamanın çalışmasına izin verir. Arkaplanda ise belli sayıda uygulamalar çalışabilir hatta bu uygulamalar ön planda çalışmasalar bile yapacakları işlemleri arka planda yapabilirler. Arka planda çalışabilecek program çeşitleri aşağıdaki gibi gruplandırılabilir.

Arka Planda Yürütülen Müzikler

Kullanıcılar bir müziği seçerek arka planda yürütebilirler, böylece müzik dinlerken mesaj atabilir, oyun oynayabilir veya istedikleri herhangi başka bir uygulamayı aynı anda kullanabilirler.

Planlanmış Görevler

Planlanmış görevler uygulama ön planda koşmasa bile kodlarını icra etmelerine izin verilen görevlerdir. Bu görevleri periyodik görevler ve kaynak gereksinimli görevler olarak iki gruba ayırabiliriz. Periyodik görevler kısa sürelidirler, 30 dakikada bir çalışırlar ve çalışma süreleri 25 saniyedir. Eğer kullanıcı telefonunu az şarj kullanım moduna almış ise periyodik görevler telefonun şarjını tüketmemek için çalışmazlar. Bu görevlere örnek olarak cihazın konumu bilgilerinin güncellenmesi, küçük veri senkronizasyonları verilebilir.

Arka Plan Dosya Transferleri

Arka plan transfer servisi bir uygulamanın çoklu http isteklerini kuyruğa atmasına izin verir, böylece uygulama ön planda çalışmasa bile dosya transferleri devam eder buna dosya indirme ve yükleme dahildir.

Planlanmış Bildirimler

Windows Phone alarmlar ve hatırlatıcılar oluşturulmasına izin verir, böylece alarm yada hatırlatıcıların zamanı geldiğinde ekranda belirerek kullanıcıya bildirim sunulur, kullanıcı bu bildirim üzerinde işlem yapmak isterse üzerine dokunabilir.

Uygulamalar Arası Hızlı Geçişlerin Sağlanması

Windows Phone 7.1 de bir uygulamadan çıkıldığında uygulama sonlanırken 7.5 ile birlikte bu uygulama **dormant** adı verilen bir aşamada tutulmaya başlandı yani uygulama dormant' ta iken tekrar çağırılabilceği düşüncesi ile kullanıcıyı beklemektedir fakat bu süre sınırlıdır ve bu şekilde arkada kullanıcıyı bekleyecek uygulama sayısı da sınırlıdır. Dormant durumunda bulunan uygulamalar hakkındaki uygulama verileri bellekte tutulur fakat bu ortamdaki uygulamalar bir kod icra etmezler kullanıcı tekrar bu uygulamaya dönmek istediğinde uygulamaya dair veriler zaten bellekte olduğu için uygulamalar arası hızlı geçiş sağlanır.



Yazardan...

*Merhaba,
Bu bölümde WP7 uygulamalarınızın YallaApps sitesine gönderim aşamasından sonra tabi tutuldukları uygunluk testlerinden bahsetmek istedim. Bu testlerin içeriği hakkında bilgi sahibi olmak, uygulamalarınızın test aşamasını daha hızlı geçmesini, dolayısıyla daha hızlı görücüye çıkmasını sağlayabilir.
Keyifli okumalar..*

Cem YAMANYILMAZ
Neslişah ÇELİK

16. Marketplace

a. Uygulamaların YallaApps/Marketplace Üzerinde Yayınlanması

Windows Phone Marketplace' e Türkiye' den şimdilik direkt olarak uygulama göndermek mümkün olmadığından, Türkiye' nin de içinde bulunduğu Orta Doğu & Afrika bölgesindeki Windows Phone uygulaması geliştiricileri uygulamalarını YallaApps (<http://www.yallaapps.com>) adlı Dubai merkezli bir şirket aracılığıyla Marketplace' e yüklemeliler. Başka bir deyişle YallaApps yazılım geliştiriciler ve Microsoft arasında bir elçi görevi görüyor. YallaApps bizim yerimize gerekli sertifikasyon sürecini yürütüyor diyebiliriz.

YallaApps' in yapısını daha iyi anlayabilmek için aşağıdaki soruları cevaplandırmakta fayda var.

- YallaApps' e kayıt olabilmenin ilk şartı nedir?

-Orta Doğu & Afrika bölgesinde yaşıyor olmak.

- YallaApps yazılım geliştiricilere teknik bir destek sunuyor mu?

Hayır. YallaApps yazılım geliştiricilere teknik bir destek sunmuyor, ayrıca onlardan son kullanıcıya teknik destek sunmalarını bekliyor.

- Uygulamamızı birden fazla dilde sunabiliyor muyuz?

Evet. Fakat uygulamanın tanımı başvuru sırasında hem uygulamanın ana dilinde hem de İngilizce olarak yazılmalı.

- YallaApps' in fiyatlandırma sistemi nasıl çalışıyor?

YallaApps' e uygulama yükleyebilmeniz için kredi almak zorundasınız.

Sunulan Uygulama	Ücretli Uyg.	Ücretsiz Uyg.
Yeni başvuru	0 Credits	1 Credit
Sertifikasyon reddi sonrası yapılan başvuru	0 Credits	1 Credit
Güncelleme başvurusu	0 Credits	1 Credit

- Öğrenciler ücretsiz kayıt olup, uygulama yükleyebiliyorlar mı?

Evet. Dreamspark üyeliği olan öğrenciler uygulamalarını ücretsiz olarak yükleyebiliyorlar.

- *Bir uygulamayı birden fazla kategoride aynı anda yayınlatabiliyor muyum?*

Hayır. Uygulamanızı sadece tek bir kategoride yayınlatabilirsiniz.

- *Uygulamamı hangi formatta göndermeliyim?*

Uygulamanız **.xap** formatında olmalı.

- *Uygulamamı birden çok markete aynı anda sürebiliyor muyum?*

Evet. Siz aksini belirtmedikçe uygulamanız standart olarak bütün marketlerde satılabilir şekilde yayınlanıyor.

- *Sertifikasyon süreci nedir?*

Uygulamamızın Windows Mobile 7 uygulama yeterliliklerini karşılayıp karşılamadığının kontrol edildiği sürece sertifikasyon deniyor. Mevzubahis yeterlilik şartları için: [http://msdn.microsoft.com/en-us/library/hh184843\(v=VS.92\).aspx](http://msdn.microsoft.com/en-us/library/hh184843(v=VS.92).aspx)

- *Uygulamam sertifikasyon sürecini gecemezse ne oluyor?*

Böyle bir durumda YallaApps size bir hata raporu gönderiyor ve hatalarınızı düzeltmenizi talep ediyor. Bu hataları YallaApps forumlarında tartışarak da çözebilirsiniz.

- *Sertifikasyon sürecini hızlandırmak için ne yapmalıyım?*

1. Uygulama ikon listenizdeki bütün ikonlar(large mobile app icon, small mobile app icon, large PC app icon and background icon) aynı olmalı.
2. İkonlarınız standart WindowsPhone ikonları olmamalı ve transparan olmayan PNG formatında olmalı.
3. Uygulama screenshotlarınızda emulator ekranınızla ilgili şeyler gözükmemeli, yani uygulama içeriği açıkça görülebilir olmalıdır.
4. Geri tuşuna bir önceki sayfayı göstermesini sağlaması dışında başka bir özellik verilmemeli. Örneğin uygulama giriş sayfasındaysa, geri tuşuna basıldığında uygulama kapanmalı.
5. Eğer uygulama İngilizceden başka bir dil de içeriyorsa, bu dilde de detaylı bir tanımlama olmalı.
6. Uygulamanızın teması okunabilir ve anlaşılabilir olmalıdır.

- Uygulamam satıldığında paramı nasıl alacağım?

Satışlardan elde ettiğiniz gelir PayPal hesabınıza yükleniyor.

- Uygulamanın satışından ne kadar para alacağım?

25% Windows Marketplace'e komisyon ödeyeceksiniz. Hesabınızda 400\$ biriktiğinde ya da aylık olarak YallaApps size ödemenizi yapıyor olacak.

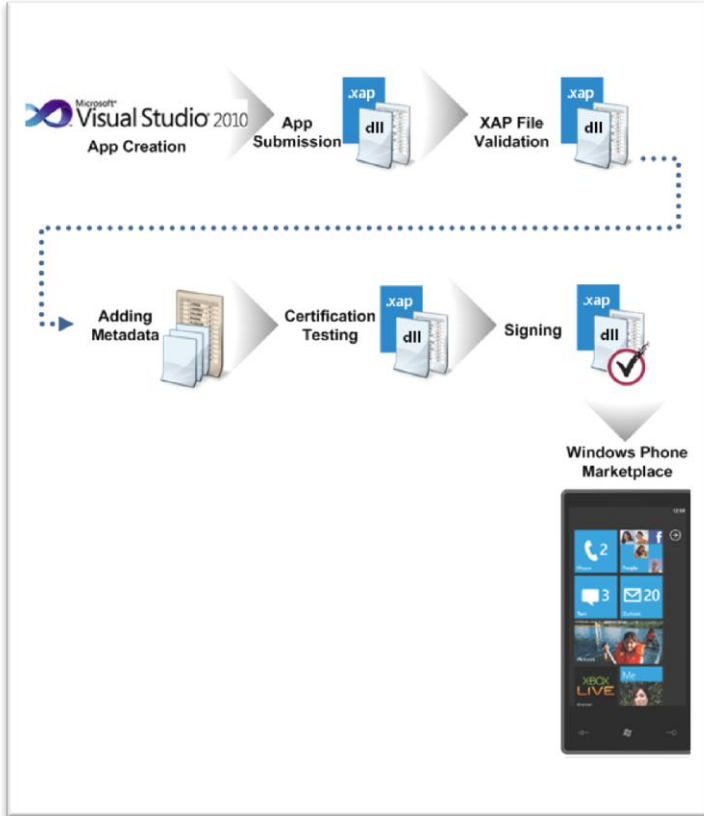
b. MarketPlace Uygunluk testleri

Marketplace' de bir uygulamanın yayınlanması için belirli uygunluk testlerinden geçerek sertifikasyon sürecini tamamlaması gerekmektedir. Bu sürecin tasarımını oluşturan temel prensipler tüm şartların anlaşılabilir, tarafsız ve uygulanabilir olmasıdır. Böylelikle kod geliştiricilerinin kolaylıkla bu süreci tamamlamaları hedeflenmektedir.

Sertifikasyon süreci, uygulamanızın aşağıdaki beş ana kategoriye uyumluluğunu test eder:

- Uygulama Politikaları
- İçerik Politikaları
- Uygulamanın Sunum Gereklilikleri
- Teknik Sertifikasyon Gereklilikleri
- Özel Uygulama Çeşitleri için Ek Gereklilikler

Sertifikasyon sürecinin gidişatı aşağıdaki gibidir:



- 1- App Hub hesabınıza giriş yapın ve yeni bir uygulama yükleme girişi yapın.
- 2- XAP dosyasını yükleyin.
- 3- Uygulama için tanımlayıcı bilgi girin; başlık, açıklama, kategori ve simge gibi. XAP dosyası siz bilgileri girerken onaylanma sürecindedir.
- 4- Eğer onay verilirse; sunum süreci 8. Adıma geçer, onaylanmazsa süreç sona erer ve bir bildirim alırsınız.
- 5- Onaylandıktan sonra XAP dosyanız yeniden paketlenir ve sertifikasyon testi için bir telefona yüklenir. Bu testte yukarıda belirtilen 5 kategorinin olup olmadığı kontrol edilir.
- 6- Uygulama tüm gereklilikleri sağlıyorsa, tekrar oluşturulmuş XAP ve assembly dosyaları onaylanır ve yayına hazır hale gelir.

Uygulama için herhangi bir güncelleştirme yüklerken de yukarıdaki süreçler uygulanır.

Dilerseniz bu konuyla ilgili [http://msdn.microsoft.com/en-us/library/hh184843\(v=VS.92\).aspx](http://msdn.microsoft.com/en-us/library/hh184843(v=VS.92).aspx) adresine de girip göz atabilirsiniz.



Bu bölümde, Windows Phone 7' nin bize gerek uygulama geliştirici, gerek de kullanıcı olarak sağladığı güvenlik özelliklerini inceleyeceğiz. Hem cihazımızın, hem de verilerimizin güvenliği ile ilgili pek çok özelliğin bulunduğu Windows Phone 7' nin, Windows Phone Marketplace' deki sıkı denetimleri, Çember Mimarisi ve "Capability" tabanlı erişim izinleri ile bizi zararlı yazılımlardan etkili bir şekilde nasıl koruduğunu öğreneceğiz. Bunun yanı sıra, Find My Phone özelliği sayesinde Windows Phone 7 işletim sistemine ait cihazımızın internet üzerinden sadece tek bir tık ile nerede olduğunu öğrenip, çalınma durumunda ister bilgilerimizi silip, istersek de telefonu kilitlememiz mümkün.

Bir antivirüs yazılımına ihtiyaç duymayan bu işletim sistemini kullanırken, güvenliğiniz ile ilgili en ufak bir şüphemiz olmayacak.

17. Windows Phone 7'de Güvenlik

Mobil cihazların yaygınlığı ve kullanım alanları her geçen gün artmakta ve bu artış sonucunda bilgilerimizin güvenliği git gide daha önemli bir konu haline gelmekte. Bu durumda gerek cihazımızın, gerekse de verilerimizin kötü amaçlı yazılımlar, web siteleri ve hackerlardan uzak durduğundan emin olmayı istemek ise en doğal hakkımız. Bu fikri göz önüne alarak, başından beri geliştirilmesinde güvenliğin ön planda tutulduğu Windows Phone 7; veri koruma alanındaki yenilikleri ve yaratıcı bir mimariye sahip güvenlik modeli ile telefonun özelliklerini ve kullanılabilirliğini azaltmadan gerekli güvenliği sağlamayı tasarlıyor.

Çember Mimarisini

Windows Phone 7'nin güvenlik modeli, uygulamaların telefonun kaynaklarından yalıtımını ve sıkı bir ayrıcalık kontrolünü ön plana çıkaran “çember” konseptini kullanır. Her çember, bir güvenlik sınırı oluşturur, ve uygulamalar belirlenmiş çemberleri içerisinde bu sınırlara uygun olarak çalıştırılır. Çemberler, bir güvenlik poliçesi sistemi ile bünyesinde çalışan uygulamaların işletim sisteminin hangi özelliklerini kullanabileceklerini belirler.

Toplamda dört adet çember bulunmaktadır. Bu dört çemberin üçünün verdikleri izinler sabittir. Dördüncü çember ise “Capabilities” denilen, gerekli ayrıcalıkların yükleme ve çalışma zamanlarında verilmesini sağlayan bir güvenlik özelliğini kullanır.

Bu dört çember şu şekilde adlandırılır:

- **Trusted Computing Base (TCB):** Bu çember, en çok yetkiye sahip olan çemberdir. Windows Phone 7'nin neredeyse tüm kaynaklarına kısıtlama olmadan erişim imkanı sağlar. Bu çember, güvenlik poliçelerinin değiştirilmesini ve güvenlik modelinin uygulanmasını sağlar. Kernel ve kernel modunda çalışan sürücüler bu çemberde yer alırlar. TCB çemberinde çalışan yazılım miktarının azaltılması, Windows Phone 7'nin potansiyel saldırı yüzeyinin azaltılması için hayati önem taşır.
- **Elevated Rights Chamber (ERC):** Bu çember, güvenlik poliçeleri haricindeki tüm kaynaklara erişebilir. Bu çemberi diğer mobil uygulamaların ihtiyaç duyacağı fonksiyonları sağlayan servisler ve kullanıcı modunda çalışan sürücüler kullanır. ERC, TCB'den daha az yetkiye sahiptir.
- **Standard Rights Chamber (SRC):** Cihaza hazır yüklü gelen uygulamaların çalışması için varsayılan çemberdir. Telefona diğer uygulamaların kullanabileceği bir servis sağlamayan tüm uygulamalar bu çemberde çalıştırılır. Bu çemberde çalışan bir uygulamaya örnek olarak Microsoft Outlook Mobile 2010'u verebiliriz.
- **Least Privileged Chamber (LPC):** Bu çember, Windows Phone 7 Marketplace'den indirilen, Microsoft tarafından geliştirilmeyen tüm uygulamaların çalışacağı varsayılan çemberdir. LPC çemberi her uygulama için Capabilities özellikleri kullanılarak ayarlanır.

Capabilities

Bir “capability”, Windows Phone 7 üzerinde kullanıcı gizliliği, güvenlik, maliyet ya da ticari açılardan kullanımı endişe teşkil edebilecek bir kaynaktır. Bu kaynaklara örnek olarak coğrafi konum bilgisi, kamera, mikrofon ve sensörleri verebiliriz. LPC, varsayılan olarak minimum erişim ayrıcalığı sağlar. Ancak LPC dinamik bir yapıya sahiptir, ve capabilities kullanılarak genişletilebilir. Capabilities, uygulamalar telefona yüklenirken atanır, ve uygulamaların sahip olduğu ayrıcalıklar çalışma esnasında arttırılmaz.

Capability temelli, uygulamalara en az ayrıcalığı verme modeli şu avantajlara sahiptir:

- Saldırı yüzeyi azaltılır. Her uygulama, sadece çalışması için gerekli olan ayrıcalıklara sahiptir. Uygulamalar, kullanmayacakları kaynaklara erişemez.
- Her uygulama, hangi kaynakları kullanacağını kullanıcıya açıkça belirtir.
- Windows Phone Marketplace' de uygulama detayları sayfasında gerekli capabilities gösterilir.
- Uygulamanın satın alımı sırasında, yasal gereksinimleri bulunan ve kullanıcının açıkça iznini gerektiren capabilities varsa kullanıcının onayı alınır.
- Uygulama içerisinde, coğrafi konum bilgisi ilk kez kullanılırken kullanıcıdan onay alınır.

Uygulama geliştiriciler, Windows Phone Developer Tools ile birlikte gelen capability bulma aracını kullanarak kendi uygulamaları için capability listesi oluştururlar. Bu liste, uygulama paketinin içerisindeki uygulama manifestosunda bulunur (WMAAppManifest.xml).

Uygulamalar Arası İletişim ve Multitasking

Windows Phone 7'deki her uygulama, kendisine ait yalıtılmış bir çember içerisinde çalışır, ve bu çemberin sağlayacağı ayrıcalıklar uygulamanın doğru çalışabilmesi için gereken kabiliyetler ile belirlenir. Her uygulamaya, isolated storage dosyaları kullanabilmek gibi bir grup temel izin verilir. Ancak, uygulamaların bulut (cloud) dışarısında birbirleri ile iletişim kurmaları mümkün değildir. Uygulamalar, birbirlerinden tamamen izole edilmişlerdir. Bir uygulama başka bir uygulamanın kullandığı hafızaya (memory) ve dosyalara erişemez.

Ayrıca, Windows Phone Marketplace' de yayınlanan üçüncü parti uygulamalar arka planda çalışmaya devam edemezler. Telefonda başka bir uygulamaya geçiş yapıldığı anda, önceki uygulama o anki durumu kaydedilerek kapatılır. Bu sayede, bir uygulama kullanılmadığı zaman telefonun kaynaklarını kullanamaz veya internet tabanlı servislerle iletişime geçemez.

Uygulama Kurulumu

Uygulama geliştiriciler, Windows Phone Marketplace' de uygulamalarını yayınlamak için öncelikle buraya üye olurlar. Her üyeliğe kabul edilmeden önce bir kimlik kontrolü yapılır. Bütün uygulamalar, VeriSign tarafından dijital olarak imzalanır ve bu imza her uygulama geliştiriciye özeldir. Dijital imzaya sahip olmayan uygulamalar Windows Phone 7'de çalışamaz.

Microsoft dışındaki firmalar tarafından geliştirilen uygulamalar sadece Windows Phone Marketplace üzerinden yüklenebilir. Mobil operatörler bir telefona

önceden altı taneye kadar uygulama yükleyebilirler, ancak bu uygulamaların da öncelikle Microsoft tarafından incelenerek dijital imza sahibi olması gerekir.

Windows Phone Marketplace' de uygulamaların yayınlanması titizlikle yürütülen bir işlemdir. Bu işleme uygulama geliştiricinin kayıt esnasındaki incelenmesi, uygulamanın teknik ve içeriksel incelenmesi, ve Windows Phone Marketplace ve genel uygulama davranış kurallarına uyumluluğunun kontrol edilmesi dâhildir.

Windows Phone 7 güvenlik modelinin en az ayrıcalık ve izolasyon mantığının yanı sıra sadece managed-code kullanılması da güvenliği arttıran bir başka etkidir. Eğer uygulamalarda bir açık tespit edilirse, bir takım güncelleştirmeler ile, ya da en kötü durumda, uygulamanın tamamen kaldırılmasıyla bu durum düzeltilir.

Windows Phone 7 kullanıcılarına indirilmeye hazır bir güncelleştirmeler mesajlarla bildirilir. Kullanıcılar isterlerse bu güncelleştirme bildirimlerini kapatabilirler.

Veri Korunumu

Windows Phone 7'nin tasarımı verilerin korunması ile ilgili pek çok özellik içermektedir. Temel olarak Windows Phone 7 bir PIN kilidi ve ilgili EAS (Exchange ActiveSync) şifre poliçeleri ile korunabilir.

Veri korunumu için kullanılan diğer yöntemler:

- **İletim halindeki veri şifrelenir.** Windows Phone 7, bütün verileri Secure Socket Layer (SSL) kullanarak şifreler. Bağlı olan sunucuya göre şifrelemede kullanılan anahtar 128-bit ya da 256-bit olabilir. Endüstri standartlarını sağlaması Windows Phone 7'nin gerek yerel, gerekse de bulut tabanlı servislere (Exchange Server ya da SharePoint Server gibi) bağlanabilmesini sağlar.
- **PC'den dosya sistemine erişim yoktur.** Microsoft Zune Software, Windows Phone 7 ile Windows işletim sistemli bilgisayarlar arasında bağlantı kuran yazılım olarak, bilgisayardan telefonun dosya sistemine erişim sağlamamaktadır. Bu sayede telefonda dosya kopyalamak ya da silmek mümkün değildir. Zune yazılımı sadece müzik, resim ve video gibi medya dosyalarını bir PC ile senkronize etmekte kullanılabilir. Windows Phone 7 RAPI kullanımını desteklemez.
- **Kaldırılabilir veri depolama aygıtları desteklenmez.** Güvenlik risklerini daha da azaltmak için, Windows Phone 7 verilerin telefon dışına aktarılmasını engellemek için kaldırılabilir veri depolama aygıtlarını desteklemez. Windows Phone 7 cihazları en az 8 gigabyte hafızaya sahiptir. Eğer bir Windows Phone 7 cihazına SD kart takılırsa, bu SD kart kilitletir. SD kartı kilitlemek için, 128-bitlik bir anahtar kullanılır ve bu anahtar telefonun içerisindeki hafızada kart ile telefonu özgün bir şekilde eşleştirir. Sonuç olarak, eğer bu kart telefonda çıkartılırsa, SD karta doğru 128-bitlik şifre sağlanmadığı sürece erişim engellenir. Bu şekilde bir Windows Phone ile eşleştirilen SD kartlar başka telefonlarda ya da bilgisayarlarda kullanılamazlar.

Microsoft Exchange ActiveSync (EAS)

EAS, Windows Phone 7 kullanıcılarına dünyadaki en iyi email senkronizasyon servislerinden birini sunan güçlü bir iletişim protokolüdür. Windows Phone 7 telefonları EAS protokolünün 14.0 versiyonuna sahiptir.

EAS, güvenlik için, işletim sistemleri ve uygulamaların kullandıkları grup poliçelerine benzer bir şekilde, Windows Phone 7 kullanıcıları için güvenlikle ilişkili poliçelerin kullanımına ve yönetimine imkân sağlar. EAS'nin güvenlikle ilgili poliçeleri, IT departmanları tarafından düzenlenebilir. Bu poliçeler sayesinde:

- Telefonun mail, takvim ve rehber bilgilerini Microsoft Exchange Server ile senkronize etmeden, kullanıcının telefona bir PIN kodu ataması,
- Kullanıcının belirli periyodlarla PIN kodunu yenilemesi,
- Kullanıcının aynı PIN kodunu tekrar kullanmasının engellenmesi,
- Basit PIN kodlarının (Ör: 1111) kullanımının engellenmesi,
- PIN kodunun minimum uzunluğunun belirlenmesi,
- Telefon kullanılmadığında kilitlemesi için gerekli zamanın belirlenmesi,
- Telefonun içeriğinin silinip fabrika ayarlarına geri dönmesi için kaç kez PIN kodunun yanlış girilmesi gerektiğinin belirlenmesi,

gibi düzenlemeler yapılabilir.

Ayrıca, Microsoft Outlook Web App kullanarak ya da bir Exchange yöneticisi aracılığıyla, telefonun uzaktan içeriğinin silinip fabrika ayarlarına döndürülmesi de (Remote Device Wipe) mümkündür.



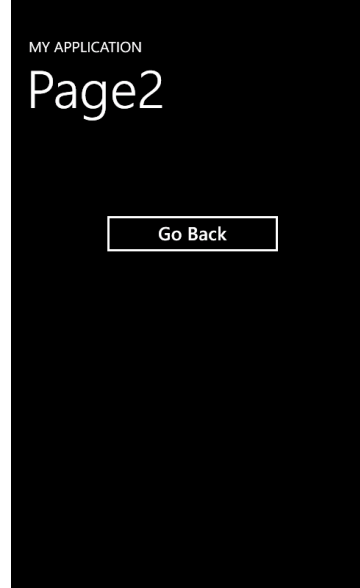
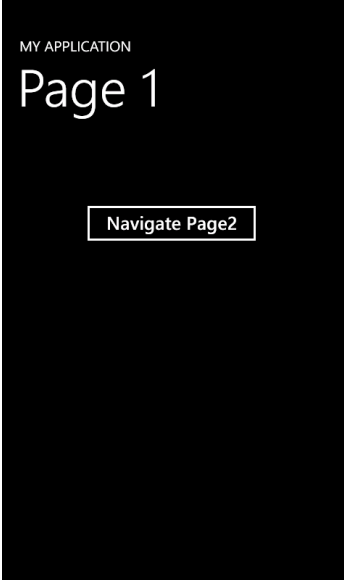
Kitabımızın bu bölümünde ise Windows Phone uygulamalarında en çok ihtiyaç duyulan konulardan biri sayfalar arası geçişe değiniyor olacağız. Ayrıca sayfalar arasında bu geçişi sağlarken veri transferinin nasıl yapıldığını ve hangi yöntemlerin kullanılıyor olduğunu inceliyor olacağız.

Yazardan...

18. Sayfalar Arası Geçiş ve Veri Transferi

Windows Phone uygulaması geliştirirken uygulamalar birden fazla sayfa içerebilir ve bunlar arasında geçiş yapılabilir. Bu geçiş işlemleri için `NavigationService` sınıfını kullanılmaktadır. Uygulamamız için `NavigationService` sınıfının bir nesnesi aynı isimle oluşturulmuştur.

`Page1.xaml` ve `Page2.xaml` isimli iki sayfanın bulunduğu bir projede sayfalar arası geçiş işlemini sağlamak için `NavigationService` sınıfının `Navigate` Metodunu kullanılmaktadır.



```
private void btn_NavigatePage2_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/Page2.xaml",
UriKind.Relative));
}
```

`Page1.xaml` içindeki `btn_NavigatePage2` butonun `Click` Event'inde `Navigate` metodu kullanılmaktadır. `Navigate` Metodu içine bir `Uri` nesnesi almaktadır. `Uri` nesnesi oluşturulurken `string` tipinde yönlendirme yapılacak olan sayfanın ismi ve `Uri`'nin tipi belirlenir.

`Uri` tipinin `Relative` olarak yapılmasının sebebi ise bu `Uri`'nin uygulama içinde bir yönlendirme için kullanılacağını belirtir.

`Page2.xaml` içinden `Page1.xaml`'a geri dönüş için herhangi bir kod yazılmadığı takdirde `Back` butonu ile geri dönebilir. Ancak sayfa sayısının ikiden fazla olduğu projelerde bazı yönlendirmeler kod ile yapılmaktadır.

```
private void btn_GoBackPage1_Click(object sender, RoutedEventArgs e)
{
    NavigationService.GoBack();
}
```

Page2.xaml içindeki btn_GoBackPage2 isimli butonun Click Event' inde Navigation sınıfının GoBack() metodu kullanılmıştır. GoBack metodu sayfa tarihinde bulunan bir önceki sayfaya geri dönmektedir. NavigaitonService sınıfı içinde bulunan GoForward metoduna ise dikkat edilmelidir. Silverlight içinde kullanılabilen GoForward metodu Windows Phone için desteklenmemektedir. Bu metodu kullandığını uygualama expectation atacaktır.

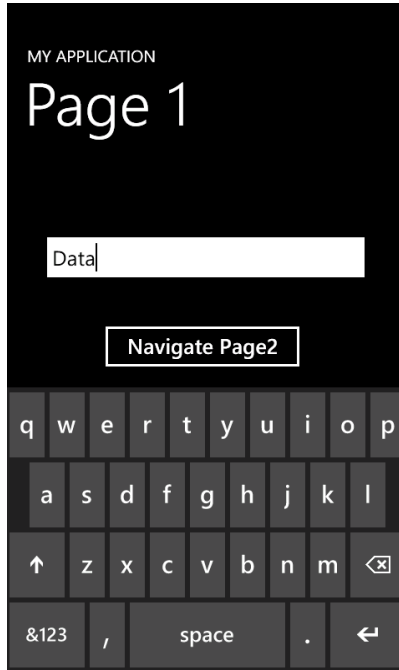
GoBack() ve GoForward() metodlarının kullanmadan önce NavigationService sınıfının CanGoBack ve CanGoForward özelliklerinden yararlanabilirsiniz.

```
private void btn_GoBackPage1_Click(object sender,
RoutedEventArgs e)
{
    if (NavigationService.CanGoBack)
    {
        NavigationService.GoBack();
    }
}
```

Sayfalar arasında geçiş yaparken veri taşımak zorunda kalınabilir. Bu durumlarda birkaç farklı yöntem izlenebilir. Uygulamanın yapısına göre istenilen bir yöntem kullanılabilir.

Sayfalar arası veri transferinde QueryString en yaygın olarak kullanılan yöntemdir. ASP.NET' teki bir Query oluşturulur ve veri diğer sayfada anahtar isimleri ile QueryString' ten alınır.

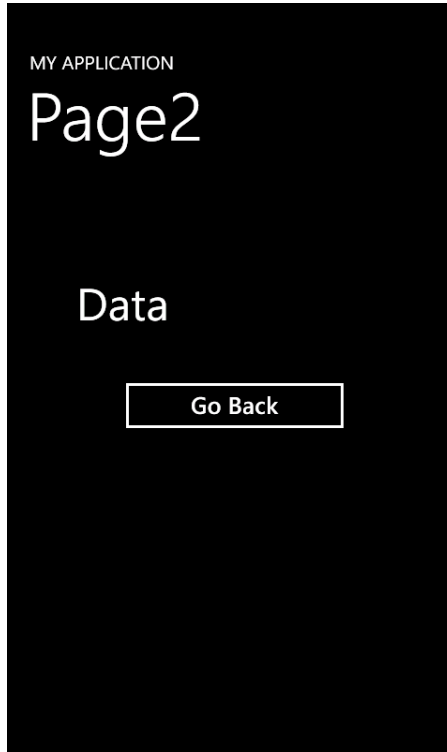
Örneğin; Page1.xaml sayfasındaki bir TextBox' a yazılan değer Page2.xaml' ın içindeki bir TextBlock' un Text' ine aktarılabilir.



```
private void btn_NavigatePage2_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new
    Uri(String.Format("/Page2.xaml?Query={0}", tbx_ForQuery.Text),
    UriKind.Relative));
}
```

Bilindiği gibi Navigite metodu kullanırken metoda parametre olarak bir Uri verilmektedir. Query' i Uri oluşturulurken kullanılan string ifade ile ayarlanmaktadır. "Page2.xaml" yazıldıktan sonra her soru işareti kullandığımızda Query String' e yeni bir değer eklenmektedir. Bu örnekte; Anahtar "Query" olan bir değer QueryString' e eklenmiştir.

QueryString' e eklenen veri uygulamaya göre istenilen bir metodun içine alınabilmektedir. Bu örnekte; veri Page2.xaml' ın Loaded eventi içinde alınmıştır.



```
private void Page2_Loaded(object sender, RoutedEventArgs e)
{
    if (NavigationContext.QueryString.Count!=0)
    {
        tbl_QueryText.Text =
        NavigationContext.QueryString["Query"];
    }
}
```

Bir sayfadan diğerine geçildiğine QueryString içinde bulunan değerlere sayfanın NavigationContext özelliğindeki QueryString ile ulaşılabilir.

İlk sayfadan yollanılan veriye erişmeden önce QueryString içinde herhangi bir veri olup olmadığını kontrol etmekte fayda vardır. Aynı sayfaya farklı metotlardan veri gönderilmeksizin yönlendirme yapılabilir.

[“Anahtar”] şeklinde Query’ i içinden verilen anahtar ile veri alınırken ilk sayfadan anahtarın yazılışına dikkat edilmelidir. “Data”, ”DATA” farklı anahtarlardır. Ayrıca QueryString ile taşıdığımız değerler string olarak gelmektedir. Kullanım yerine göre veri tipleri arasında dönüşüm yapmanız gerekebilir.

QueryString dışında sayfaları veri transferi için birkaç yöntem daha bulunmaktadır. Kelime karşılığı olarak parça olarak geçen Fragment metodunu da kullanabilirsiniz.

Aynı Page1.xaml ve Page2.xaml sayfaları arasında veri transferini Fragment ile gerçekleştirmek istenildiğinde birkaç yerde değişiklik yapmak yeterli olacaktır.

```
private void btn_NavigatePage2_Click(object sender,
RoutedEventArgs e)
{
    NavigationService.Navigate(new
Uri(String.Format("/Page2.xaml#{0}", tbx_ForFragment.Text),
UriKind.Relative));
}
```

Aynı QueryString' ten farklı olarak "?" karakteri yerine "#" işareti kullanılmıştır. Ancak dikkat edilmesi gereken en önemli nokta Fragment ile tek bir veri taşınmaktadır ve herhangi bir anahtara ihtiyaç duymamaktadır. "#" karakterinden sonra gelen tüm karakterler fragment a atanır.

İkinci sayfada veri alınmak istenildiğinde QueryString' den farklı olarak Fragment' ı her yerden alamayız. Bunun için OnFragmentNavigation metodunu override etmemiz gerekmektedir.

```
protected override void
OnFragmentNavigation(System.Windows.Navigation.FragmentNavigationEve
ntArgs e)
{
    base.OnFragmentNavigation(e);
    tbl_Fragment.Text = e.Fragment;
}
```

Event argümanının Fragment özelliği ile diğer sayfadan yollanan veri alınmaktadır. QueryString'e benzer olarak burada da alınan değerın string tipinde olduğu unutulmamalıdır.

Veri transferi için statik değişkenleri de kullanabilmektedir. App sınıfının içinde statik bir değişken oluşturup veri transferi sırasında bu değişken kullanılabilir. Statik bu değişkeni uygulama için yazılmış olan bir sınıfta da kullanmak mümkündür.

```
public partial class App : Application
{
    /// <summary>
    /// Provides easy access to the root frame of the Phone
Application.
    /// </summary>
    /// <returns>The root frame of the Phone
Application.</returns>
    public PhoneApplicationFrame RootFrame { get; private set; }

    public static string Data { get; set; }

    /// <summary>
```

```
/// Constructor for the Application object.  
/// </summary>  
public App()  
{  
    // Global handler for uncaught exceptions.  
    UnhandledException += Application_UnhandledException;  
  
    // Standard Silverlight initialization  
    InitializeComponent();  
}
```

İlk sayfadan geçiş işlemi yapılmadan önce istenilen değer statik değişkene atılır ve diğer sayfada bu statik değişkenden alınır.

```
private void btn_NavigatePage2_Click(object sender, RoutedEventArgs e)  
{  
  
    App.Data = tbx_ForStatic.Text;  
    NavigationService.Navigate(new Uri("/Page2.xaml",  
    UriKind.Relative));  
  
}  
  
private void Page2_Loaded(object sender, RoutedEventArgs e)  
{  
    tbl_Static.Text = App.Data;  
}
```



Bu bölümde Windows Azure ve SQL Azure kavramlarından yüzeysel olarak bahsedip ardından bu servisleri Visual Studio 2010 ortamında Windows Phone 7 programlamamız da kullanabilmemiz için gereken kurulumlara değiniyor olacağız.

Yazardan...

19. Cloud ile İşlemler

Bir Application Tile programladınız ya da bir Notification Service kullanıyorsunuz, işte bu nokta da size yardımcı olabilecek bir çözümü bu kısımda bulabilirsiniz. Sadece Application Tile ya da Notification Service' lerle sınırlamayıp daha da basite indirelim isterseniz. Şöyle düşünün, öyle bir uygulama yazdınız ki, harika ötesi! Herkesin indirip güzel paralar kazanacağınız bir uygulama; fakat uygulamanızın boyutu yüksek, “Benim telefonumda yeterli alan yok” ya da “3G ile bağlanıyorum bunun için kotamı dolduramam.” diyen kullanıcıların indirmeden vazgeçmesini istemezsiniz sanırım. İşte bu nokta da yardımımıza Windows Azure ve SQL Azure yetişiyor. Peki nedir bu Windows Azure, SQL Azure.



Windows® Azure™



Uygulamalarınıza, dosyalarınıza, arşivlerinize, web sitelerinize ve benzeri bir çok veriye dünyanın herhangi bir yerinden kullandığınız farklı bilgisayarlar üzerinden erişebildiğiniz bir sistem. Bunun için ise herhangi bir uygulama ya da kurulum gibi bir şey ihtiyacınızın yok. Bütün bunlar bize bulut bilişim hakkında fikir verirken; Windows Azure ise Microsoft tarafından geliştirilen bir bulut bilişim servisedir.

Daha da genişletecek olursak; Windows Azure basit bir işletim sistemidir. Bu sistem üzerinden bulut bilişim deyiminin kapsadığı olayları gerçekleştirmemizi sağlar. Kapsadığı kavramlar içerisinde yazılım, platform, güvenlik ve dosya erişimi vardır.

Peki bu Windows Azure nerede? Dosyalarımızı Windows Azure' a attık; onlar nereye gidiyor.



Dünya üzerinde Microsoft' un pek çok veri merkezi bulunmakta. Ve yanda görmüş olduğunuz bunlardan sadece bir tanesi. Hepsi de olası bir hata, çökme durumunda eş zamanlı olarak otomatik yedeğinin alındığı bir sistem. Yandaki veri merkezinin içinde milyonlarca konteyner ve bu konteynerların içerisinde ise milyarlarca

bilgisayar bulunmakta.

Yandaki resimde ise bu veri merkezinin içindeki bir konteynerın içerden görüntüsü...



SQL Azure ise, SQL Server veritabanının bir bulut hizmeti olarak genel ağ(internet) üzerinden sunulmasıdır. Windows Azure Platformunun bir parçasıdır.

Bu kavramları anladığımızı varsayarak, Windows Phone 7 üzerinde yazdığımız uygulamaları bu sistemleri kullanarak nasıl programlayacağımızı görelim.

Öncelikle Azure' un da Windows Phone gibi bir SDK(Software Development Kit)' i mevcut ve Azure kurulumumuzu yapmadan önce, <http://www.microsoft.com/download/en/details.aspx?id=15658> adresinden Visual Studio 2010 için Azure SDK' sını kurmamız lazım. Ardından ise Azure Toolkit' ini indirip kurduktan sonra örnek uygulamamızı yapıp bu bölüme noktayı koyacağız.

Download Center

Windows Azure SDK and Windows Azure Tools for Microsoft Visual Studio (March 2011)

Quick links

- Overview
- System requirements
- Instructions

Looking for support?

Visit the Microsoft Support site now >

Microsoft

Try Windows Azure free >

Build, host, and scale applications. In the cloud.

Windows Azure

Quick details

Version: 1.4.20227.1419 Date published: 4/9/2011

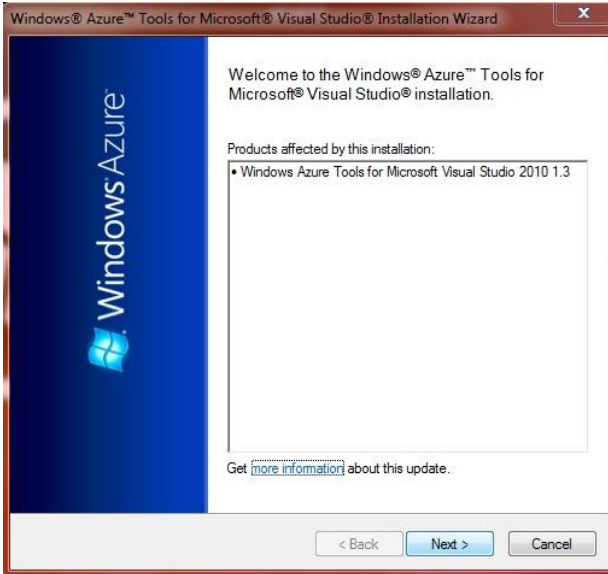
Change language: English

Files in this download

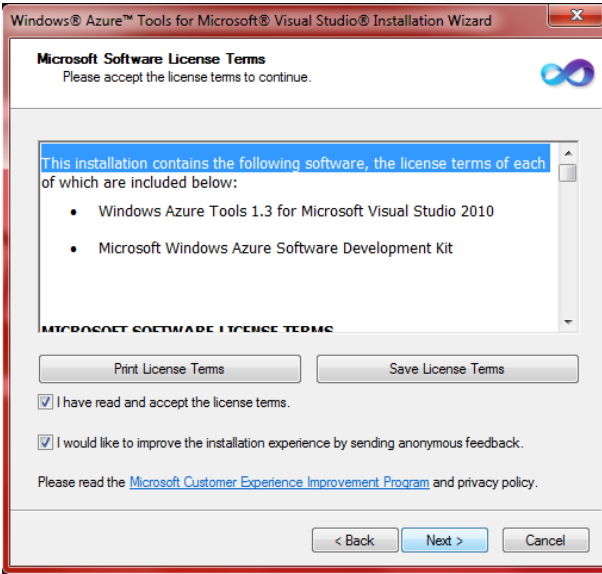
The links in this section correspond to files available for this download. Download the files appropriate for you.

File name	Size	
VSCloudService.exe	18.9 MB	DOWNLOAD
WindowsAzureSDK.chm	4.8 MB	DOWNLOAD
WindowsAzureSDK-x64.exe	9.0 MB	DOWNLOAD
WindowsAzureSDK-x86.exe	9.1 MB	DOWNLOAD

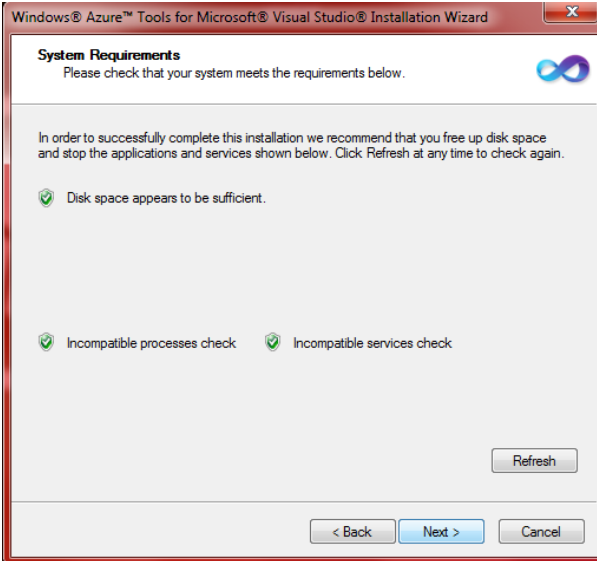
VSCloudService.exe ve WindowsAzureSDK(x64 ya da x86 hangi işletim sistemine sahipseniz) adlı dosyaları indirip VSCloudService.exe yi çalıştırılm.



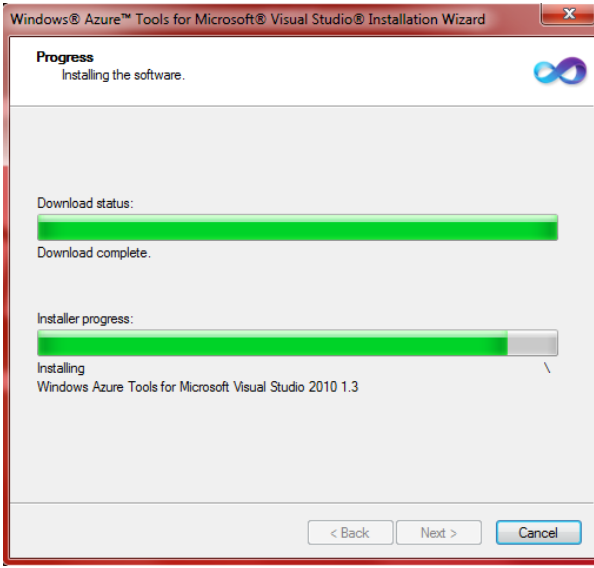
Karşımıza gelen bu pencerede Next deyiş ilerliyoruz.



Ardından “I have read and accept the license terms” u ve altındaki check box’ ı işaretliyoruz. Tabi yukardaki lisans koşullarımızı okuduğumuzu varsayıyorum. Ardından next deyip bu kısımda geçiyoruz.

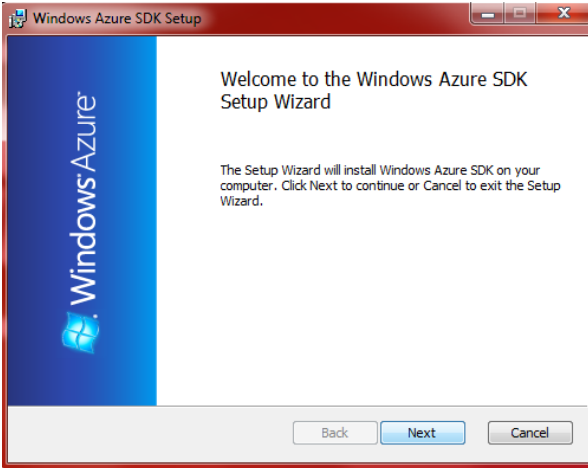


Kurulum bilgisayarınızın durumuna bakıyor bu kısımda, eğer Visual Studio açık ise onu kapatmanızı istiyor. Aynı zamanda yeterli alana sahip olmalısınız. Burayı da Next deyip ilerliyoruz.

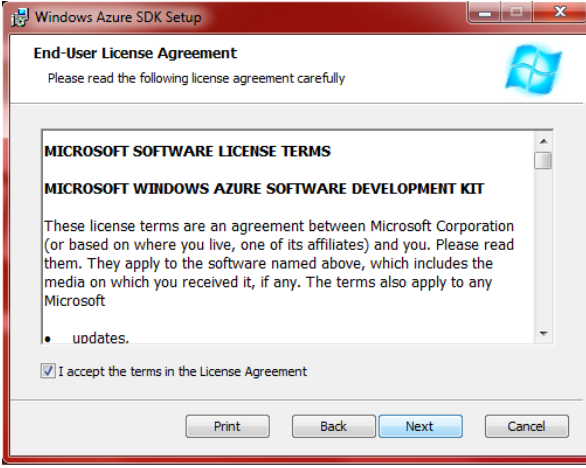


Kurulum başladı.

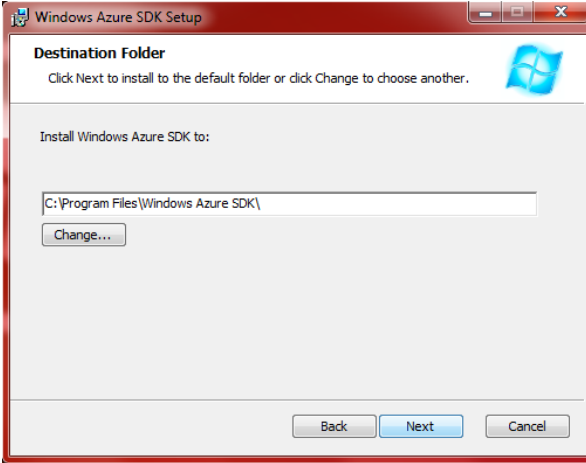
İşlemler tamamlandıktan sonra diğer indirdiğimiz Windows Azure SDK' yı kuralım.



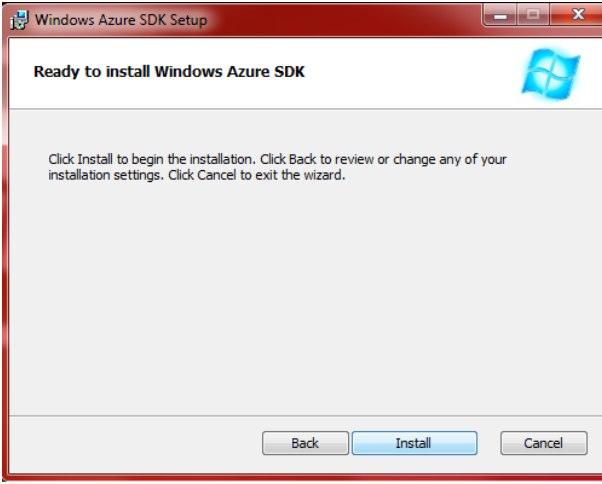
Yine Next deyip hemen geçiyoruz.



“I accept the terms of the License Agreement” i işaretleyerek lisans sözleşmesini kabul ettiğimizi söyleyip Next diyoruz.



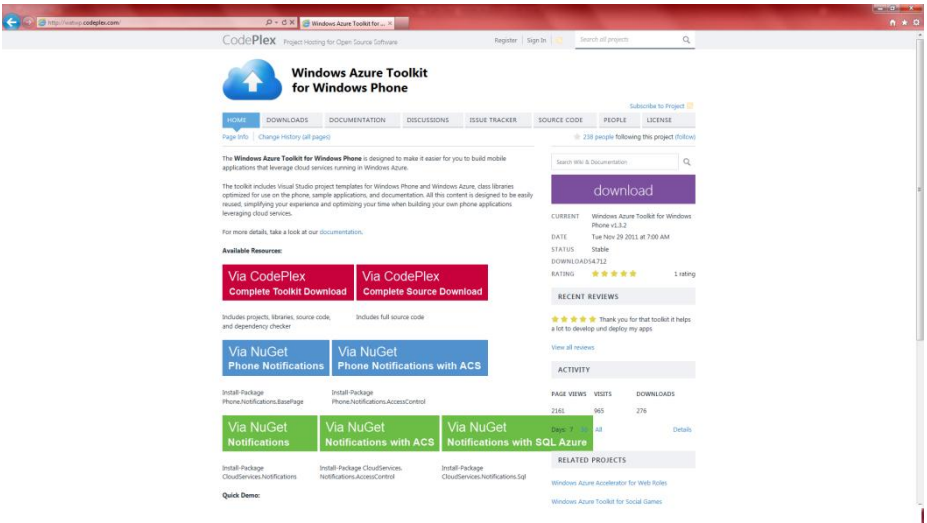
SDK'nın kurulmasını istediğiniz yolu veriyoruz. Burayı hiç değiştirmeden devam edebilirsiniz. Yine Next diyerek devam ediyoruz.



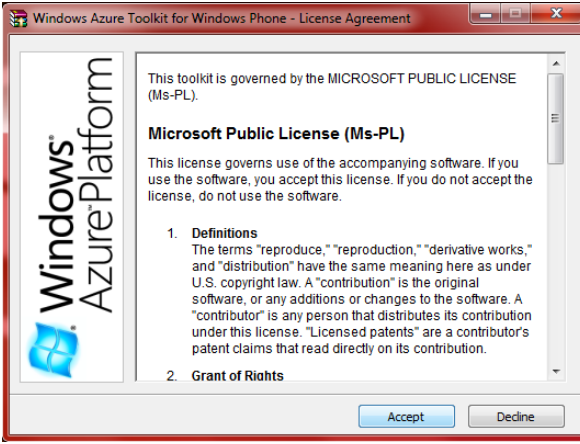
Son olarak Install diyerek kurulumu başlatıyoruz ve artık Windows Azure' da proje geliştirebiliriz.

Gelelim Windows Phone 7 için Azure Toolkit' ine. Bizim asıl konumuz olan Windows Phone 7' de Cloud servislerini kullanabilmemize olanak sağlayacak olan bu kiti, Codeplex' den indirebilirsiniz. Azure' un diğer kaynaklarına ulaşmak için yine Codeplex' ten

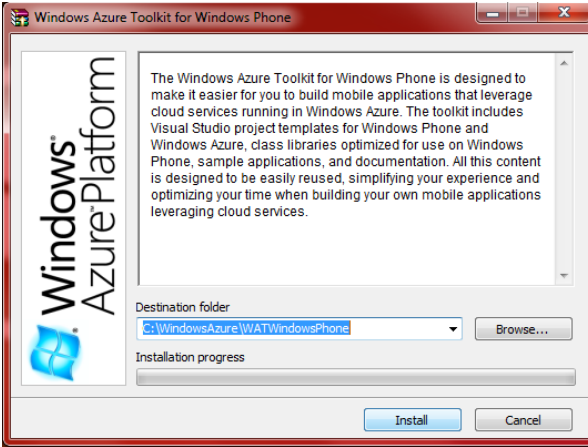
ulaşabilirsiniz dipnotunu da verdikten sonra isterseniz kurulumu geçelim:



<http://watwp.codeplex.com/> adresine giriyoruz ve Windows Azure Toolkit for Windows Phone' u indiriyoruz. Şunları söylemekte fayda var; indirme işlemleri için bu adresleri ezberlemenize gerek yok, Bing' e ilgili anahtar kelimeleri girdiğinizde yine karşınıza bu sayfaların çıkacağından emin olabilirsiniz. Bu sırada gerekli olan bütün dosyaları CodePlex' ten indirebilirsiniz.

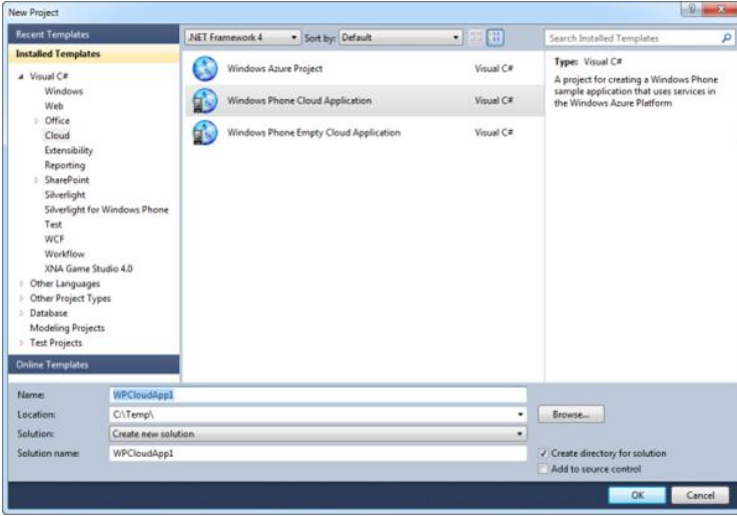


İndirdiğimiz kurulumu çalıştırarak "Accept" diyerek kabul ediyoruz ve ardından,



Kurulacak yerin yolunda değişiklik yapmayacasınız Install diyerek kurulumu bitiriyoruz.

Kurulum işlemi tamamlandıktan sonra Visual Studio 2010' u açarak File > New Project diyerek, şablonlardan Cloud' u seçerek Windows Phone Cloud Application' a tıklıyoruz ve aşağıda uygulamamızın ismini yazıyoruz. Biz WPCloudApp1 diyerek devam edelim.



Hemen ardından depolama birimimizi seçiyoruz. Windows Azure Storage ve SQL Azure Database olmak üzere iki seçeneğimiz var ve burda SQL Azure tek bir veritabanı olduğu için biz şimdi Windows Azure Storage' ı seçiyoruz.



Ardından karşımıza gelen pencerede www.windowsazure.com adresinden full ya da deneme sürümü hesabımızın bilgilerini giriyoruz.

New Windows Phone Cloud Application Project

Enter your Windows Azure storage account information

Account name:

Account key:

Use Storage Emulator Use HTTPS

< Prev Next > Cancel

Sadece tek bir database işimizi görecekse eğer SQL Azure Database' ı seçebiliriz. Eğer SQL Azure Database' ı seçseydik karşımızda şu şekilde bir pencere gelecekti:

New Windows Phone Cloud Application Project

Enter your SQL Azure Database Server information

Server Name

User

Password

[How to create a SQL Azure Database Server?](#)

Use local SQL Server instance

< Prev Next > Cancel

Ardından Push Notification servislerinden Microsoft Push Notification' ı seçiyoruz. Burada Microsoft' un dışında diğer firmalarında servisleri destekleniyor.

New Windows Phone Cloud Application Project

Choose the Push Notification Services you want to support

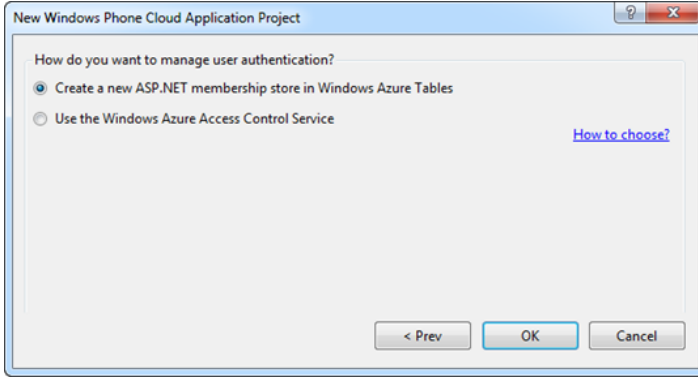
Microsoft Push Notification Service

Apple Push Notification Service

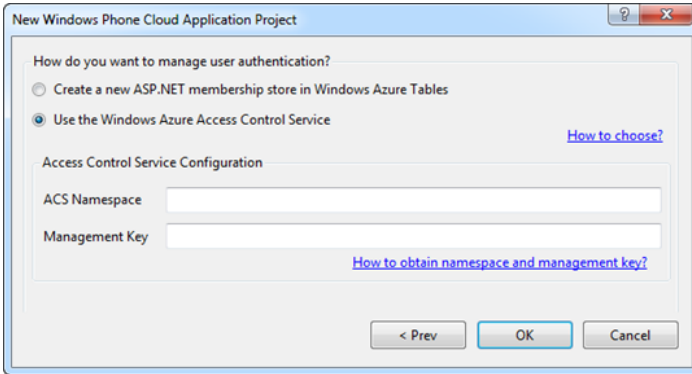
Android Push Notification Service

< Prev Next > Cancel

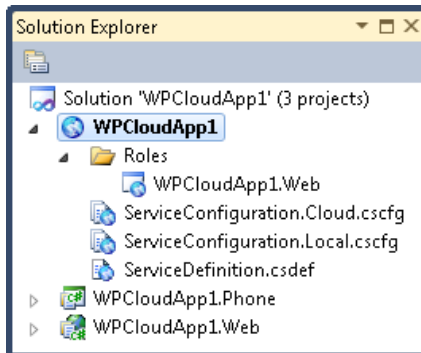
Şimdi ise kimlik doğrulama için Windows Azure’ da Asp.Net üyeliği yaratmasını seçiyoruz.



Diğer seçeneği seçmiş olsaydık, Windows Azure hesabınızdan edindiğiniz ACS Hizmet Alan Adı ve yönetim anahtarını girmemiz gerekiyordu.



Ve Windows Azure’ da bir Windows Phone 7 uygulamamız hazır kodları ile karşımıza geliyor... İyi kodlamalar!





Yazardan...

Bu bölümde akıllı telefon kullanıcıları tarafından sıklıkla kullanılan ve çok sayıda indirme alan harita uygulamalarından, özelliklerinden ve nasıl programlama yapacağından bahsediyor olacağız. Biliyoruz ki insanoğlu hayatı boyunca mutlaka bilmediği, daha önce hiç gidip görmediği kasabalara, şehirlere, ülkelere, eyaletlere mutlaka gitmiş, gitmek istemiş, gitmek zorunda kalmıştır. Teknolojinin en önemli yararlarından birisinin insan hayatını kolaylaştırmak olduğunu biliyoruz. Bu yolda icat edilen navigasyon cihazlarının revaçta olduğunu inkar edemeyiz. Windows Phone 7 ile navigasyon cihazlarına ihtiyacımızın olmadığını, cihaz ile beraber gelişmiş GPS alıcısının telefon içinde barındığını, ve bunu Visual Studio 2010 ve Windows Phone Emulatrör' ü ile nasıl programlayacağımızı bu bölümde bulabilirsiniz... Yararlı olması dileğiyle...

20. Harita İle İşlemler

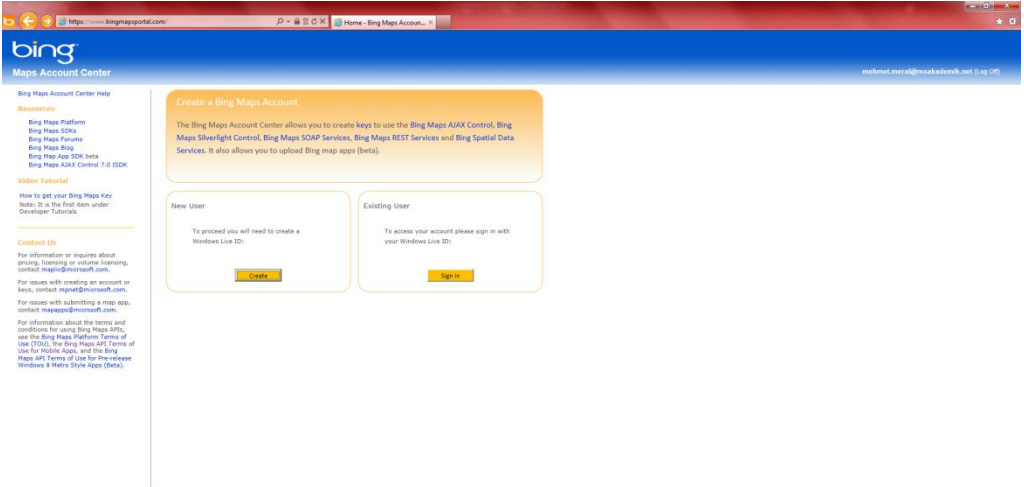
Aylarca çalıştınız, yorulduunuz. Tatili hak ettiğimize inanıyorsunuz. Eşinizle, dostunuzla ya da arkadaşlarınızla tatil planı yapmaya koyulduunuz. Fakat hiç birimizde önceden herhangi bir otel rezervasyonu ya da valiz hazırlıkları yapmadan yola çıkmayız. İşte tam bu noktada bilmediğiniz, daha önce hiç görmediğiniz yerlere gidecekseniz, hem de eğer orası yurtdışında bir yer ise, mutlaka haritadan nerde ne varmış diye bakarsınız. Bir yerden bir yere gitmeyi önce ister, sonra nasıl gidileceğini öğreniriz. Aksi takdirde eğer şoför biz isek otobanlardaki karışık tabelaların içinden çıkabilmek için epey yakıt sarf etmemiz gerekir. Bunu sadece yol tarihi ve navigasyon olarak algılamayın. Örneğin yaptığınız bir uygulama ile belki de onlarca insanın hayatını kurtarabilmek istemez miydiniz? Şöyle düşünün, doğal afetler sırasında(deprem, sel ve benzeri) belki telefonumuz kırılabilir belki internetimiz kesilebilir. İşte bu durumda önceden yazmış olduğunuz uygulama telefonun bilgilerini sürekli bir yere post ediyor olsun, bu da yaklaşık yarım saatte bir aralıkta devam etsin. Buna göre sisteme bakılarak o telefondaki kişinin en son yarım saat önce nerede görüldüğüne dair bilgiler edinilerek arama kurtma çalışmaları başlatılabilir.

Bu uğur da pek çok teknolojiler geliştirildi. Günümüzde navigasyon cihazları olarak bilinen cihazlar GPS(Global Positioning System ; Küresel Konumlama Sistemi) alıcıları ile uyduya bağlanıp uyduya göre konumumuzu bularak haritanın üzerine yansıtmasından ibaret.

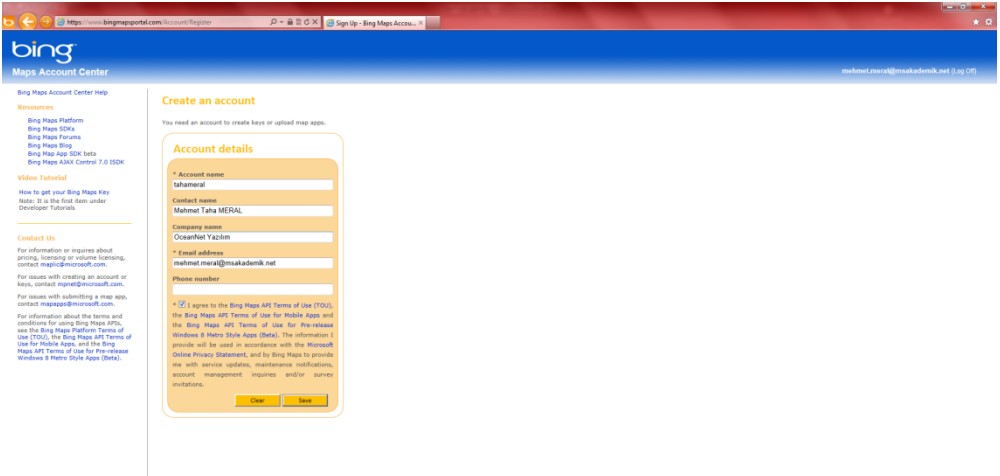
Peki bu konumu nasıl bulacağız? Latitude ve Longitude değerleri ile dünya yüzeyindeki yerimiz koordinat bazında bulunabilir. Bu bilgiler harita üzerinde anlam kazanır. GPS alıcılarının yaptığı tek şey Latitude ve Longitude' ü hesaplamak. Bizim yazdığımız yazılımlar ise eğer bir navigasyon uygulaması ise bu koordinatları olarak, diğer koordinatlara göre yön göstermek.

Windows Phone 7 üzerinde GPS alıcıları mevcuttur. Bizlerde bu alıcılar ile Latitude ve Longitude adreslerine ulaşacağız. Visual Studio 2010 ile Windows Phone 7 üzerinde harita oluşturmak, Toolbox' taki Map template' ini sürükleyip bırakmak kadar kolay.

Uygulamamıza geçmeden önce <http://www.bingmapsportal.com> adresinden kullanacağımız Bing Map için Live ID' miz ile yeni bir hesap oluşturuyoruz. Buradan pek çok şey yapmak mümkün, örneğin api keyi alarak uygulmamız kaç kişi tarafından çalıştırılmış öğrenebiliriz. Ayrıca uygulamalarımızı buradan da submit edebiliyoruz fakat Marketplace' te yayınlamıyor.



New User kısmında Create butonuna tıklıyoruz.



Ardından karşımıza gelen pencerede Account Name' imizi ve E-mail adresimizi girmek zorundayız. Diğer kısımları boş bırakabilirsiniz. Bing Map lisans sözleşmesini okuduktan sonra seçeneği işaretleyip Save diyoruz ve hesabımız oluşuyor.

Bing Maps Portal' a yüklemiş olduğunuz uygulamalarınız görebilirsiniz. Resources bölümünden kaynak dosyalara ulaşabilirsiniz.

Gelelim kodlama kısmına. File > New Project deyip Silverlight for Windows Phone kısmından yeni bir Windows Phone uygulaması oluşturalım. Adı PhoneApp3 olsun ve OS 7.1 platformunda çalışsın. Toolbox' ımızdan "Map" ı sürükleyip telefonumuzun üzerine bırakıyoruz. Ve karşımıza şu şekilde hazır bir kod geliyor.

```
<my:Map Height="601" HorizontalAlignment="Left"
Margin="0,6,0,0" Name="map1" VerticalAlignment="Top" Width="462" />
```

Height ile haritamızın ekranda kaplayacağı uzunluğu width ile de genişliğini değiştirebilirsiniz. Yatay hizalamayı sola göre, dikey hizalamayı ise yukardan özelliğini kendisi oluşturuyor. Bunu da istersek değiştirebiliriz. Bu şekilde uygulamamızı çalıştırdığımızda sorunsuz bir şekilde çalışacaktır fakat haritanın üzerinde bize Bing Map' e geliştirici olarak kayıt olmalısınız yazısı gözükmemektedir. Bunun için Bing Maps' ten aldığımız API Key' ini yerleştirelim. CredentialsProvider yazarak tırnak içerisinde Keyimizi girelim. Tekrar çalıştırdığımızda o yazının gözükmediğini fark edeceksiniz. XAML Kodu:

```
<my:Map CredentialsProvider="Au-MFFI75E2aUst-
XEgaMhul2dPF2RtW06ZMelEtt53Fdn82PH600JewaKf12efa" Height="601"
HorizontalAlignment="Left" Margin="0,6,0,0" Name="map1"
VerticalAlignment="Top" Width="462" />
```

şeklinde olacaktır.

Ardından 4 adet buton oluşturalım ve bu butonların ikisi map' in şekliyle ilgili olsun. Yani yol haritası olarak mı gösterilsin uydu haritası olarak mı. Diğer ikisinde ise Zoom artırma ve azaltma yapalım. XAML Kodu:

```
<my:Map CredentialsProvider="Au-MFFI75E2aUst-
XEgaMhul2dPF2RtW06ZMelEtt53Fdn82PH600JewaKf12efa" Height="462"
HorizontalAlignment="Left" Margin="6,6,0,0" Name="map1"
VerticalAlignment="Top" Width="444" />
<Button Content="Yakınlaştır" Height="72"
HorizontalAlignment="Left" Margin="6,535,0,0" Name="buttonZoomIn"
VerticalAlignment="Top" Width="207" Click="yakinlastirClick" />
<Button Content="Yol Modu" Height="72"
HorizontalAlignment="Left" Margin="6,474,0,0" Name="buttonRoad"
VerticalAlignment="Top" Width="207" Click="yolModuClick" />
<Button Content="Uzaklaştır" Height="72"
HorizontalAlignment="Left" Margin="243,535,0,0" Name="buttonZoomOut"
VerticalAlignment="Top" Width="207" Click="uzaklastirClick" />
<Button Content="Uydu Modu" Height="72"
HorizontalAlignment="Left" Margin="243,474,0,0" Name="buttonAerial"
VerticalAlignment="Top" Width="207" Click="uyduModuClick" />
```

Kodlamaya geçmeden önce birkaç referans eklememiz gerekiyor. Solution Explorer’ da References klasörünün üzerine gelip sağ tıklayarak Add References diyerek, System.Observable, Microsoft.Phone.Interop ve Microsoft.Phone.Reactive’ i ekledikten sonra kodumuz sadece butonların click eventlerinden oluşuyor.

```
using System.Windows;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Maps;
```

Bunları kullanıyor olduğumuz belirtmemiz gerek.

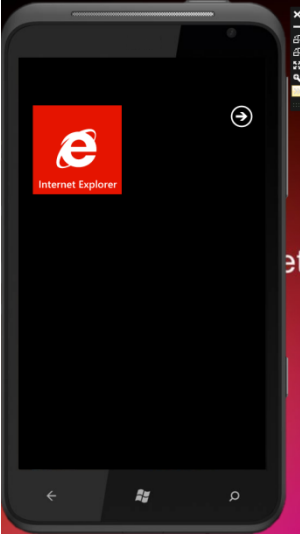
```
private void yolModuClick(object sender, RoutedEventArgs e)
{
    map1.Mode = new RoadMode();
}

private void uyduModuClick(object sender, RoutedEventArgs e)
{
    map1.Mode = new AerialMode();
}

e) private void yakinlastirClick(object sender, RoutedEventArgs
{
    double zoom;
    zoom = map1.ZoomLevel;
    map1.ZoomLevel = ++zoom;
}

e) private void uzaklastirClick(object sender, RoutedEventArgs
{
    double zoom;
    zoom = map1.ZoomLevel;
    map1.ZoomLevel = --zoom;
}
```

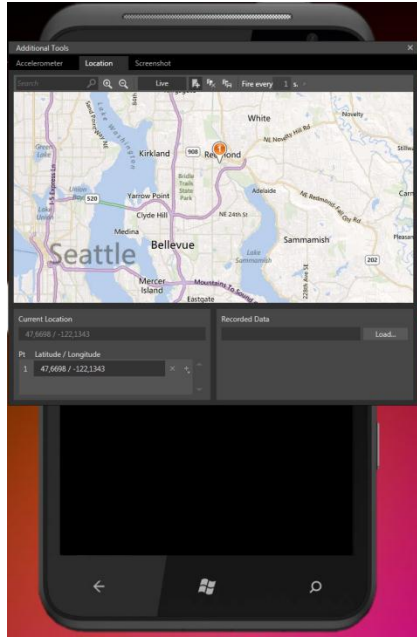
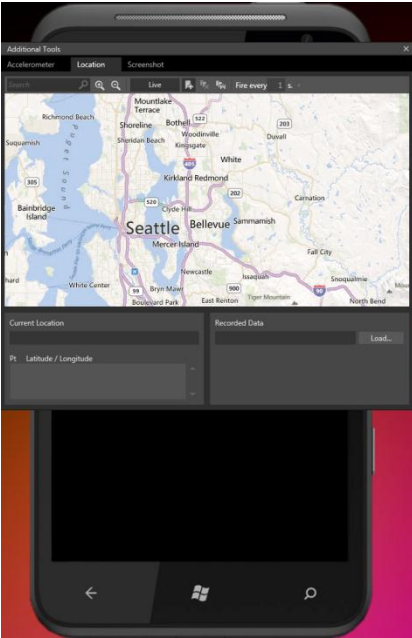
Bu kısmı kavradıktan sonra artık lokasyon bulma işlemlerine geçebiliriz. Öncelikle elinizde Windows Phone 7 cihazı bulunuyorsa debug yaparken Windows Phone Device’ ı seçip telefon üzerinden debug yapın. Eğer Windows Phone 7 cihazınız yoksa ise, başarılı emulatörümüz üzerinden önceden bir lokasyon belirleyelim. Emulatörümüzü açalım.



Yanda görmüş olduğunuz emülatörümüzün sağ üst köşesinde önceki bölümlerde anlatıldığı gibi yatay ve düşü konumu emülatörümüzün büyüklüğünü artırıp azaltabildiğimiz seçenekler mevcuttu, en attaki seçeneğe tıkladığımızda, ek toollar karşımıza çıkıyor ve burada accelerometer ile ivme simülasyonu yapabiliyor, screenshot ile emulatorekran görüntüsünü alabiliyorsunuz.

Location' a geldiğimizde karşımıza haritamız çıkıyor. İster harita üzerinden konumumuzun olmasını istediğimiz yere tıklayıp seçiyoruz isterseniz ise aşağıdan longitude ve latitude adreslerini girerek konumuzu verebiliyorsunuz.

Emulatorekran ise bu konuma göre davranıyor. Bizi oradaymış gibi gösteriyor.



Sağdaki resimde gördüğünüz gibi ben haritaya tıklayarak konumumu verdim.

Geçelim kodlamaya;

Arayüzde yapmamız gereken fazla bir şey yok. Sadece bir Map kontrolünü sürükleyip bırak ardından BingMap Api Key' imizi girerek hemen C# tarafına geçelim. XAML Kodu:

```
<my:Map CredentialsProvider="Au-MFFI75E2aUst-
XEgaMhul2dPF2RtW06ZMe1Ett53Fdn82PH600JewaKf12efA" Height="622"
HorizontalAlignment="Left" Margin="6,6,0,0" Name="map1"
VerticalAlignment="Top" Width="444" />
```

C# tarafında yine using lerimizi belirtelim;

```
using Microsoft.Phone.Controls;
using System.Device.Location;
using Microsoft.Phone.Reactive;
```

Ardından yazmamız gereken sadece 3 metod var. Öncelikle konumumuzu izleyen yeni bir GeoCoordinateWatcher nesnesi oluşturalım ve adına izleyici diyelim.

```
private readonly GeoCoordinateWatcher izleyici = new
GeoCoordinateWatcher();
```

Yapılandırıcımızın içerisinde Loaded' ın sürekli yenilenmesi için Loaded = Loaded + OnLoaded diyerek Loaded' a OnLoaded' ı ekleyerek ilerleyeceğiz. Ve gelelim 3 metoda. Bunlardan birincisi OnLoaded metodumuz. Bu metodda yazacağımız “pozisyonumuz” adlı metoddan alacağımız pozisyonumuzu observable adında bir nesneye atayacağız ve bu observable' ı System.Observable' ı kullanarak OnDispatcher' dan pozisyon değişikliğini yakalayacağız. Ve buradaki değişikliklerde OnLoaded metodumuz ile Loaded' a aktarılıp ekranda yenilecek.

```
private void OnLoaded(object sender, RoutedEventArgs e)
{
    var observable = pozisyonumuz();
    observable.ObserveOnDispatcher().Subscribe(PozisyonDegisikligi);
}
```

Pozisyon değişikliğini alacağımız metodda ise sadece GeoCoordiante tarafından aldığımız bir lokasyonu harita üzerinde ortalayarak yapacağız.

```
private void PozisyonDegisikligi(GeoCoordinate lokasyon)
{
    map1.Center = lokasyon;
}
```

Son olarak pozisyonumuzu da aşağıdaki metod ile alıyoruz.

```
private IObservable<GeoCoordinate> pozisyonumuz()  
{  
    var observable =  
Observable.FromEvent<GeoPositionChangedEventArgs<GeoCoordinate>>(e  
=> izleyici.PositionChanged += e  
    , e => izleyici.PositionChanged -= e).Select(e =>  
e.EventArgs.Position.Location);  
    izleyici.Start();  
    return observable;  
}
```

Hepsi bu kadar. Uygulamamızı çalıştırdığımızda harita açılır ve konumumuza göre haritayı ekranda ortalar. Farklı yerlere doğru yakınlaştırma yaptığınızda ilk başta oraya yakınlaşmaya başlar ve yakınlaşma bittiğinde yine konumumuza göre harita ekranımızda ortalanır.