



AHMET YESEVİ ÜNİVERSİTESİ

2013-2014

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜKSEK LİSANS

TBİL - 699 - DÖNEM PROJESİ ÖDEVİ



Windows Phone

MOBİL UYGULAMA TASARIMLARI

VE

WINDOWS PHONE 8 İÇİN ÖRNEK BİR UYGULAMA

HAZIRLAYAN : *MEHMET AKİF SÖNMEZ*

ÖĞRENCİ NO : *122173032*

DERS SORUMLUSU : *Yrd. Doç. Dr. Hacer KARACAN*

ARALIK 2013

ÖZET

Akıllı sistemlerin günlük hayatımızdaki yeri ve önemi, tartışılmayacak kadar büyüktür. Bu teknolojiler arasında da en büyük dilimlerde biriside cep telefonu pazarı ve akıllı telefonlar almaktadır. Telefonlar akıllandıkça ihtiyaç duydukları görevler çoğalmakta ve donanım ihtiyacının yanında işletim sistemlerinin yeri artmaktadır. Bu anlamda kullanılan İOS, Android ve Windows Phone işletim sistemleri bu görevi önde götürenlerdir Bu projede Türkiye’de ve dünya akıllı telefon piyasasına daha güçlü girdiğine inandığım Windows Phone İşletim sistemleri, Windows kullanan PC kullanıcılarının çok olduğu bu dünyada akıllı telefonu daha kolay kullanana bilir hale getirecektir. Ama Windows market, Google gibi uygulamalarına çok ihtiyaç duyulacak firmaların official uygulama yüklememesi marketi biraz kısır bırakıyor. Ancak Windows marketin bu durumu absorbe edecek çok artı tarafı vardır.

- DevCenter’a program koymak isteyen öğrencilere uygulama eğitimlerinin, yazılım yapmak için gereken programların öğrencilere ücretsiz verilmesi.
- DevCenter’a yazılan yazılımların basit, modern, genel amaçlı, nesne yönelimli programlama dili olan c# ile yazılabiliyor olması.
- Windows 8 işletim sistemli bilgisayarlar ile olan kolay bütünleşmiş
- Buda farklı yazılımcıların uygulama yazmasına ve aranılan doğrultuda uygulama eksikliğini kapatıyor.

Bu nedenle Windows Phone sektördeki yerini hızlı bir şekilde ilerleyecektir. Bu projede herkes için Windows Phone 8 uygulama yazmanın kolaylığı ve nasıl yapıldığı ele alınmaktadır.

Anahtar Sözcükler: Windows Phone 8, Visual Studio, C#, Silverlight, XAML, Windows 8

İçindekiler

ÖZET	i
İçindekiler	ii
Şekiller	v
1. GİRİŞ	7
2. BÖLÜM 1: WINDOWS PHONE İÇİN UYGULAMA GELİŞTİRME	7
2.1. Windows Phone ve Uygulamaları ile Tanışmak	7
2.2. Windows Phone Nedir?	7
2.3. Teknoloji Seçimi	8
Silverlight ve XNA	8
Silverlight ile Oyun Geliştirmek	9
XNA ile Oyun Geliştirmek	9
Silverlight ile XNA 'i Birlikte Kullanmak	10
2.4. Geliştirme Ortamını Hazırlamak	11
Kullanılacak Yazılım Gereksinimleri	11
Kullanılacak Donanım Gereksinimleri	12
Uygulamaları Gerçek Bir Telefonda Test Etmenin Faydaları	12
3. BÖLÜM 2: UYGULAMA GELİŞTİRME	13
Yeni Proje Oluşturmak	13
İlk Uygulamayı Telefon Üzerinde Çalıştırmak	15
4. BÖLÜM 3: WINDOWS PHONE KONTROLLERİ, SENSÖRLERİ VE SERVİSLERİ	17
4.1. XAML ve Kontrolleri Tanımak	17
Kullanıcı Arayüzü (User Interface) Oluşturmak	17
XAML ve Görsel Ağaç (Visual Tree)	21
Statik Kullanıcı Ara yüzünden Fazlası Gerektiğinde	23
4.2. Kontrollerle Çalışmak (Toolbox Kullanımı)	23
Kontrol Örneği	23
Kontrol Ekleme	24
Kontrolle İsim Vermek	25
4.3. Kontrol Tipleri	27
Sık Kullanılan Kontroller	31
TextBlock Kontrolü:	31
TextBox Kontrolü:	32
Örnek Uygulama	33
Button Kontrolü:	36

<i>CheckBox Kontrolü</i>	39
<i>ListBox Kontrolü:</i>	40
<i>Popup Kontrolü</i>	41
<i>İvme Ölçer (Accelerometer)</i>	52
<i>Konum Servisi Örnek Uygulama 14: (Microsoft Açık Akademi, 2013)</i>	60
5. BÖLÜM 4: WINDOWS PHONE İLE UYGULAMA GELİŞTİRME VE VERİ İŞLEMLERİ ..	64
5.1. <i>Uygulama Mimarisi ve Isolated Storage</i>	64
<i>Windows Phone Silverlight Uygulama Yaşam Döngüsü(Application Life Cycle)</i>	64
<i>Windows Phone Çalışma Modeli (Execution Model)</i>	64
<i>Terminoloji</i>	66
5.2. <i>Windows Phone Uygulama Yaşam Döngüsü (Application Life Cycle)</i>	68
<i>The Launching Event</i>	68
<i>Running</i>	69
<i>The OnNavigatedFrom Method</i>	69
<i>The Deactivated Event</i>	69
<i>Dormant</i>	70
<i>Tombstoned</i>	70
<i>The Activated Event</i>	71
<i>The OnNavigatedTo Method</i>	71
<i>The Closing Event</i>	71
5.3. <i>Isolated Storage</i>	72
<i>Isolated Storage Nasıl Kullanılır?</i>	72
5.4. <i>Data İşlemleri</i>	76
<i>Veri İle Çalışmak</i>	76
<i>Lokal Dosyalar (Local Files)</i>	77
<i>Kaynak Dosyaları (Resource Files)</i>	77
<i>Isolated Storage</i>	79
<i>Web Servisler</i>	79
<i>Web Servis Teknolojileri</i>	80
<i>WCF Servisleri</i>	81
<i>WCF Data Services (OData Services)</i>	81
<i>Windows Azure Storage Services</i>	81
<i>Hangi Yaklaşımı Kullanacağına Karar Vermek</i>	82
5.5. <i>Kontrollere Veri Bağlamak(Data Binding To Controls)</i>	83
<i>Bir Kontrolü Tek Bir Nesneye Bağlamak</i>	83
<i>Bir Kontrolü Nesne Listesine Bağlama</i>	86

<i>Lokal(Yerel) Veri Tabanı ile Çalışmak.....</i>	<i>91</i>
<i>İş Sınıflarını Tablolarla Eşleştirmek</i>	<i>93</i>
6. BÖLÜM 5: WİNDOWS PHONE UYGULAMALARINI DEVCENTER'A GÖNDERMEK	99
<i>Dev Center Uygulamaları</i>	<i>102</i>
<i>Dev Center Üyeliği Oluşturmak.....</i>	<i>103</i>
<i>Dev Center'a Uygulama Yükleme</i>	<i>103</i>
7. BÖLÜM 6: ÖRNEK UYGULAMAR	110
<i>Windows Phone 8 ile telefonun operatörünü bulma</i>	<i>110</i>
<i>Windows Phone 8 için Basit bir Hesap makinesi örneği.....</i>	<i>113</i>
8. SONUÇ:.....	118
<i>Kaynakça</i>	<i>119</i>

Şekiller

Şekil 1 – Kurulması gereken 3 önemli program (Microsoft, 2013).....	11
Şekil 2 - Örnek bir uygulama görüntüsü	13
Şekil 3 - Emülatör çalıştırma işlemi.....	15
Şekil 4 – Emulator yardımıyla çalışan bir başka uygulama örneği.....	15
Şekil 5 – Örnek 3’ün ekran çıktısı.....	19
Şekil 6 – Örnek 4’ün ekran çıktısı.....	21
Şekil 7 – Örnek 4’ün ağaç yapısı	21
Şekil 8 – Örnek 5’in ekran çıktısı.....	22
Şekil 9 – Örnek 5’in olay ağacı	23
Şekil 10 – Textbox Kontrol örneği.....	24
Şekil 11 – Visual studio ekranı	24
Şekil 12 - Visual Studio IntelliSense penceresi	25
Şekil 13 – Properties (özellikler) araç kutusu	26
Şekil 14 - Frame ve Page modeli gösterimi	28
Şekil 15 – Layout kullanımı (dikey ve yatay örneği).....	30
Şekil 16 – Yeni proje açma ekranı.....	33
Şekil 17 – Textbox örneği görüntüsü	35
Şekil 18 – Buton kontrolü ekran görüntüsü.....	36
Şekil 19 – Checkbox kullanım örneği	39
Şekil 20 – Listbox kullanım örneği	40
Şekil 21 – Windows phone panorama page kontrol görüntüsü	42
Şekil 22 – Panorama örneği.....	43
Şekil 23 - Genel Panorama görüntüsü	44
Şekil 24 – Panorama örnek 11 ‘in ekran görüntüsü	46
Şekil 25 - Pivot kullanım örneği	46
Şekil 26 – Örnek 12 ekran görüntüsü	47
Şekil 27 – Örnek 12 ekran görüntüsü	49

Şekil 28 - İvme Ölçerin Genel Şeması	53
Şekil 29 – Emulator başlatma ekranı	57
Şekil 30 – Örnek 13 deki İvmeölçer örneğinin ekran görüntüsü	57
Şekil 31 – Emülatör ve Örnek 13 Kullanımı	58
Şekil 32 – Konum Servisinin kod olarak eklenmesi.....	60
Şekil 33 – Windows Phone uygulama yaşam döngüsü.....	68
Şekil 34 – Örnek 14 Ekran görüntüsü	74
Şekil 35 – Veri kaynakları.....	77
Şekil 36 – Örnek 15 ekran görüntüsü	85
Şekil 37 – Örnek 17 Ekran görüntüsü	91
Şekil 38 – Projeye referans ekleme işlemi.....	94
Şekil 39 - WMAAppManifest.xml dosyası görünümü	99
Şekil 40 - Capabilities penceresi	100
Şekil 41 - Requirements penceresi	101
Şekil 42 – Packaging penceresi	101
Şekil 43 – Dashboard ekranı	103
Şekil 44 – Dashboard karşılama ekranı ve yapılması gereken 4 adım	104
Şekil 45 – Appinfo – uygulamaya ait açıklamaların yazıldığı ekran	105
Şekil 46 – Dashboard’ XAP dosyasını yüklenmesi.....	106
Şekil 47 – XAP Store Listing ekranı	107
Şekil 48 – Uygulamanın ekran görüntülerini yükleme ekranı	108
Şekil 49 – Uygulamanın fiyatlandırma ekranı	108
Şekil 50 – Bing Maps kullanım ekranı	109
Şekil 51 – Uygun market seçimi ve harita servis ekleme ekranı.....	109
Şekil 52 – Proje oluşturma ekranı.....	110
Şekil 53 – Uygulamanın ekran görüntüsü.....	111
Şekil 54 – Uygulamada butana tıklandıktan sonraki görünüm.....	112
Şekil 55 - Uygulamanın ekran görüntüsü	113
Şekil 56 - Uygulamanın çalışır görüntüsü	117

1. GİRİŞ

2. BÖLÜM 1: WINDOWS PHONE İÇİN UYGULAMA GELİŞTİRME

2.1. Windows Phone ve Uygulamaları ile Tanışmak

2.2. Windows Phone Nedir?

Microsoft'un mobil cihazlar ile ilgili ilk ürünü, 1992 yılında geliştirilmeye başlanan ve 1996 yılında piyasaya sürülen Windows CE 1.0'dır. Elbette, Microsoft o dönemde Windows 95'i piyasaya sürmek üzere olduğundan, Windows CE'de işletim sistemi olarak Windows 95 mimarisi ve kullanıcı arabirimi olarak da WinPad kullanıldı.

2000 yılında Microsoft, Pocket PC 2000'i piyasaya sürdü. Windows CE 3.0 tabanlı bu sistem Palm tarzı cihazlara yönelikti. Pocket PC 2000 Windows 98, Windows ME ve Windows 2000 işletim sistemlerindeki gibi bir görünüme sahipti.

2001 yılına geldiğimizde Microsoft, Pocket PC 2002'yi piyasaya sürdü ve bu sistem, Microsoft'un "akıllı telefonlara" uygulanabilen ilk modeli olma özelliğine sahip oldu.

2003 yılında piyasaya sürülen Windows Mobile 2003, Windows CE 4.2.

2004 yılında Windows Mobile 2003 SE, 2005 yılında ise Windows Mobile 5 piyasaya sürüldü.

2007 yılında Windows CE 5.0 tabanlı Windows Mobile 6;

2009'da, Windows Mobile 6.5 piyasaya sürüldü.

2010'da tasarımsal değişikliğe gidilerek, Metro UI arayüzünü kullanan Windows Phone 7 piyasaya sürüldü.

Windows Phone 7 ile beraber; uygulama geliştiricilerin Silverlight, oyun geliştirenlerinse XNA platformunu kullanması gerekiyordu.

Microsoft'un son mobil işletim sistemi olan Windows Phone 8'de HTML 5 ve Native desteği sunuldu. HTML5 halen gelişmekte olan bir standarttır ve 2024-2025 yıllarında standart halini alması planlanan bir teknolojidir. HTML5 ile Flash ve Silverlight gibi eklenti (plugin) tabanlı sistemlerde yapılabilecek her şey yapılabilecektir. HTML5 fikrinin çıkış noktasının "bir kere yaz, her yerde çalışsın" ifadesi olmasından dolayı bir platformda geliştirdiğiniz HTML5 uygulamalarını ekstra iş gerektirmeden farklı platformlara taşına bilinir.

Native ise, C++ programlama dilini kullanarak geliştirilen bir uygulama geliştirme yöntemidir. Geliştirilen kodlar, Java ve C# dillerinde olduğu gibi herhangi bir ara dile dönüştürülmeden çalışır. Windows Phone 8’de C++ kullanarak oyun veya uygulama yazılabilir.

Native ve HTML5 desteği sadece Windows Phone 8 uygulamalarında olduğu için, Windows Phone 7’de bu uygulama geliştirme yöntemlerini kullanamazsınız (Microsoft).

2.3. Teknoloji Seçimi

Silverlight ve XNA

Windows Phone, geliştiricilere 2 ana programlama modeli sunuyor: Silverlight ve XNA. Silverlight, ilk olarak web tarayıcılar için bir eklenti (plug-in) olarak tasarlandı. Amaç, zengin internet uygulamaları geliştirme sürecine yeni bir bakış getirmektir. Silverlight bizlere deklaratif bir kullanıcı ara yüzü sundu. Kolayca oluşturulabilen ve temalandırılabilen bu ara yüz, animasyon, veri bağlama (veri binding), vektör grafikler ve çeşitli kontroller ile destekleniyordu. Silverlight’ın, Windows Phone tarafından kullanılmakta olan mevcut sürümü, Windows ve Mac üzerinde kullanılan güncel Silverlight sürümü ile aynı etkinliğe sahiptir. Fakat telefon ile alakasız bazı özellikler kaldırılmıştır ve telefona özgü bazı şeyler eklenmiştir. Ayrıca performans ayarlaması yapılmış durumdadır.

XNA, ilk olarak XBOX odaklı olarak tasarlanmış olmasına rağmen, ayrıca Windows, Zune HD ve tabii ki Windows Phone sürümleri de mevcuttur. XNA’nın geliştirilme amacı yüksek performanslı, 2D sprite-based ya da 3D oyunlar oluşturmaktır.

Hangisini Seçmeli? İki model hakkında da genel fikir sahibi olduktan sonra, gelelim seçim yapmaya. Temel seviyede, geliştiricilere yapılan tipik tavsiyem şudur: “Uygulama geliştirirken Silverlight, oyun geliştirirken XNA kullanın.” Ama detaylar için içerisine girdikçe, gerçek dünyada durum biraz daha farklı olabilir ve tercihler değişebilir.

İki model hakkında da genel fikir sahibi olduktan sonra, gelelim seçim yapmaya. Temel seviyede, geliştiricilere yapılan tipik tavsiye şudur: “Uygulama geliştirirken Silverlight, oyun geliştirirken XNA kullanın.” Ama detaylar için içerisine girdikçe, gerçek dünyada durum biraz daha farklı olabilir ve tercihler değişebilir (Microsoft Açık Akademi, 2013).

Silverlight ile Oyun Geliřtirmek

Kesinlikle Silverlight ile harika oyunlar yapabilirsiniz. Sonu olarak oyun da bir programdır. DevCenter'te Silverlight ile geliřtirilmiř “Darts” isminde mkemmek bir rnek bulabilirsiniz. Bu oyunun ilk srm Windows Phone DevCenter zerinde yayınlandığında, ok kısa zamanda cretli uygulamalar arasında 11. sıraya kadar ykseldi. Darts, XNA kullanan ok sayıda Xbox LIVE oyunu arasından sıyrıldı ve bu konuma ykseldi. Bir oyunu Silverlight ile kodlamak, geliřtiriciye eřitli avantajlar saėlar. rneėin; Facebook ve Twitter gibi servisler ile kolay uyum ve bunun yanında, menler ve skorboardlar gibi noktalarda tm standart Silverlight kontrollerini kullanabilme yeteneėi. Diėer yandan, Silverlight ile kompleks oyunlar oluřturmaya alıřman, performans aısından ok akıllıca olmayabilir. Xbox LIVE zelliklerini, sadece XNA oyunları iin kullanılabilir (Microsoft Aık Akademi, 2013).

XNA ile Oyun Geliřtirmek

XNA kullanarak oyun dıřında bir uygulama kodlanabilir. Button ve ListBox gibi en temel kontrolleri yeniden yaratma ihtiyaının doėmasının dıřında, XNA uygulamaları řu anda kullanıcının telefon teması, uygulama ubuėu (application bar), tarayıcı kontrol (web browser control) ve benzeri ok sayıda zelliėin avantajlarından yararlanabilme yeteneėine sahip deėildir. Ayrıca XNA iin ok sayıda nc parti kontrol ktphaneleri (control libraries) mevcuttur (Microsoft).

Silverlight ile XNA 'i Birlikte Kullanmak

Bir Windows Phone uygulaması içerisinde *Silverlight* ve XNA'nın fonksiyonelliğini harmanlanabilir. *DevCenter*'te yer alan *Silverlight* uygulamalarının bir kısmı, mikrofonu kullanmak, ses efektleri çalmak gibi XNA fonksiyonelliklerinden de faydalanmıştır. Ayrıca XNA uygulamaları içinde bir web tarayıcı kullanılmamasına rağmen, *Silverlight*'ın ağ ile ilgili sınıflarını kullanarak bir web kaynağına talepte yapabilirsin. Bu konudaki tek kısıt, *Silverlight* ara yüzü ile XNA ara yüzünün aynı anda kullanılamamasıdır. Bir uygulama için seçilecek olan kullanıcı ara yüzü, bu teknolojilerden yalnızca biri olabilir. *DevCenter* sertifikasyonu, *Silverlight* ve XNA'nın uygun olmayan bir biçimde harmanlanmasını yasaklar! XNA kullanıcı ara yüzü parçalarını, bir *Silverlight* uygulaması içerisinde kullanmanın bir yolunu bulup uygulamayı sorunsuz bir şekilde çalıştırmayı başarlarsa bile, iki ara yüz birbirine karıştırıldığı için, *Windows Phone DevCenter* uygulamasını onaylamayacaktır.

Bir *Silverlight* uygulaması içerisinde *Microsoft.Xna.Framework.Game.dll* ve *Microsoft.Xna.Framework.Graphics.dll* kütüphanelerini referans etmediğiniz sürece sorun yok. *Silverlight* ve XNA arasındaki ilişki, olması gerektiğinden çok daha karışıktır. Bu durum, *Silverlight* tarafındaki hedef ile XNA tarafındaki hedeflerin birbirine oldukça uzak olmasından kaynaklanmaktadır. Windows Phone için geliştirilen *Silverlight* altyapısının, Windows ve Mac için geliştirilen *Silverlight*'a mümkün olan en yüksek ölçüde uyumlu olması arzulanıyor. XNA tarafındaki hedef ise Windows Phone'un XBOX, Windows ve Zune HD için olan XNA ile olabildiğince uyumlu olmasıdır.

Sonuç olarak iki tarafın hedefleri doğrultusunda bazı fonksiyonellikler platformda zorluk çıkarmaktadır. Bu riskten dolayı ve aralarındaki belirgin farklılıklar sebebi ile iki teknolojinin iyi ilişki içerisinde değildir. Örneğin; Windows Phone mikrofon ile haberleşmek için tek bir sınıfa sahiptir ve bu bir XNA özelliğidir. Bu sınıf, bir XNA kütüphanesi içerisinde yer alır. Çünkü XNA'nın daha önceden geliştiricilere sunmuş olduğu, eski sürümler ile olan uyumluluğun korunması hedeflenmiştir (Microsoft).

2.4. Geliştirme Ortamını Hazırlamak

Kullanılacak Yazılım Gereksinimleri

- Geliştirme ortamını kurmak için Windows 8 veya 8.1 işletim sistemi
- Windows Phone Yazılım Geliştirme Kiti (SDK) 8
- (<http://developer.windowsphone.com> adresinden ücretsiz indirilebilir.)

Paket içerisinde:

- Windows Phone için Visual Studio 2013 Express
- Windows Phone için Expression Blend 5 (Silverlight-tabanlı vektör grafikler, animasyonlar ve kontrol şablonları tasarlamak için alternatif olarak kullanabilirsiniz)
- XNA Game Studio 4.0 (XNA uygulamaları ve oyunları geliştirmek için kullanılır)
- Windows Phone Emulator (Uygulamalarını gerçek bir telefon yerine bilgisayarda çalıştırıp test etmeye yarar)
- Silverlight for Windows Phone Toolkit (<http://silverlight.codeplex.com> adresinden ücretsiz indirilebilir) Bu kit, çok sayıda kontrol içerir.
- Silverlight Toolkit (<http://silverlight.codeplex.com> adresinden ücretsiz indirebilirsiniz) Chart ve Graph gibi, Windows Phone ile kullanılabileceğin ek kontroller içerir.

Eğer Visual Studio'nun daha üst bir sürümünü kullanıyorsa, SDK ile gelen Express sürümü kullanman gerekmiyor. SDK'nın kurulumu, Windows Phone'a özgü özellikleri diğer sürümlere de kurar. Visual Studio'nun Express sürümü ile ücretli sürümleri arasındaki farkların Windows Phone uygulaması geliştirmek için bir etkisi yoktur. Şu anki SDK sürümü, yalnızca C# ve XAML ile uygulama geliştirmeye izin verilir. Fakat Windows Phone için Silverlight uygulamaları geliştirirken, Visual Studio Professional ve üzeri sürümler için eklenti indirerek C# diline alternatif olarak Visual Basic de kullanılabilir (Microsoft).



Şekil 1 – Kurulması gereken 3 önemli program (Microsoft, 2013)

Kullanılacak Donanım Gereksinimleri

- İşlemciniz SLAT (Second Level Address Translation) destekli olmalıdır.
- BIOS ayarlarında Sanallaştırma (Virtualization) aktif (enabled) hale getirilmelidir.
- İşletim sisteminiz Windows 8 Pro 64-bit olmalıdır (Hyper-V sadece bu sürümle birlikte gelir).
- Windows bileşenlerinden Hyper-V aktif hale getirilmelidir.
- Bilgisayarınızın RAM'i 4GB ve üzeri olmalıdır.

İşlemciniz SLAT'i desteklemese bile, kurulumu gerçekleştirebilirsiniz. SLAT, Windows Phone 8 emulatör imajları kurulurken gereklidir. Windows Phone 8 emulatörleri Hyper-V üzerinde çalıştığından buna ihtiyaç duyulur. İşlemciniz SLAT'i desteklemiyorsa, Windows Phone 8'e özel emulatör imajları kurulmaz ama yine de Windows Phone 7 uygulamaları geliştirebilir ve test edebilirsiniz. Ayrıca, Windows Phone 8 uygulamaları da geliştirebilirsiniz ama test etmek istediğinizde, mutlaka bir Windows Phone 8 cihazının bilgisayara bağlanması gerekir (Microsoft).

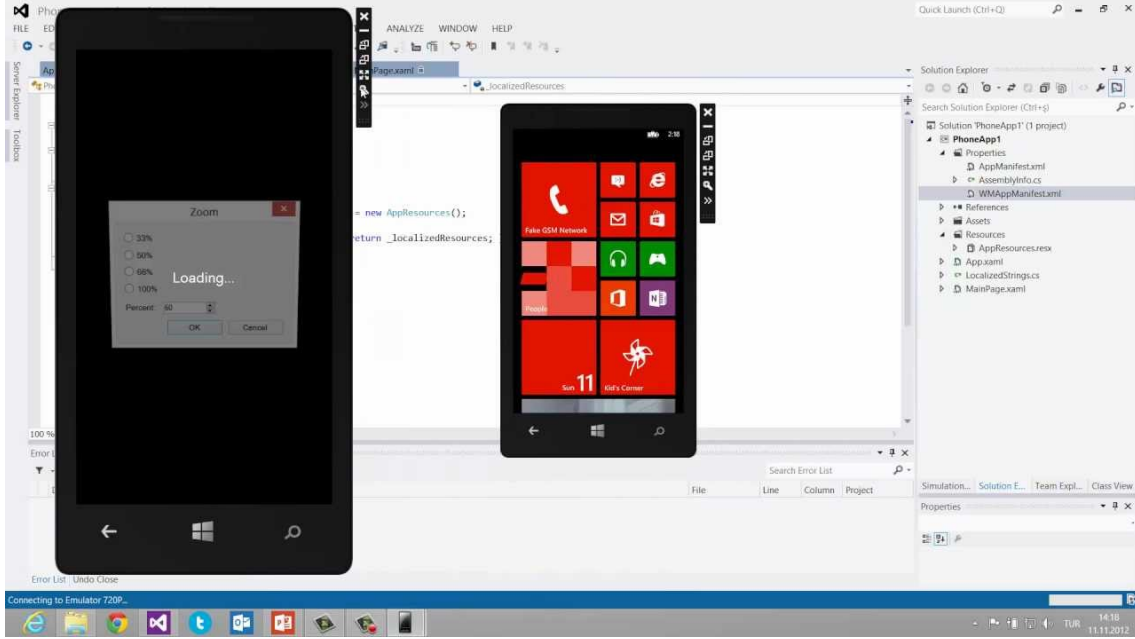
Uygulamaları Gerçek Bir Telefonda Test Etmenin Faydaları

Dokunmatik hedeflerin çok küçük, birbirine çok yakın veya ekranın kenarlarına çok yakın olmadığından emin olmak için, uygulamaları gerçek bir telefon üzerinde çalıştırmak faydalıdır. Örneğin, tasarlayıp emulatör üzerinde test ettiğin bir oyunda, pratikte gerçek telefon üzerinde oyun oynanırken kullanıcının parmağının ekrandaki önemli bilgileri bloklayabileceğini önceden düşünememiş olabiliriz. Bu açıdan, Windows Phone DevCenter üzerine yalnızca emulatör üzerinde test etmiş olduğumuz uygulamaları göndermek risklidir. Pratikte de testleri yapmamız gereklidir (Microsoft).

3. BÖLÜM 2: UYGULAMA GELİŞTİRME

Yeni Proje Oluşturmak

Windows Phone Geliştirici Araçları'nı kurduktan sonra, ilk uygulamanızı oluşturmak için en kolay yol Visual Studio kullanmaktır. Microsoft Visual Studio 2013 Express for Windows Phone uygulamasını başlatılır ve File menüsünden, New Project seçeneği tıklanır.



Şekil 2 - Örnek bir uygulama görüntüsü

Bu işlemin ardından, yeni proje (New Project) diyalog penceresi açılır. Bu pencerenin sol tarafında çeşitli proje şablonları çıkar. Sol taraftan bir şablon grubunu seçildiğinde, diyalog penceresinin orta bölümünde, oluşturabilecek farklı uygulama tiplerini görülür. Sol taraftan, Silverlight for Windows Phone şablon grubu seçilir. Orta bölümden de, Windows Phone Application şablonu seçilir. Name bölümünden, projeye MerhabaPhone adı verilir ve OK butonuna tıklanır. Bu işlemlerin ardından, yeni bir Silverlight for Windows Phone projesi oluşturulup açılacaktır.

Varsayılan olarak, Visual Studio üç parçaya bölünür. Sol tarafta tasarım penceresi (Design view, ortada XAML penceresi (XAML view) ve sağ tarafta *Solution Explorer*. Solution Explorer içerisinde projeye ait olan çok sayıda dosya bulunur. Bu örnekte MainPage.xaml ve MainPage.xaml.cs dosyalarını kullanılır. MainPage.xaml dosyası, uygulamanın kullanıcı arayüzünü tanımlamak için kullanılır. XAML, XML-tabanlı deklaratif bir dildir ve uygulamanın kullanıcı arayüzündeki elementleri oluşturup uygun bir düzende yerleştirmek için kullanılır. MainPage.xaml dosyasının yanındaki ok simgesine tıklayarak ilgili alanı

geniřlettiđinizde, MainPage.xaml.cs adındaki C# kod dosyası g r l r. Kod dosyaları, par ası oldukları XAML dosyası ile birleřtirilir. Par alı sınıf olarak ele alınan bu dosyalar XAML dosyalarının iřlevselliđine ait kodları barındırırlar (Microsoft A ık Akademi, 2013).

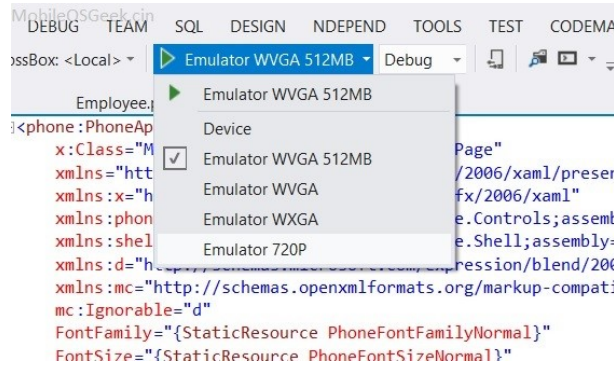
İlk Uygulamayı  alıřtırmak

Windows Phone i in geliřtirilen ilk Silverlight uygulamasını  alıřtırmak i in Windows Phone cihazını simule eden bir *Windows Phone Emulator * kullanılır. Windows Phone emulator  kullanarak, uygulamayı telefona deploy etmeden masa st  bilgisayarında hızlıca test edilebilir. Uygulamayı klavyeden F5 ya da Ctrl+F5 ile de  alıřtırıřabilir. Alternatif olarak men den Debug - Start Debugging ya da Debug - Start Without Debugging se erek bařlatılabilir. Visual Studio emulator  bařlatıp, uygulamayı emulator i erisine y kleyecektir. Uygulamanın derlenmesine engel bir hata var ise, Visual Studio hata bilgisini g r nt leyecektir. Herhangi bir hata bulunmaması durumunda, emulator penceresi a ılır.

İlk sefer i in emulator n a ılıřı bařlaması biraz zaman alabilir. Sonraki oturumları hızlandırmak i in emulator penceresi kapatılmamalıdır. Bunun yerine, Visual Studio i erisinde *Debug* men s nden *Stop Debugging* se eneđi ile debug oturumunu sonlandır. Bu sayede emulator halen  alıřır durumda olacak ve bir sonraki debug oturumunda emulator n yeniden a ılması gerekmeyeceđi i in oturum daha  abuk bařlatılacaktır (Microsoft A ık Akademi, 2013).

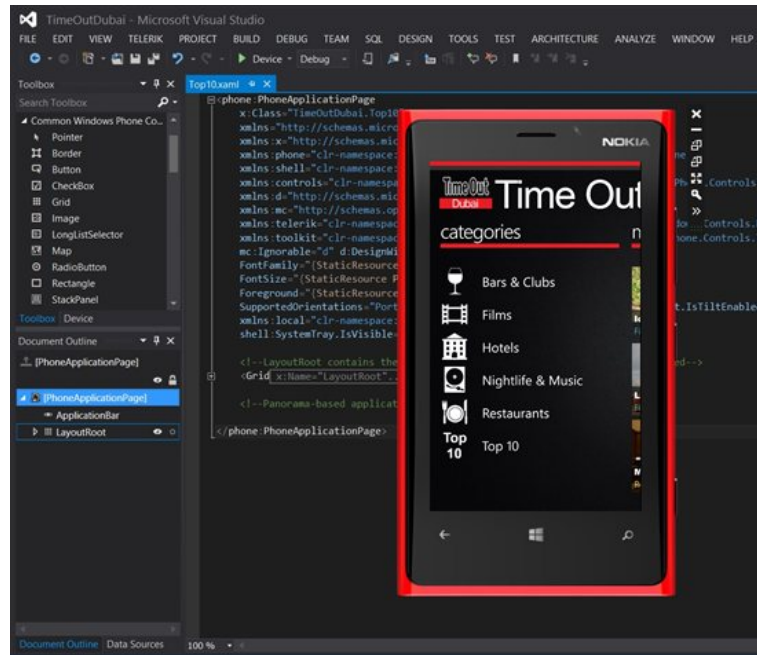
Emulator Kullanımı

Geliřtirme sırasında emulator kullanırken, emulator  s rekli a ık tutulabilir. Her deđiřiklik yapıldıđında ve dađıtım yaparken emulator  kapatmak gerekmez, kapatılmamalı   nk  yeniden bařlaması olduk a uzun zaman alabiliyor. Emulator  a ık tutarak uygulamayı yeniden dađıtabilir ve testleri 1-2 saniye i erisinde bařlatılabilir (Microsoft). (Lecrenski, Watson, & Ensor, 2011) (Delprato) (Basu, 2013)



Şekil 3 - Emülatör çalıştırma işlemi

- F1 tuşu donanım üzerindeki Back butonuna karşılık gelir.
- F2 tuşu (ya da Windows tuşu) donanım üzerindeki Home butonuna karşılık gelir.
- Pause tuşu klavyeyi değiştirir. Aktive edildiğinde, ekrandaki tuşlara tıklamak yerine bilgisayarın klavyesi kullanılır.
- Ayrıca, *Page Up* tuşu ile bilgisayarının klavyesini kullanılır ve *Page Down* tuşu ile yeniden dâhili klavyeye geçiş yapılabilir.



Şekil 4 – Emulator yardımıyla çalışan bir başka uygulama örneği

İlk Uygulamayı Telefon Üzerinde Çalıştırmak

Uygulamayı Windows Phone işletim sistemine sahip bir telefonda çalıştırmak için öncelikle telefonun kilidi açılmalıdır. Bu iş için Windows Phone Developer Registration aracı kullanılır. Bu araca Start Menüünde yer alan *Windows Phone Developer Tools* altında bulunmaktadır. Ayrıca, ücretli bir *App Hub* hesabına sahip olunmalıdır.

- App Hub hesabı için: <https://windowsphone.create.msdn.com/Register/>
- Bilgisayarda Zune yazılımı başlatılır.
- Telefon bilgisayara bağlanır.
- *Windows Phone Developer Registration* aracı başlatılır, *App Hub* hesabıyla ilişkili olan *Windows Live ID* kullanıcı adı ve şifre girilir.
- Kayıt sihirbazı (registration wizard) içerisinde telefonunla ilgili gereken tanımlama bilgileri girilir. Telefonun kilidi açılacak ve Visual Studio içerisinde deploy edilecek uygulamaları almaya hazır olunacaktır.
- Visual Studio içerisinde projeyi telefona yüklemek çok kolaydır. Tek yapmak gereken dağıtım hedefi olarak emulätör yerine "Windows Phone Device" seçilmelidir.

Açılan menüden "Windows Phone Device" seçtikten sonra kilidi açılmış bir telefona uygulamayı yükleyebiliriz. Telefon üzerinde çalışırken, emulätörde yaptığımız işlemlerin aynısını yapıp aynı seçenekleri kullanabiliriz (Basu, 2013).

4. BÖLÜM 3: WINDOWS PHONE KONTROLLERİ, SENSÖRLERİ VE SERVİSLERİ

4.1. XAML ve Kontrolleri Tanımak

Kullanıcı Arayüzü (User Interface) Oluşturmak

Genel olarak, Windows Phone uygulamaları geliştirirken *Silverlight*, Windows Phone oyunları geliştirirken de *XNA* kullanılır. *XAML*, Silverlight tarafında kullanıcı ara yüzü oluştururken kullanılan deklaratif bir dildir ve Windows Phone uygulamaları geliştirirken de ara yüz için bu dil kullanılır. Kontroller (controls), şekiller (shapes), metinler (text), ve ekran üzerinde sunulan diğer içerikler bu dil kullanılarak ele alınır. *XAML*, HTML ye göre çok daha kuvvetli bir dildir. *XAML*, aynı HTML gibi elementler (elements) ve niteliklerden (attributes) oluşur. *XAML*, XML-tabanlıdır ve bu yüzden kodlarken XML kurallarına uyulmalıdır. Kafanızda, "Kullanıcı ara yüzünü oluşturmak için *Visual Studio* ve *Expression Blend* gibi araçlar kullanacaksa neden *XAML*'i öğreneyim ki?" sorusu belirebilir. *XAML*'i otomatik olarak oluşturan çeşitli araçlar bulunmasına rağmen, gerektiğinde *XAML* üzerinde ince ayarlamaları yapabilmek için en iyi yol onu anlamaktan geçer. Ayrıca, bazen ince ayar yapmak istendiğinde ya da arka planda neler olup bittiğine daha fazla hâkim olmak istendiğinde kullanıcı ara yüzünü (UI) elle kodlamak daha kolay gelir (Delprato).

Örnek 1:

XAML Örneği:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"> <Button Height="72" Width="160"
Content="Tıkla" />
```

XAML içerisinde, Button kontrolü, <Button> elementi ile tanımlanır. Genişlik (Width) ve Yükseklik (Height) nitelikleri (attributes) ile butonun boyutları belirlenir. Visual Studio içerisinde yeni bir Windows Phone uygulaması oluşturduğunda, hazır bir <Grid> elementi sunulur. Bu element nesneleri pozisyonlamak/yerleştirmek için kullanılır.

XAML oluşturmak için Visual Studio'dan faydalanılabilir. Örneğin, Toolbox'tan tasarım ekranına bir buton sürükleyip bırakıldığında, buna ait *XAML* kodu otomatik olarak oluşturulacaktır.

XAML gibi deklaratif bir dil kullanmanın en iyi yönlerinden birisi kullanıcı ara yüzü ve iş mantığını birbirinden net bir şekilde ayırma imkânına sahip olmaktır. Örneğin, ekibinizdeki

bir tasarımcı kullanıcı ara yüzünü (UI) XAML ile tasarlayıp sonra iş mantığını kodlama kısmını uygulama geliştiriciye bırakabilir. Tasarımı ve kodlamayı aynı kişi yapıyor olsa bile (ülkemizde maalesef çoğunlukla bu şekilde), kullanıcı arayüzünü bir XAML dosyasında (.xaml) tutup, iş mantığına ait kodu code-behind dosyalarda (.cs ve .vb) tutmak, süreçte rahatlık sağlar (Delprato).

XAML prosedürel koddur (Sadece daha kolayı)

<Button /> gibi XAML elementlerinin kullanımı, kodlama tarafındaki nesne örneklemeye benzer.

Örnek 2:

XAML kodu:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"> <Button Height="72" Width="160"
Content="Tıkla" />
```

Bu XAML kodunu, C# kullanarak aşağıdaki şekilde de yazılabilir.

```
// Buton nesnesinin örneklenmesi Button btn = new Button();
// Butona ait özelliklerin değerlerinin atanması
    btn.Width = 160;
    btn.Height = 72;
    btn.Content = "Tıkla";
// Butonun, 'ContentPanel' isimli Grid elementinin alt üyesi olarak arayüze eklenmesi
// Bir başka deyişle butonun kullanıcı arayüzüne eklenip pozisyonlanması.
    ContentPanel.Children.Add(btn);
```

Kullanıcı arayüzü (UI) tarafında, XAML'in kolay okunabilirlik ve prosedürel koda göre daha kompakt olma avantajı vardır. Bununla birlikte, bazı durumlarda kullanıcı arayüzünü dinamik olarak oluşturmak gerektiğinde, prosedürel kodlama yoluna gitmek gerekecektir.

Özellikler (Properties)

XAML içerisinde özellik (property) değerleri iki farklı yol ile belirlenebilir.

- Nitelik (Attribute) elementi sentaksı ile
- Özellik (Property) elementi sentaksı ile

Nitelik (Attribute) elementi sentaksı daha önceki örneklerde de ele alındığı gibi nitelik="değer" şeklindedir. Bu kullanım şekline HTML'den de aşinasın zaten. Sıradaki

örnekte, kırmızı bir dikdörtgen (Rectangle) oluşturuluyor. Fill niteliği üzerine, öntanımlı bir renk olan “Red” (kırmızı) atanıyor.

Alternatif olarak, renk değerini, özellik (property) elementi sentaksı kullanarak da belirleyebilirsin. Bu örnekte, Fill özelliğinin ihtiyaç duyduğu SolidColorBrush nesnesi, sadece "Red" metinsel ifadesini kullanmak yerine açık bir şekilde tanımlanıyor. Bu örneğe baktığında, property elementi sentaksının aynı işi yapmaya yarayan gereksiz bir alternatif olduğu düşünülebilir. Fakat daha basit olan nitelik değeri atama yöntemini kullanarak her türlü değerin atanması mümkün değildir. Örneğin, bir nesnenin özelliğine atanacak bilgi, tek bir değer ile ifade edilemediğinde, property elementi sentaksına ihtiyaç duyulur. Aşağıdaki örnek bir dikdörtgen oluşturuyor. Burada basit kırmızı renkte bir dolgu yerine, geçişli bir renk (gradient) kullanılıyor.

Örnek 3:

<!--Bu dikdörtgen çaprazçizgisel geçişli (diagonal linear gradient) renklerle boyanmıştır.-->

```
<Rectangle Width="200" Height="200">
```

```
  <Rectangle.Fill>
```

```
    <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
```

```
      <GradientStop Color="Yellow" Offset="0.0" />
```

```
      <GradientStop Color="Red" Offset="0.25" />
```

```
      <GradientStop Color="Blue" Offset="0.75" />
```

```
      <GradientStop Color="LimeGreen" Offset="1.0" />
```

```
    </LinearGradientBrush>
```

```
  </Rectangle.Fill>
```

```
</Rectangle>
```



Şekil 5 – Örnek 3’ün ekran çıktısı

Şekil 5 de görüldüğü gibi, Fill özelliği, geçişleri (gradient) oluşturmak için complex bir nesne olan LinearGradientBrush nesnesini kullanıyor. Bu gibi durumlarda, bir niteliğe basitçe

metinsel bir deęer ataması yapmak yerine, property elementi sentaksını kullanılmalıdır (Lecrenski, Watson, & Ensor, 2011).

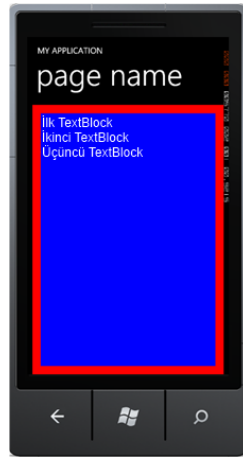
XAML ve Görsel Ağaç (Visual Tree)

XAML içerisindeki <Button> ve <Grid> gibi elementler, diğer elementleri birer düğüm (node) şeklinde kendi alt elementleri (children) olarak kabul edebilirler. Bu üst/ast (parent/child) ilişkisi nesnelerin ekrana ne şekilde yerleştirileceğini ve kullanıcı tarafından tetiklenen olaylara nasıl cevap verileceğini belirler.

Örnek 4:

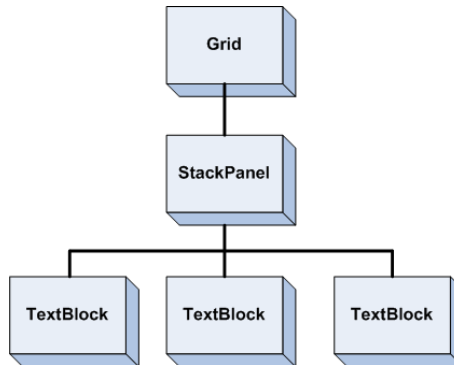
```
<Grid x:Name="ContentPanel" Background="Red" Grid.Row="1" Margin="12,0,12,0">
    <StackPanel Margin="20" Background="Blue" >
        <TextBlock Name="TextBlock1" FontSize="30">İlk TextBlock</TextBlock>
        <TextBlock Name="TextBlock2" FontSize="30">İkinci TextBlock</TextBlock>
        <TextBlock Name="TextBlock2" FontSize="30">İkinci TextBlock</TextBlock>
    </StackPanel>
</Grid>
```

Uygulamayı çalıştırdığında şu şekilde görünecek:



Şekil 6 – Örnek 4’ün ekran çıktısı

Aşağıda yer alan ağaç yapısındaki diagram, elementler arasındaki ilişkileri gösteriyor.



Şekil 7 – Örnek 4’ün ağaç yapısı

Görsel ağaç (visual tree), içeriğin nasıl sunulduğunun anlaşılması dışında, olayların (events) nasıl ele alınıp işlendiğinin anlaşılmasında da sana yardımcı olacaktır.

Birçok olay (event), tetiklendiğinde, ağaçtaki hiyerarşi üzerinde yukarı doğru diğer olayları da tetikler. Örneğin, StackPanel üzerinde sol fare tuşunun tıklanmasını (MouseLeftButtonDown event) ele alan bir olay yakalayıcı, TextBlock nesneleri tıklandığında da tetiklenecektir. Aşağıdaki XAML kodu, StackPanel nesnesinin MouseLeftButtonDown olayı için, commonMouseHandler isimli bir olay yakalayıcının nasıl ele alınması gerektiğini gösteriyor.

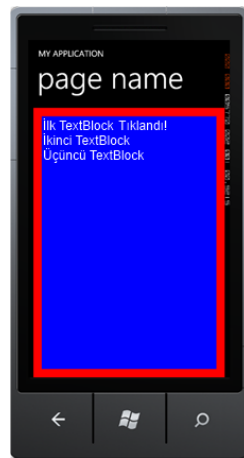
Örnek 5:

XAML kodu:

```
<Grid Background="Red" x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<StackPanel Margin="20" Background="Blue" MouseLeftButtonDown="commonMouseHandler">
<TextBlock Name="TextBlock1" FontSize="30" >İlk TextBlock</TextBlock>
<TextBlock Name="TextBlock2" FontSize="30" >İkinci TextBlock</TextBlock>
<TextBlock Name="TextBlock3" FontSize="30" >Üçüncü TextBlock</TextBlock>
</StackPanel>
</Grid>
```

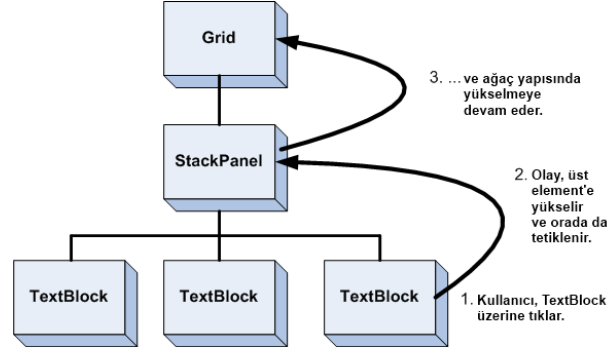
C# Kodu:

```
private void commonMouseHandler(object sender, RoutedEventArgs e)
{ FrameworkElement feSource = e.OriginalSource as FrameworkElement;
switch (feSource.Name)
{ case "TextBlock1": TextBlock1.Text = TextBlock1.Text + " Tıklandı!"; break;
case "TextBlock2": TextBlock2.Text = TextBlock2.Text + " Tıklandı!"; break;
case "TextBlock3": TextBlock3.Text = TextBlock3.Text + " Tıklandı!"; break; } }
```



Şekil 8 – Örnek 5’in ekran çıktısı

Aşağıdaki diyagram, olayların yukarıya doğru diğer nesneler üzerinde de tetiklenmesini gösteriyor. Olay (evet), ağaç üzerinde yukarı doğru tetiklenmeye devam ettiğinden dolayı, MouseLeftButtonDown olayının Grid elementi üzerinde dahi ele alınması mümkündür.



Şekil 9 – Örnek 5'in olay ağacı

Statik Kullanıcı Ara yüzünden Fazlası Gerektiğinde

XAML ile statik bir kullanıcı ara yüzü görüntülemekten fazlası yapılabilir. Sadece işaretleme dilini kullanarak, animasyonlar oluşturabilir, video ekleyebilir ve kontrollere veri bağlanması mümkündür.

4.2. Kontrollerle Çalışmak (Toolbox Kullanımı)

Windows Phone Silverlight uygulamaları için kullanıcı ara yüzü, butonlar, metin kutuları ve aşağı-açılır listeler gibi kontroller kullanılarak hazırlanır. Windows Phone kontrolleri, birçok ön tanımlı özellik sunacak şekilde tasarlanmıştır. Kontroller ile çalışırken genel olarak şu işlemler yapılır:

- Uygulamanın kullanıcı ara yüzüne ilgili kontrol eklenir.
- Kontrolün, genişlik, yükseklik ve renk gibi özellikleri ayarlanır.
- Kontrolün olaylarını ele alacak iş mantığı kodlanır.

Kontrol Örneği

Aşağıdaki ekran görüntüleri, bir TextBox kontrolü ve kontrolün TextChanged olayına bağlı bir olay yakalayıcısı (event handler) ele alan örneğe aittir. Bu bölümdeki örnekte TextBox içerisine herhangi bir şey yazıldığında, TextBox kontrolünün hemen altında bir mesaj görüntülenecek.



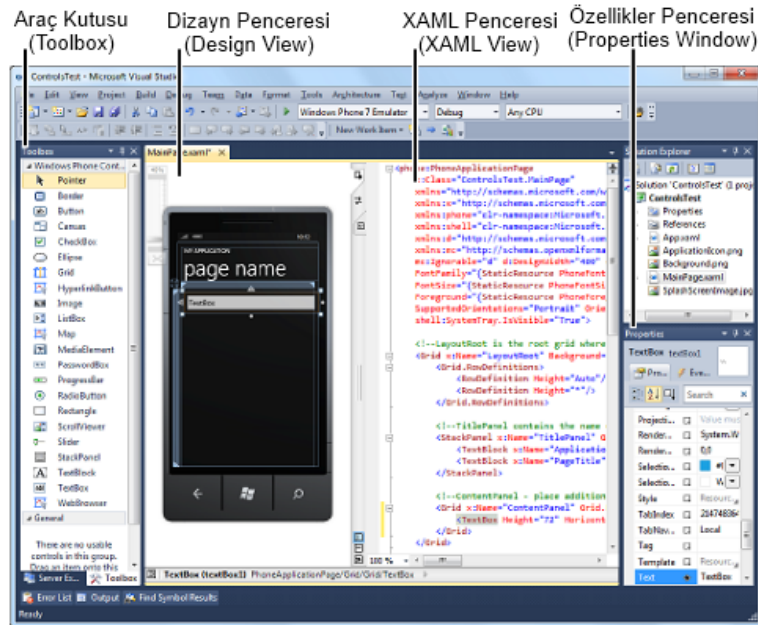
Şekil 10 – Textbox Kontrol örneği

Kontrol Ekleme

Bir Windows Phone uygulamasına kontrol eklemenin çeşitli yolları vardır:

- Visual Studio içerisindeki Toolbox'tan tasarım ekranına sürükleyip bırakmak.
- XAML penceresinde eklemek.
- Kod tarafında eklemek.

Aşağıdaki resim, Visual Studio kullanarak, Windows Phone için Silverlight uygulaması geliştirilen bir projeden alınmıştır. Visual Studio içerisinde, geliştireceğin uygulama için kontroller ekleyip ayarlarken, araç kutusu (Toolbox), tasarım penceresi (Design view), XAML penceresi (XAML view), ve özellikler penceresi (Properties window) kullanılabilir.



Şekil 11 – Visual studio ekranı

Visual Studio araç kutusu (Toolbox), uygulama içerisinde kullanılabileceğin birçok hazır kontrol barındırır. Uygulama geliştirirken çeşitli ihtiyaçları karşılayacak bazı ek kontroller

barındıran Silverlight for Windows Phone Toolkit paketini <http://silverlight.codeplex.com> adresinden indirip kurulabilir (Microsoft, 2013).

Uygulamaya bir kontrol eklemek için, Toolbox içerisinde ilgili kontrole çift tıkla. Alternatif olarak kontrolü tasarım penceresi içerisine sürükleyip bırakılabilir.

Kontrolle İsim Vermek

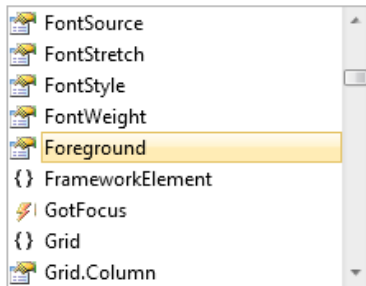
Her kontrolün bir adı olur. Kod içerisinde bir kontrol ile çalışırken kontrolün adı kullanılır. Bir kontrolün Name özelliğine (property) değer atayarak, kontrolün adını değiştirilebilir. Bu işlem Visual Studio Properties penceresinde ya da XAML penceresi içerisinde yapılabilir. Kontrol ismini XAML penceresinde değiştirmek istenirse, Name isimli niteliğe (attribute) yeni bir değer atanmalıdır (Basu, 2013).

Kontrol Özelliklerini Atamak

Bir kontrolün görünümü, içeriği ya da diğer niteliklerini belirlemek için özellikler (properties) kullanılır. Kontrol özelliklerini, Properties penceresinde, XAML içerisinde, ya da kod tarafında C# ile tanımlayabilirsin. Bir TextBox'ın içerisindeki yazı rengini değiştirmek için, kontrolün Foreground özelliğini (property) değiştirilmelidir. (Basu, 2013).

Foreground özelliğini XAML tarafında değiştirmek için niteliğin adını yazıp değer ataması yapılabilir. Visual Studio IntelliSense penceresi burada da açılarak sentaks ile ilgili yardımcı olur. (Delprato)

```
<TextBox Height="72" HorizontalAlignment="Left" Margin="113,127,0,0"
Name="tb1" Text="TextBox" VerticalAlignment="Top" Width="260"
Fore />
```

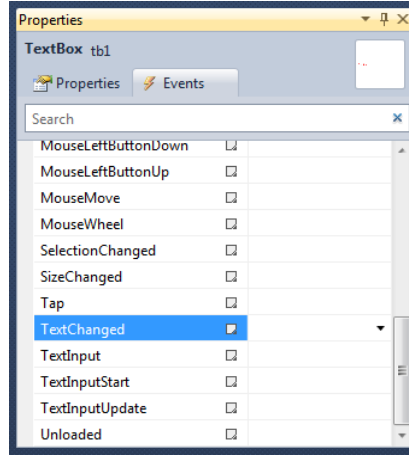


Şekil 12 - Visual Studio IntelliSense penceresi

Olay Yakalayıcı (Event Handler) Oluşturmak

Her kontrolün, kullanıcı eylemlerine cevap vermeyi sağlayacak çeşitli olayları (event) vardır. Örneğin, bir Button kontrolü Click isimli bir olaya sahiptir ve bu olay, buton tıklandığında tetiklenir. Olay yakalayıcı (event handler) tarafından kullanılacak olan bir metot yazarak, olay gerçekleştiğinde yapılması istenen işler bu metot içerisinde kodlanır. Olay yakalayıcı metotlar, Properties penceresinden ya da XAML tarafında oluşturulabilir. Ayrıca kod yazarak manuel bir şekilde oluşturulmaları da mümkündür.

Properties penceresinin Events sekmesini kullanarak bir olay yakalayıcı (event handler) oluşturulabilir. Oluşturmak için, ilgili kontrolü seçilir ve Properties penceresinin üst bölümündeki Events sekmesi tıklanır. Properties penceresinde ilgili kontrolün tüm olayları listelenir. Şekil 13 deki resimde TextBox'ın bazı olaylarını (events) görüntülenmektedir. (Microsoft Açık Akademi, 2013).



Şekil 13 – Properties (özellikler) araç kutusu

Properties penceresindeki olay adına (event name) çift tıkladığında, varsayılan isim kullanılarak bir olay yakalayıcı metot oluşturulur. Ardından, code-behind dosyası ekranda açılır. Aşağıdaki kod, tb1 isimli TextBox kontrolünün TextChanged olayını (event) ele alan olay yakalayıcı metoda aittir. TextBox içerisindeki yazı değiştiğinde, blok1 isimli TextBlock nesnesinin Text özelliği “Bilgi girildi!” olarak değiştirilecek.

Örnek 6:

```
private void tb1_TextChanged(object sender, TextChangedEventArgs e) { blok1.Text = "Bilgi Girildi!"; }
```

Olay yakalayıcıyı XAML tarafında oluşturmak istersen, XAML penceresine geç ve kontrolün niteliklerinin bulunduğu yerde ele almak istediğin olayın adını yazmaya başla. XAML ekranında *IntelliSense* penceresi açılacak. Olayı belirttikten sonra, *IntelliSense* penceresinde

gözükün <New Event Handler> üzerine çift-tıklayarak varsayılan isimle bir olay yakalayıcı oluştur.

Aşağıdaki XAML kodu TextChanged olayının, tb1_TextChanged isimli bir olay yakalayıcı metot ile ilişkilendirildiğini gösteriyor.

XAML Kodu;

```
<TextBox Height="72" HorizontalAlignment="Left" Margin="10,10,0,0" Name="tb1" Text="TextBox"
VerticalAlignment="Top" Width="260" Foreground"Red" TextChanged="tb1_TextChanged" />
```

C# kodu;

```
tb1.TextChanged +=new TextChangedEventHandler(tb1_TextChanged);
```

4.3. Kontrol Tipleri

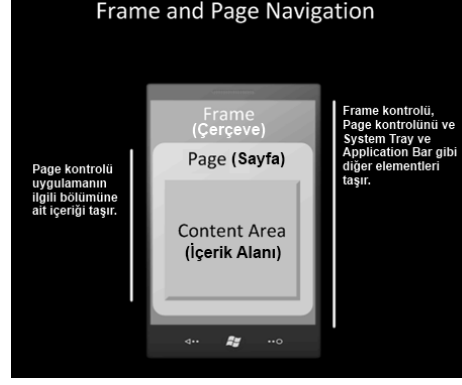
Silverlight for Windows Phone uygulamalarında kullanılabilecek çeşitli tiplerde kontroller vardır. Bu kontrollerin bazıları Silverlight masaüstü uygulamaları için de mevcutken bazıları ise Windows Phone uygulamalarına özeldir.

Navigasyon Kontrolleri (Navigation controls) Nedir?

Silverlight for Windows Phone uygulamaları kullanıcıların çeşitli içerik sayfaları arasında gezinmelerini sağlayan bir sayfa modelini temel alır. Bu model, bir çerçeve (Frame) kontrolü üzerine kurulmuştur. Bu kontrol, kullanıcıların aralarında gezinebilecekleri sayfa (Page) kontrollerini içerir. Sayfalar (Pages), yerleşim (layout) kontrollerini taşırlar. Bu kontroller de içlerinde diğer kontrolleri barındırır. Aşağıda, Windows Phone uygulamalarında navigasyon için kullanılan kontrolleri görülebilir. (Lecrenski, Watson, & Ensor, 2011)

- PhoneApplicationFrame: Silverlight for Windows Phone uygulamalarının ana kontrolüdür. Bir sayfaya ve o sayfadan geri navigasyona destek sunar.
- PhoneApplicationPage: PhoneApplicationFrame tarafından geçiş yapılacak içeriği kapsüller/taşır.

Bu iki kontrol de Microsoft.Phone adlı assembly içerisinde yer alıyor. Şekil 14 de Frame ve Page modelini görebilirsiniz.



Şekil 14 - Frame ve Page modeli gösterimi

Yerleşim (Layout) Kontrolleri

İçlerinde diğer kontrolleri içerebilen/barındırabilen kontroller, sıklıkla yerleşim (layout) kontrolleri olarak anılırlar. Yerleşim kontrolleri (Layout controls), diğer kontroller ve görsel nesneler (visual objects) için taşıyıcı olarak kullanılırlar. Adından da anlaşıldığı üzere, yerleşim (layout) kontrolleri, içerdikleri bu nesneleri ekran üzerinde pozisyonlamak için kullanılırlar. Yerleşim (layout) kontrolleri, uygulamanızın sayfa içerisindeki yerleşim kökü (layout root) olarak hareket eder. Kullanıcı arayüzündeki tüm diğer nesneler bu kök içerisinde yer alır. Bir yerleşim kontrolünün içerisinde, başka yerleşim kontrolleri kullanılabilir.

Yeni bir Windows Phone uygulaması oluşturduğunda, yerleşim kökü olarak kullanılmak üzere bir Grid kontrolü hazır gelir. Yerleşim kökü olan bu grid, kendi içerisinde başlık paneli ve içerik paneli olarak çalışan başka yerleşim kontrolleri barındırır. İhtiyaç duyulan yeni yerleşim kontrolleri olursa, bunlar içerik paneline eklenebilir. Ayrıca geliştireceğiniz uygulama için yerleşim kontrollerini seçerken uygulamanın *landscape* (yatay) ya da *portrait* (dikey) *orientation* destekleyip desteklemeyeceğine de karar vermelisin.

Yerleşim kontrollerinin büyük çoğunluğu Panel sınıfından üretilmiştir. Örneğin; *StackPanel*, *Canvas*, ve *Grid*. Ayrıca, telefon uygulamalarında kullanılabilecek olan iki kontrol daha var. Bunlar, *Panorama* ve *Pivot*. *Pivot* ve *Panorama* kontrolleri *ItemsControl* sınıfından kalıtılmıştır ve özel olarak Silverlight for Windows Phone uygulamalarında kullanılmak üzere geliştirilmişlerdir. Bu kontroller, geniş listeler barındırabilen tipik items controls'den farklıdır. Çünkü az sayıda eleman içermeleri gerekir. Bu kontroller kullanıcılara, içerdikleri elemanlar arasında kolayca swipe ve pan imkânı verir.

Windows Phone uygulamalarında yerleşim (layout) ve eleman gruplama işlemleri için kullanılan kontrollerin listesini aşağıda bulabilirsin.

Border: Başka bir kontrol için çerçeve (border) ve/veya fon (background) sağlamak için kullanılır.

Canvas: Alt elementlerinin kendisi üzerinde belirli koordinatlarda gösterilebilmesi için ortam sağlar.

ContentControl: Tek bir alt element barındıran taşıyıcı bir kontroldür. Alt element herhangi bir nesne olabilir. Ayrıca, ContentControl içerisindeki alt element de, kendi içerisinde alt elementlere sahip bir yerleşim kontrolü barındırabilir.

Grid: Satırlar ve kolonlardan oluşan bir yüzey sunar. Satırlar ve kolonları tanımlayıp, grid içerisinde istediğin satır ve kolona herhangi bir nesne atayabilirsin.

Panorama: Kenardan kenara sürüklenebilen elemanların panoramik bir görünümünü oluşturur. Silverlight for Windows Phone'a özel bir kontroldür.

Pivot: Uygulama içerisinde büyük boyutlardaki verinin sunumunu yönetmek için hızlı bir yol sağlar. Filtreleme ya da farklı sayfa görünümleri arasında geçiş yapma amacıyla navigasyon arayüzü olarak kullanılabilir. Silverlight for Windows Phone'a özel bir kontroldür.

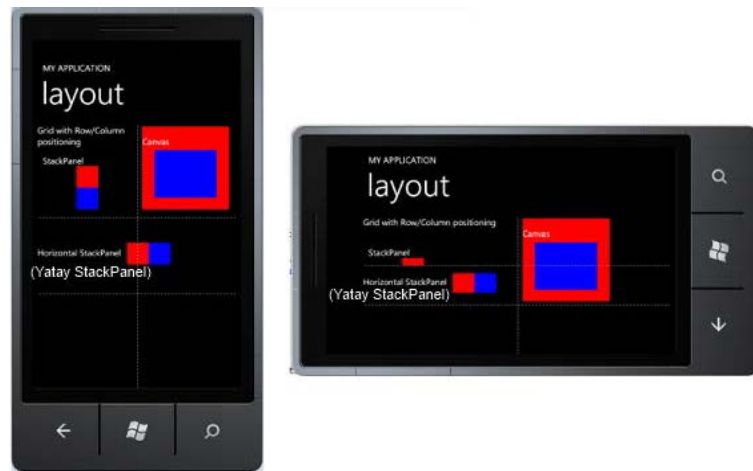
StackPanel: İçerisindeki elementleri yatay veya dikey bir çizgi halinde görüntüleme imkanı sunar.

VirtualizingStackPanel: İhtiyaç duyulduğunda ek kullanıcı arayüzü kontrolleri oluşturarak, ekranda görünür durumda olan içeriği düzenleyen bir StackPanel olarak çalışır. Kalabalık eleman kümeleri ile çalışırken, StackPanel'den daha iyi bir performans sunar.

ScrollViewer: İçerdiği elementi görüntülemek için kaydırılabilir (scrollable) bir ortam sunar.

Microsoft.Phone.Controls assembly'si içerisinde yer alırlarken, listedeki diğer kontroller ise Silverlight System.Windows assembly'si içerisinde bulunuyor.

Şekil-14 deki görsellerde bazı yerleşim kontrolleri (layout controls) hem portrait (dikey) hem de landscape (yatay) orientation'da sunulmuştur. Bu görselde, kolon ve satır tabanlı bir yerleşim sistemi kullanan Grid içerisinde StackPanel ve Canvas kontrolleri yer alıyor.



Şekil 15 – Layout kullanımı (dikey ve yatay örneği)

Sık Kullanılan Kontroller

TextBlock Kontrolü:

TextBlock windows phone uygulamalarında metin göstermek için tercih edeceğin ilk kontroldür. Başka bir kontrol için başlık ya da açıklama göstermeye ihtiyacın olduğunda etiket olarak kullanabilirsin.

TextBlock, diğer kontrollerin durumundan bağımsız olarak değişmeyen, sabit boyutlu dikdörtgen bir kutudur. Bu kontrol birden fazla satırda yazı yazma alanı sunar. Son kullanıcılar, TextBlock kontrolleri ile etkileşim içine giremezler.

Windows Phone cihazının güçlü yanlarından bir tanesi bilgiyi kullanıcıya temiz, güzel bir tipografi ile göstermektir. TextBlock kontrolünü kullanıcıya sunarken bu tasarım özelliğini avantaja çevirmek için az, öz, sade, okunaklı kelime ve etiketler kullanmalısın.

TextBlock kontrolünde gösterilecek metinsel bilgiyi Text özelliği ile belirleyebilirsin.

```
<TextBlock Text="Merhaba Windows Phone!"/>
```

Alternatif olarak etiketlerin arasında göstermen de yeterli olur.

```
<TextBlock> Merhaba Windows Phone! </TextBlock>
```

Kendisini kapsayan panel, grid vb. kontrollerin içinde yerleşimlerini ayarlamak için şu üç özelliğini kullanabilirsin:

- TextAlignment
- HorizontalAlignment
- VerticalAlignment

XAML içinde TextBlock kontrolünün alt öğeleri olarak Run ve LineBreak nesneleri kullanılabilir. Bunlardan metinleri formatlanmış olarak göstermek için faydalanırsın. Run nesnesi, TextBlock içeriğindeki diğer bölümlerden bağımsız olarak formatlanabilen ayrı bir bölgeyi temsil eder. LineBreak nesnesi ise TextBlock içinde özellikle koymak istediğin bir satır atlama nesnesidir.

Örnek olarak aşağıdaki XAML kodlarını inceleyebilirsiniz:

```
<TextBlock FontFamily="Times New Roman" Width="380" Text="Örnek metin">
<LineBreak/>
<Run Foreground="Cyan" FontFamily="Calibri" FontSize="22">Calibri 22</Run>
<LineBreak/>
<Run Foreground="Teal" FontFamily="Comic Sans MS" FontSize="20" FontStyle="Italic">Comic Sans MS
Italic 20</Run>
<LineBreak/>
<Run Foreground="CadetBlue" FontFamily="Tahoma" FontSize="16" FontWeight="Bold">Tahoma 16</Run>
</TextBlock>
```

TextBox Kontrolü:

Son kullanıcıdan e-posta, adres, fiyat vb. bilgileri almak için TextBox’a ihtiyaç duyacaksınız. Silverlight Textbox kontrolü, Web ve masaüstü programlama yaparken kullandığınız TextBox ile benzer niteliklere sahiptir. İçeriğini elde edip kullanmak ve güncellemek için yine Text özelliğini kullanırsın.

Bu kontrole telefon ekranında dokunduğunda önce bir imleç (cursor) görünür ve ardından içini doldurmak için ekran üzerindeki klavye ortaya çıkar. Klavye ile işin bittiğinde Return butonuna basarsan TextBox yazdıklarını ele alır ve artık uygulamanın kullanması için hazır hale getirir. TextBox içeriğini tek satır ya da çok satırda gösterebilirsiniz. Çok satırlı textbox, kontrolün genişliğine göre içindeki yazıyı alt satırlara kaydıracaktır.

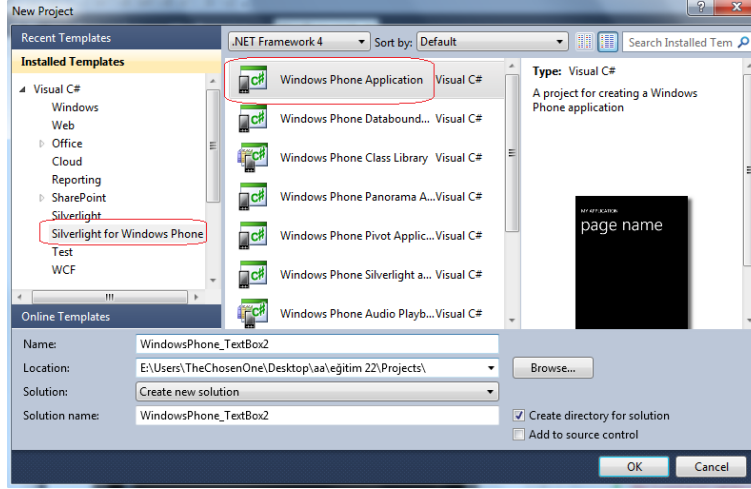
Uzunluğunu kısıtlamak dışında, TextBox girilmesi gereken geçerli değer için ipucu veren bir görselliğe sahip değildir. Hemen yanında, girilmesi gereken değer ne olması gerektiği ile ilgili bir ifade yazmalısın. Anlaşılması kolay görsel ipuçları, daha az hata mesajı ile karşılaşmanı sağlar ve uygulamadaki veri girişinin hızlı ve sorunsuz olduğu algısını yaratmaya yardım eder.

TextBox kontrolü sadece okunabilir olarak belirlenebilse de genellikle metin ifadelerin girilmesi ve güncellenmesi için kullanılır. Veri girişi yaparken 2 ayrı hareket desteklenir:

- Dokunma: Fokuslanmak ve seçim yaparak ekran-içi klavye çıkmasını sağlar.
- Dokunma ve 1-2 saniye tuttuktan sonra bırakma: İçine yazılan yazının karakter arasına girmeyi sağlar.

Örnek Uygulama

Adım 1: Yeni bir proje aç. Visual Studio ile File - New - Project adımlarını takip et. Karşına New Project penceresi açılacak. Solda Silverlight for Windows Phone grubunu, ortada Windows Phone Application şablonunu seç.



Şekil 16 – Yeni proje açma ekranı

Adım 2: Uygulama ve sayfa başlıklarını değiştir. Sayfanın en üstünde iki tane TextBlock kontrolü yer alır. Uygulama başlığı en yukarda yer alır ve küçük boyuttadır. Sayfa başlığı yine telefon ekranının üst kısmındadır ve büyük boyuttaki başlıktır. Bu bilgileri değiştirmek için iki yol vardır. Birincisi görsel arayüzden, ikincisi xaml kodundandır.

TitlePanel isimli *StackPanel* içindeki bu TextBlock'ların üstteki küçük boyutta olanını yani uygulama başlığını xaml kodu içinden değiştir. Bunun için xaml kodunu aşağıdaki şekilde değiştir ve uygulama adı olarak WINDOWS PHONE yaz.

```
<TextBlock x:Name="ApplicationTitle" Text="WINDOWS PHONE" Style="{StaticResource PhoneTextNormalStyle}"/>
```

Büyük fontlu olan sayfa başlığına ait TextBlock kontrolünü ise görsel arayüzden değiştir. Bunun için varsayılan olarak arayüzde page name yazan PageTitle isimli kontrolü seçip F4'e bas. Açılan Properties penceresinde Text özelliğini değiştir ve buraya TextBox Örnek yaz.

Adım 3: Hem yazılabilir hem okunabilir bir TextBox ve bu TextBox kontrolüne ait açıklama için bir TextBlock kontrolü ekle. Görsel arayüze ekleyeceğin kontrolleri ContentPanel Grid'inin içine eklemelisin. ToolBox penceresinden bir tane TextBlock, yanına bir tane TextBox sürükleyip bırak.

TextBlock, yanına ekleyeceğin kontrollerin ne işe yaradığını son kullanıcıya açıklayan ifadelerin yazılması için kullanılır. Bunun Text özelliğine Birşeyler Yaz olarak güncelle. Bunu yaparken her zaman olduğu gibi iki seçeneğin var:

- XAML kodu içinden
- Görsel arayüzden kontrolü seçip Properties penceresinden

Varsayılan olarak TextBox kontrolünün içeriğini hem değiştirebilirsin hem de elde edebilirsin. TextBox ile ilgili ilk alman gereken aksiyon Name özelliğini değiştirmektir. Arayüze eklediğin bir kontrolü eğer kod tarafında kullanman gerekiyorsa ilk iş olarak Name özelliğini değiştirmen gerekir. Böylece kod yazarken hangi kontrolü yönettiğini kolayca anlayabilirsin. TextBlock kontrolünün hemen sağına sürükleyip bıraktığın TextBox için Name özelliğini txtGuncellenebilir olarak belirle. Bunu en kolay Properties penceresinin en üst kısmından yapabilirsin.

Son olarak TextBox için Text özelliğini temizlemen gerekir, burası kullanıcının girmesini istediğin yerdir.

Adım 4: Sadece okunabilir bir TextBox ve bu TextBox kontrolüne ait açıklama için bir TextBlock kontrolü ekle. Önceki adımda eklediğin kontrollerin hemen altına ToolBox penceresinden birer tane daha TextBlock ve TextBox ekle. TextBlock için Text özelliğini Sadece okunabilir olarak güncelle. TextBox için ise Text özelliğini temizle. Ayrıca TextBox kontrolünün Name özelliğini txtOkunabilir olarak belirle.

Ekrandaki ikinci TextBox kontrolünün genişlik ve yüksekliğini üstteki TextBox gibi azaltabilirsin. Bunun için genişliği 279, yüksekliği 59 olarak belirle. Küçülen ekrana yazıların sığması için FontSize özelliğini 16 yap.

TextBox kontrolüne son kullanıcının yazı yazmasını engellemek için IsReadOnly özelliğine true değer ataman gerekiyor. Properties penceresinden yapabileceğin gibi xaml kodundan da bu kısıtlamayı gerçekleştirebilirsin. Bu durumda txtOkunabilir TextBox kontrolü için xaml şöyle olacaktır:

```
<TextBox Height="59" HorizontalAlignment="Left" Margin="177,107,0,0" Name="txtOkunabilir" Text="" VerticalAlignment="Top" Width="279" FontSize="16" IsReadOnly="True" />
```

Uygulamayı çalıştırıp üstteki TextBox kontrolüne dokunduğunda yazı yazmak için klavye çıkarken, sadece okunabilir olarak belirlediğin alttaki için klavye çıkmaz.

Adım 5: TextChanged olayını ele al. Son kullanıcının ekranda üstte yer alan TextBox kontrolüne yazdığı yazıyı alttaki TextBox kontrolünde göstermek gerekiyor. Bunun için ilk yapmak gereken şey txtGuncellenebilir kontrolünün olay listesine gelmektir. Kontrolü seçtikten sonra F4'e bas, Properties penceresinin üst kısmında Events tabını seçilir ve listenin alt kısımlarında yer alan TextChanged olayına çift tıklanır.

Bu olay bir TextBox içine yazılan veya silinen her karakter için tetiklenir.

Bunu yaptığında tasarım ekranında 2 şey otomatik olarak gerçekleşir. Senin ekstra birşey yapmaya gerek yoktur.

- Birinci olarak MainPage.xaml.cs içinde olayı ele alacağın olay metodu otomatik olarak yazılır.

```
private void txtGuncellenebilir_TextChanged(object sender, TextChangedEventArgs e)
{ txtOkunabilir.Text = txtGuncellenebilir.Text; }
```

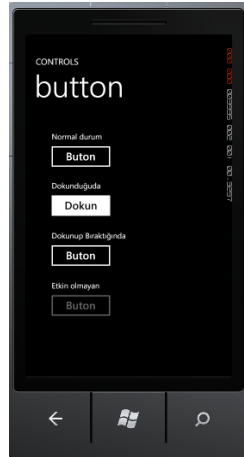
- İkinci olarak txtGuncellenebilir TextBox kontrolü için xaml kodunda ilgili bildirimin yapıldığı anahtar-değer çifti eklenir.



Şekil 17 – Textbox örneği görüntüsü

Buton Kontrolü:

Bir buton kontrolü, son kullanıcı üzerine dokunduğunda uygulamanın bir aksiyon almasına sebep olur. Son kullanıcının buton ile ekran etkileşimlerine nasıl cevap verdiğini burada görüntülenir (Lecrenski, Watson, & Ensor, 2011).



Şekil 18 – Buton kontrolü ekran görüntüsü

Butonun üzerinde görünecek olan yazı ya da içerik Text değil Content özelliği ile atanır. Bunun temel sebebi Button sınıfının ContentControl tipinden türemesidir. Örnekte ilk verilen Content özelliğidir.

Örnek 7:

```
<Button Content="Dokun" Height="72" HorizontalAlignment="Left" Margin="37,172,0,0" Name="button2"
VerticalAlignment="Top" Width="160" Background="White" Foreground="Black"></Button>
```

Butona dokunduğunda kod tarafında Click olayı tetiklenir. Varsayılan olarak eğer butona fokuslanılmışsa klavyede ENTER ya da SPACE tuşlarına basmak da Click olayını tetikler.

Butonun 3 farklı ClickMode değeri vardır.

- Release
- Press
- Hover

ClickMode özelliği varsayılan, Release olarak gelir. Bunun anlamı, son kullanıcı butona bastığında değil bıraktığında Click olayı metodu için yazdığın kod çalışır. Eğer bastığında anda çalışmasını istenirse değeri Press yapılmalıdır. Hover ise ekranın başka bir yerine

dokunup, bırakmadan butonun üzerine gelene kadar dokunmaya devam etmesi durumunda Click olayının tetiklenmesine olanak tanır. ClickMode özelliğinin değeri Hover olduğunda, olayın tetiklenmesi için klavyedeki tuşları kullanılamaz.

ClickMode özelliklerini daha yakından tanıman için yeni bir proje aç ve ContentPanel isimli Grid içine aşağıdaki xaml kodunda yer alan üç tane butonu hazırla. Butonların her biri farklı ClickMode değerlerine sahiptir.

Örnek 8:

```
<Button Content="Parmağını üzerine sürükle" Height="72" HorizontalAlignment="Left" Margin="38,46,0,0"
Name="btnHover" VerticalAlignment="Top" Width="337" FontSize="16" ClickMode="Hover"
Click="btnHover_Click" />
```

```
<TextBlock Height="30" HorizontalAlignment="Left" Margin="54,124,0,0" Name="txtHover" Text=""
VerticalAlignment="Top" />
```

```
<Button Content="Dokunur dokunmaz" FontSize="16" Height="72" HorizontalAlignment="Left"
Margin="38,173,0,0" Name="btnPress" VerticalAlignment="Top" Width="337" ClickMode="Press"
Click="btnPress_Click" />
```

```
<TextBlock Height="30" HorizontalAlignment="Left" Margin="54,251,0,0" Name="txtPress" Text=""
VerticalAlignment="Top" />
```

```
<Button Content="Dokunup Bıraktığında" FontSize="16" Height="72" HorizontalAlignment="Left"
Margin="38,304,0,0" Name="btnRelease" VerticalAlignment="Top" Width="337" ClickMode="Release"
Click="btnRelease_Click" />
```

```
<TextBlock Height="30" HorizontalAlignment="Left" Margin="54,382,0,0" Name="txtRelease" Text=""
VerticalAlignment="Top" />
```

Butonların Click olaylarını içeren kodda ilgili metotlardan TextBlock kontrollerine modları açıklayan yazılar yazılır.

```
public partial class MainPage : PhoneApplicationPage
{
    // Constructor
    public MainPage()
    {
        InitializeComponent();
    }
    private void btnHover_Click(object sender, RoutedEventArgs e)
```

```

{
txtHover.Text = "Butonun üstüne kadar dokunmaya devam ettin";
//Diğer textblock kontrollerini temizle
txtPress.Text = string.Empty;
txtRelease.Text = string.Empty;
}
private void btnPress_Click(object sender, RoutedEventArgs e)
{
txtPress.Text = "Butona dokunur dokunmaz Click tetiklendi.";
//Diğer textblock kontrollerini temizle
txtHover.Text = string.Empty;
txtRelease.Text = string.Empty;
}
private void btnRelease_Click(object sender, RoutedEventArgs e)
{
txtRelease.Text = "Butona dokunup bıraktığında Click tetiklendi.";
//Diğer textblock kontrollerini temizle
txtHover.Text = string.Empty;
txtPress.Text = string.Empty;
}
}

```

Dizayn Önerileri

Windows Phone işletim sistemli telefonlar üzerinde yazılım geliştirirken buton ile ilgili dikkat etmek gereken bazı tasarım prensipleri vardır (Lecrenski, Watson, & Ensor, 2011).

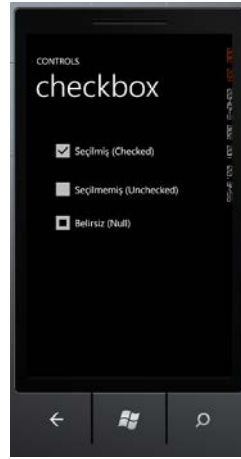
- Butonun üzerine yazdığın yazı (Content özelliği) 2 kelimeden fazla olmamalıdır.
- Mümkün olduğunca sistem fontlarını kullanmaya çalışmalıdır.
- Butonun üzerindeki yazı, kısa ve genellikle bir fiil olmalıdır.
- Eğer Metro stili dizaynına sahip bir uygulama geliştiriyorsan, butonun görünümünü varsayılan halinde bırakmak, görsel açıdan en uygun çözüm olur.
- Özelleştirilmiş bir butonu, ancak belli bir senaryo gereği farklı bir buton mekanizması gerekiyorsa tercih et.

Checkbox Kontrolü

CheckBox kontrolü 3 farklı duruma sahip olabilir:

- Checked (Seçili)
- Unchecked (Seçili değil)
- Indeterminate (Belirsiz)

Bu durumları gösteren ekran görüntüsünü Şekil 19 da görülmektedir.



Şekil 19 – Checkbox kullanım örneği

CheckBox kontrolünü kullanıcıya bir listeden seçim yapabilmesi adına seçenek listesi sunmak için kullanırsın. Örneğin bir ankette ilgilendiği teknolojileri işaretlemesini istediğin zaman checkbox ile kullanıcıya seçenekler sunabilirsin. Kullanıcı seçimini değiştirmek için ekranda kontrolü üstüne dokunmak yeterlidir (Lecrenski, Watson, & Ensor, 2011).

CheckBox kontrolünün yanına yazacağın açıklayıcı etiketi Content özelliği ile belirlenir.

Örnek 9:

```
<CheckBox Content="Seçilmiş (Checked)" Height="72" HorizontalAlignment="Left" Margin="45,44,0,0" Name="checkBox1" VerticalAlignment="Top" IsChecked="True" />
```

gSeçili ve seçilmemiş durumları dışında belirsiz durumundan faydalanmak için kontrolün IsThreeState özelliğine True değer vermen yeterlidir.

```
IsThreeState="True"
```


CheckBox kontrolünün RadioButton ve ListBox ile belli karakteristik özellikleri aynıdır. Ancak pratikteki kullanımları farklılıklar gösterir.

ListBox Kontrolü:

Listbox, bir dizi öge gösteren bir kontroldür. Bir ListBox içinde aynı anda birden fazla sayıda öge görünür olabilir. SelectionMode özelliğini kullanarak kullanıcıya birden fazla seçim yapma imkânı tanır.

Items ya da ItemsSource özelliklerini kullanarak içeriğini belirleyebilirsiniz. Bunu dersen dizayn zamanında dersen çalışma zamanında yapman mümkündür. ListBox kontrolünü doğrudan veri bağlayabileceğin gibi, bir koleksiyon içeriğine bağlaman da mümkündür (ObservableCollection<T> gibi).

Kullanıcı ListBox kontrolünün içeriğini görüntülemek için dikey olarak parmağını yukarı ya da aşağıya kaydırabilir. Listedeki bir seçim yapmak için ilgili ögenin üzerine gelip dokunması yeterlidir.

ListBox kontrolünü aşağıya doğru kaydırıldığında sağ tarafta küçük bir kaydırma çubuğu (scrollbar) belirir. Bu çubuktan, listedeki öğeler içinde ne kadar yukarıda ya da aşağıda olduğunu göstermek için faydalanır (Lecrenski, Watson, & Ensor, 2011).



Şekil 20 – Listbox kullanım örneği

Örnek 10:

Şehir listesi için ContentPanel isimli Grid içinde şu xaml içeriği:

```
<ListBox Height="245" HorizontalAlignment="Left" Margin="12,34,0,0" Name="listBox1"
VerticalAlignment="Top" Width="436" FontSize="32" />
```

ListBox içeriğini kodla doldurmak için ekran yüklendikten sonra çalışan Loaded olayını ele almak gerekir.

```
private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
{
    listBox1.Items.Add("İstanbul");
    listBox1.Items.Add("Ankara");
    listBox1.Items.Add("Antalya");
    listBox1.Items.Add("İzmir");
    listBox1.Items.Add("Aydın");
    listBox1.Items.Add("Konya");
    listBox1.Items.Add("Kocaeli");
    listBox1.Items.Add("Bursa");
}
```

ListBox kontrolünü ekranda kalıcı olarak görünmesini istediğin kelime, sayı ya da görsel öğelerden oluşan uzun bir listeden kullanıcının seçim yapmasına ihtiyaç duyduğunda tercih edilir.

Genellikle kullanıcı 8 ya da daha fazla öğe içinden seçim yapacaksa ListBox kontrolünü kullan. Eğer seçtirilecek 4 ya da daha az öğe varsa RadioButton daha doğru bir tercih olacaktır.

ListBox dikey olarak bir liste sunar. Eğer yatay olarak bir liste sunmak istiyorsan ve bu öğeler özellikle grafik ya da fotoğraf ise ScrollViewer kontrolünü kullanılır.

Popup Kontrolü

Genellikle belirli bir işi sonuçlandırdıktan sonra Popup kontrolünü kullanılır. Örneğin kullanıcı ekran üzerindeki işaretçiyi parmağıyla belirli bir kontrol üzerine getirdiğinde yardım bilgisi çıkarmak için Popup kontrolünden faydalanılır.

Popup içeriğini hazırlamak için Child özelliğini kullanman gerekir. Zorunda olmamakla birlikte içeriği bir UserControl olarak hazırlayabilirsin; böylece farklı ihtiyaçlarda kullanman mümkün olur (Lecrenski, Watson, & Ensor, 2011).

Popup kontrolünün ekrandaki pozisyonunu Silverlight ekranının sol üst köşesine göre ayarlarsın. Bunu yaparken VerticalOffset ve HorizontalOffset özelliklerini kullanılır.

Popup içeren dışardaki kontrolü kullanıcının ölçeklendirmesi ve döndürmesi durumunda Popup kontrolü de bu değişiklikleri uygular. Popup kontrolünü kullanıcıya göstermek için çalışma zamanında IsOpen özelliğine true değer vermek gerekir. Aynı şekilde kapatman gerektiğinde false değer atanılır.

```
Popup.IsOpen = true;
```

Popup kontrolünü kullanmak için Mainpage.xaml.cs dosyasına şu isim alanını eklemelisin:

```
using System.Windows.Controls.Primitives;
```

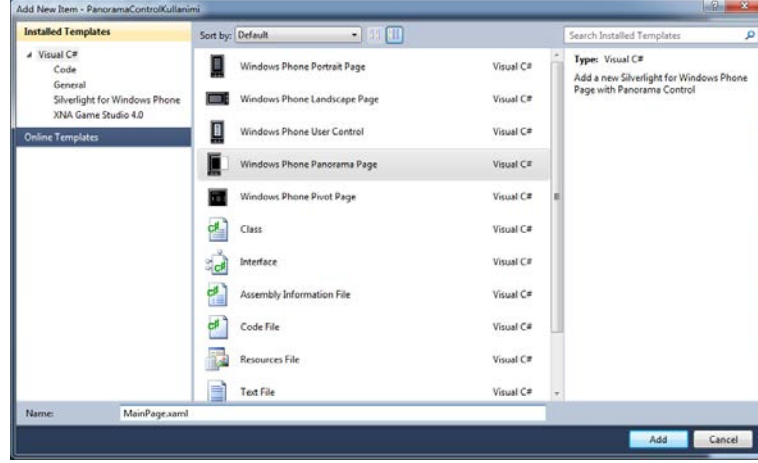
Windows Phone Panorama Page Kontrolü:



Şekil 21 – Windows phone panorama page kontrol görüntüsü

Panorama kontrolü, arka planda yer alan panoramik bir görsel üzerinde birden fazla sayfa görünümü barındırabilir. Her bir sayfa içeriği birbirinden farklı tasarımda olabilir, farklı nesneler barındırabilir ve veri görüntüleyebilir. Parmak ile yatay düzlemde sağa ya da sola yapılan hareketler ile sayfalar arasında geçiş sağlanır. Bu sırada arka plandaki görsel de ekranda otomatik olarak kaydırılır. Arka plan olarak kullanılan ve ekran sınırlarının dışına taşan bu büyük boyuttaki fotoğraf ve resimler, görselin görüntü kalitesini kaybetmeden görüntülenmiş olur. (Lecrenski, Watson, & Ensor, 2011)

Şimdi, Visual Studio'yu açılır ve File menüsü altındaki New Project seçeneğini kullanarak Silverlight For Windows Phone şablonlarından bir Windows Phone Application projesi oluşturulur.



Şekil 22 – Panorama örneği

Proje adı olarak “PanoramaControlKullanimi” kullanabilirsiniz.

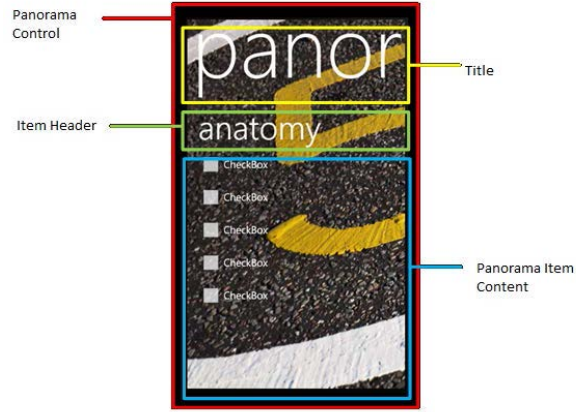
Proje açıldıktan sonra, Solution Explorer penceresinde yer alan MainPage.xaml dosyasına sağ tıklayıp Delete seçeneğini kullanarak bu dosya silinir.

Ardından yine Solution Explorer penceresinde PanoramaControlKullanimi adlı projeye sağ tıklayıp, Add > New Item seç ve açılan pencereden bir Windows Phone Panorama Page eklenir.

Ekleyeceğin sayfa için “MainPage.xaml” adını kullan. Böylece uygulamanın bu sayfa ile başlatılarak açılmasını sağlar.

Panorama kontrolü ile panoramik bir fotoğraf görüntülerken, aynı zamanda bu kontrolün üzerinde, veri kontrolleri, listeleme kontrolleri ya da fotoğraf ve animasyon gibi diğer nesneleri de kullanılabilir. Bu kontrolü kullanmak için parmağını bir yandan diğer yana kaydırarak nesneler üzerinde dolaşılabilir.

Panorama kontrolüne ait bir görsel şekil 23 de yer almaktadır. Bu kontrol, arka planda panoramik bir görsel barındırıyor ve sayfanın üst kısmında başlık yer alıyor. Başlığın altında 'Panorama Item' bölümü bulunuyor. Bu bölüm, başlık (Item Header) ve içerik (Panorama Item Content) üyelerinden oluşur. Parmak ile ekranı sağa veya sola doğru kaydırma hareketi sıradaki Item'a geçiş yapılmasını sağlar.



Şekil 23 - Genel Panorama görüntüsü

Başlık tanımlamasını yaptıktan sonra kontrolün arka planına bir görsel eklenmelidir. Bunun için öncelikle projeye sağ tıklayıp Add > Existing Item seçeneği ile sayfanın arka planında yer almasını istediğin görseli projeye eklenir. Bu görseli Panorama sayfana eklemek için aşağıda yer alan kodları, XAML kod sayfadaki `<!--Panorama item one-->` ifadesinin hemen öncesinde yazılmalıdır.

Örnek 11:

```
<controls:Panorama.Background>
<ImageBrush ImageSource="/PanoramaControlKullanimi;component/BrisbanePanorama.png" />
</controls:Panorama.Background>
```

Bu örnekte “*BrisbanePanorama.png*” projeye eklediğimiz görselin dosya adıdır. Görseli ekledikten sonra tasarım ekranındaki görüntü aşağıdaki gibi olacaktır.

Şimdi, varsayılan (default) olarak iki item barındıran bu kontrol içerisine item eklenmelidir. İlk iş olarak `<!--Panorama item one-->` altında yer alan kodları sil ve onların yerine aşağıdaki kodları eklenir.

```
<controls:PanoramaItem Header="Tarihi Bilgi" Orientation="Horizontal">
<Grid>
<StackPanel>
<TextBlock Text="1824 yılında kurulmuştur." FontSize="22"/>
<TextBlock Text=" " />
<TextBlock Text="1834'te Lord Brisbane'ın adını aldı." FontSize="22" TextWrapping="Wrap"/>
```

<TextBlock Text=" "/>

<TextBlock Text="1842'de serbest sömürge merkezi oldu." FontSize="22" TextWrapping="Wrap"/>

<TextBlock Text=" "/>

<TextBlock Text="Tarım üretimi ile hızla gelişti." FontSize="22" TextWrapping="Wrap"/>

<TextBlock Text=" "/>

<TextBlock Text="Limanın trafiği 10 megaton civarındadır." FontSize="22" TextWrapping="Wrap"/>

</StackPanel>

</Grid>

</controls:PanoramaItem>

Aşağıda yer alan ve ikinci item'a ait olan kodları da <!--Panorama item two--> ifadesinin altında yer alan kodları sildikten sonra aynı alana eklenir.

<controls:PanoramaItem Header="Demografi" Orientation="Horizontal">

<Grid>

<StackPanel>

<TextBlock Text="Nüfusu 2,043,185'in üzerindedir." FontSize="22"/>

<TextBlock Text=" "/>

<TextBlock Text="Avustralya'nın üçüncü en büyük şehridir." FontSize="22"/>

<TextBlock Text=" "/>

<TextBlock Text="Nüfusun %1,7'sinin etnik kökeni yerlidir." FontSize="22"/>

<TextBlock Text=" "/>

<TextBlock Text="Nüfusun %21,7'si yurtdışında doğdu." FontSize="22"/>

<TextBlock Text=" "/>

<TextBlock Text="Evlerin %16,1'inde yabancı bir dil konuşuluyor." FontSize="22"/>

</StackPanel>

</Grid>

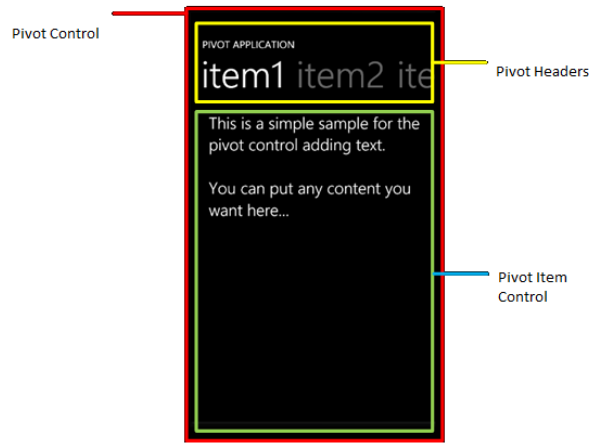
</controls:PanoramaItem>



Şekil 24 – Panorama örnek 11 ‘in ekran görüntüsü

Pivot Kullanım Örneği:

Pivot kontrolü, bir önceki konuda incelediğimiz panorama kontrolü ile benzerlik göstermektedir. Burada ana taşıyıcı kontrol Pivot kontrolüdür ve bu kontrol kendi içerisinde Item'lar barındırır. Aynı panoramada olduğu gibi parmakla sağa sola kaydırma yaparak Item'lar arasında geçiş sağlanır.



Şekil 25 - Pivot kullanım örneği

Pivot kontrolünde üstte başlık (Pivot Headers) kısmında Pivot uygulamasının adı yer alır ve yine başlık içerisinde Pivot uygulamasının adının hemen altında da Item isimleri bulunur. Header bölümünün altındaki bölümde ise Item içeriği görüntülenir (Lecrenski, Watson, & Ensor, 2011).

Örnek 11 de kullanılan Item'ları Pivot projesine eklenir. Bu amaçla öncelikle “PivotControlKullanimi” isimli yeni bir proje oluşturulur ve daha önce panorama kontrolünü incelerken yapılan gibi MainPage.xaml dosyasını silerek projeye yine MainPage.xaml

isminde bir Windows Phone Pivot Page kontrolü eklenir. Bu işlemleri gerçekleştirdikten sonra projenin XAML kodlarında Pivot kontrolünün aşağıdaki gibi yer aldığı görülebilir.

Örnek 12:

```
<!--Pivot Control-->
<controls:Pivot Title="MY APPLICATION">
<!-- Pivot item one-->
<controls:PivotItem Header="item1"><Grid/>
</controls:PivotItem>
<!-- Pivot item two-->
<controls:PivotItem Header="item1"><Grid/>
</controls:PivotItem>
</controls:Pivot>
```

Kodlarda görüldüğü üzere Pivot kontrolünün başlık (Title) bilgisi MY APPLICATION değerini taşıyor ve içerisinde de varsayılan (default) olarak iki Item bulunuyor. Yeni bir proje açtığında bu kodlara karşılık gelen görsel tasarım aşağıdaki gibi olacaktır.



Şekil 26 – Örnek 12 ekran görüntüsü

İlk iş olarak Title bilgisini “Brisbane Australia” olarak ayarlanır. Ve <!--Pivot item one--> altında yer alan PivotItem’ı sil ve aşağıda kodları yer alan Item’ı bu alana eklenir.

Örnek 12 devam:

```
<controls:PivotItem Header="Tarihi Bilgi">
<Grid>
<StackPanel Margin="10">
<TextBlock Text="1824 yılında kurulmuştur." FontSize="22"/>
<TextBlock Text=" "/>
<TextBlock Text="1834'te Lord Brisbane'ın adını aldı." FontSize="22" TextWrapping="Wrap"/>
```



```

<TextBlock Text=" "/>
<TextBlock Text="1842'de serbest sömürge merkezi oldu." FontSize="22" TextWrapping="Wrap"/>
<TextBlock Text=" "/>
<TextBlock Text="Tarım üretimi ile hızla gelişti." FontSize="22" TextWrapping="Wrap"/>
<TextBlock Text=" "/>
<TextBlock Text="Limanın trafiği 10 megaton civarındadır." FontSize="22" TextWrapping="Wrap"/>
</StackPanel>
</Grid>
</controls:PivotItem>

```

Hemen ardından `<!--Pivot item two-->` altında yer alan kodları da aşağıdaki kodlarla değiştir.

```

<controls:PivotItem Header="Demografi">
<Grid>
<StackPanel Margin="10">
<TextBlock Text="Nüfusu 2,043,185'in üzerindedir." FontSize="22"/>
<TextBlock Text=" "/>
<TextBlock Text="Avustralya'nın üçüncü en büyük şehridir." FontSize="22"/>
<TextBlock Text=" "/>
<TextBlock Text="Nüfusun %1,7'sinin etnik kökeni yerlidir." FontSize="22"/>
<TextBlock Text=" "/>
<TextBlock Text="Nüfusun %21,7'si yurtdışında doğdu." FontSize="22"/>
<TextBlock Text=" "/>
<TextBlock Text="Evlerin %16,1'inde yabancı bir dil konuşuluyor." FontSize="22"/>
</StackPanel>
</Grid>
</controls:PivotItem>

```

Son olarak projeye üçüncü bir PivotItem eklenir. Bunu yapmadan önce kullanılacak görseli projeye sağ tıklayarak Add > Existing Item seçeneği ile projeye dâhil et. Sonrasında da aşağıdaki kodları kullanarak projeye üçüncü PivotItem'ı eklenir.

```

<!--Pivot item three-->
<controls:PivotItem Header="Harita">
<Grid>
<StackPanel Margin="10">
<Image Margin="10" Source="/PivotControlKullanimi;component/BrisbaneHarita.png" />
</StackPanel>
</Grid>
</controls:PivotItem>

```

Bu işlemlerin ardından tasarım ekranı aşağıdaki görsellerde sunulduğu gibi olacaktır.



Şekil 27 – Örnek 12 ekran görüntüsü

Silverlight For Windows Phone Toolkit Controls:

Windows Phone SDK ile gelen kontroller dışında, <http://silverlight.codeplex.com> adresinde yer alan Silverlight for Windows Phone Toolkit paketini kurarak ek kontroller edinilebilir. Bu sitede kontrollerin dışında, kaynak kodlar (source code) ve örnekler de yer alıyor. Toolkit ayrıca, sayfa geçişleri (*page transitions*) ve *gesture service* gibi ek bileşenler de içerir (Lecrenski, Watson, & Ensor, 2011).

AutoCompleteBox: Kullanıcı girişi için bir TextBox sunar. Bu textbox, girilen değere benzer kayıtları liste halinde sunar. Silverlight kontrollerinden birisi olan AutoCompleteBox ile benzer bir kontroldür.

ListPicker: Bir metin kutusu ve bir açılır menüden oluşan bu kontrol, kullanıcıya listede yer alan üyelerden birini seçme imkânı verir.

LongListSelector: Gelişmiş bir listeleme kontrolüdür. Üye gruplama ve sanallaştırmayı destekler.

ContextMenu: Kullanıcı bir kontrole dokunup basılı tuttuğunda görüntülenecek olan bir menü kontrolüdür.

DatePicker: Kullanıcıya tarih seçme imkânı veren bir kontroldür.

TimePicker: Zaman seçimi için kullanılabilecek olan kontroldür.

ToggleSwitch: Açma/kapama ayarı ve bir etiketten oluşan kontroldür.

WrapPanel: Yeterli alan bulunduğu sürece içerisindeki elemanları yan yana sıralayan, yer kalmadığında yeni bir satır oluşturarak elemanlarını listelemeye devam eden yerleşim kontrolüdür.

Bu kontroller, Microsoft.Phone.Controls.Toolkit assembly'si içerisinde yer alır.

Sensörler ve Servisler:

Bir windows phone cihazı, birçok donanım özelliğini barındırmak zorundadır. Bu donanım özellikleri çoğu zaman sensör olarak adlandırılır (Lecrenski, Watson, & Ensor, 2011).

Bunlara örnek olarak şunlar gösterilebilir:

- *Wi-Fi*: Telefonda GSM operatörlerinin sunduğu 3G veri erişimi hizmeti yanında kablosuz internet erişimi bulunur. Ayrıca işletim sistemi ile birlikte Internet Explorer varsayılan olarak gelmektedir.
- *Kamera*: Windows Phone içeren cihazlar en az 5 megapixel kameraya sahiptir. Yazdığınız uygulama, ihtiyacı olan görsel için kamerayı çağırıp ekrana getirebilir, kameranın çektiği imajı kullanabilir.
- *İvmeölçer*: Cihazı hareket ettirdiğinde bunu hangi yöne doğru yaptığının bilgisini elde edilir.
- *Lokasyon*: Kullanıcı izin verirse, coğrafik olarak nerede olduğunun bilgisini elde edilir. Telefon içindeki GPS donanımından faydalanarak web ya da GSM vericilerinden edindiği bilgilerle bunu yapar.
- *Titreşim*: Yazdığınız program içerisinde telefonun titreşim özelliğini tetikleyebilir.
- *FM radyo*: Program kontrolü ile aracılığıyla radyo erişilebilirdir.
- *Push Notification*: Bu servis, yazdığınız uygulamaya gelen mesaj vb. yeni taleplerin telefon dışında bir yerde birikmesini sağlar. Sadece veri değiştiği zaman telefonun bildirim almasını sağlar.

İvme Ölçer (Accelerometer)

Windows Phone işletim sistemine sahip cihaz içinde bir ivmeölçer bulunur. İvmeölçer, küçük bir donanım parçasıdır ve temel fiziğin bize orantısal hızlanma olarak söylediği gücü ölçer. Telefonu tutmaya devam ettiğinde ivmeölçer yerçekimi kuvvetine cevap verir; böylece uygulamaya telefonun göreceli olarak dünyadaki yönünü aktarır (Lecrenski, Watson, & Ensor, 2011).

Bubble Level simülasyonu, ivmeölçer kullanımı için biçimsel bir uygulamadır. Bunun yanında interaktif özellik ve animasyonlar için bir altyapı sağlar.

Örneğin kullanıcı:

- Formula 1 pistinde telefonunu sağa ve sola yatırarak direksiyonu simüle edebilir ve aracına yön veren bir sürücü olabilir.
- Topu deliklere düşürmeden aralardan geçirmeye çalışarak bitiş noktasına getirmek için telefonu sağa sola oynatarak oyun oynayabilir.
- Telefonun çaldığında ekranı aşağıya gelecek şekilde çevirdiğinde sessize alabilir.

İvmeölçer aynı zamanda sallama ya da titretme gibi ani hareketlere de cevap verebilir. Bu yolla zar ya da diğer tipte rasgele aktiviteler için faydalanılabilirsin. Müzik çalar uygulaması ile müzik dinlerken telefonu sağa çevirdiğinde sonraki, sola çevirdiğinde önceki şarkıya geçebilirsin. Telefon üzerinde yazılım geliştirmenin en zor yanlarından biri de ivmeölçer ile yaratıcı kullanım alanları bulmaktır.

İvmeölçerden gelen çıktıyı üç boyutlu bir vektör ile temsil etmen gerekir : (x, y, z) Bu değerleri, kullanıcının telefonu hangi yöne doğru hareket ettirdiğini belirlemek için kullanabilir.



Şekil 28 - İvme Ölçerin Genel Şeması

İvmeölçer sensörü, kullanıcının hareketlerinden telefona uygulanan yerçekimi gücünü tespit eder. Bunu yaparken kullanacağın en temel tip Accelerometer sınıfıdır. Sınıf olaylarından CurrentValueChanged, sensörün değeri değiştiği anda tetiklenir ve CurrentValue özelliği ile sensörün o andaki değerini elde edebilirsin.

Örnek uygulama 13 (Yöndem, 2013)

1.adım: Yeni bir proje aç. Windows Phone Application şablonunu ile yeni bir uygulama aç. Proje adı olarak “wp_ivmeolcer” verelim.

2.adım: Gerekli referansları ekleyelim. Uygulama içinde sensörleri kullanmak ve ivmeölçerden dönen üç boyutlu veriyi ele alabilmek için 2 tane kütüphane eklemek gerekiyor. Bunlardan Microsoft.Devices.Sensors kütüphanesine, cihaz içindeki küçük donanım parçasından veri akışı sağlamak için ihtiyaç vardır. Microsoft.Xna.Framework kütüphanesini ise Vector3 sınıfı ile ivmeölçerden gelen veriyi elde etmek için kullanılacaktır.

Solution Explorer içinde projenin altında References menüsüne sağ tıklayarak Add Reference öğesine tıkla ve şu 2 kütüphaneyi eklenir:

- *Microsoft.Devices.Sensors*
- *Microsoft.Xna.Framework*

3.adım: Görsel arayüzü hazırla. MainPage.xaml sayfası içinde varsayılan olarak gelen ContentPanel isimli Grid elementi içine aşağıdaki xaml kodunu yapıştırılabilir.

```
<Button Content="Başlat" Height="72" HorizontalAlignment="Left" Margin="20,10,0,0"
Name="baslatButonu" VerticalAlignment="Top" Width="160" Click="baslatButonu_Click" />
```

```

<Button Content="Durdur" Height="72" HorizontalAlignment="Right" Margin="0,10,20,0"
Name="durdurButonu" VerticalAlignment="Top" Width="160" Click="durdurButonu_Click"/>
<TextBlock Height="30" HorizontalAlignment="Left" Margin="20,100,0,0" Name="txtX" Text="X: 1.0"
VerticalAlignment="Top" Foreground="Red" FontSize="28" FontWeight="Bold"/>
<TextBlock Height="30" HorizontalAlignment="Center" Margin="0,100,0,0" Name="txtY" Text="Y: 1.0"
VerticalAlignment="Top" Foreground="Green" FontSize="28" FontWeight="Bold"/>
<TextBlock Height="30" HorizontalAlignment="Right" Margin="0,100,20,0" Name="txtZ" Text="Z: 1.0"
VerticalAlignment="Top" Foreground="Blue" FontSize="28" FontWeight="Bold"/>
<TextBlock Height="30" HorizontalAlignment="Center" Margin="6,571,6,0" Name="txtDurum"
Text="TextBlock" VerticalAlignment="Top" Width="444" />

```

Bu kodlar, ivmeölçeri başlatmak ve durdurmak için 2 tane Button ve ivmeölçerden okunan sayısal değerleri sayfada göstermek 3 tane TextBlock tanımlamaktadır. Son olarak bir TextBlock nesnesi, uygulamanın güncel durumunu göstermek için kullanılır.

4.adım: Kod dosyasına gerekli isim alanlarını ekle.

MainPage.xaml.cs kod dosyasını aç ve projeye önceden referans olarak eklediğin kütüphaneleri kullanmak için gerekli isim alanlarını ekle.

```

using Microsoft.Devices.Sensors;
using Microsoft.Xna.Framework;

```

5.adım: İvmeölçer ile çalışmak için gerekli tipte bir değişken tanımla. İvmeölçer ile çalışmak için kullanacağın temel tip Accelerometer sınıfıdır. Bu tipten sınıf düzeyinde bir değişken tanımla. Böylece sayfa içindeki bütün olay metotları ve kendi yazacağın metotları içinden rahatça bu nesneye erişebilirsin.

```

namespace wp_sensorler
{
    public partial class MainPage : PhoneApplicationPage
    {
        Accelerometer ivmeO;
    }
}

```

6.adım: Kullanılan cihazın ivmeölçeri destekleyip desteklemediği kontrol edilmelidir. Bütün cihazlar ivmeölçer desteği ile gelmezler. Bu yüzden senin uygulamanın üzerinde kurulduğu cihazın ivmeölçeri destekleyip desteklemediği kontrol edilmelidir. Sayfaya ait yapıcı metot içine bunun için şöyle bir kod yazabilirsin. Eğer cihaz ivmeölçeri desteklemiyorsa, durum bilgisini ilgili TextBlock kontrolüne yazıp, başlat ve durdur butonlarını de-aktif hale getir.

```

...
InitializeComponent();
if (Accelerometer.IsSupported == false)

```

```
{
txtDurum.Text = "Cihaz, ivmeölçeri desteklemiyor.";
baslatButonu.IsEnabled = false;
durdurButonu.IsEnabled = false;
}
```

...

7.adım: Accelerometer nesnesini hazırla Accelerometer sınıfına ait sayfa düzeyinde tanımlanan alanın nesnesini örnekleyip kullanıma hazır hale getirmek lazımdır. Bunun için yine sayfanın yapıcı metodu kullanılabilir.

- Accelerometer sınıfına ait nesneyi örnekle.
- Ardından TimeBetweenUpdates özelliğine değer vermelisin. Bu özellik, CurrentValueChanged olayının ne kadar aralıkla ivmeölçerden veri alacağını belirler. Buna 20 milisaniye gibi kısa bir aralık verilir.
- CurrentValueChanged olayına abone ol ve olay metodunu oluştur. Bu olay, ivmeölçerden her yeni veri geldiğinde tetiklenir. Böylece sen istediğin kontrolleri burada yapabilirsin ve telefonun hareketine göre yaptırmak istedikleri kodlanabilir.

```
//Yapıcı metot
public MainPage()
{ InitializeComponent();
if (Accelerometer.IsSupported == false)
{ txtDurum.Text = "Cihaz, ivmeölçeri desteklemiyor.";
baslatButonu.IsEnabled = false;
durdurButonu.IsEnabled = false;
}
ivmeO = new Accelerometer();
ivmeO.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
ivmeO.CurrentValueChanged += new
EventHandler<SensorReadingEventArgs<AccelerometerReading>>(ivmeO_CurrentValueChanged);
}
```

8.adım: CurrentValueChange olay metodunu kodlanır. Kullanıcı telefonu her hareket ettirdiğinde ivmeölçerden veri gelir. Accelerometer.TimeBetweenUpdates özelliği ile programcının verdiği zaman aralığında CurrentValueChanged olayının tetiklenmesi ile programcı bu bilgileri okuyup aksiyon alabilir.


```
void ivmeO_CurrentValueChanged(object sender, SensorReadingEventArgs<AccelerometerReading> e)
{
    Dispatcher.BeginInvoke(() => Guncelle(e.SensorReading));
}
```

Olay metodunda kullanılan Dispatcher nesnesi BeginInvoke metodu ile Guncelle isimli metodu çağırılmaktadır.

CurrentValueChanged olay metodu ana iş parçacığı (thread) dışında çalışıp arka planda veri toplama işini devam ettirir. Guncelle metodu ile farklı bir iş parçacığından ana iş parçacığında yer alan kontrol içeriklerini güncellenmelidir. Bu yüzden olay metodu veri toplama işini yaparken sayfadaki TextBlock kontrolüne bir metin yazdırma işini doğrudan yapılmamalı, yapılırsa hata alınır: *Invalid cross-thread access*.

İş parçacıkları arasındaki erişim problemini çözmek için Dispatcher nesnesi üzerinden BeginInvoke metodu ile asenkron çağrılar yapılmalıdır.

9.adım: Guncelle metodunu kodlamak. Guncelle metodu, ivmeölçerden 20 milisaniyede bir gelen anlık bilgileri sayfadaki TextBlock kontrollerine yazar. Bu metoda parametre olarak alınan AccelerometerReading yapısı (struct), telefona ait hareketin x,y,z koordinatlarını içerir.

Bu bilgiyi *CurrentValueChaned* olay metodunda *SensorReadingEventArgs<AccelerometerReading>* tipindeki e parametresinden gönderildi.

```
private void Guncelle(AccelerometerReading ioOkuma)
{
    txtDurum.Text = "veri toplanıyor...";
    Vector3 acc = ioOkuma.Acceleration;
    txtX.Text = "X : " + acc.X.ToString("0.00");
    txtY.Text = "Y : " + acc.Y.ToString("0.00");
    txtZ.Text = "Z : " + acc.Z.ToString("0.00"); }
}
```

10.adım: Başlat butonunun arkasını kodlanması. Başlat butonuna basıldığı zaman cihazın ivmeölçerden veri toplamaya başlaması sağlanmalıdır. Ayrıca güncel durumu txtDurum TextBlock kontrolüne yazdırılacak. Bunun için arayüzde btnBaslat üzerine çift tıklayarak click olay metodunun açılır ve kodlanır.

```
private void baslatButonu_Click(object sender, RoutedEventArgs e)
{
    Try
    {
        txtDurum.Text = "ivmeölçer başlatılıyor...";
    }
}
```

```

        ivmeO.Start();
    }
catch (InvalidOperationException exc)
{
    txtDurum.Text = "ivmeölçer başlatılamadı...";
}
}

```

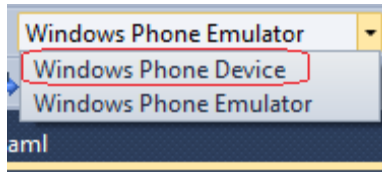
11.adım: Durdur butonunun arkasını kodlama. Durdur butonuna basıldığı zaman cihazın ivmeölçerden veri toplamayı bırakması sağlanmalıdır. Ayrıca güncel durumu txtDurum TextBlock kontrolüne yazdırılır. Bunun için ara yüzde btnDurdur üzerine çift tıklayarak click olay metodunun açılır ve kodlanır.

```

private void durdurButonu_Click(object sender, RoutedEventArgs e)
{
    ivmeO.Stop();
    txtDurum.Text = "ivmeölçer durduruldu...";
}

```

12.adım: Uygulamayı çalıştır. Uygulamayı çalıştırıp test edilebilir. Eğer cihaz üzerinde çalıştırılacaksa Visual Studio'da Windows Phone Device seçip çalıştırılabilir (developer unlock yapmak gereklidir)



Şekil 29 – Emulator başlatma ekranı

Uygulamayı emülatör üzerinde çalıştıracaksa uygulama açıldıktan sonra emülatörün sağ üst köşesindeki araç çubuğunu görünür hale getirip Additional Tools tıklanır.



Şekil 30 – Örnek 13 deki ivmeölçer örneğinin ekran görüntüsü

Bunu yaptığında ivmeölçeri emülatör üzerinde hareket ettirebileceğin bir ara yüz açılır. Telefonun ortasındaki kırmızı daireyi tutup sürükleyerek hareket ettirebilir ve uygulama burada test edilebilir.



Şekil 31 – Emülatör ve Örnek 13 Kullanımı

4.4. Konum Servisleri:

Microsoft konum servisi, Windows Phone için lokasyon odaklı uygulama geliştirmeni sağlar. GPS, kablosuz internet ve mobil internet gibi farklı kaynaklardan coğrafik konum verisi elde edebilir. Uygulamanın ihtiyaçlarına bağlı olarak güç kullanımını dengelemek için Windows Phone bu kaynaklardan bir ya da birkaçını kullanabilir. Konum, uygulamaya olay tabanlı kod ara yüzleri ile sunulur. (Microsoft Açık Akademi, 2013)

Konum Servisi Mimarisi

Cihaz için konum servisi mimarisinde ilk katman donanım bileşenlerinden oluşur. Bu bileşenler arasında GPS alıcısı, kablosuz bağlantı alanı ve mobil radyo alıcısı bulunmaktadır. Bunların hepsi konum verisi sağlayıcısı olarak çalışabilirler. Ayrıca hepsinin doğruluk seviyesi ve güç kullanımı farklıdır.

Donanım bileşenlerinin üstünde makine kodu katmanı yer alır. Bu katman, konum verisi sağlayan kaynaklar ile doğrudan iletişim halindedir ve cihazın konumunu belirlemek için hangi kaynağın kullanılacağına karar verir. Bu kararı verirken uygulama tarafından belirlenen performans kriterleri ve verinin kullanılabilirliği önemlidir. Bunların yanında makine kodu

katmanı, internet üzerinden Microsoft tarafından barındırılan web servisine bağlanır ve bir veri tabanından konumla ilgili bilgiyi araştırır.

Konum servisinin en üst katmanında NET ile geliştirilmiş ara yüz bulunur. Bu ara yüz, Windows Phone SDK içinde gelen bir kütüphane (DLL) aracılığıyla sunulur. Bir uygulama içinde bu kütüphanedeki kodları şu temel sebeplerle kullanırsın:

- Konum servisini başlatmak ve durdurmak
- Uygulamanın ihtiyaç duyduğu doğruluk seviyesini belirlemek
- Veri mevcut olduğunda makine kodu katmanından konum verisini almak

Güç Kullanımını Azaltma

Konum odaklı bir proje oluştururken doğruluğu yüksek konum verisine ihtiyaç duymak ile güç kullanımını minimize etmek istemek arasındaki dengeyi iyi kurmak gerekir. Mobil cihazlarda bu iki uygulama ihtiyacı ters yönlü ilişkiye sahiptir. Kablosuz bağlantı alanı ve mobil internet, GPS alıcısına göre daha az güç kullanır; ancak aynı zamanda daha az doğruluğa sahip konum verisi üretir.

Uygulama geliştirirken takip etmen gereken iki temel kural ortaya çıkar:

- Yazılan uygulama, daha yüksek doğruluk seviyesine ihtiyaç duymadığı sürece konum servisini güç kullanımını optimize edecek ayarlarla yani daha düşük doğruluğa sahip konum bilgileri elde edecek şekilde kullanılır.
- Konum servisini (Location Service) sadece uygulama ihtiyaç duyduğunda açılır, iş bittiğinde kapatılır.

Konum servisi, lokasyon bilgisi için çoklu kaynakları kullanmasına rağmen belli bir zamanda bu kaynakların hiç biri erişilebilir durumda olmayabilir. Bu durumda makine kodu katmanı, uygun veriyi hesaplama işini ve en doğru kaynağın ne olduğunu seçer. Senin uygulamanın ihtiyaç duyduğu en önemli şey varsayılan olarak ayarlı gelen güç kullanımı ayarları ile yüksek doğruluk seviyesi arasındaki seçimi yapmaktır. Bu ayarı ana konum servisi sınıfı başlatılırken yapılabilir: GeoCoordinateWatcher

```
GeoCoordinateWatcher watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);
```

Default	GeoPositionAccuracy GeoPositionAccuracy.Default
High	Default accuracy.

Şekil 32 – Konum Servisinin kod olarak eklenmesi

Mobil cihazlardaki GPS bileşeninin anteni olmadığı için sensörler genellikle oldukça hassas tasarlanırlar. Bu hassaslık yüzey yansıması ve diğer ortam efektlerinden dolayı az da bir ses çıkarabilir. Ana konum sınıfı olan `GepCoordinateWatcher`, `MovementThreshold` özelliğini sunar. Bu özellik, `PositionChanged` olayı tetiklenmeden önce gerçekleşmesi gereken minimum pozisyon değişikliğini belirler. Eğer `MovementThreshold` özelliğine çok düşük bir değer verirsen, uygulamanın sinyallerinden ses gelmesine sebep olacak kadar çok olay tetiklenmesine sebep olursun. Bu sorunun oluşmasının önüne geçmek ve sadece pozisyonadaki küçük değişiklikleri temsil etmek için `MovementThreshold` özelliğini en az 20 metre olarak belirleyebilirsin. Bu aynı zamanda uygulamanın daha düşük güç kullanmasını sağlar. Hareket eşiğini 0 olarak belirlemek nerdeyse her saniye olayın tetiklenmesine sebep olur. Bu şekilde bir ayar yapmak, navigasyon uygulamaları için faydalı olacaktır.

Konum servisi (Location Service), cihazın yer bilgisini elde ederken birden fazla kaynaktan faydalanmasına rağmen bu bilginin elde edilemeyeceği durumlar oluşabilir. Örneğin GSM ve GPS alıcılarına erişimin olmadığı bir ortamda olabilirsin. Programının bu veri eksikliğini hissettirmeden çözmesi gerekir. Konum servisinin durumu değiştiği anda `StatusChanged` olayı tetiklenir. Uygulaman bu olay metodunu kullanabilir ve kullanıcıya konum verisinin erişim durumu hakkında bilgi verebilir ve uygun bir şekilde uygulamanın davranışını güncelleyebilir.

Konum servisi, yer bilgisini elde edebiliyor olmasına rağmen, ilk okumayı başlatmak ve elde etmek, ortalama olarak 15 saniye sürer; bu süre 120 saniyeye kadar uzayabilir. Uygulamayı bu bilgiyi aklında tutarak modellemen gerekir. Bu süreyi beklerken uygulamanın cevap veremeyebileceği hakkında kullanıcıyı uyarman gerekebilir.

Konum Servisi Örnek Uygulama 14: (Microsoft Açık Akademi, 2013)

1.adım: Yeni bir proje açılır.

Visual Studio kullanarak Silverlight for Windows Phone - Windows Phone Application şablonunu ile yeni bir uygulama aç. Proje adı olarak `wp_locationservice` verebilirsin.

2.adım: Gerekli referansları ekle.

Uygulama içinde konum bilgisini alabilmek için eklemen gereken bir kütüphane var: “System.Device.dll”. Bunun için Solution Explorer içinden projeye “References-Add Reference...” tıklaman gerekiyor; ardından. NET tabından System.Device kütüphanesini seç.

3.adım: Kod dosyasına gerekli isim alanlarını ekle. “xaml.cs” dosyasının üst kısmında yer alan isim alanı tanımlamalarına bir isim alanı eklemelisin. Bu isim alanını, projeye yeni eklediğin kütüphaneden bazı veri tiplerini kullanabilmek için ekle.

```
using System.Device.Location;
```

4.adım: Görsel arayüzü hazırla. MainPage.xaml sayfası içinde varsayılan olarak gelen ContentPanel isimli Grid elementi içine aşağıdaki xaml kodunu yapıştırabilirsin.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<StackPanel Orientation="Vertical">
<TextBlock HorizontalAlignment="Left" Name="lblDurum" Text="durum" Width="134"
Style="{StaticResource PhoneTextLargeStyle}"/>
<TextBlock HorizontalAlignment="Left" Name="txtDurum" Text="konum servisi kapalı" Width="436"
Style="{StaticResource PhoneTextNormalStyle}" Foreground="{StaticResource PhoneAccentBrush}" />
<TextBlock HorizontalAlignment="Left" Name="lblLatitude" Text="latitude" Width="134"
Style="{StaticResource PhoneTextLargeStyle}"/>
<TextBlock HorizontalAlignment="Left" Name="txtLatitude" Text=" " Width="436" Style="{StaticResource
PhoneTextLargeStyle}" Foreground="{StaticResource PhoneAccentBrush}" />
<TextBlock HorizontalAlignment="Left" Name="lblLongitude" Text="longitude" Width="134"
Style="{StaticResource PhoneTextLargeStyle}" />
<TextBlock HorizontalAlignment="Left" Name="txtLongitude" Text=" " Width="436" Style="{StaticResource
PhoneTextLargeStyle}" Foreground="{StaticResource PhoneAccentBrush}" />
</StackPanel>
</Grid>
```

5.adım: Konum servisi ile çalışmak için gerekli sınıftan bir değişken tanımla. GeoCoordinateWatcher sınıfından bir değişken tanımla. Bu değişkeni MainPage sınıfı düzeyinde tanımlayabilirsin; böylece sayfa içinde her yerden kolayca erişebilirsin.

```
namespace wp_locationservice
{
public partial class MainPage : PhoneApplicationPage
{
GeoCoordinateWatcher watcher;
...
}
```

6.adım: Konum servisi ile çalışmak için gerekli sınıftan bir değişken tanımla. *GeoCoordinateWatcher* sınıfının veri toplamaya başlaması için bir metod yaz: İsmi *KonumServisiniBaslat* olsun.

Bu metod *GeoPositionAccuracy* tipinde bir parametre alsın. Bu parametre, konum servisinin yüksek oranda doğrulukla mı, yoksa güç tasarrufu yaparak düşük doğruluk oranı ile mi çalışacağını belirler.

MovementThreshold özelliği ile her pozisyon değişikliğinde ne kadarlık bir mesafe kat edileceğini belirtmelisin. Optimum performans için ortalama 20 metre verebilirsin.

GeoCoordinateWatcher sınıfının *Start* metodu ile konum servisinden veri almaya başlayabilir.

Ayrıca bu temel sınıfın 2 önemli olayı (event) var. Bu olayları ele almak için gerekli kod parçalarını da yazmak gerekir:

StatusChanged: Konum servisinin durumu değiştiğinde tetiklenir. Kullanıcıyı bilgilendirmek için kullanılabilir. Örneğin çevrimiçi konum bilgisi alırken, veri kaynaklarına erişilemeyecek bir yerde olduğunda artık yer bilgisi alamamaya başladığında bunu kullanıcıya bir mesaj ile gösterilebilir.

PositionChanged: Konum servisi pozisyonda bir değişiklik tespit ettiğinde tetiklenir. Burada güncel konum bilgisi alınıp kullanılabilir.

```
private void KonumServisiniBaslat(GeoPositionAccuracy accuracy)
{
    txtDurum.Text = "konum servisi başladı";
    watcher = new GeoCoordinateWatcher(accuracy);
    watcher.MovementThreshold = 20;
    watcher.StatusChanged += new
    EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
    watcher.PositionChanged += new
    EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
    watcher.Start();
}
```

KonumServisiniBaslat metodunun, uygulama ile birlikte çalışmasını sağlamak için sayfanın yapıcı metodu (constructor) içinde çağır.

```
public MainPage()
{
    InitializeComponent();
    KonumServisiniBaslat(GeoPositionAccuracy.Default);
}
```

7.adım: Durum değişikliklerini takip etmek için olay metodunu kodla. Konum servisinin çalışma ve yer bilgisine erişme durumu değiştiğinde çalışacak olan *StatusChanged* olay metoduna bir önceki adımda *watcher_StatusChanged* ismini vermiştin. Şimdi bu metodun içeriğini kodla. Kod içerisinde xaml ara yüzündeki txtDurum kontrolüne durum bilgisini göster.

```
void watcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
{
    switch (e.Status)
    {
        case GeoPositionStatus.Disabled:
            txtDurum.Text = "konum bilgisi bu cihazda desteklenmiyor.";
            break;
        case GeoPositionStatus.Initializing:
```


5. BÖLÜM 4: WINDOWS PHONE İLE UYGULAMA GELİŞTİRME VE VERİ İŞLEMLERİ

5.1. Uygulama Mimarisi ve Isolated Storage

Windows Phone Silverlight Uygulama Yaşam Döngüsü(Application Life Cycle)

Windows Phone Silverlight uygulamalarında gerçekleşen olayların (events) sıralı bir şekilde tam olarak anlaşılmış olması, kullanıcılara sağlam, istikrarlı bir biçimde çalışan, esnek uygulamalar sunabilmek için kritik derecede önem taşır (Basu, 2013).

Windows Phone Çalışma Modeli (Execution Model)

Windows Phone çalışma modeli (execution model), Windows Phone üzerinde çalışan uygulamaların yaşam döngüsünü, yönetir. Bu süreç, uygulamanın çalışması (launch) ile başlar ve yok edilmesi (terminate) ile son bulur.

Çalışma modeli, son kullanıcılara hızlı ve her an interaktif bir deneyim sağlayacak şekilde tasarlanmıştır. Bunu sağlamak için, Windows Phone, anlık olarak sadece bir aktif uygulama çalıştıracak şekilde dizayn edilmiştir. Bir uygulama ekranda ön planla aktif olarak kullanılmaktan çıktığında, işletim sistemi, ilgili uygulamayı geçici olarak *deaktif (dormant)* duruma alır. Telefon hafızası, ön planda çalışan uygulamanın performanslı bir şekilde çalışıp, iyi bir kullanıcı deneyimi sağlaması için yetersiz olduğunda, işletim sistemi, geçici olarak *deaktif (dormant)* durumda bulunan uygulamaları *yok etmeye (terminate)* başlayacaktır. Bu süreç başladığında, ilk *deaktif (dormant)* duruma geçen uygulama ilk olarak yok edilecektir (terminate). Uygulamaların *deaktif (deactivated)* ve tekrar aktif (reactivated) olduklarındaki durumlarını yönetmek için bir programlama altyapısı (framework) sağlanmıştır. Bu altyapı, uygulamaların kullanıcıya tek örnek (single instance) olarak sunulmasını sağlar ve iyi bir kullanıcı deneyimi sağlanmasına yardım eder. Ayrıca, uygulama yok edildiğinde (terminated) veya yeniden aktif edildiğinde (reactivated), kalınan yerden devam edilmesini sağlayacak imkânları da sunar.

Çalışma modeli, ayrıca kullanıcıya uygulamalar arasında consistent bir navigasyon deneyimi sağlar. Windows Phone üzerinde, kullanıcılar ileri yönlü navigasyonlarını, kurulu uygulamalar listesinden bir uygulama başlatarak veya Start ekranındaki bir Tile'dan yapabilecekleri gibi, bir uygulama ile ilişkili toast notification üzerine dokunarak da yapabilirler. Kullanıcılar ayrıca çalışmakta olan bir uygulamanın sayfaları arasında ya da

önceden çalıştırılmış olan uygulamalar arasında donanımsal geri tuşunu kullanarak, geri yönlü navigasyon yapabilirler. Windows Phone 7.5 ile birlikte gelen yeni bir özellik sayesinde, telefon üzerindeki donanımsal geri tuşunu basılı tutarak, daha önceden çalışmakta olan uygulamalara geçiş yapılabilir.

Bu konuda, Windows Phone uygulamalarının yaşam döngüsü detaylıca ele alındı ve responsive ve consistent navigation experience sağlayan uygulamalar geliştirmek için ihtiyaç duyulan adımlardan bahsedildi.

Terminoloji

Windows Phone uygulama yaşam döngüsünün detaylarına girmeden önce bazı term ve konseptlere aşina olmakta fayda var (Microsoft).

Application State (Uygulama Durumu): Bir uygulama içerisinde birden fazla sayfa tarafından kullanılan veri (data). Örneğin; bir web servisinden obtain edilen yapısal veri. Farklı sayfalarda bu veriyi çeşitli görünümlemlerle sunmak isteyebilirsiniz. Bu gibi bir durumda kullanılan verinin herhangi bir sayfa ya da sayfa topluluğuna değil, uygulamanın bütününe ait olduğu düşünülmelidir. Uygulama durumu ile ilgili sıklıkla yapılan hatalardan birisi, uygulamaya ait tüm durum verisinin uygulamadan çıkarken kaydedilmesi gerektiğinin düşünülmesidir. Oysaki bu tip veriyi, kullanıcı bir sayfadan diğerine geçerken kaydetmek daha doğru olacaktır. Böylece, uygulama kapatılırken yapılması gereken durum yönetimi işlemleri de azalacaktır ve süreç kolaylaşacaktır.

Page State (Sayfa Durumu): Tek bir uygulama sayfasına ait mevcut görsel durum. Kullanıcıların bilgi gireceği kontroller içeren bir sayfan varsa ve kullanıcı senin uygulamandan dışarı gidip tekrar geri dönerse, sayfa üzerindeki tüm kontrollerin, son girilen değerleri içermesini bekleyecektir. Sayfa durumunun yönetilmesi gereken uygulamalarda, sayfa üzerindeki kontrollerin değerleri sayfa yüklendiğinde/açıldığında atanır. Böylece persistent bir uygulamanın kullanıcı deneyimi sağlanmış olur. Fakat kullanıcı, uygulamayı kapatıp açtığında, sayfanın arayüz durumu (Page UI State) uygulamanın baştan açıldığını yansıtabilecek şekilde boş olarak gelmeli ve uygulamanın bir önceki çalışmasından kalan arayüz durumunu (UI state) yüklememelidir.

Application Events (Uygulama Olayları): Uygulamanın durum yönetimini sağlarken dört ana olay kullanılır: Uygulama, kullanıcı tarafından başlatılırken tetiklenen Launching, kullanıcı uygulamayı terk ettiğinde ya da herhangi bir Launcher ya da Chooser ile çalışmaya başladığında tetiklenen Deactivated, kullanıcı uygulamayı terkettikten sonra geri döndüğünde tetiklenen Activated, ve uygulamadan çıkışta tetiklenen Closing. Bu olayları ele alacak olan olay yakalayıcılar, Application nesnesi içerisinde yer alır. Uygulamanın durumunu yönetebilmek için, bu olayları ele alacak metotlar yazılıp, içleri kodlanır. Uygulama olaylarını ele alan işlemleri tamamlamak için 10 saniyelik bir limit var. Uygulama, bu limitleri aştığında derhan sonlandırılır (terminate). Bundan dolayı, uygulama olaylarını ele alırken Isolated Storage üzerinden bilgi okumak ya da kaydetmek gibi kaynak kullanımı yoğun işlemler

yapmaktan kaçınılmalıdır. Bu işlemler uygulamanın çalışması sırasında arka planda ele alınarak yapılmalıdır. Uygulama verilerini, uygulama yaşam döngüsü içerisinde, değişiklik meydana geldiği zaman kaydetmek, uygulama olayları sırasında sarf edilmesi gereken durum yönetimi eforunu azaltacaktır.

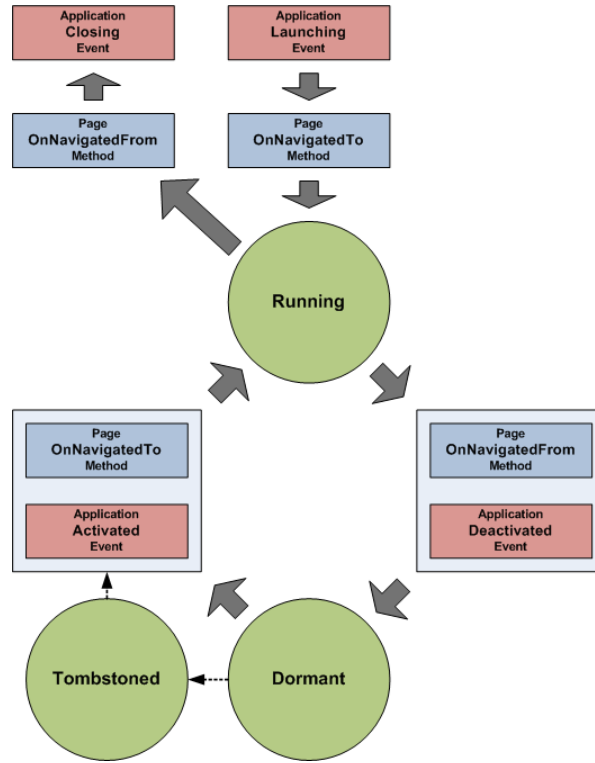
Page Events (Sayfa Olayları): Tüm Windows Phone Sayfaları (pages), *PhoneApplicationPage* nesnesinden kalıtılır ve iki metot sunarlar; *OnNavigatedTo(NavigationEventArgs)* ve *OnNavigatedFrom(NavigationEventArgs)*. Uygulama içerisinde bu metotlar ezilerek (override) sayfa durumu yönetilebilir.

Tombstoning: Bu aşamada uygulama kapanmış (terminated), fakat uygulama durumu (*application state*) ve uygulama içerisindeki sayfaların durumları henüz kaybolmamıştır. Mevcut görüntülenmekte olan uygulama sayfası ve uygulama içerisinde ziyaret edilen önceki sayfaların datası kayıtlı durumdadır. Eğer kullanıcı tombstoned konumdaki bir uygulamaya geri döners, uygulama bellekte yeniden oluşturulur ve mevcut sayfa ile birlikte sayfa geçmişi (page history) otomatik olarak geri yüklenir.

State Dictionaries: Her Windows Phone uygulaması uygulama sayfası, anahtar/değer (*key/value*) çiftlerini saklayabileceği bir Dictionary nesnesi kullanır. Bu dictionary nesneleri, uygulama tombstoned konuma geçtiğinde korunur ve yok olmaz. Uygulama tombstoned konumdan aktif konuma geçtiğinde (activated), bu dictionary nesneleri kullanılarak uygulama durumu (*application state*) restore edilir. Dictionary nesneleri içerisindeki tüm veri serileştirilebilir (serializable) olmalıdır.

5.2. Windows Phone Uygulama Yaşam Döngüsü (Application Life Cycle)

Aşağıdaki görselde, bir Windows Phone uygulamasının yaşam döngüsü gösterilmiştir. Diyagramdaki daireler, uygulama durumunu (application state) belirtiyor. Dikdörtgenler ise, uygulamanın durum yönetimini sağlamak için kullanılabilecek olan uygulama ya da sayfa seviyesindeki olayları (events) temsil ediyor.



Şekil 33 – Windows Phone uygulama yaşam döngüsü

Windows Phone uygulamalarının yaşam döngüsü, hem işletim sistemi hem de kullanıcının, hangi durumda, ne yaparak, uygulama durumunda (application state) meydana gelen çeşitli değişiklikleri tetikleyeceği ele alınır (Yöndem, 2013).

The Launching Event

Bu olay (event), kullanıcı tarafından uygulama başlatıldığında tetiklenir. Örneğin; kurulu uygulamalar listesinden ya da başlangıç (Start) ekranında yer alan bir Tile kullanarak. Bir uygulama bu yolla başlatıldığında, kullanıcıya temiz bir başlangıç sunulmalıdır. Önceki kullanımlardan herhangi bir kalıntı olmamalıdır. Uygulamanın hızlıca yüklenmesini garanti altına almak için, bu olayı ele alırken, olabildiğince az kod çalıştırmalısın. Başka bir deyiş ile dosya (file) ve ağ (network) işlemleri gibi yoğun kaynak kullanımı gerektiren işlemlerden kaçınmak gerekir. İyi bir kullanıcı deneyimi sağlamak için, bu tarz işlemler, uygulama

başlatıldıktan sonra arka planda başka bir thread üzerinde gerçekleştirilmelidir (Microsoft Açık Akademi, 2013).

Running

Bir uygulama başlatıldıktan (launched) sonra, çalışır (running) duruma geçer. Kullanıcı, uygulamadan dışarı çıkana kadar ya da uygulamanın ilk sayfasında donanımsal geri butonunu kullanana kadar uygulama çalışmaya devam eder. Windows Phone uygulamaları, kullanıcının uygulamadan çıkış yapması için herhangi bir mekanizma sağlamamalıdır. Ayrıca önlem alınmadığı sürece, telefonun kilit ekranı geldiğinde de uygulamalar running durumunu terkeder (Microsoft Açık Akademi, 2013).

The OnNavigatedFrom Method

OnNavigatedFrom(NavigationEventArgs) metodu, kullanıcı, uygulama sayfasını (application page) terkettiğinde çağırılır. Bu işlem uygulaman içerisindeki normal sayfa geçişlerinde de (page navigation) gerçekleşebilir. Ayrıca, uygulama deactivated konuma geçtiğinde de bu metod çağrı alır. Bu metod her çağrıldığında uygulamana ait sayfa durumu (page state) kaydedilmelidir ki kullanıcı sayfaya geri döndüğünde bu bilgiler tekrar elde edilebilsin. NavigationMode özelliği (property) kullanılarak ayrıldığı sayfadan geri yönlü mü ileri yönlü mü hareket ettiğini öğrenebilirsin. Eğer hareket geri yönlü ise sayfa durumunu kaydetmene gerek olmaz. Çünkü bu sayfaya tekrar gelindiğinde sayfa, sıfırdan oluşturulacaktır (Microsoft Açık Akademi, 2013).

Bazı senaryolarda, durum verilerini OnNavigatingFrom(NavigatingCancelEventArgs) metodunda kaydetmek isteyebilirsin. Bu metodun diğerinden farkı sayfa değiştikten sonra değil, değişirken çalışmasıdır. MediaElement kontrolüne ait durum datası burada kaydedilmelidir (Microsoft Açık Akademi, 2013).

The Deactivated Event

Deactivated olayı (event) , kullanıcı, uygulama içerisinde ileri yönlü ya da başka bir uygulama çalıştırarak veya Başlat butonuna basarak uygulamadan dışarı doğru hareket ettiğinde tetiklenir. Deactivated olayı (event) , ayrıca, uygulama içerisinde bir Chooser başlatıldığında da tetiklenir. Ayrıca telefonun kilit ekranı devreye girdiğinde de bu olay tetiklenir (Microsoft Açık Akademi, 2013).

Uygulama içerisinde Deactivated olayı ele alınırken uygulama durumu (application state) ile ilgili herşey kaydedilmelidir ki bu veriler daha sonra tekrar kullanılabilirsin. Windows Phone uygulamaları, uygulama durumunu (application state) kaydetmek için, dictionary tipli State nesnesini kendi içerisinde barındırır. Eğer uygulama tombstoned konumdayken tekrar aktive edilirse (reactivated), bu sözlük (dictionary) tipli durum (state) nesnesi kullanılarak Deactivated olayı sırasında kaydedilen veriler elde edilir. Bu veri, hali hazırda hafızada (memory) mevcut olduğundan dolayı, uygulama durumunu yeniden elde etmek için, yoğun kaynak kullanımı gerektiren dosya işlemleri ile uğraşmadan da kullanılabilir.

Deactivated olayından sonra, bir uygulamanın tamamen kapatılması (terminated) ihtimali de vardır. Bir uygulama tamamen kapatıldığında (terminated), dictionary tipindeki state, artık hafızada korunmaz ve silinir. Bu yüzden, daha sonra ihtiyaç duyulacak ve kaydedilmemiş olan tüm durum datasını, Deactivated olayı sırasında Isolated Storage’a ayrıca kaydetmelisin.

Dormant

Kullanıcı uygulamadan ileri yönlü hareket ederek çıktığında, Deactivated olayı (event) tetiklendikten sonra, işletim sistemi, uygulamayı dormant duruma (state) geçirmeye çalışır. Uygulama bu durumda iken, Uygulamaya ait tüm thread’ler durdurulur ve hiç işlem yapılmaz/işlemci gücü kullanılmaz. Fakat uygulama halen hafızadadır. Eğer uygulama bu durumdayken, tekrar aktif hale (reactivated) getirilir ise, uygulamanın herhangi bir durum yönetimine ihtiyacı yoktur. Çünkü tüm durum verileri zaten hafızada korunmuştur.

Bir uygulama dormant duruma geçtikten sonra yeni uygulamalar çalıştırılırsa ve bu yeni uygulama veya uygulamalar sağlıklı çalışıp iyi bir kullanıcı deneyimi sunmak için daha fazla hafızaya ihtiyaç duyarlarsa, işletim sistemi dormant durumdaki uygulamaları tombstone konumuna alarak hafızada yer açacaktır (Microsoft Açık Akademi, 2013).

Tombstoned

Tombstoned durumdaki bir uygulama kapatılmış, hafızadan silinmiştir (terminated). Fakat, Deactivated olayı sırasında uygulama tarafından navigasyon durumu (navigation state) ve dictionary tipindeki durum datasına (state dictionaries) ait bilgiler kaydedilerek ihtiyaç halinde tekrar ulaşılabilir. Telefonunuz, anlık olarak 5 uygulamaya kadar tombstoning bilgilerini tutabilir. Eğer bir uygulama tombstoned konuma geçmişse ve kullanıcı uygulamaya

geri dönerse, uygulama tekrar başlatılır ve korunmuş olan datayı kullanarak uygulama durumunu geri yükleyebilir (restore) (Microsoft Açık Akademi, 2013).

The Activated Event

Activated olayı (event), kullanıcı dormant ya da tombstoned konumdaki bir uygulamaya dönüş yaptığında tetiklenir. Olay parametreleri (event args) içerisindeki IsApplicationInstancePreserved özelliği (property) kontrol edilerek, uygulamanın dormant durumdan mı yoksa tombstoned durumdan mı döndüğü tespit edilebilir. IsApplicationInstancePreserved değeri true olduğunda, uygulama dormant durumdan geri gelmiş demektir ve durum datası işletim sistemi tarafından otomatik olarak korunmuştur. Değerin false olması, uygulamanın tombstoned durumdan geri geldiğini gösterir. Bu durumda, uygulama state dictionary'yi kullanarak uygulama durumunu (application state) geri getirmelidir (restore). Uygulamalar, bu olayı ele alırken, IsolatedStorage kullanımı ya da ağ kaynakları kullanımı gibi yoğun kaynak kullanımı gerektiren işlemler yapmamalıdır. Aksi takdirde bu işlemler uygulamanın geri dönüş (resume) süresini uzatabilir. Bu işlemlerin yapılması gerekiyorsa, tercih edilecek yer uygulama yüklendikten sonra arka planda çalışan bir thread olmalıdır (Microsoft Açık Akademi, 2013).

The OnNavigatedTo Method

OnNavigatedTo(NavigationEventArgs) metodu, kullanıcılar bir sayfaya yönlendiğinde (navigate) çağrılır. Uygulama ilk çalıştığında veya sayfalar arasında geçiş yapıldığında ya da uygulama dormant veya tombstoned durumda iken yeniden çalıştırıldığında bu metod çağrılır. Bu metod içerisinde, sayfanın ilk defa mı açıldığı yoksa geri dönüş ile mi sayfaya ulaşıldığı kontrol edilmelidir. Eğer sayfaya geri dönüş ile ulaşılmış ise, durumun (state) restore edilmesi gerekli değildir. Sayfa ilk defa açılıyor ise ve state dictionary içerisinde veri var ise, o zaman veriler kullanılmalı ve sayfa arayüzünün durumu restore edilmelidir (Microsoft Açık Akademi, 2013).

The Closing Event

Closing olayı (event), bir uygulamanın ilk sayfasındayken geri yönlü hareket edildiğinde tetiklenir. Bu durumda, uygulama kapatılır (terminated) ve durum (state) bilgisi saklanmaz. Closing olayı ele alınırken, uygulama tekrar açıldığında kullanılması gerekecek

olan veriler kaydedilebilir. Uygulama (application) ve sayfa (page navigation) olaylarının (events) tamamlanması için 10 saniyelik bir limit var. Bu süre dolduğunda uygulama hafızadan indirilir (terminated). Bundan dolayı, durum (state) datalarını uygulamanın yaşam süreci içerisinde kaydederek, Closing olayı ele alınırken yüksek miktarda dosya okuma yazma işlemi yüzünden 10 saniyelik limitin aşılması riskinin önüne geçmek iyi bir fikir olacaktır (Microsoft Açık Akademi, 2013).

5.3. *Isolated Storage*

Silverlight uygulamalarında, işletim sistemi ve diğer uygulamalardan izole olarak sadece bir uygulamanın kendisine özel kullanabileceği, veri saklamak için faydalanan bir alandır (Microsoft Açık Akademi, 2013).

Isolated Storage’a Neden İhtiyaç Duyulur? Uygulaman içerisinde kaydedip daha sonra tekrar kullanmak isteyeceğin bilgiler olabilir. Ya da uygulaman çalışırken telefon çaldığında, o an için hafızada olan bilgileri kaybetmek istemeyebilirsin. Bu ve benzeri durumlarda, uygulama ayarlarını ve kullanıcı dosyalarını kaydetmek için IsolatedStorage kullanarak ihtiyaçlarını karşılayabilirsin.

Isolated Storage Nasıl Kullanılır?

IsolatedStorageFile ya da ApplicationSettings koleksiyonunu kullanarak IsolatedStorage’dan faydalanabilir, dosya ve klasör saklayabilir ya da uygulama ayarlarını kaydedebilirsin.

Isolated Storage içerisinde Dosya ve Klasör Saklamak

Bu konuyu bir örnek ile ele alacağız. Şimdi, bu örneği hazırlamak için öncelikle görsel arayüzü hazırla. Başlangıç olarak tasarım ekranına bir TextBox ve üç Button sürükleyip bırak. TextBox ve butonları “txtDeger, btnKaydet, btnGetir, btnSil” şeklinde isimlendirilir.

Bu işlemin ardından butonların Content property’lerine de kaydet, getir ve sil değerlerini vererek bu ifadelerin butonlar üzerinde gözükmesini sağla.

İlk buton, TextBox içerisinde yazılı bulunan bilgiyi IsolatedStorage'a kaydedecek.

- İkinci buton, IsolatedStorage içerisinde kayıtlı bulunan bilgiyi TextBox'a getirecek.
- Üçüncü buton ise IsolatedStorage içerisindeki bilgiyi silecek.

Isolated Storage içerisinde direkt olarak dosya ve klasör saklamak mümkün. Biz de örneğimizde, saklayacağımız değeri *.txt dosyası olarak bu alana depolayacağız. Dosya adı olarak “data.txt” kullanalım. IsolatedStorage alanı ile çalışmak için projemize using System.IO.IsolatedStorage; ifadesini ile ilk satırı ekle.

Uygulama açılır açılmaz, ilk iş olarak IsolatedStorage içerisinde "data. txt" dosyasının bulunup bulunmadığını denetleyeceğiz. Dosyanın bulunması durumunda içeriğini okuyup TextBox içerisinde sunalım. Dosya bulunamadığında ise TextBox'a String.Empty değeri atanır.

Bu işlemleri yapmak üzere code-behind dosyasında DepoyuOku isimli bir metot yazıp, MainPage isimli yapıcı metodun son satırında bu metodu çağırılır.

DepoyuOku methodunun içeriği aşağıdaki gibi olmalı.

Örnek 14:

```
private void DepoyuOku()
{
    IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();
    if (storage.FileExists("data.txt"))
    {
        IsolatedStorageFileStream fs = storage.OpenFile("data.txt", System.IO.FileMode.Open,
        System.IO.FileAccess.Read);
        StreamReader sr = new StreamReader(fs);
        txtDeger.Text = sr.ReadToEnd();
        sr.Close();
    }
    else
    { txtDeger.Text = String.Empty;
    }
}
```

IsolatedStorage ile çalışmak için, öncelikle IsolatedStorageFile.GetUserStoreForApplication() metodu ile ilgili depolama alanının referansı alınıyor. Ardından, FileExists metodu ile “data.txt” dosyasının varlığı kontrol ediliyor. Dosya bulunursa, OpenFile metodu kullanılarak dosya açılıyor ve StreamReader aracılığı ile okunuyor. Dosya bulunamazsa da TextBox içerisine boş string atanıyor. Biz örnekte direkt olarak bir dosya adı kullandık ama eğer dosya bir klasör içerisinde olsaydı, dosyanın yolunu klasör adı ile beraber girmek gerekecekti.

IsolatedStorage içerisinde klasör oluşturulduğunda, her klasör 1KB yer kaplar. IsolatedStorage içerisinde oluşturulan tüm dosya ve klasörler, uygulamana özel bir IsolatedStorage alanında depolanacağı için diğer uygulamalardan erişilmesi mümkün olmayacak.

Uygulama açılışında, IsolatedStorage alanında “data.txt” dosyasının bulunması durumunda yapılacak işlemleri tamamladıktan sonra, şimdi sıra geldi ilk buton ile TextBox içerisindeki bilgiyi *.txt dosyası olarak IsolatedStorage içerisine kaydetmeye. Aşağıdaki kodları, kaydet butonu tıklandığında çalışacak şekilde projene ekle.



Şekil 34 – Örnek 14 Ekran görüntüsü

```
IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();  
IsolatedStorageFileStream fs = storage.CreateFile("data.txt");  
StreamWriter sw = new StreamWriter(fs);  
sw.Write(txtDeger.Text);  
sw.Close();
```

Daha önceki örnekte olduğu gibi IsolatedStorage alanının referansını aldıktan sonra, CreateFile metodu kullanılarak “data.txt” dosyası oluşturuluyor. Ardından, TextBox içerisindeki değer, StreamWriter aracılığıyla “data.txt” dosyasına yazılıyor.

Kaydetme ile ilgili işlemlerimiz de hazır. İkinci butonumuzda yapılacak olan veri getirme işlemine ait kodu daha önce DepoyuOku isimli bir metot içerisinde ele almıştık. Getir butonu tıklandığında bu metodu çağırarak satırı projene ekle. Son olarak üçüncü butona basıldığında “data.txt” dosyasını IsolatedStorage’den silecek işlemleri yapacağız. Bunu sağlamak için üçüncü butona basıldığında aşağıdaki kodun çalışabileceği şekilde kodları projene ekle.

```
IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();  
if(storage.FileExists("data.txt"))  
{ storage.DeleteFile("data.txt"); }
```

IsolatedStorage referansını aldıktan sonra, FileExists metodu ile dosyanın alanda bulunup bulunmadığı kontrol ediliyor ve varsa DeleteFile metodu ile dosya siliniyor.

Son olarak, IsolatedStorage kapasitesini ve üzerinde kalan boş alanı göstermek üzere, tasarım ekranına tbBosAlan adında bir TextBlock ekle. Ardından, aşağıdaki BosAlanGoster isimli metodu projene ekle ve kaydet butonunun son satırında bu metodu çağırarak boş alanın görüntülenmesini sağla.

```
private void BosAlanGoster()  
{  
IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();  
tbBosAlan.Text = String.Format("Boş Alan: {0}/{1}", storage.AvailableFreeSpace, storage.Quota);  
}
```

IsolatedStorage üzerindeki Quota özelliği ile IsolatedStorage üzerindeki kullanılabilir maksimum alan miktarı elde edilebilir. AvailableFreeSpace özelliği ise byte cinsinden kullanılabilir boş alanı verecektir.

Isolated Storage içerisinde birden fazla dosya saklamak mümkündür. CreateDirectory, MoveDirectory, DeleteDirectory gibi metotlar aracılığıyla klasör oluşturup, taşıyabilir veya silebilir, GetFileNames ve GetDirectoryNames metotları ile de IsolatedStorage içerisinde mevcut bulunan dosya ve klasörlerin listesini çekebilirsin.

IsolatedStorage tarafından sunulan alan sana yetmediğinde IncreaseQuotaTo metodunu aşağıdaki gibi kullanarak kullanıcının iznini isteyip kullanılacak depolama alanı arttırabilirsin.

```
storage.IncreaseQuotaTo(1000000);
```

IncreaseQuotaTo metodu, kullanmak istenilen toplam alanı byte cinsinden parametre olarak alır. Kullanıcıya uygulamanın daha fazla depolama alanına ihtiyaç duyduğunu bildiren bir mesaj gösterir ve onay ister. Kullanıcı onay verirse bu metot geriye True değer döndürür. Onay alınamaması durumunda ise dönen değer False olacaktır.

Sadece basit bir değer tutmak için dosya mı oluşturacağım? (ApplicationSettings Kullanımı)
Isolated Storage, uygulamalara hard disk gibi kullanılabilecek bir alan sağlar. Fakat uygulama içerisinde minik bir ayar yapmak için bir dosya oluşturup saklamak çok da efektif bir yol

değildir. Önce dosya oluşturup sonra bilgileri bu dosyaya kaydetmek kodlama açısından zaman alıcı ve yorucu olacaktır. Bu gibi durumlarda kullanmak üzere IsolatedStorage sınıfının ApplicationSettings özelliğinden faydalanabiliriz. Buradaki durumu, Windows uygulamalarında, uygulama ayarlarını app.config içerisinde saklamaya benzetebiliriz. Örnekle kod satırları ile birlikte durumu daha net anlamaya çalışalım.

```
IsolatedStorageSettings iss = IsolatedStorageSettings.ApplicationSettings;  
iss.Add("Tercih", "İngilizce");  
iss["Tercih"] = "Türkçe";  
string renk = iss["Tercih"].ToString();  
iss.Remove("Tercih");
```

IsolatedStorageSettings içerisindeki ApplicationSettings özelliği bir değişken üzerine alınıyor. Sonrasında sırası ile yeni bir ayar ekleme, mevcut bir ayarı güncelleme, ayar bilgisini geri okuma ve ilgili ayarı silme işlemlerini gerçekleştirecek satırlar kullanılıyor.

5.4. Data İşlemleri

Veri İle Çalışmak

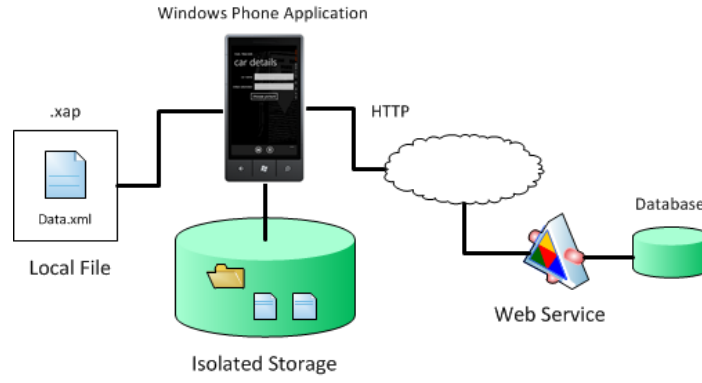
Windows Phone Uygulamasına Veri Getirmek

Windows Phone uygulaman için veri saklamak ve elde etmek için birçok yol bulunmaktadır. Bu bölüm, uygulamanın ihtiyaçlarına bağlı olarak hangi yaklaşımı kullanman gerektiğine rehberlik etmeyi hedeflemektedir (Lecrenski, Watson, & Ensor, 2011).

Veri Kaynakları

Veri, birçok farklı lokasyonda olabilir. Örneğin;

1. Bütün kullanıcıların sadece okumak için kullanacağı bir veriyi, uygulamanın içinde lokal bir dosyada saklayabilirsin.
2. Kullanıcıya özel veriyi, cihaz içinde Isolated Storage adı verilen bölgede tutabilirsin.
3. Veriyi web servisleri aracılığıyla erişeceğin internet ortamında saklayabilirsin.



Şekil 35 – Veri kaynakları

Lokal Dosyalar (Local Files)

Sadece okunabilir veriyi cihaz içinde yerel olarak saklamak için text ve xml gibi dosyaları kullanabilirsiniz. Lokal dosyalar iki şekilde derlenebilir:

- Kaynak dosyaları (Resource Files)
- İçerik dosyaları (Content Files)

Kaynak Dosyaları (Resource Files)

Kaynak dosyaları, proje assembly dosyası içine gömülüdür; bu dosyaları uygulama ve kütüphane projeleri ile birlikte kullanabilirsiniz. Assembly'leri uygulama paketi (.xap) içinde ya da dışında yayınlayabilirsiniz.

Kaynak dosyasının avantajı, dosyanın uygulama için her zaman erişilebilir olmasıdır. Bununla birlikte kaynak dosyasından faydalandığında uygulamanın açılması normalden biraz uzun sürebilir.

Kaynak Dosyası ile Çalışmak

Visual Studio içinde Solution Explorer penceresinde projeye eklediğin bir dosyayı seçip klavyeden F4 butonuna tıklayarak Properties penceresine giderek Build Action özelliğini Resource olarak belirlemelisin.

Kaynak dosyalarına programatik olarak erişmek için `Application.GetResourceStream` metodunu kullanırsın.

Şu durumlarda kaynak dosyalarını kullanabilirsiniz:

- Uygulamanın açılış süresi senin için önemli değilse.
- Assembly içerisine derlenip gömüldükten sonra kaynak dosyasını güncelleme ihtiyacı duymazsan
- Dosyalara olan bağımlılıklarını azaltarak uygulamanın dağıtımını basitleştirmek istersen.

İçerik Dosyaları (Content Files)

Performans sebebi ile Windows Phone uygulamalarında kaynak dosyalarına göre içerik dosyaları tercih edilir. İçerik dosyaları, proje kütüphaneleri içine gömülmeden uygulama paketi (.xap) içine dahil edilir. Projeye eklediğin imaj, metin vb. dosyalar, varsayılan olarak bu yolla eklenirler. İçerik dosyaları, assembly içerisine derlenmemesine rağmen her bir içerik dosyası ile ilişki kuran metadata bilgisini dâhil ederek derlenirler. İçerik dosyaları paket içerisinde yer alan birden fazla assembly tarafından paylaşımlı olarak kullanılabilir (Microsoft Açık Akademi, 2013).

İçerik Dosyaları ile Çalışmak

Visual Studio içinde Solution Explorer penceresinde projeye eklediğin bir dosyayı seçip klavyeden F4 butonuna tıklayarak Properties penceresine giderek Build Action özelliğini Content olarak belirlemelisin. Varsayılan olarak dosyaların Build Action özellikleri zaten bu şekilde gelir.

İçerik dosyalarına programatik olarak erişmek için paket dosyasına göre adresini kullanırsın. Örneğin xml dosyayı uygulama ortamına yüklemek ve içindeki verilerle çalışmak için şu kod parçasını kullanabilirsin:

```
XElement myElement = XElement.Load("veri.xml");
```

Burada veri.xml, xml dosyanın adıdır. XElement sınıfını kullanarak xml verilerle çalışmak için projene referans olarak System.Xml.Linq kütüphanesini eklemen ve kod dosyasının en üstüne ilgili isim alanını eklemen gerekir :

```
using System.Xml.Linq
```

İçerik dosyalarını şu durumlarda kullanabilirsin:

- Uygulamanın açılış süresi senin için önemliyse

- İçerik dosyalarını kullanan kütüphaneyi yeniden derlemeye gerek olmadan içerik dosyalarını güncelleyebilmek istersen.

Isolated Storage

Kullanıcıya özel bilgiler saklamak ve kullanma ihtiyacın varsa, isolated storage kullanabilirsin. Silverlight for Windows Phone uygulamalarında işletim sisteminin dosya sistemine doğrudan erişimin yoktur. Bununla birlikte kullanıcının cihazında lokal olarak veri saklayabilir ve elde etmek için sana sunulan isolated storage mekanizmasını kullanabilirsin. Isolated Storage kullanmak için iki yolun var:

- Birinci yol IsolatedStorageSettings sınıfını kullanarak anahtar/değer (key/value) çiftleri saklamak ve elde etmektir.
- İkinci yol IsolatedStorageFile sınıfı kullanarak bilgileri dosyaya kaydetmek ve dosyadan elde etmektir.

Isolated Storage ile ilgili daha fazla bilgi almak için önceki eğitimlere geri dönebilirsin.

Web Servisler

Veriyi internet ortamında saklamak ve farklı uygulamaların ortak veriye ulaşarak kullanmasını sağlamak için web servislerinden faydalanabilirsin. Farklı tipte servis, format ve teknolojiler bulunduğu için web servisler ile çalışmak biraz kafa karıştırıcı olabilir (Yöndem, 2013).

SOAP : (Simple Object Access Protocol) Merkezi olmayan, dağıtık ortamlardaki yapısal bilgiyi transfer etmek amacıyla olan basit bir protokoldür.

JSON : (JavaScript Object Notation) Veri transferi yaparken kullanılan basit bir formattır. İnsanlar tarafından okunması için dizayn edilen JSON, aynı zamanda bilgisayar sistemleri tarafından da kolayca ayrıştırılabilir bir formattır.

OData : (Open Protocol Data) Veriyi sorgulamak ve güncellemek için kullanılan bir web protokolüdür.

REST : (Representational State Transfer Protocol) İstemciler tarafından ulaşılması için web üzerindeki kaynakları sunan bir protokol.

Web Service: Birçok uygulamanın ortak erişimi için veri ve servis sağlayan kodları içerir. Uygulamalar, her bir web servisinin nasıl yazıldığından bağımsız olarak, standart web protokollerini ve HTTP, XML, SOAP gibi veri formatlarını kullanarak web servislerine erişirler.

Web Servis Teknolojileri

Veri ile çalışmak için Silverlight for Windows Phone uygulamada kullanabileceğin çok sayıda ağ ve web servis teknolojisi vardır. Bunlardan en sık kullanacaklar :

- HTTP sınıfları
- WCF servisleri
- WCF Data Services (OData services)
- Windows Azure Services

HTTP Sınıfları

Silverlight for Windows Phone uygulamadan System.Net isim alanı altındaki HttpWebRequest / HttpWebResponse ya da WebClient sınıflarını kullanarak bir web servisine ya da kaynağa doğrudan erişebilirsin. Bu sınıflar, HTTP protokolü üzerinden ulaşılabilen herhangi bir web servise istekte bulunmak için gerekli fonksiyonalliteyi sağlar.

Silverlight, HTTP tabanlı servislerin barındırılmasına destek vermez; dolayısıyla bu sınıflar uygulanabilir var olan bir web servisi kullanırken faydalı olacaktır. İlgili kaynağa yapılacak talebi oluşturmak ve servis tarafından beklenen formatla yapılacak isteğin formatının eşleştiğinden emin olmak için ihtiyaç duyarsın. Bu sınıfları genellikle şu durumlarda kullanırsın:

- Bir HTTP servisi senin uygulamanla alakası olmayan üçüncü parti bir kaynak tarafından sunuluyorsa
- Web servis senin kontrolünde değilse
- Servisin geriye döndürdüğü veri XML ya da JSON formatında ise

Eğer elde var olan veri modelinin üzerine kendin bir servis yazacaksan, Silverlight daha üretken uçtan uca bir çözüm sunar. Bu çözüm, WCF servisleri kullanılarak geliştirilir.

WCF Servisleri

Windows Communication Services (WCF), web servisleri oluşturmak ve web servislerine erişmek için var olan bir altyapı sunar. WCF, yazdığın bir sınıfı servis olarak dış dünyaya açmanı ve Silverlight ile bu servis arasında veri transferi yapmanı sağlar. Silverlight for Windows Phone uygulamanda Visual Studio içinde Solution Explorer ile Add Service Reference özelliği ile servisi kullanmak için gerekli olan sınıfın üretilmesini sağlayabilirsin. Üretilen yerel sınıf genellikle proxy class olarak adlandırılır. Bu sınıf, web servise sanki lokal bir sınıfın metodunu çalıştırıyormuşsun gibi erişmeni sağlar. WCF servisleri HTTP ve TCP protokolleri dahil olmak üzere geniş yelpazede protokolleri ve SOAP, XML gibi çok çeşitli veri formatlarını destekler (Yöndem, 2013).

WCF Data Services (OData Services)

Önceleri ADO.NET Data Services olarak bilinen WCF Data Services, var olan veri modelindeki veriye erişmek için kullanılan bir altyapıdır. WCF veri servisleri, veriyi Open Data Protocol (OData) beslemesi olarak sunar. Buna ek olarak eğer Silverlight for Windows Phone uygulaman Sharepoint ile iletişim halindeyse Sharepoint 2010 veriyi WCF Data Services olarak verir. WCF veri servisleri, bütün HTTP iletişimini, serileştirmeyi ve normalde senin yaptığın diğer bütün görevleri kendisi halleder. Bunun anlamı uygulamalar, standart HTTP protokolü aracılığıyla buradaki veriye erişirler, sorgu çalıştırırlar, silerler ve güncellerler. Bunu yaparken aynı ağ altında ya da farklı bir ağdan geliyor olmasının bir önemi yoktur. Windows Phone için OData fonksiyonelliği CodePlex üzerindeki istemci kütüphanesi tarafından sağlanır (Yöndem, 2013).

Windows Azure Storage Services

Windows Phone uygulamaların içinde veri saklamak ve elde etmek için Windows Azure kullanabilirsin. Özellikle cihaz içindeki saklama alanı limitli olduğu durumlar için bu seçeneği değerlendirebilirsin. Windows Azure platformu, windows phone uygulamaları için çok sayıda veri saklama seçeneği sunmaktadır. Windows Azure depolama servisleri, bulut üzerinde kalıcı, uzun ömürlü veri saklamayı sağlar ve artan/azalan taleplere cevap verebilmek için esnek bir şekilde ölçeklenebilir. Windows Azure depolama servislerine erişimin, web servislerine erişimine çok benzerdir. Windows Azure, veriyi Microsoft veri merkezlerinde saklar.

Hangi Yaklaşımı Kullanacağına Karar Vermek

İstek: Assembly içine sadece okunabilir veri gömmek *Önerilen çözüm:* Lokal kaynak dosyaları (Local resource file)

İstek: Yeniden derleme yapmadan veri güncellemek

Önerilen çözüm: Lokal içerik dosyaları (Local content file)

İstek: Kullanıcıya özel bilgi saklamak ve elde etmek

Önerilen Çözüm: Isolated Storage

İstek: Üçüncü parti bir REST servisindeki veriye erişmek

Çözüm: HTTP sınıfları

İstek: RSS beslemesine erişmek

Çözüm: HTTP sınıfları

İstek: Bir SOAP servisindeki veriye erişmek

Çözüm: WCF servisi, proxy sınıf kullanarak

İstek: Silverlight tarafından kullanılmak üzere var olan bir sınıfı servis olarak sunmak

Çözüm: WCF servisi

İstek: OData beslemesi sunan bir servisteki veriye erişmek

Çözüm: WCF Data Services (WCG Veri Servisleri)

İstek: Büyük miktarda veriyi saklamak ve o veriye erişmek

Çözüm: Windows Azure Storage Services (Windows Azure depolama servisleri)

5.5. *Kontrollere Veri Bağlamak(Data Binding To Controls)*

Birçok windows phone uygulaması kontrollerde veri gösterir. Çoğu durumda veri, bir iş nesnesi ya da stok kotası, makaleler, imaj listesi gibi iş nesnelerinden oluşan bir koleksiyon olarak ele alınır. Bunun yanında bazen son kullanıcının bir listeden öge seçmesini sağlamak; ardından başka bir kontrolde kullanıcının yaptığı seçimle ilgili detayları göstermek istersin (Microsoft Açık Akademi, 2013).

Bir Kontrolü Tek Bir Nesneye Bağlamak

Veri bağlama, hedef (target) ve kaynaktan (source) oluşur. Burada hedef genellikle bir kontrolün özelliğidir. Hedef olarak belirlenen bu özellik DependencyProperty sisteminde hazırlanmış olmalıdır.

Kod içinde TextBox sınıfı yazıp klavyeden F12 tuşuna basarsan örneğin Text özelliğinin hem özellik olarak hem de DependencyProperty olarak hazırlandığını görürsün. Sıradaki örnek bir kontrolü tek bir özelliğe nasıl bağlayacağını anlatıyor.

- Hedef olarak, TextBox kontrolünün Text özelliği kullanılabilir.
- Kaynak ise basit bir Kitap sınıfı olsun.

Örnek 15:

1.adım: Projeyi aç

Visual Studio içinde File - New Project - Visual C# - Silverlight for Windows Phone - Windows Phone Application adımlarını takip ederek yeni bir proje aç. Projenin ismine BindingSingleItem verebilirsin.

2.adım: XAML ile görsel arayüzü hazırla

Arayüze bir tane Textbox kontrolü sürükley bırak. Ardından şu özelliklerini sırayla değiştir. Bu değişiklikleri dersen F4 ile kontrolün Properties penceresinden, dersen doğrudan XAML içinden gerçekleştirebilirsin.

- *IsReadOnly = True*
- *Margin = 10*
- *VerticalAlignment=Top*
- *HorizontalAlignment=Left*
- *TextWrapping = Wrap*

- *Height* = 130
- *Width* = 350

Ayrıca kontrolün Text özelliğinin veri bağlama ile yapılacağını belirt.

- *Text* = {Binding}

XAML'in son hali şöyle olmalı:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<TextBox      Height="130"      Width="350"      VerticalAlignment="Top"      HorizontalAlignment="Left"
IsReadOnly="True" Margin="10"
TextWrapping="Wrap" Name="textBox1" Text="{Binding}" />
</Grid>
```

3.adım: Kitap sınıfını kodla

Projeye sağ tıklayış bir sınıf (class) ekle. İsmine Kitap.cs verelim.

Kitap.cs içine TextBox kontrolüne bağlayacağın Kitap sınıfını kodla ve 3 tane özellik ekle. Kodları şu şekilde olsun:

```
//Basit bir iş nesnesi
public class Kitap
{
public Kitap() { }
public Kitap(string adi, string yazari, DateTime yayinTarihi)
{
this.Adi = adi;
this.Yazari = yazari;
this.YayinTarihi = yayinTarihi;
}
public string Adi { get; set; }
public string Yazari { get; set; }
public DateTime YayinTarihi { get; set; }
public override string ToString()
{
return string.Format("{0} yazarının {1} isimli kitabı {2} tarihinde raflardaki yerini alacak.", this.Yazari,
this.Adi,this.YayinTarihi.ToShortDateString());
}
}
```

Kitap sınıfında 3 tane özellik, nesne kullanıcısından bu 3 özelliği alan yapıcı metot ve kitap nesnesi hakkında özet bilgi verecek şekilde ezilmiş ToString metodu yer almaktadır.

4.adım: Kitap nesnesinin kontrole bağlanması

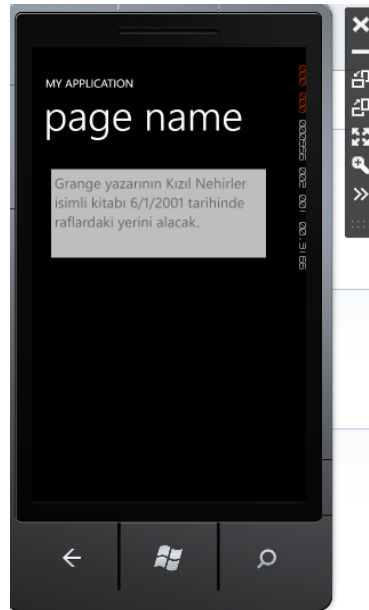
Bu adımda TextBox kontrolüne veri bağlama işini gerçekleştireceksin. Uygulamanın ana sayfasına ait yapıcı metot içinde (MainPage.xaml.cs) yapacaksın. Bunun için kontrolün DataContext özelliğine Kitap sınıfından bir nesne örneği atamalısın.

namespace BindingSingleItem

```
{  
public partial class MainPage : PhoneApplicationPage  
{  
    // Constructor  
    public MainPage()  
    {  
        InitializeComponent();  
        textBox1.DataContext = new Kitap("Kızıl Nehirler", "Grange", new DateTime(2001, 6, 1));  
    }  
}}
```

5.adım: Uygulamayı çalıştır

Uygulamayı bu haliyle çalıştırıp test ettiğinde şöyle bir çıktı elde etmen gerekiyor:



Şekil 36 – Örnek 15 ekran görüntüsü

TextBox kontrolüne bir kitap bilgisi göstermek için kontrolün Text özelliğine süslü parantezler arasına işaretleme eklentisi kullanarak {Binding} atadık. Bu örnekte veri bağlama modu, varsayılan değer olan BindingMode.OneWay olarak belirlenmiştir. Bunun anlamı veri kaynaktan elde edilir; ancak veri kaynağına güncelleme yapılamaz.

BindingMode özelliğini değiştireceğin yer, Text özelliğidir. Eğer xaml tarafındakodlarken otomatik kod tamamlayıcıdan yardım alabilirsin.

Text="{Binding Mode=TwoWay}"

ToString() metodu, bağlanan nesne üzerinden gösterim amacıyla çağrılır. Örnekte veri bağlayacağın Text özelliği için doğrudan Binding.Source özelliğine atama yapmadın, bunun yerine TextBox kontrolünün DataContext özelliğine yeni bir kitap nesnesi atadın.

Bir Kontrolü Nesne Listesine Bağlama

The following example binds a collection of music Recording objects to a ComboBox. To try this example, click the down arrow in the combo box to see the list of bound recordings.

Önceki örnek, kontrollere tek bir nesne olarak veri bağlayacağın zaman takip edeceğin yolu göstermektedir. Şimdi sırada daha yaygın olarak kullanacağın bir senaryo var. İş nesnelerinden oluşan bir liste ile çalışmak. Generic ObservableCollection sınıfı, veri bağlama için güzel bir seçim olur; çünkü INotifyPropertyChanged ve INotifyCollectionChanged arayüzlerini uygulamıştır. Bu arayüzler, listedeki öğelerden birinde değişiklik gerçekleştirğinde kontrole verinin değişen halini yeniden bağlamak için bildirim yaparlar. Eğer koleksiyon içindeki nesnelerin özelliklerinde meydana gelen değişikliklerde, veri bağladığın kontrollerin güncellenmesini istiyorsan iş nesnesi olarak kullandığın sınıfın da InotifyPropertyChanged arayüzünü uygulaması gerekir.

ÖRNEK 16: ComboBox kontrolüne Kitap nesnelerinden oluşan bir koleksiyon bağlar (Microsoft Açık Akademi, 2013).

1.adım: Projeyi aç

Visual Studio içinde File - New Project - Visual C# - Silverlight for Windows Phone - Windows Phone Application adımlarını takip ederek yeni bir proje aç. Projenin ismine BindingCollection verelim.

2.adım: XAML ile görsel arayüzü hazırla

Uygulamada liste halinde veri bağlama amacıyla kullanmak üzere bir ListBox kontrolü olacak. Yeni proje oluşturduğunda varsayılan olarak gelen ContentPanel isimli Grid içerisine yerleştirebilirsin.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">  
  
<ListBox VerticalAlignment="Top" Margin="15" x:Name="ListBox1" ItemsSource="{Binding}"  
FontSize="20" Height="80" Width="400"/>  
  
</Grid>
```

Veri bağlama ile ilgili önemli olan ItemSource özelliğidir. Buna değer olarak {Binding} vermen, kod tarafında ya da Properties penceresinde bir veri kaynağı atayacak olman anlamına gelmektedir.

3.adım: İş nesnesini hazırla

Önceki örnekte yazdığın Kitap sınıfını bu örnek için de kullanacaksın. Dilersen projeye sağ tıklayarak Add Existing Item ile önceki proje içinden Kitap.cs dosyasını bulup bu projeye ekleyebilirsin. Dilersen Kitap sınıfını bu projede yeniden hazırlayabilirsin.

```
public class Kitap  
{  
    public Kitap() { }  
    public Kitap(string adi, string yazari, DateTime yayinTarihi)  
    {  
        this.Adi = adi;  
        this.Yazari = yazari;  
        this.YayinTarihi = yayinTarihi;  
    }  
    public string Adi { get; set; }  
    public string Yazari { get; set; }  
    public DateTime YayinTarihi { get; set; }  
    public override string ToString()  
    {  
        return string.Format("{0} yazarının {1} isimli kitabı {2} tarihinde raflardaki yerini alacak.", this.Yazari,  
            this.Adi, this.YayinTarihi.ToShortDateString());  
    }  
}
```


4.adım: Veri kaynağı olarak kullanacağın koleksiyonu hazırla

Veri kaynağı olarak ObservableCollection<Kitap> koleksiyonunu kullanacaktır.

- Bu koleksiyonu kullanmak için ilgili isim alanını kod dosyasının en üstüne eklemen gerekir:

```
using System.Collections.ObjectModel;
```

- MainPage.xaml.cs dosyasında sınıf düzeyinde bir üye olarak koleksiyonu hazırla.
- Ardından sınıfın yapıcı metodunda koleksiyona 3 tane Kitap nesnesi ekle.
- Ve son olarak ListBox kontrolünün veri kaynağı olarak koleksiyon nesnesi ata.

```
using System.Collections.ObjectModel;
namespace BindingCollections
{
    public partial class MainPage : PhoneApplicationPage
    {
        ObservableCollection<Kitap> kutuphane = new ObservableCollection<Kitap>();
        //Yapıcı metot
        public MainPage()
        {
            InitializeComponent();
            kutuphane.Add(new Kitap("Kızıl Nehirler", "Grange", new DateTime(2001, 6, 1)));
            kutuphane.Add(new Kitap("Şu Çılgın Türkler", "Turgut Özakman", new DateTime(2006, 1, 24)));
            kutuphane.Add(new Kitap("Ütopya", "Thomas More", new DateTime(1516, 1, 1)));
            ListBox1.DataContext = kutuphane;
            ListBox1.DisplayMemberPath = "Adi";
        }
    }
}
```

Hazırlanan koleksiyon, ListBox kontrolünün DataContext özelliğine veri kaynağı olarak atandıktan sonra DisplayMemberPath özelliği ile kontrolde görünecek kolon belirlenir. Bu atamayı yapmazsan her bir satırda varsayılan olarak nesnenin ToString metodundan dönen değer görüntülenir.

Örnek 17: Listedeki Öğelerin Detayını Göstermek (Yöndem, 2013)

ListBox içinde kitap adlarını listelerken seçtiğin öğenin diğer detaylarını farklı bir alanda göstermek var. Bunun için öncelikle xaml tarafında bir hazırlık yapman gerekiyor.

1.adım: Projeyi aç

Visual Studio içinde File - New Project - Visual C# - Silverlight for Windows Phone - Windows Phone Application adımlarını takip ederek yeni bir proje aç. Projenin ismine DetailViewForCollection verebilirsin.

2.adım: İş nesnesini hazırla

Örnek 16 da yazdığımız Kitap sınıfını Projeye sağ tıklayarak Add Existing Item ile önceki proje içinden Kitap.cs dosyasını bulup bu projeye eklenir.

```
public class Kitap
{
    public Kitap() { }
    public Kitap(string adi, string yazari, DateTime yayinTarihi)
    {
        this.Adi = adi;
        this.Yazari = yazari;
        this.YayinTarihi = yayinTarihi;
    }
    public string Adi { get; set; }
    public string Yazari { get; set; }
    public DateTime YayinTarihi { get; set; }
    public override string ToString()
    {
        return string.Format("{0} yazarının {1} isimli kitabı {2} tarihinde raflardaki yerini alacak.", this.Yazari,
            this.Adi,this.YayinTarihi.ToShortDateString());
    }
}
```

3.adım: XAML ile görsel arayüzü hazırla

Listeden seçilen öğenin detay bilgilerini ListBox altında göstereceğin şekilde xaml kodunu güncellenir. Buradaki kodları ContentPanel isimli Grid içine koyulur.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <ListBox VerticalAlignment="Top" Margin="15" x:Name="ListBox1" ItemsSource="{Binding}"
        FontSize="20" Height="80" Width="400"/>
    <StackPanel x:Name="KitapDetayi" Margin="15,200,0,0" Orientation="Vertical">
        <TextBlock FontWeight="Bold" Text="{Binding Adi}" />
```

```

<TextBlock FontStyle="Italic" Text="{Binding Yazari}" />
<TextBlock Text="{Binding YayinTarihi}" />
</StackPanel>
</Grid>

```

Listbox altına 3 tane TextBox içeren bir StackPanel kontrolü yerleşiyor. StackPanel için Orientation özelliğine Vertical değerini veriyorsun; böylece altında yer alan TextBlock içerikleri alt alta görüntüleniyor. Bir diğer önemli ayrıntı ise TextBlock kontrollerinin Text özellikleri {Binding} ile işaretleniyor ve hemen yanında göstereceği nesne özelliğinin adını alıyor. Burada verdiği özellikler, Kitap sınıfının özellikleridir. ListBox ile kullanıcının seçtiği öğeye ait hangi detay bilgileri göstermek istiyorsan burada onları bildiriyorsun.

4.adım: Veri kaynağını hazırla

Yapılan bu görsel hazırlıkların yanında veriyi bağlarken de bazı değişiklikler yapman gerekiyor. ListBox ile seçtiğin öğenin detaylarını StackPanel içindeki TextBlock kontrollerinde göstermek için veri kaynağı olarak önceki örneklerde olduğu gibi koleksiyonun kendisini değil CollectionViewSource nesnesini kullanman gerekiyor. Bu sınıfı kullanmak için ilgili isim alanını kod dosyasının en üstüne eklenmelidir;

```
using System.Windows.Data;
```

CollectionViewSource sınıfı, ListBox kontrolünden seçilen öğenin detayının sayfaya bağlanmasını sağlar.

Detay görünümü ile çalışmak için bu örnekte uyguladığın diğer bir değişiklik, hazırladığın veri kaynağını xaml içinde en dışardaki LayoutRoot isimli Grid kontrolüne veriyor olmandır. Önceki örneklerde ListBox kontrolüne verdiğin veri kaynağı bilgisini, hem ListBox hem de StackPanel kontrollerini kapsayan daha dışardaki bir kontrole verirdin; böylece sayfadaki bütün kontroller seviyesinde veri bağlama sürecini yönetebilir ve görüntüleyebilirsin.

Bunun için hazırlaman gereken kod şöyledir:

```

namespace DetailViewForCollection
{
    public partial class MainPage : PhoneApplicationPage
    {
        ObservableCollection<Kitap> kutuphane = new ObservableCollection<Kitap>();
        // Constructor
    }
}

```

```

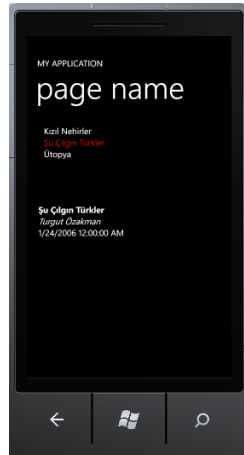
public MainPage()
{
    InitializeComponent();
    kutuphane.Add(new Kitap("Kızıl Nehirler", "Grange", new DateTime(2001, 6, 1)));
    kutuphane.Add(new Kitap("Şu Çılgın Türkler", "Turgut Özakman", new DateTime(2006, 1, 24)));
    kutuphane.Add(new Kitap("Ütopya", "Thomas More", new DateTime(1516, 1, 1)));
    LayoutRoot.DataContext = new CollectionViewSource { Source = kutuphane };
    ListBox1.DisplayMemberPath = "Adi";
}
}
}

```

Örnekte bütün sayfa içeriklerini kapsayan LayoutRoot Grid kontrolüne DataContext olarak CollectionViewSource nesnesi verildiğini görüyorsun. Nesnenin Source özelliğine, hazırladığın kutuphane koleksiyonunu atarsın.

5.adım: Uygulamayı test et

Uygulamayı çalıştırdığında şöyle bir görüntü ile karşılaşman gerekir. Test etmek için ListBox kontrolüne listelenen kitaplardan her birini seç ve aşağı kısımda ilgili kitabın diğer detaylarının görüntülendiğine dikkat et.



Şekil 37 – Örnek 17 Ekran görüntüsü

Lokal(Yerel) Veri Tabanı ile Çalışmak

Windows Phone ile programlama yapmaya başlayıp birkaç uygulama ile deneyim kazandıktan sonra yazacağın gerçek hayat uygulamalarında veriyi yerel olarak saklayacağın bir çözüme ihtiyaç duyarsın. Windows Azure servisleri veya bir web servis arkasından erişebileceğin uzak bir sql sunucusu, ilişkisel veri saklama ihtiyacını karşılayabilir. Bu çözümler büyük miktarda veri ile çalışırken belki ilk tercihin de olabilir. Ancak her ne kadar

elindeki telefon bir mobil cihaz olsa ve çoğu zaman internete bağlı olduğunu düşünsen de, uzak bir lokasyonda saklayamayacağın ve çevrimdışı olduğunda dahi her zaman erişmen gereken bilgiler olacaktır. Tabi ki uygulama ile ilgili ayarlar bu tarz verilere örnek olarak gösterilebilir; ancak burada esas anlatılmak istenen yerel olarak cihaz içinde saklayabileceğin ilişkisel veridir. Bu şekilde saklayabileceğin veri ile birlikte uygulamayı hızlı bir şekilde çalıştırabilirsin ve sürekli internete bağlı olmana gerek kalmaz (Microsoft Açık Akademi, 2013).

Önceki Windows Phone versiyonu 7.0 için tek yerel veri saklama yeri isolated storage, normal bir dosya sistemi gibi kullanılıyor ve iş nesnelerini serileştirmek ve de-serileştirmek için azımsanmayacak bir kod yazmanı gerektiriyor. Bu problemi daha kolay aşmak için bazı üçüncü parti çözümler oluşturuldu. Yeni Windows Phone 7.1 versiyonu geldiğinde Microsoft, iş uygulamalarına yardımcı olacak ve SQL Server Compact Edition (SQL Server CE) üzerine kurulu lokal veritabanı uygulamasını eklemeye karar verdi. Bu yeni çözüm, ilişkisel veritabanına ihtiyaç duyduğunda cihaz içinde kullanabileceğin veri saklama çözümlerini tamamladı. Üstelik kurması ve kullanması oldukça kolay gerçek bir ilişkisel veritabanına sahip olmak, birçok senaryo için gerçekten çok büyük rahatlık.

Windows Phone 7.1 versiyonunda kullanabileceğin yeni lokal veritabanı desteği, SQL Server ürününün popüler versiyonu SQL Server Compact Edition üzerine inşa edilmiştir. SQL Server CE, eski Windows Mobile ve Windows CE için ilişkisel veritabanı desteği sağlamak için oluşturulmuş bir versiyondur. Windows Phone 7 ve sonrası versiyonlarında Microsoft.NET Compact Framework 3.7 olduğu için ilişkisel veritabanı olarak SQL Server CE olması gayet normaldir ve gerçek hayat uygulamalarında kullanman için sağlam ve güvenilir araç olduğu kesindir.

Fakat şöyle bir gerçek var ki, Windows Phone 7 için var olan Silverlight uygulama programlama arayüzü, ADO.NET bileşenlerini içermemektedir; bu yüzden SQL CE veritabanına erişimi, normal T-SQL sorguları ile yapamazsın. Bu amaçla Windows Phone 7 ekibi, var olan LINQ to SQL altyapısının aldılar ve telefon içinde kullanılmak üzere entegre ettiler. Bunun anlamı, SQL CE veritabanına erişim için LINQ sorguları, veritabanındaki bilgileri elde etmek ve değiştirmek için DataContext nesnesini kullanmak zorundasın. LINQ to SQL, gerçek bir nesne yönelimli veritabanı ortamı sunmaz. Nitelik (attribute) tabanlı eşleştirme kullanır. Ancak windows phone üzerinde yerel veritabanı oluşturmak ve veritabanına erişmek amacıyla iyi, kolay ve güvenilir bir seçimdir.3 / 8

Bir SQL CE veritabanı genellikle SDF dosyaları ile ilişkilendirilir. Windows Phone ortamında da yine SDF dosyası vardır; ancak burada isolated storage içine yerleştirilir. Bu yüzden ilk dikkat etmen gereken şey bağlantı için özel connectionstring bilgisinin formatıdır (Microsoft Açık Akademi, 2013).

isostore://veritabanim.sdf

Dikkat etmek gereken bir diğer nokta sdf dosyasına isolated storage içinde sahip olmak, bu dosyaya erişim için sadece onu oluşturan uygulama ile ilişkilendirilmesi anlamına gelir. Windows Phone 7.1 için birden fazla uygulamanın lokal veritabanını ortak kullanması için herhangi bir yol yok. Bunun için başvuracağın yöntem, web servis arkasındaki uzak bir sql sunucusu ya da Windows Azure saklama servisleridir.

Windows phone uygulaman içinde SQL Server CE ile çalışmak için System.Data.Linq.dll kütüphanesini proje referanslarına eklemen gerekir. Bu referansı ekledikten sonra SQL CE çalışma zamanını kullanmak için başka bir şey yapmana gerek yok; çünkü SQL CE zaten Windows Phone 7.1 versiyonunun dahili bir parçasıdır; bu yüzden uygulamanın boyutunu artırmaz.

İş Sınıflarını Tablolarla Eşleştirmek

LINQ to SQL içinde tablolar ile kod içindeki sınıfların arasında doğrudan ilişki kurulur. Her tablo için eşleştirilmiş bir sınıfın olur. İş sınıflarını tablolar ile eşleştirmek oldukça kolaydır. Bu amaçla kullanmak üzere sınıf tanımlamasının üzerine atayabileceğin birkaç tane nitelik (attribute) vardır (Delprato).

- TableAttribute, sınıfın tablo ile ilişkilendirildiğini gösterir.
- ColumnAttribute, sınıf içindeki özelliklerin tablodaki kolonlar ile ilişkilendirildiğini gösterir.
- AssociationAttribute, tablolar arasındaki ilişkiyi gösterir.

İş sınıfların varsayılan olarak uygulamanın ihtiyaç duyduğu kendi kodlarını ekleyebileceğin, kalıtım mimarisine sokabileceğin şekilde tasarlanır. Bu şekildedeki sınıflara POCO nesneleri de denir (Plain Old C# Object). Örneğin veri bağlama ihtiyacı için sınıfına, InotifyPropertyChanged arayüzünü uygulayabilir, aynı zamanda bazı özel senaryolar için başka bir sınıftan türetebilirsin.

Örnek 18:

1.adım: Yeni proje aç

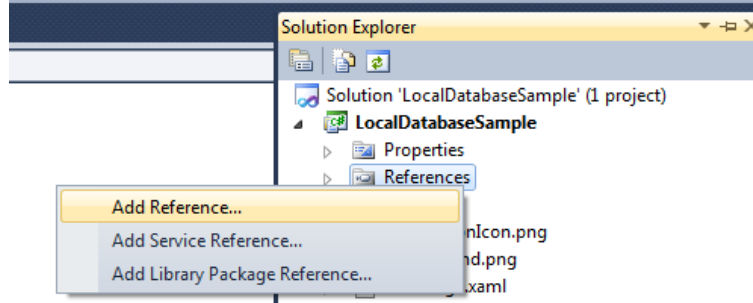
Visual Studio içinde File

☐ New Project

Windows Phone Application adımlarını takip ederek yeni bir proje aç. Projenin ismine BindingSingleItem verebilirsin.

2.adım: Gerekli referansları projeye ekle

Visual Studio içinde Solution Explorer penceresi içinde Add Reference menüsünü tıklayarak projeye System.Data.Linq.dll kütüphanesini ekle.



Şekil 38 – Projeye referans ekleme işlemi

3.adım: İş nesnesini hazırla

Visual Studio içinde Solution Explorer penceresinde projeye bir sınıf (class) dosyası ekle ve dosya içine UrunNo, UrunAdi, CikisTarihi özelliklerine sahip Urun isimli bir sınıf yaz. Ardından bu sınıfın ilişkisel veritabanı içinde tablo ismi ve kolon isimleri karşılıklarını belirle. Bunu yaparken TableAttribute ve ColumnAttribute niteliklerden faydalanabilirsin. Bu nitelikleri kullanmak için Urun.cs kod dosyasının en üstüne ilgili isim alanını eklemen gerek.

```
using System.Data.Linq.Mapping;
namespace LocalDatabaseSample
{
    [Table(Name="Urunler")]
    public class Urun
    {
        [Column(IsPrimaryKey=true,Name="ProductId")]
        public int UrunNo { get; set; }
        [Column]
        public string UrunAdi { get; set; }
    }
}
```

```
[Column(Name="SalesDate")]
public DateTime CikisTarihi { get; set; }
}
}
```

Burada ColumnAttribute niteliğini, atadığın özelliğin davranışını belirlemek için kullanırsın. Örneğin UrunNo özelliğinin tablodaki adı ve tablodaki primary key kolonu olacağını belirtebiliyorsun. Eğer UrunAdi özelliğinde olduğu gibi özel olarak kolon adı belirlemezsek ilgili kolon, veritabanında özellik adı ile aynı isme sahip olur.

4.adım: Veritabanını oluşturmak

Urun sınıfı gibi istediğin sayıda iş nesnesini hazırladıktan sonra fiziksel olarak veri tabanını oluşturmadan önce yazman gereken son bir sınıf daha var. Bu, DataContext sınıfından türeyen kendi yazacağın özel bir sınıf olacak. DataContext sınıfı, veri ekleme, güncelleme, silme gibi veri tabanında yapabileceğin bütün işlemleri bünyesinde barındırır ve aynı zamanda veri tabanındaki her tablonun bütün verilerine erişmek için birer liste sunar.

Bu adımda projeye sağ tıklayarak yeni bir sınıf oluştur, kod dosyasına UrunlerDataContext.cs ismini verebilirsin. DataContext ile çalışabilmek için kod dosyasının en üstüne ilgili isim alanını eklemen gerekir.

```
using System.Data.Linq;
namespace LocalDatabaseSample
{
    public class UrunlerDataContext : DataContext
    {
        public const string ConnectionString = "isostore:/urunler.sdf";
        public Table<Urun> Urunler { get; set; }
        public UrunlerDataContext(string connectionString) : base(connectionString)
        {
            this.Urunler = this.GetTable<Urun>();
        }
    }
}
```

UrunlerDataContext sınıfı içinde sabit olarak tanımlanan connectionstring bilgisi var. Bu bilgiyi ilerleyen adımlarda kullanacaksın.

Urunler isimli listeyi, veritabanında oluşacak tabloda yer alan bütün kayıtlarına erişmek için kullanacaksın. Bu özelliğin veri tipinin Table<T> olmasının sebebi, LINQ to SQL altyapısının verileri bu tipte sunmasından kaynaklanmaktadır.

Yapıcı metodun parametresinde, nesne kullanıcısından alacağın connectionstring bilgisini temel sınıf DataContext yapıcı metoduna aktarır. Yapıcı metodun kod bloğunda ise veritabanında yer alan Urunler tablosundaki bütün kayıtların GetTable<T>() metodu ile yazdığın Urunler özelliğine atanmasını sağlıyorsun.

5.adım: Veritabanını oluşturan kodu yaz

Eğer veritabanı yoksa oluşturulmasını sağlayacak kodu yazma vakti geldi. Bunun için projeye sağ tıklayıp yeni bir sınıf ekle. Kod dosyasının adını UrunDeposu.cs olarak belirleyebilirsin. Kod dosyasının üst kısmına 2 tane isim alanı eklemek gerekiyor:

```
using System.Collections.Generic;  
using System.Linq;
```

Bunlardan ilki generic koleksiyonların sunduğu yeteneklerden faydalanmak; ikincisi ise LINQ sorguları yazabilmek içindir.

```
public class UrunDeposu  
{  
    public static void Baslat()  
    {  
        UrunlerDataContext ctx = new UrunlerDataContext(UrunlerDataContext.ConnectionString);  
        if (ctx.DatabaseExists() == false)  
        {  
            ctx.CreateDatabase();  
            //Yaptığın değişiklikleri veri kaynağına yansıtırsın.  
            ctx.SubmitChanges();  
            //Burada diğer başlangıç satırlarını kodlayabilirsin. Örneğin statik bazı verileri, veritabanındaki b,r tabloya  
            ekleyebilirsin.  
            VeriEkle();  
        }  
    }  
    private static void VeriEkle()  
    {  
        UrunlerDataContext ctx = new UrunlerDataContext(UrunlerDataContext.ConnectionString);
```

```

ctx.Urunler.InsertOnSubmit(new Urun { UrunNo = 1, UrunAdi = "Notebook", CikisTarihi = new
DateTime(2012, 6, 23) });
ctx.Urunler.InsertOnSubmit(new Urun { UrunNo = 2, UrunAdi = "Kinect", CikisTarihi = new DateTime(2010,
2, 17) });
ctx.Urunler.InsertOnSubmit(new Urun { UrunNo = 3, UrunAdi = "Televizyon", CikisTarihi = new
DateTime(2010, 2, 17) });
ctx.SubmitChanges();
}
public static List<Urun> UrunListesiAl()
{
UrunlerDataContext ctx = new UrunlerDataContext(UrunlerDataContext.ConnectionString);
List<Urun> urunListesi = (from u in ctx.Urunler
orderby u.CikisTarihi descending
select u).ToList();
return urunListesi;
}
}

```

Baslat() metodu, veritabanını oluşturmak için gereken kodu içermektedir. Eğer uygulama ilk defa çalıştırılıyorsa Baslat() metodu içinde veritabanı oluşturulduktan hemen sonra VeriEkle() metodunu çağırıp Urunler tablosunda bulunması gereken örnek verileri de bu aşamada eklenir.

Dolayısıyla bu kodu uygulama ilk açıldığında çalıştırmak yeterli olacaktır; bunun için en uygun yer App.xaml dosyasıdır. Yeni bir proje oluşturduğunda varsayılan olarak projene eklenen bu dosyadaki App sınıfı, uygulaman ilk açıldığında çalışan metotlara ev sahipliği yapar. Bu metotlardan bir tanesi varsayılan olarak içi boş olan Application_Launching metodudur. Senin tek yapman gereken içine statik Baslat() metodunu çağırarak kodu yazmaktır. Böylece uygulaman ilk defa açıldığında önceden veritabanının oluşturulup oluşturulmadığı kontrol edilir.

```

private void Application_Launching(object sender, LaunchingEventArgs e)
{
UrunDeposu.Baslat();
}

```

6.adım: Tablodaki verileri kullan

Test etmek amacıyla tablodaki verileri bir ListBox kontrolünde kullan. Bunun için MainPage.xaml sayfasına ToolBox penceresinden bir tane ListBox sürükleyip bırak. Xaml kodundan ya da Properties penceresinden kontrole veri bağlanacağını belirtecek şekilde ItemSource özelliğini değiştir.

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

MainPage sayfasında görsel arayüzün son hali aşağıdaki gibi olmalıdır:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<ListBox Height="100" HorizontalAlignment="Left" Margin="12,133,0,0" Name="listBox1"
VerticalAlignment="Top" Width="394" ItemsSource="{Binding}" />
</Grid>
```

Son olarak UrunDeposu sınıfına yazdığın UrunListesiAl metodundan dönen listeyi ListBox1 kontrolüne veri kaynağı olarak ver. Listede görünecek kolon olarak UrunAdi kolonunu belirleyebilirsin. Bu işler için MainPage.xaml.cs dosyasında MainPage sınıfının yapıcı metodunu kullanabilirsin.

```
public partial class MainPage: PhoneApplicationPage
{
//Yapıcı metod
public MainPage()
{
InitializeComponent();
listBox1.DataContext = UrunDeposu.UrunListesiAl();
listBox1.DisplayMemberPath = "UrunAdi";
}
}
```

7.adım: Uygulamayı test edelim.

6. BÖLÜM 5: WINDOWS PHONE UYGULAMALARINI DEVCENTER'A GÖNDERMEK

Windows Phone uygulamalarını geliştirdikten sonra, uygulamanıza ait bilgileri girerek Dev Center'a göndermeye hazırlamanız gerekir.

WMAppManifest.xml dosyası işlemleri

WMAppManifest.xml dosyasında düzenleme yapabilmek için, Properties klasörüne girdikten sonra WMAppManifest.xml dosyasını açmanız gerekir.


Application UI | Capabilities | Requirements | Packaging




Use this page to set the UI details that identify and describe your application.

Display Name: OrnekUygulama

Description: Sample description

Navigation Page: MainPage.xaml




App Icon: 

Supported Resolutions:  ☒ wvga  ☒ wxga  ☒ 720p

Tile Template: TemplateFlip

☐ Support for large Tiles

Tile Title: OrnekUygulama

Tile Images: Small:  Medium:  Large: 

Şekil 39 - WMAppManifest.xml dosyası görünümü

Application UI penceresinde, uygulamanıza ait simge dosyalarını, desteklediği çözünürlükleri ve uygulama yüklendiğinde görünecek ismi belirleyebilirsiniz.

Eğer uygulamanızda büyük bir uygulama alanı kullanacaksanız, Support for large Tiles seçeneğini işaretlemeniz ve Tile Images kısmındaki Large bölümünden resmimizi seçmeniz gerekir.

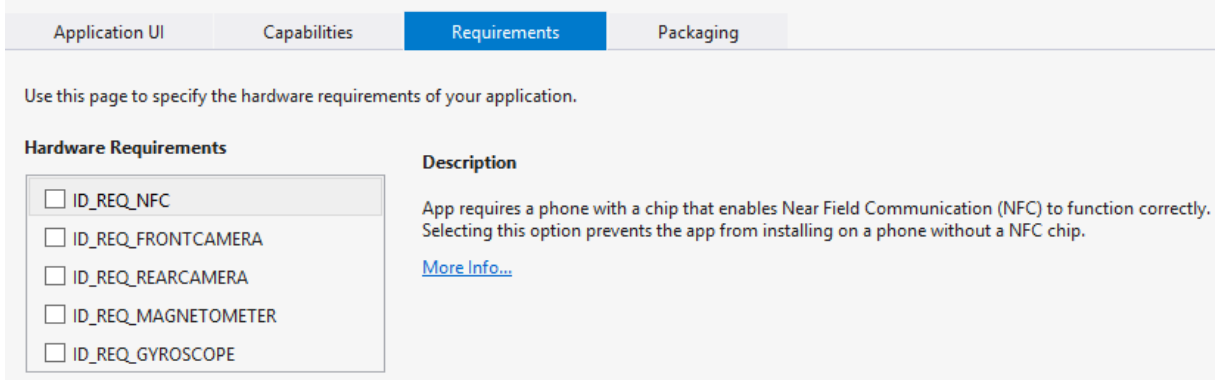
Capabilities penceresinde ise uygulama içinde kullanacağınız Cihaz Yeteneklerini belirleyebilir ve buna göre kullanıcıdan izin alma işlevlerini otomatik olarak yerine getirebilirsiniz.

Capabilities	Description
<input type="checkbox"/> ID_CAP_APPOINTMENTS	Provides access to appointment data. More Info...
<input type="checkbox"/> ID_CAP_CONTACTS	
<input type="checkbox"/> ID_CAP_GAMERSERVICES	
<input type="checkbox"/> ID_CAP_IDENTITY_DEVICE	
<input type="checkbox"/> ID_CAP_IDENTITY_USER	
<input type="checkbox"/> ID_CAP_ISV_CAMERA	
<input type="checkbox"/> ID_CAP_LOCATION	
<input type="checkbox"/> ID_CAP_MAP	
<input checked="" type="checkbox"/> ID_CAP_MEDIALIB_AUDIO	
<input type="checkbox"/> ID_CAP_MEDIALIB_PHOTO	
<input checked="" type="checkbox"/> ID_CAP_MEDIALIB_PLAYBACK	
<input type="checkbox"/> ID_CAP_MICROPHONE	
<input checked="" type="checkbox"/> ID_CAP_NETWORKING	
<input type="checkbox"/> ID_CAP_PHONEDIALER	
<input type="checkbox"/> ID_CAP_PROXIMITY	
<input type="checkbox"/> ID_CAP_PUSH_NOTIFICATION	
<input type="checkbox"/> ID_CAP_REMOVABLE_STORAGE	
<input checked="" type="checkbox"/> ID_CAP_SENSORS	
<input checked="" type="checkbox"/> ID_CAP_WEBBROWSERCOMPONENT	

Şekil 40 - Capabilities penceresi

Örneğin, cihazın internet bağlantısını kullanacaksanız, ID_CAP_NETWORKING seçeneğini veya bir numarayı aramanız gerekiyorsa, ID_CAP_PHONEDIALER seçeneğini işaretlemeliyiz.

Requirements penceresinde, bu uygulamayı yükleyecek olanlar için kısıtlamaları belirtebilirsiniz:



Application UI Capabilities **Requirements** Packaging

Use this page to specify the hardware requirements of your application.

Hardware Requirements

- ☐ ID_REQ_NFC
- ☐ ID_REQ_FRONTCAMERA
- ☐ ID_REQ_REARCAMERA
- ☐ ID_REQ_MAGNETOMETER
- ☐ ID_REQ_GYROSCOPE

Description

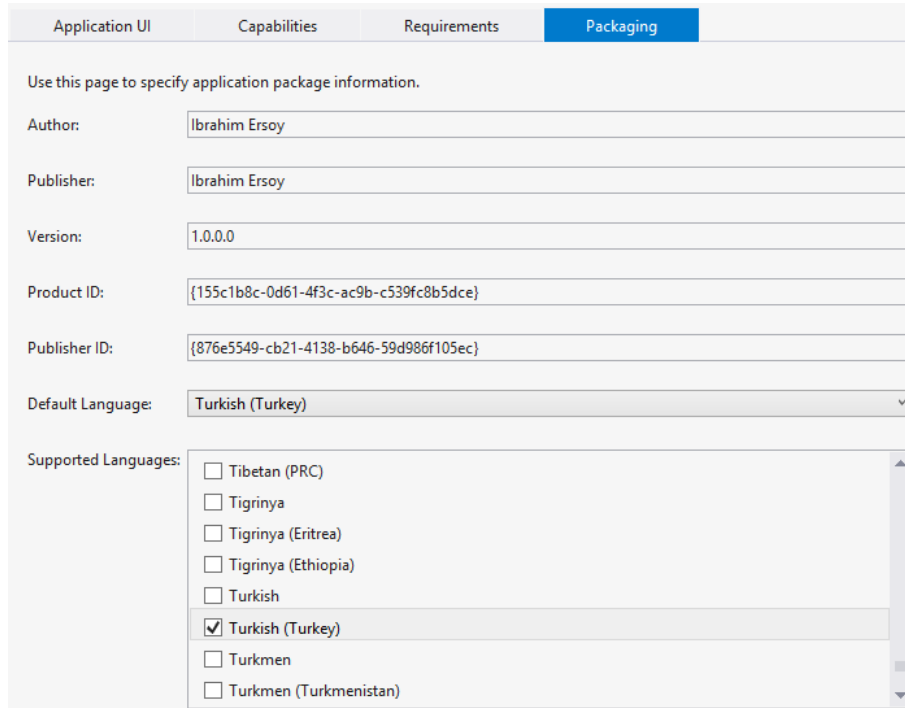
App requires a phone with a chip that enables Near Field Communication (NFC) to function correctly. Selecting this option prevents the app from installing on a phone without a NFC chip.

[More Info...](#)

Şekil 41 - Requirements penceresi

Örneğin, eğer kullanıcının cihazı NFC'yi desteklemiyorsa ID_REQ_NFC seçeneğini işaretleyerek kullanıcının bu uygulamayı yüklemesini engelleyebilirsiniz. Kullanıcının cihazında olmayan her özelliği buradan belirtebilir ve yüklenme işlemini kısıtlayabilirsiniz.

Packaging kısmında, uygulama paketine yönelik bilgileri görebilir, düzenleyebilir veya kullanılan dilleri ve desteklenen dilleri belirtebilirsiniz.



Application UI Capabilities Requirements **Packaging**

Use this page to specify application package information.

Author: Ibrahim Ersoy

Publisher: Ibrahim Ersoy

Version: 1.0.0.0

Product ID: {155c1b8c-0d61-4f3c-ac9b-c539fc8b5dce}

Publisher ID: {876e5549-cb21-4138-b646-59d986f105ec}

Default Language: Turkish (Turkey)

Supported Languages:

- ☐ Tibetan (PRC)
- ☐ Tigrinya
- ☐ Tigrinya (Eritrea)
- ☐ Tigrinya (Ethiopia)
- ☐ Turkish
- ☒ Turkish (Turkey)
- ☐ Turkmen
- ☐ Turkmen (Turkmenistan)

Şekil 42 – Packaging penceresi

Şekilde, varsayılan durumda Turkish (TR-tr) dili kullanılırken, desteklenen dillerde de Turkish seçilmiştir. Burada yapılan işlem, aslında dil ayarı Türkçe olmayan cihaz sahiplerinin bu uygulamayı yüklemesini ve kullanmasını engellemeye yönelik bir işlemdir. Eğer farklı dillerin de desteklenmesini istiyorsanız, Supported Languages kısmından istediğiniz bir dili seçerek, o dil ayarı yüklü cihazların da uygulamanızı kullanmasını sağlayabilirsiniz.

DevCenter

Dev Center, Microsoft'un Windows Phone uygulamaları için oluşturduğu bir mağazadır. Windows Store ile sürekli karıştırılır. Windows Store sadece Windows 8 uygulamalarına yönelik bir mağaza iken, Dev Center Windows Phone uygulamalarına yöneliktir. İki farklı mağaza olmasının nedeni, Windows Phone ve Windows 8 platformlarında uygulama geliştirirken; her ne kadar aynı dil yapısını, XAML'i ve Windows 8 Core'u kullansalar da bir platforma yönelik geliştirilen bir uygulamanın birebir diğerinde çalışmayacak olmasıdır. Uygulama geliştirenler geliştirdikleri uygulamalarını ister tam, ister deneme sürümü olsun, ücretsiz veya ücretli bir şekilde yayınlayabilirler.

Dev Center'a üye olup uygulama göndermeden önce yapılması gereken ilk işlem bir geliştirici lisansı satın almaktır. Microsoft'un Dev Center için biçtiği geliştirici lisansı ücreti 99\$'dır. Ancak eğer MSDN üyeliğiniz varsa, Dev Center'ı 1 yıl süresince ücretsiz kullanabilirsiniz. MSDN üyeliğinizdeki "Special Offers" kısmına geldiğinizde, Dev Center'da ücretsiz hesap açmak için size 1 yıllık ücretsiz lisans için bir şifre verilir. Bu şifreyi Dev Center üzerinde yeni bir geliştirici lisansı oluştururken kullanırsanız, 1 yıl boyunca tamamen ücretsiz bir şekilde Dev Center ve sunduğu tüm imkânlardan yararlanabilirsiniz. Ancak bu bir yılsonunda Dev Center sizden üyeliği yenilenmesini isteyecektir. Aksi belirtilmediği sürece, yenileme ücreti her yıl için 99\$'dır (Microsoft Açık Akademi, 2013).

Dev Center Uygulamaları

Dev Center'daki uygulamaları görmenin 2 farklı yolu vardır: Windows Phone üzerinden ya da Web üzerinden Store'a girerek. Web üzerinden: <http://www.windowsphone.com/tr-tr/store> adresinden Store'a erişebilirsiniz. Eğer beğendiğiniz bir uygulama olursa buradan yükleyebilir ya da ilgili uygulama destekliyse, "Sonradan kur" seçeneği sayesinde. XAP uzantılı Windows Phone uygulamasını makinenize indirip Windows Phone Application Deployment vasıtasıyla ister aygıt üzerine, ister Emulator'e kurup aygıtı aktarmadan önce test edebilirsiniz. Bu da beraberinde. XAP uzantılı dosyanın güvenlik konusunu gündeme getirir.

Eğer siz uygulamanızı çeşitli programlar vasıtasıyla şifrelemezsensiz, Reflector gibi uygulamalar aracılığıyla bu işi hiç bilmeyen birisi bile kodunuzu tersine derleyerek tersine mühendislik işlemi gerçekleştirebilir. Bu noktada uygulama geliştirdikten sonra ve markete atmadan hemen önce tanınmış şifreleyici uygulamalar ile projenizi şifreleyip o şekilde markete göndermeniz gerekir. Bu konuya özellikle dikkat edilmesi gerekir (Microsoft Açık Akademi, 2013).

Dev Center Üyeliği Oluşturmak

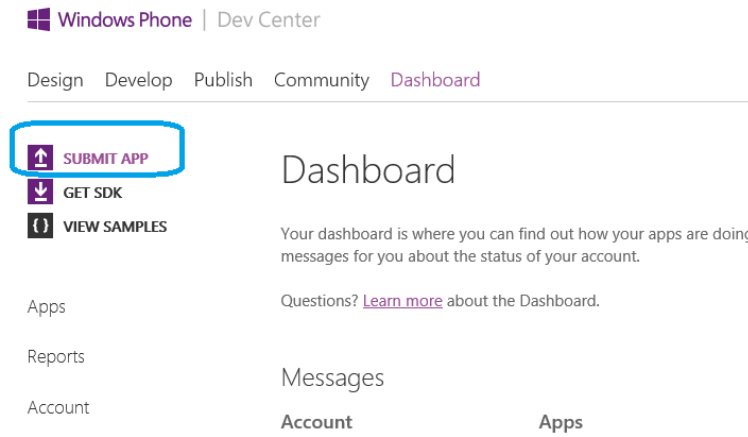
<http://dev.windowsphone.com> üzerinden üyelik oluşturulmaktadır.

Dev Center'a Uygulama Yükleme

Diyelim ki, bir uygulama geliştirdiniz ve uygulamaya ait tüm nesneleriniz, resimleriniz, simgeleriniz hazır bir şekilde mevcut. Dev Center'a sahip olduğunuz geliştirici lisansı ile uygulama göndermek için aşağıdaki işlemleri gerçekleştirin.

Dashboard'a gir:

Submit App seçeneğini tıklayın.



Şekil 43 – Dashboard ekranı

Dashboard'da ana başlık gelecektir: App Info, XAP Dosyasını yükleyeceğiniz alan, Market Seçimi, Harita Servisleri.



Submit app

You've spent hours developing and designing your app, and now it's time for the rest of the world to experience your masterpiece. In just two steps we'll gather the information we need to successfully launch your app in the Windows Phone Store. [Learn more](#) about the steps for successfully submitting your app.

Required

- 1** App info
Give your app a Dev Center alias, price it, and enter other relevant info
- 2** Upload and describe your XAP(s)
For each XAP in your app, this is where you'll enter descriptions and upload screenshots that will showcase your app in the Store.

Optional

-  Market selection and custom pricing
For apps, you have the option to define different pricing and availability for different countries/regions.
-  Map services
Get the token required to use map services in your app.

[Review and submit](#)

Şekil 44 – Dashboard karşılama ekranı ve yapılması gereken 4 adım

Uygulamanıza ait açıklama bilgilerini girmek için App Info kısmına girdikten sonra, yapılan değişiklikleri kaydetmek için Save seçeneğini tıklayıp devam edilir.

App info

The info on this page is used to refer to your app here in the Dev Center, and also controls how it appears in the Store.

App info

App alias*

This name is used to refer to your app here on Dev Center. The name your customer sees is read directly from your XAP file.

Category*

Subcategory

Pricing

Base price*

Free or paid? If paid, how much? [Learn how](#) this affects pricing in different countries/regions.



TRY

You'll need to provide your tax or bank info (or both) if you want to submit paid apps.

- ☐ Offer free trials of this app. Before you select this option, make sure you've implemented a trial experience in your app. [Learn more.](#)

Market distribution

- ☒ Distribute to all available markets at the base price tier
☐ Distribute to all markets except China. [Learn more.](#)
☐ Continue distributing to current markets

More options ▼

Şekil 45 – Appinfo – uygulamaya ait açıklamaların yazıldığı ekran

Ardından, uygulamanızı karşıya yüklemek için ikinci seçeneğe girin. Burada ilk yapılacak işlem, XAP uzantılı ve Release modda Build edilmiş Windows Phone uygulamanızı yüklemektir. Bu XAP dosyasının yüklemesi bitince ek içerik girebileceğiniz alanlar açılacaktır.

Upload and describe your XAP

Hello World

If your app contains more than one XAP, you can upload additional files as soon as you upload the first one here. [Learn more.](#)

XAPs

This is an important page, because in addition to uploading your XAP, you're also creating your customer's first impression of your app. The info you provide will be part of the Store's listing of your app. If you're updating an existing app, this page will also include XAPs that you've already uploaded. All these XAPs will be available in the Store after you've published your submission.

XAP name	Version	OS	Resolution	Language		
<input checked="" type="radio"/> PhoneApp1_Debug_AnyCPU.xap	1.0.0.0	8.0	WVGA, 720P, WXGA	English	Replace	Delete

[Add new](#)

Select a XAP above to view or edit its Store listing and other info.

XAP version number*

1

.

0

.

0

.

0

More XAP options 

XAP details detected from file 

File name	PhoneApp1_Debug_AnyCPU.xap
File size	42 KB
Supported OS	8.0
Resolution(s)	WVGA 720P WXGA
Language(s)	English
Capabilities	ID_CAP_MEDIALIB_AUDIO ID_CAP_MEDIALIB_PLAYBACK ID_CAP_NETWORKING ID_CAP_SENSORS ID_CAP_WEBBROWSERCOMPONENT ID_RESOLUTION_HD720P ID_RESOLUTION_WVGA ID_RESOLUTION_WXGA

Şekil 46 – Dashboard’ XAP dosyasını yüklenmesi

XAP's Store listing info

A XAP file can contain multiple languages. Select a language to add Store listing info specific for that language.

English

More options per language

Description for the Store*

2000 characters remaining.

Specify keywords

Used as exact words or phrases in the Store's search to help people find your app

Upload images*

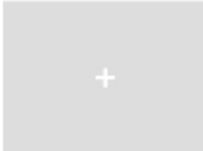
Use this link to upload all the images in one step, or upload them separately by clicking on each image placeholder below.

[Upload all](#)

★ App tile icon 300 x 300 px



Background image 1000 x 800 px



Şekil 47 – XAP Store Listing ekranı

☒ Automatically create lower resolution screenshots from WXGA

WXGA | 720p | WWGA

★ We'll use these screenshots to showcase your app.

For WXGA they must be 768 x 1280 px or 1280 x 768 px. [Learn more.](#)



Save

Şekil 48 – Uygulamanın ekran görüntülerini yükleme ekranı

Bu alanları da doldurup Save seçeneğini tıkladıktan sonra, Submit App kısmına geri dönersiniz. Eğer DevCenter seçimi yapmak isterseniz, üçüncü seçenek olan DevCenter Selection'ı seçmeniz gerekir. Ülkelere göre değişik ücretler belirleyebilir ya da birinci seçenek olan “Distribute to all available markets at the base price tier” ifadesini seçerek tüm marketlerde aynı fiyat üzerinden yayınlanmasını belirtebilirsiniz.

Define market pricing

Hello World

Change where your app is available and how much it will cost in those individual markets.

- ☐ Distribute to all available markets at the base price tier
☐ Distribute to all markets except China. [Learn more.](#)
☒ Continue distributing to current markets

ⓘ These countries/regions have more restrictions for app or in-app product content. [Learn more.](#)

<input checked="" type="checkbox"/> Afghanistan ⓘ	0.00 ▼ USD	<input checked="" type="checkbox"/> Lithuania	0.00 ▼ LTL
<input checked="" type="checkbox"/> Albania ⓘ	0.00 ▼ USD	<input checked="" type="checkbox"/> Luxembourg	0.00 ▼ EUR
<input checked="" type="checkbox"/> Algeria ⓘ	0.00 ▼ DZD	<input checked="" type="checkbox"/> Macao SAR	0.00 ▼ USD
<input checked="" type="checkbox"/> Andorra	0.00 ▼ EUR	<input checked="" type="checkbox"/> Macedonia FYRO	0.00 ▼ USD
<input checked="" type="checkbox"/> Angola	0.00 ▼ USD	<input checked="" type="checkbox"/> Madagascar	0.00 ▼ USD

Şekil 49 – Uygulamanın fiyatlandırma ekranı

Eğer uygulamanızda Bing Maps kullandıysanız dördüncü seçenek olan Maps Services’a girmeniz ve bir token (jeton) almanız gerekir. Bu token’ı uygulamanızda kullanmanız gerekeceğinden, projenizi Build > Debug > Build > Release komutunu seçerek tekrar karşıya yüklemeniz gerekir.

Map services

Hello World

To use map services in your app, you need a map service application ID and a token to include in your app’s code. We may share with Nokia the developer IDs that are using the map services, because Nokia supplies some of these services. [Learn more](#).

By clicking **Get token**, I agree to be bound by the [Terms of Use](#). Further, if accepting on behalf of a company, then I represent that I am authorized to act on my company’s behalf.

[Get token](#)

Close

Şekil 50 – Bing Maps kullanım ekranı

Özellikle gerekli değilse, son iki seçenek olan Market Seçimi ve Harita Servisleri bölümlerini doldurmak zorunda değilsiniz.

Submit app

Hello World

You’ve spent hours developing and designing your app, and now it’s time for the rest of the world to experience your masterpiece. In just two steps we’ll gather the information we need to successfully launch your app in the Windows Phone Store. [Learn more](#) about the steps for successfully submitting your app.

Required



App info

Give your app a Dev Center alias, price it, and enter other relevant info



Upload and describe your XAP(s)

For each XAP in your app, this is where you’ll enter descriptions and upload screenshots that will showcase your app in the Store.

Optional



Market selection and custom pricing

For apps, you have the option to define different pricing and availability for different countries/regions.



Map services

Get the token required to use map services in your app.

Review and submit

Şekil 51 – Uygun market seçimi ve harita servis ekleme ekranı

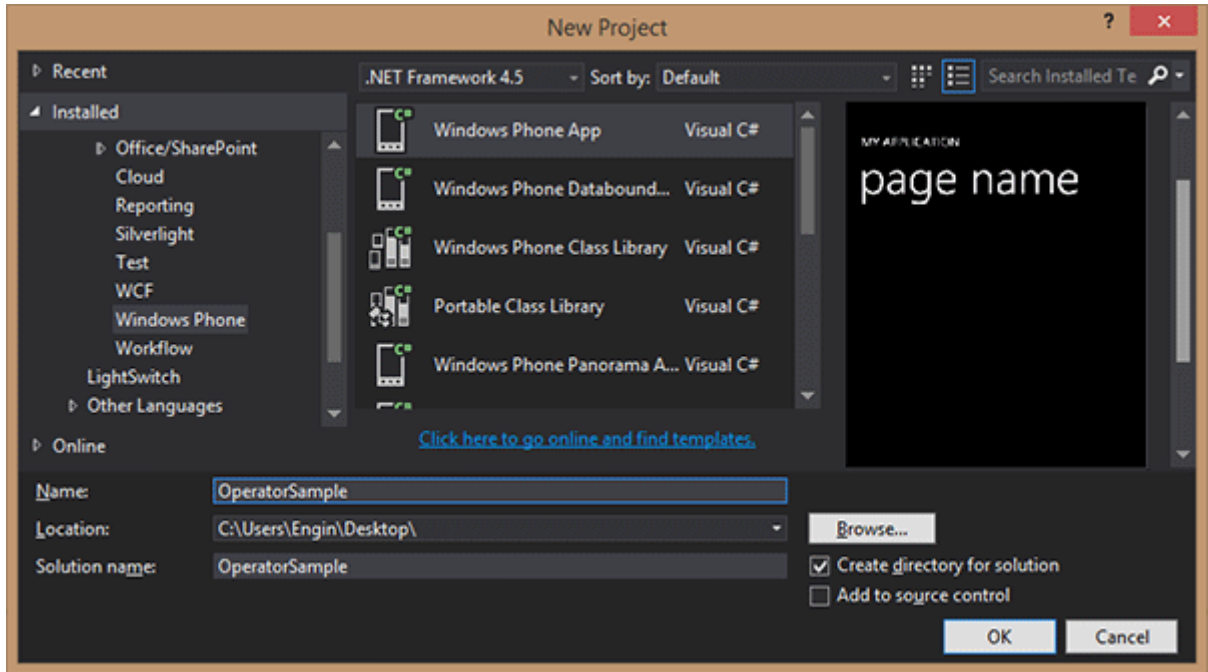
Uygulamanızı DevCenter’te yayınlamanın son adımı Submit düğmesini tıklamaktır. Tıklandıktan sonra işleminiz sıraya alınacak ve birkaç testten geçtikten sonra, eğer testlerde başarısız olarak geri dönmezse Dev Center DevCenter’te yerini alacaktır.

7. BÖLÜM 6: ÖRNEK UYGULAMALAR

Windows Phone 8 ile telefonun operatörünü bulma

Windows Phone 8 için geliştirdiğiniz uygulamanın çalıştığı telefonun bağlı olduğu operatör'ü bulmak isteyebilirsiniz. Özellikle operatör'e özel seçenekler çıkartmak veya kısıtlamalar getirmek için bunu yapmaya ihtiyacınız olabilir. Windows Phone 8 SDK içerisinde yer alan DeviceNetworkInformation sınıfında Mobil Operator bilgisini kolaylıkla almanızı sağlayan static tanımlı CellularMobileOperator özelliği mevcuttur.

Hemen yeni bir proje oluşturarak kullanımını inceleyelim;



Şekil 52 – Proje oluşturma ekranı

MainPage.xaml dosyasını açarak sayfanın tasarımını değiştirelim. StackPanel içerisine bir Button nesnesi ekleyelim, Content özelliğine *Tıkla*, Background özelliğine *StaticResource PhoneAccentBrush*, Height özelliğine *120* değerlerini verelim;

```
<phone:PhoneApplicationPage
    x:Class="OperatorSample.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
```

```

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
shell:SystemTray.IsVisible="True">

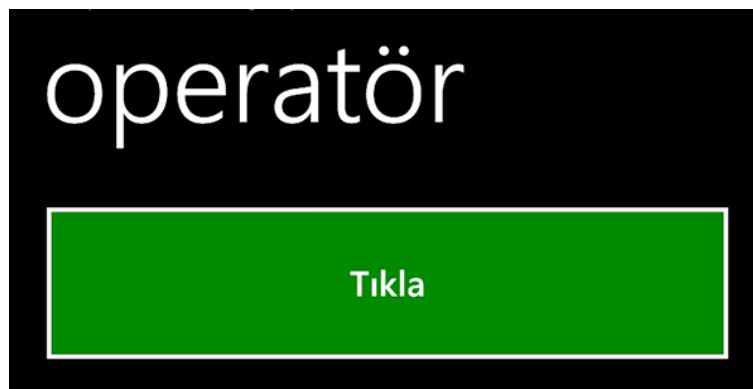
<StackPanel>
    <StackPanel x:Name="TitlePanel" Margin="12,17,0,28">
        <TextBlock Text="http://www.enginpolat.com" Style="{StaticResource
PhoneTextNormalStyle}" Margin="12,0"/>
        <TextBlock Text="operatör" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
    </StackPanel>

    <StackPanel Margin="12,0,12,0">
        <Button Content="Tıkla" Background="{StaticResource PhoneAccentBrush}"
Height="120" Click="Button_OnClick" />
    </StackPanel>
</StackPanel>
</phone:PhoneApplicationPage>

```

Background özelliğine StaticResource listesinden PhoneAccentBrush değerini vererek, buton'un arkarengini kullanıcının tercih ettiği tema rengine ayarlamış olduk;

Button'a tıklandığında *Button_OnClick()* method'u tetiklenecek;

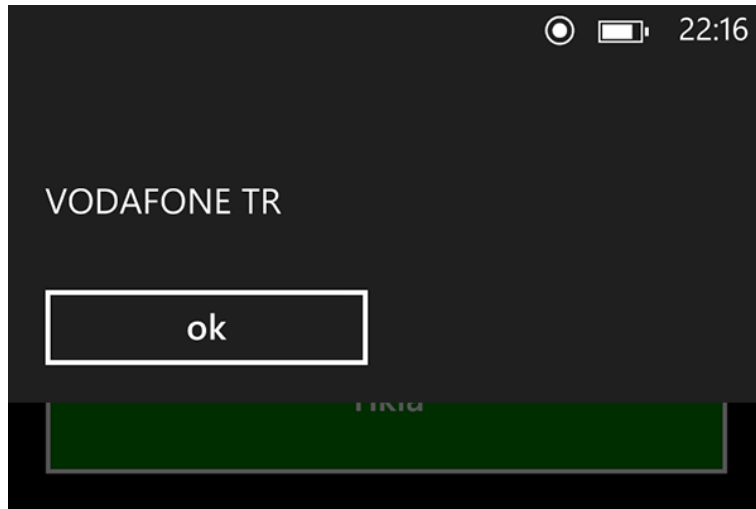


Şekil 53 – Uygulamanın ekran görüntüsü


```
private void Button_OnClick(object sender, RoutedEventArgs e)
{
    var operatorAdi = DeviceNetworkInformation.CellularMobileOperator;

    MessageBox.Show(operatorAdi);
}
```

Butona tıkladığımızda ekrana, uygulamanın kurulu olduğu telefonun *mobil operatörü* gelmeli;



Şekil 54 – Uygulamada butana tıklandıktan sonraki görünüm

Windows Phone 8 için Basit bir Hesap makinesi örneği

Bu projede basit matematiksel işlemleri kullanarak girilen 2 değeri toplama, çıkarma, bölme, çarpma, mod alma işlemlerine tabii tutarak işlem sonucunu ekrana getirecektir. Öncelikle yeni bir proje açalım.

Başlık Panelimizdeki kodu (StackPaneldeki) aşağıdaki kodla değiştirelim.

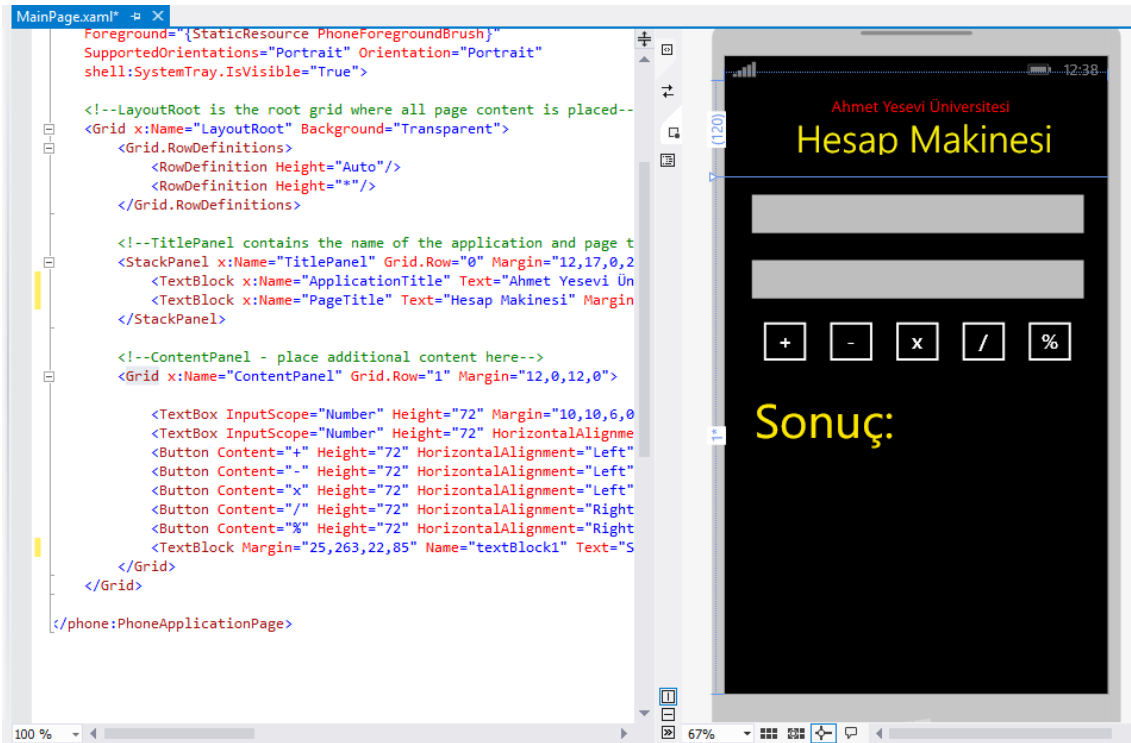
```
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">

    <TextBlock x:Name="ApplicationTitle" Text="Ahmet Yesevi Üniversitesi" Style="{StaticResource
PhoneTextNormalStyle}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="Red"/>

    <TextBlock x:Name="PageTitle" Text="Hesap Makinesi" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}" FontSize="48" Height="55" HorizontalAlignment="Center"
Foreground="#FFDF20F"/>

</StackPanel>
```

Şimdi Projemize 2 adet TextBox, 5 adet Buton ve ekrana sonucu yazabilmek için TextBlock ekleyelim. Nesneleri eklendikten sonra kodumuz aşağıdaki gibi olmalıdır. Projemizin son görüntüsü Şekil-55 de olduğu gibi olmalıdır.



Şekil 55 - Uygulamanın ekran görüntüsü

```

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <TextBox InputScope="Number" Height="72" Margin="10,10,6,0" Name="textBox1" Text=""
    VerticalAlignment="Top" />
    <TextBox InputScope="Number" Height="72" HorizontalAlignment="Left" Margin="10,92,0,0"
    Name="textBox2" Text="" VerticalAlignment="Top" Width="440" />
    <Button Content="+" Height="72" HorizontalAlignment="Left" Margin="25,170,0,0"
    Name="Toplama" VerticalAlignment="Top" Width="77" Click="Toplama_Click" />
    <Button Content="-" Height="72" HorizontalAlignment="Left" Margin="108,170,0,0"
    Name="Çıkartma" VerticalAlignment="Top" Width="77" Click="Çıkartma_Click" />
    <Button Content="x" Height="72" HorizontalAlignment="Left" Margin="191,170,0,0"
    Name="Çarpma" VerticalAlignment="Top" Width="77" Click="Çarpma_Click" />
    <Button Content="/" Height="72" HorizontalAlignment="Right" Margin="0,170,105,0"
    Name="Bölme" VerticalAlignment="Top" Width="77" Click="Bölme_Click" />
    <Button Content="%" Height="72" HorizontalAlignment="Right" Margin="0,170,22,0"
    Name="Kalan_Bulma" VerticalAlignment="Top" Width="77" Click="Kalan_Bulma_Click" />
    <TextBlock Margin="25,263,22,85" Name="textBlock1" Text="Sonuç:" FontSize="60"
    Foreground="#FFFE800" />
</Grid>

```

Şimdi gerekli matematiksel işlemleri yapmak için c# kodunu aşağıdaki gibi yapalım. Öncelikle kullanacağımız değişkenleri tanımlayalım. Bu değişken her yerde geçerli olması için global tanımlanmalıdır.

```

// Bu alanda değişken tanımlamalarımızı yapıyoruz. 3 değişken tanımlıyoruz.
double BirinciSayı = 0, İkinciSayı = 0, ToplamSayı = 0;

```

Toplama işlemini gerçekleştirebilmek için gereken c# kodu:

```

private void Toplama_Click(object sender, RoutedEventArgs e)
{
    try
    {
        // İlk tanımladığımız değişkene veriyi Convert fonksiyonunu kullanarak
        // textBox1 objesinin .Text özelliğinden string olan veriyi alıyoruz.
        BirinciSayı = Convert.ToDouble(textBox1.Text);
        // İlk sayıda yaptığımız işlemin aynısını diğer değişkenimiz için yapıyoruz.
        İkinciSayı = Convert.ToDouble(textBox2.Text);

        // ToplamSayı değişkenimize BirinciSayı ve İkinciSayı değişkenimizin toplamını atıyoruz.
    }
}

```

ToplamSayı = BirinciSayı + İkinciSayı;

```
// textBlock1.Text objemize bulduğumuz sonucu başına "Sonuç: " yazdırarak atıyoruz.  
// Aynı zamanda Toplamsayı'yı .ToString() yaparak string'e çeviriyoruz.  
textBlock1.Text = "Sonuç: " + ToplamSayı.ToString();  
}  
catch  
{  
}  
}
```

Çıkarma işlemini gerçekleştirebilmek için gereken c# kodu:

```
private void Çıkartma_Click(object sender, RoutedEventArgs e)  
{  
    try  
    {  
        // İlk tanımladığımız değişkene veriyi Convert fonksiyonunu kullanarak  
        // textBox1 objesinin .Text özelliğinden string olan veriyi alıyoruz.  
        BirinciSayı = Convert.ToDouble(textBox1.Text);  
        // İlk sayıda yaptığımız işlemin aynısını diğer değişkenimiz için yapıyoruz.  
        İkinciSayı = Convert.ToDouble(textBox2.Text);  
  
        // ToplamSayı değişkenimize BirinciSayı ve İkinciSayı değişkenimizin çıkartmasını atıyoruz.  
        ToplamSayı = BirinciSayı - İkinciSayı;  
  
        // textBlock1.Text objemize bulduğumuz sonucu başına "Sonuç: " yazdırarak atıyoruz.  
        // Aynı zamanda Toplamsayı'yı .ToString() yaparak string'e çeviriyoruz.  
        textBlock1.Text = "Sonuç: " + ToplamSayı.ToString();  
    }  
    catch  
    {  
  
    }  
}
```

Çarpma işlemini gerçekleştirebilmek için gereken c# kodu:

```
private void Çarpma_Click(object sender, RoutedEventArgs e)  
{  
    try  
    {  
        // İlk tanımladığımız değişkene veriyi Convert fonksiyonunu kullanarak
```

```

// textBox1 objesinin .Text özelliğinden string olan veriyi alıyoruz.
BirinciSayı = Convert.ToDouble(textBox1.Text);
// İlk sayıda yaptığımız işlemin aynısını diğer değişkenimiz için yapıyoruz.
İkinciSayı = Convert.ToDouble(textBox2.Text);

// ToplamSayı değişkenimize BirinciSayı ve İkinciSayı değişkenimizin çarpımını atıyoruz.
ToplamSayı = BirinciSayı * İkinciSayı;

// textBlock1.Text objemize bulduğumuz sonucu başına "Sonuç: " yazdırarak atıyoruz.
// Aynı zamanda Toplamsayı'yı .ToString() yaparak string'e çeviriyoruz.
textBlock1.Text = "Sonuç: " + ToplamSayı.ToString();
}
catch
{

}
}

```

Bölme işlemini gerçekleştirebilmek için gereken c# kodu:

```

private void Bölme_Click(object sender, RoutedEventArgs e)
{
    try
    {
        // İlk tanımladığımız değişkene veriyi Convert fonksiyonunu kullanarak
        // textBox1 objesinin .Text özelliğinden string olan veriyi alıyoruz.
        BirinciSayı = Convert.ToDouble(textBox1.Text);
        // İlk sayıda yaptığımız işlemin aynısını diğer değişkenimiz için yapıyoruz.
        İkinciSayı = Convert.ToDouble(textBox2.Text);

        // ToplamSayı değişkenimize BirinciSayı ve İkinciSayı değişkenimizin bölümünü atıyoruz.
        ToplamSayı = BirinciSayı / İkinciSayı;

        // textBlock1.Text objemize bulduğumuz sonucu başına "Sonuç: " yazdırarak atıyoruz.
        // Aynı zamanda Toplamsayı'yı .ToString() yaparak string'e çeviriyoruz.
        textBlock1.Text = "Sonuç: " + ToplamSayı.ToString();
    }
    catch { }
}

```

Mod alma işlemini gerçekleştirebilmek için gereken c# kodu:

```

private void Mod_Bulma_Click(object sender, RoutedEventArgs e)
{

```

```

try
{
    // İlk tanımladığımız değişkene veriyi Convert fonksiyonunu kullanarak
    // textBox1 objesinin .Text özelliğinden string olan veriyi alıyoruz.
    BirinciSayı = Convert.ToDouble(textBox1.Text);
    // İlk sayıda yaptığımız işlemin aynısını diğer değişkenimiz için yapıyoruz.
    İkinciSayı = Convert.ToDouble(textBox2.Text);

    // ToplamSayı değişkenimize BirinciSayı ve İkinciSayı değişkenimizin kalan verisini atıyoruz.
    // Bu şöyle çalışır. birinci sayıyı ikinci sayıya bölüp o bölümden kaç kaldığını bulur.
    // Yani 5 / 2 yaparsak kalan 1 olur.
    ToplamSayı = BirinciSayı % İkinciSayı;

    // textBlock1.Text objemize bulduğumuz sonucu başına "Sonuç: " yazdırarak atıyoruz.
    // Aynı zamanda Toplamsayı'yı .ToString() yaparak string'e çeviriyoruz.
    textBlock1.Text = "Sonuç: " + ToplamSayı.ToString();
}
catch
{
}
}

```



Şekil 56 - Uygulamanın çalışır görüntüsü

8. SONUÇ:

Proje; giderek artan akıllı telefon pazarına kendi uygulamasını yazabilen insanlar hazırlamak amacı hedeflenmiştir. İşletim sistemi olarak Windows Phone seçilmiştir. Bu işletim sisteminin seçilmesinin en büyük sebebi Türkiye’de bu konuda eğitilmiş kişilerin ve bu alandaki programların azlığı olarak belirlenmiştir. Böylece insanlar kendi telefonlarında kolaylıkla kendi programlarını kullanabilecek ve piyasanın ihtiyacını karşılayacak seviye geleceklerdir. Bu doğrultuda ihtiyaç duyulan programlar ve uygulama yazmak için gereken donanımlar tanıtılmıştır. Programcılar tarafından yoğun bir şekilde kullanılan C# dili ve tasarım işlemleri için XAML tasarım dili seçilmiştir. Windows Phone seçilmesinin önemli nedenlerinden biriside uygulama yazma ortamının insanlara daha alışık oldukları Windows işletim sistemi ortamını kullandırması ve Windows Phone telefona sahip olmadan emülatörler sayesinde uygulama şansları verilmesidir. Uygulama yazma aşamasında kullanılan tüm programların ve ortamların öğrencilere ve akademisyenlere ücretsiz olması da Windows Phone programlama için önemlidir.

Programlama ve tasarım yapılırken gerekli tüm aşamalar örnek uygulamalarla gösterilmiş. Uygulamanın tüm aşamaları adımlar halinde ve ekran görüntüsü şeklinde açıklanmıştır. Kullanılacak ortamın, çözünürlüğün belirlenmesi ve uygulamada kullanılacak donanımlarının seçilmesi noktasında gerekli aşamalar anlatılmıştır. Yapılan uygulamaların DevCenter aracılığıyla mağazaya gönderilmesi aşamaları da ayrıntılı bir şekilde anlatılmıştır.

Ülkemizin; Windows Phone mağazasında ismini duyurmak ve Türkçe uygulamaların sayısında artış olması hedeflenmesi sağlanmıştır.

Katkılarından dolayı Yrd. Doç. Dr. Hacer KARACAN hocamıza teşekkür ederim.

Kaynakça

- [1]. Microsoft. <http://channel9.msdn.com/> adresinden alınmıştır.(10.11.2013).
- [2]. Basu, S. Introduction to Windows 8. *Real World Windows 8 Development* (s. 3-10). içinde Apress. (2013).
- [3]. Delprato, J. M. (10.10.2013). *Windows Phone 8. Pearson Education*. France.2013.
- [4]. Lecrenski, N., Watson, K., & Ensor, R. F. (2011). *Beginning Windows Phone 7 Application Development: Building Windows Phone Applications Using Silverlight and XNA*. Wrox Press Ltd.
- [5]. Microsoft. <http://www.microsoftvirtualacademy.com/> adresinden alınmıştır. (17.11.2013).
- [6]. Microsoft. *MSDN*. <http://www.msdn.com> adresinden alınmıştır. (2013).
- [7]. *Microsoft Açık Akademi*. <http://www.acikakademi.com> adresinden alınmıştır. (2013).
- [8]. Microsoft. *Microsoft Türkiye MSDN Servisi*. 2013: <http://msdn.microsoft.com/tr-TR/> adresinden alınmıştır.(21.10.2013).
- [9]. Yöndem, D. (2013). *Microsoft MVP Uzmanı*. <http://daron.yondem.com/tr/> adresinden alınmıştır.