

Awesome Note

OneNote, EverNote gibi kisisel bir Windows Phone not tutma uygulaması

Table of Contents

Symbol Reference	1
AwesomeNote Namespace	1
AwesomeNote.Resources Namespace	2
Classes	2
AppResources Class	2
Classes	29
AboutPage Class	30
AboutPage.AboutPage Constructor	31
AboutPage Fields	31
App Class	32
App.App Constructor	32
App Properties	33
AppSettings Class	33
AppSettings Properties	34
AutoScrollBehavior Class	35
AutoScrollBehavior Methods	35
BackgroundColorSettings Class	36
BackgroundColorSettings.BackgroundColorSettings Constructor	36
BackgroundColorSettings Fields	37
ChangeNoteFolder Class	37
ChangeNoteFolder.ChangeNoteFolder Constructor	38
ChangeNoteFolder Fields	38
ChangeNoteFolder Methods	38
EnterPasswordPage Class	40
EnterPasswordPage.EnterPasswordPage Constructor	40
EnterPasswordPage Fields	41
EnterPasswordPage Methods	41
FolderOrderSettings Class	43
FolderOrderSettings.FolderOrderSettings Constructor	44
FolderOrderSettings Fields	44
FolderOrderSettings Methods	44
FolderOrderStyleSettings Class	45
FolderOrderStyleSettings.FolderOrderStyleSettings Constructor	45
FolderOrderStyleSettings Fields	46
FolderOrderStyleSettings Methods	46
FolderSettings Class	46

FolderSettings.FolderSettings Constructor	47
FolderSettings Fields	47
FolderSettings Methods	47
FontFamilySettings Class	49
FontFamilySettings.FontFamilySettings Constructor	50
FontFamilySettings Fields	50
FontFamilySettings Methods	51
FontSizeSettings Class	52
FontSizeSettings.FontSizeSettings Constructor	52
FontSizeSettings Fields	53
FontSizeSettings Methods	53
GeneralSettings Class	54
GeneralSettings.GeneralSettings Constructor	55
GeneralSettings Fields	57
GeneralSettings Methods	57
Language Class	59
Language Properties	59
LanguageSettings Class	60
LanguageSettings.LanguageSettings Constructor	60
LanguageSettings Fields	61
LanguageSettings Methods	61
LocalizedStrings Class	62
LocalizedStrings Properties	62
MainPage Class	62
MainPage.MainPage Constructor	63
MainPage Fields	63
MainPage Methods	64
Note Class	64
Note Properties	65
NoteDetail Class	66
NoteDetail.NoteDetail Constructor	67
NoteDetail Fields	68
NoteDetail Methods	69
NoteFolder Class	70
NoteFolder Properties	71
NoteFolderDataContext Class	73
NoteFolderDataContext.NoteFolderDataContext Constructor	74
NoteFolderDataContext Fields	74
NoteList Class	75
NoteList.NoteList Constructor	75

NoteList Fields	76
NoteList Methods	77
NoteOrderSettings Class	79
NoteOrderSettings.NoteOrderSettings Constructor	79
NoteOrderSettings Fields	79
NoteOrderSettings Methods	80
NoteOrderStyleSettings Class	81
NoteOrderStyleSettings.NoteOrderStyleSettings Constructor	81
NoteOrderStyleSettings Fields	82
NoteOrderStyleSettings Methods	82
Page1 Class	83
Page1.Page1 Constructor	84
Page1 Fields	84
Page1 Methods	84
PopupAddFolder Class	86
PopupAddFolder.PopupAddFolder Constructor	86
PopupAddNote Class	86
PopupAddNote.PopupAddNote Constructor	87
PopupChangeFolder Class	87
PopupChangeFolder.PopupChangeFolder Constructor	87
PopupChangeFolderName Class	88
PopupChangeFolderName.PopupChangeFolderName Constructor	88
PopupChangeNoteName Class	88
PopupChangeNoteName.PopupChangeNoteName Constructor	89
PopupEnterPassword Class	89
PopupEnterPassword.PopupEnterPassword Constructor	89
PopupSetPassword Class	90
PopupSetPassword.PopupSetPassword Constructor	90
SearchPage Class	90
SearchPage.SearchPage Constructor	91
SearchPage Fields	91
SearchPage Methods	91
SetPasswordPage Class	92
SetPasswordPage.SetPasswordPage Constructor	92
SetPasswordPage Fields	93
SetPasswordPage Methods	93
Files	94
AboutPage.xaml.cs	95
App.xaml.cs	98
AppResources.Designer.cs	107

AppSettings.cs	133
AssemblyInfo.cs	133
AutoScrollBehavior.cs	134
Awesome Note.csproj	135
AwesomeNote.sln	135
AwesomeNoteDataContext.cs	135
BackgroundColorSettings.xaml.cs	136
ChangeNoteFolder.xaml.cs	139
EnterPasswordPage.xaml.cs	141
FolderOrderSettings.xaml.cs	145
FolderOrderStyleSettings.xaml.cs	148
FolderSettings.xaml.cs	151
FontFamilySettings.xaml.cs	157
FontSizeSettings.xaml.cs	160
GeneralSettings.xaml.cs	162
Language.cs	174
LanguageSettings.xaml.cs	174
LocalizedStrings.cs	178
MainPage.xaml.cs	178
Note.cs	184
NoteDetail.xaml.cs	185
NoteFolder.cs	197
NoteFolder.xaml.cs	198
NoteInfo.xaml.cs	205
NoteOrderSettings.xaml.cs	207
NoteOrderStyleSettings.xaml.cs	210
PopupAddFolder.xaml.cs	213
PopupAddNote.xaml.cs	214
PopupChangeFolder.xaml.cs	215
PopupChangeFolderName.xaml.cs	216
PopupChangeNoteName.xaml.cs	217
PopupEnterPassword.xaml.cs	218
PopupSetPassword.xaml.cs	219
SearchPage.xaml.cs	219
SetPasswordPage.xaml.cs	224

Index

a

1 Symbol Reference












1.1 AwesomeNote Namespace










This is namespace AwesomeNote.

Namespaces

Name	Description
Resources (↗ see page 2)	This is namespace AwesomeNote.Resources.

Classes

	Name	Description
	AboutPage (↗ see page 30)	This is class AwesomeNote>AboutPage.
	App (↗ see page 32)	This is class AwesomeNote.App.
	AppSettings (↗ see page 33)	This is class AwesomeNote.AppSettings.
	AutoScrollBehavior (↗ see page 35)	This is class AwesomeNote.AutoScrollBehavior.
	BackgroundColorSettings (↗ see page 36)	This is class AwesomeNote.BackgroundColorSettings.
	ChangeNoteFolder (↗ see page 37)	This is class AwesomeNote.ChangeNoteFolder.
	EnterPasswordPage (↗ see page 40)	This is class AwesomeNote.EnterPasswordPage.
	FolderOrderSettings (↗ see page 43)	This is class AwesomeNote.FolderOrderSettings.
	FolderOrderStyleSettings (↗ see page 45)	This is class AwesomeNote.FolderOrderStyleSettings.
	FolderSettings (↗ see page 46)	This is class AwesomeNote.FolderSettings.
	FontFamilySettings (↗ see page 49)	This is class AwesomeNote.FontFamilySettings.
	FontSizeSettings (↗ see page 52)	This is class AwesomeNote.FontSizeSettings.
	GeneralSettings (↗ see page 54)	This is class AwesomeNote.GeneralSettings.
	Language (↗ see page 59)	This is class AwesomeNote.Language.
	LanguageSettings (↗ see page 60)	This is class AwesomeNote.LanguageSettings.
	LocalizedStrings (↗ see page 62)	Provides access to string resources.
	MainPage (↗ see page 62)	This is class AwesomeNote.MainPage.
	Note (↗ see page 64)	This is class AwesomeNote.Note.
	NoteDetail (↗ see page 66)	This is class AwesomeNote.NoteDetail.
	NoteFolder (↗ see page 70)	This is class AwesomeNote.NoteFolder.
	NoteFolderDataContext (↗ see page 73)	This is class AwesomeNote.NoteFolderDataContext.
	NoteList (↗ see page 75)	This is class AwesomeNote.NoteList.
	NoteOrderSettings (↗ see page 79)	This is class AwesomeNote.NoteOrderSettings.
	NoteOrderStyleSettings (↗ see page 81)	This is class AwesomeNote.NoteOrderStyleSettings.
	Page1 (↗ see page 83)	This is class AwesomeNote.Page1.

	PopupAddFolder (see page 86)	This is class AwesomeNote.PopupAddFolder.
	PopupAddNote (see page 86)	This is class AwesomeNote.PopupAddNote.
	PopupChangeFolder (see page 87)	This is class AwesomeNote.PopupChangeFolder.
	PopupChangeFolderName (see page 88)	This is class AwesomeNote.PopupChangeFolderName.
	PopupChangeNoteName (see page 88)	This is class AwesomeNote.PopupChangeNoteName.
	PopupEnterPassword (see page 89)	This is class AwesomeNote.PopupEnterPassword.
	PopupSetPassword (see page 90)	This is class AwesomeNote.PopupSetPassword.
	SearchPage (see page 90)	This is class AwesomeNote.SearchPage.
	SetPasswordPage (see page 92)	This is class AwesomeNote.SetPasswordPage.

1.1.1 AwesomeNote.Resources Namespace

This is namespace AwesomeNote.Resources.

Classes

	Name	Description
	AppResources (see page 2)	A strongly-typed resource class, for looking up localized strings, etc.

1.1.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	AppResources (see page 2)	A strongly-typed resource class, for looking up localized strings, etc.

1.1.1.1.1 AppResources Class

A strongly-typed resource class, for looking up localized strings, etc.

Class Hierarchy

```
AwesomeNote.Resources.AppResources
```

C#

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
"4.0.0.0")]
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
public class AppResources;
```

File


AppResources.Designer.cs ([see page 107](#))

Description






















This class was auto-generated by the StronglyTypedResourceBuilder class via a tool like ResGen or Visual Studio. To add


































or remove a member, edit your .ResX file then rerun ResGen with the /str option, or rebuild your VS project.

Methods












	Name	Description
	AppResources (see page 9)	This is AppResources, a member of class AppResources.
































AppResources Properties



























	Name	Description
	About (see page 9)	Looks up a localized string similar to Hakkında.
	AboutTheApp (see page 9)	Looks up a localized string similar to Uygulama Hakkında.
	AboutTheAppText (see page 9)	Looks up a localized string similar to Umarım programı kullanırken keyif almissinizdir. Çünkü ben yazarken baya keyif aldım. Program hakkında taleplerinizi, bu sayfadaki İletişim butonuna tıklayarak veya direkt olarak coderserdar@outlook.com adresine mail atarak belirtirseniz ve uygulamayı oylarsanız çok sevinirim. Sizin sayenizde bu program daha çok gelişecek. İyi günler.
	AboutTheAwesomeNote (see page 9)	Looks up a localized string similar to Awesome Note (see page 17) Uygulaması Hakkında.
	Active (see page 9)	Looks up a localized string similar to Aktif.
	AddFolder (see page 9)	Looks up a localized string similar to Klasör Ekle.
	AddItem (see page 10)	Looks up a localized string similar to Nesne Ekle.
	AddNote (see page 10)	Looks up a localized string similar to Not Ekle.
	AppFolder (see page 10)	Looks up a localized string similar to Not Klasörleri.
	ApplicationTitle (see page 10)	Looks up a localized string similar to MY APPLICATION.
	AppTitle (see page 10)	Looks up a localized string similar to Awesome Note (see page 17).
	Arabic (see page 10)	Looks up a localized string similar to Arabic.
	Ascending (see page 10)	Looks up a localized string similar to Artan Sirada.
	AtLeastFourDigits (see page 10)	Looks up a localized string similar to En Az Dört Karakter.
	Background (see page 10)	Looks up a localized string similar to Arka Plan.
	BackgroundColor (see page 11)	Looks up a localized string similar to Arka Plan Rengi.
	BackgroundFeature (see page 11)	Looks up a localized string similar to Arka Plan Özellikleri.
	BackgroundFeatureNote (see page 11)	Looks up a localized string similar to Normalde kullandığım uygulamalarda görsellikten çok işlevselliğe önem veririm. Yazarken de aynı şekilde yazdım bu programı ancak herkesin benim düşündüğüm gibi düşünmediğini de bildiğim için görselliği sade olan bu programda arka plan rengini ayarlanabilir bir şekilde yaptım. Programın Genel Ayarları kısmında Arka Plan bölümünde istenirse sadece renk seçimi yaparak, istenirse de mobil cihaz içerisindeki her hangi bir fotoğrafı arka plan olarak belirleyebilmek mümkündür. Arka plan rengi uygulamanın tamamı iç [rest of string was truncated]";.
	BackgroundImage (see page 11)	Looks up a localized string similar to Arka Plan Resmi.
	BelongFolderName (see page 11)	Looks up a localized string similar to Ait Oldugu Klasörün Adı.
	Black (see page 11)	Looks up a localized string similar to Siyah.
	Blue (see page 11)	Looks up a localized string similar to Mavi.
	Brown (see page 11)	Looks up a localized string similar to Kahverengi.
	Cancel (see page 12)	Looks up a localized string similar to İptal.
	ChangeFolder (see page 12)	Looks up a localized string similar to Klasör Degistir.
	CharacterNumber (see page 12)	Looks up a localized string similar to Karakter Sayısı.
	Chinese (see page 12)	Looks up a localized string similar to Chinese.
	ContactWithUs (see page 12)	Looks up a localized string similar to İletişime Geç.
	CreationDate (see page 12)	Looks up a localized string similar to Olusturulma Tarihi.

	Culture (🔗 see page 12)	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
	DeleteFolder (🔗 see page 12)	Looks up a localized string similar to Klasör Sil.
	DeleteFolderQuestion (🔗 see page 12)	Looks up a localized string similar to Not Klasörünüz Tüm İçeriği Ile Silinecek. Onaylıyor Musunuz?.
	DeleteNote (🔗 see page 13)	Looks up a localized string similar to Notu Sil.
	DeleteNoteQuestion (🔗 see page 13)	Looks up a localized string similar to Notu Silmek İstediginizden Emin Misiniz?.
	Descending (🔗 see page 13)	Looks up a localized string similar to Azalan Sirada.
	English (🔗 see page 13)	Looks up a localized string similar to İngilizce.
	EnterFolderName (🔗 see page 13)	Looks up a localized string similar to Lütfen Not Klasörünün Adını Yazınız.
	EnterNoteName (🔗 see page 13)	Looks up a localized string similar to Lütfen Notun Adını Giriniz.
	EnterPassword (🔗 see page 13)	Looks up a localized string similar to Şifre Girme.
	ExitApp (🔗 see page 13)	Looks up a localized string similar to Uygulamadan Çık.
	ExitAppQuestion (🔗 see page 13)	Looks up a localized string similar to Uygulamadan Çıkmak İstediginizden Emin Misiniz?.
	FolderExists (🔗 see page 14)	Looks up a localized string similar to Bu İsimde Bir Klasör Zaten Mevcut.
	FolderName (🔗 see page 14)	Looks up a localized string similar to Klasör Adı.
	FolderOrder (🔗 see page 14)	Looks up a localized string similar to Klasör Sıralama.
	FolderSettings (🔗 see page 14)	Looks up a localized string similar to Klasör Ayarları.
	Font (🔗 see page 14)	Looks up a localized string similar to Yazı Tipi.
	FontFamily (🔗 see page 14)	Looks up a localized string similar to Yazı Tipi Ailesi.
	FontFeature (🔗 see page 14)	Looks up a localized string similar to Font (🔗 see page 14) Özellikleri.
	FontFeatureNote (🔗 see page 14)	Looks up a localized string similar to Program üzerinde notlar ile ilgili olarak, her klasör için bir font ailesi ve font boyutu ayarlaması yapabilirsiniz. Bunu seçtiğiniz not klasörü içerisinde Klasör Ayarları butonuna bastıktan sonra çıkan ekranda Font (🔗 see page 14) bölümüne gelerek yapabilirsiniz. Ekranda o an aktif olarak seçtiğiniz font ailesi ve font boyutu görünecektir. Eğer siz seçmezseniz program font ayarını otomatik olarak Verdana ve 26 punto olarak ayarlayacaktır..
	FontSize (🔗 see page 15)	Looks up a localized string similar to Yazı Tipi Boyutu.
	FourDigits (🔗 see page 15)	Looks up a localized string similar to Dört Karakter.
	French (🔗 see page 15)	Looks up a localized string similar to French.
	GeneralSettings (🔗 see page 15)	Looks up a localized string similar to Genel Ayarlar.
	German (🔗 see page 15)	Looks up a localized string similar to Almanca.
	Gray (🔗 see page 15)	Looks up a localized string similar to Gri.
	Green (🔗 see page 15)	Looks up a localized string similar to Yeşil.
	Irish (🔗 see page 15)	Looks up a localized string similar to Irish.
	IsBold (🔗 see page 15)	Looks up a localized string similar to Yazı Tipi Kalın Olsun.
	IsItalic (🔗 see page 16)	Looks up a localized string similar to Yazı Tipi İtalik Olsun.
	Italian (🔗 see page 16)	Looks up a localized string similar to Italian.
	Language (🔗 see page 16)	Looks up a localized string similar to Dil.
	LanguageFeature (🔗 see page 16)	Looks up a localized string similar to Dil Özellikleri.
	LanguageFeatureNote (🔗 see page 16)	Looks up a localized string similar to Program İngilizce, Türkçe, Almanca, İspanyolca, İtalyanca, Fransızca, Portekizce, Çince, Rusça, Arapça, Farsça gibi dillere tamamen destek vermektedir. Yani uygulamada dil gerekebilecek bütün menülerde ve sayfalarda kendi dilinizi görebileceksiniz..

	LanguageWarning (see page 16)	Looks up a localized string similar to Ayarlarin etkili olmasi için uygulamayi bastan baslatmaniz gerekebilir....
	Login (see page 16)	Looks up a localized string similar to Baglan.
	ModificationDate (see page 16)	Looks up a localized string similar to Degistirilme Tarihi.
	Next (see page 16)	Looks up a localized string similar to Sonraki.
	NextNote (see page 17)	Looks up a localized string similar to Sonraki Not.
	No (see page 17)	Looks up a localized string similar to Hayir.
	Note (see page 17)	Looks up a localized string similar to Not.
	NoteExists (see page 17)	Looks up a localized string similar to Bu Isimde Bir Not Zaten Mevcut.
	NoteFeature (see page 17)	Looks up a localized string similar to Not Özellikleri.
	NoteFeatureNote (see page 17)	Looks up a localized string similar to Not içerisinde notun adi, içinde bulunduğu klasörün adi, notun olusturulma tarihi ve notun kendisi bulunmaktadır. Not içerisinde klasördeki bir önceki nota ve bir sonraki nota gidebilmek mümkündür. Notun adina tiklandigi zaman notun adini degistirme mümkündür. Not metninin tamamini seçebilmek, not hakkında detayli bilgileri görüntülemek mümkündür. Seçilen bir notu SMS veya E-Posta yoluyla gönderebilmek de mümkündür. Notu silmek veya ait oldugu klasörü degistirebilmek de mümkündür..
	NoteFolderFeature (see page 17)	Looks up a localized string similar to Not Klasörü Özellikleri.
	NoteFolderFeatureNote (see page 17)	Looks up a localized string similar to Not klasörü içerisinde, içindeki notlarin listesi bulunmaktadır. Buradan klasör içerisindeki her hangi bir nota gidebilmek mümkündür. Not klasörünü silebilmek mümkündür, ancak bu islemi gerçeklestirdiginiz zaman not klasörünü içerisindeki bütün notlarla birlikte silecektir. Bu yüzden klasör silmeden önce Senkronizasyon yaparak notlarinizi yedeklemeniz önemle tavsiye edilir. Klasöre yeni bir not ekleyebilir veya ayarlarini degistirebilirsiniz. Ayrica klasör adina tikladiginiz zaman klasörün adini degistirebi [rest of string was truncated]";.
	NotelInfo (see page 18)	Looks up a localized string similar to Not Hakkinda.
	NoteListOf (see page 18)	Looks up a localized string similar to Not Listesi.
	NoteName (see page 18)	Looks up a localized string similar to Not Adi.
	NoteOrder (see page 18)	Looks up a localized string similar to Not Siralama.
	NoteSaved (see page 18)	Looks up a localized string similar to Not Kaydedildi.
	OK (see page 18)	Looks up a localized string similar to Tamam.
	OneDrive (see page 18)	Looks up a localized string similar to One Drive.
	OneDriveSyncCompleted (see page 18)	Looks up a localized string similar to OneDrive (see page 18) Senkronizasyonu Tamamlandi.
	Orange (see page 19)	Looks up a localized string similar to Turuncu.
	OrderFeature (see page 19)	Looks up a localized string similar to Siralama Özellikleri.
	OrderFeatureNote (see page 19)	Looks up a localized string similar to Program üzerinde hem not klasörlerini hem de klasör içerisindeki notlari siralamak mümkündür. Notlari ve not klasörlerini; adina, olusturulma tarihine göre ve en son degistirilme tarihine göre artan ve azalan sekilde siralamak mümkündür. Not klasörlerinin siralama ayarlarini degistirmek için Genel Ayarlar kismina Diger Ayarlar bölümüne, klasör içerisindeki notlarin siralama ayarlarini degistirmek için ise Klasör Ayarlari kismindaki Diger Ayarlar bölümüne girmek gerekmektedir. Ekranda hali hazırda seçilen a [rest of string was truncated]";.
	OrderStyle (see page 19)	Looks up a localized string similar to Siralama Türü.
	OtherSettings (see page 19)	Looks up a localized string similar to Diger Ayarlar.
	Passive (see page 19)	Looks up a localized string similar to Pasif.
	Password (see page 19)	Looks up a localized string similar to Sifre.
	PasswordFalse (see page 19)	Looks up a localized string similar to Sifre Yanlis.

	PasswordFeature (see page 20)	Looks up a localized string similar to Sifre Özelliği.
	PasswordFeatureNote (see page 20)	Looks up a localized string similar to Seçilen her hangi bir not klasöründe Klasör Ayarları bölümünde Sifre kısmında o not klasörüne sifre atamasi yapabilmek mümkündür. Aynı bölümden mevcut sifre de kaldırılabilir. Girilen sifre de en az 4 karakterli olmalıdır. Sadece rakam olmalı gibi bir kısıtlama söz konusu değildir. Sifre atamasi yapıldıktan sonra klasöre her giriste mevcut sifreyi soracaktır. Sifre girilirken kutucuklara sifre girildikten sonra Enter tusuna basılması yeterlidir. Bu özellik Windows Phone marketinde ilk kez bu uygulamada bulu [rest of string was truncated]";.
	PasswordProtection (see page 20)	Looks up a localized string similar to Sifre Korumasi.
	PasswordState (see page 20)	Looks up a localized string similar to Sifre Durumu.
	PasswordTrue (see page 20)	Looks up a localized string similar to Sifre Dogru.
	Persian (see page 20)	Looks up a localized string similar to Persian.
	PleaseEnterPassword (see page 20)	Looks up a localized string similar to Sifreyi Giriniz.
	PleaseReEnterPassword (see page 20)	Looks up a localized string similar to Sifreyi Tekrar Giriniz.
	Portuguese (see page 21)	Looks up a localized string similar to Portuguese.
	Previous (see page 21)	Looks up a localized string similar to Önceki.
	PreviousNote (see page 21)	Looks up a localized string similar to Önceki Not.
	ProgramFeatures (see page 21)	Looks up a localized string similar to Programın Özellikleri.
	ProgramFeaturesNote (see page 21)	Looks up a localized string similar to Bu program ile not klasörü oluşturma, klasöre sifre atama, uygulamanın arka plan rengini değiştirme veya arka plana telefon üzerindeki fotoğraflardan birini yerleştirebilme, not klasörünün arka plan fotoğrafını degistirebilme, içerik sorgulaması yapabileme, OneDrive (see page 18) senkronizasyonu, herhangi bir notun ait olduğu klasörü kolaylıkla değiştirme, tutulan notları SMS veya E-Posta yoluyla gönderme, detaylı font ve sıralama ayarları, çoklu dil desteği, not hakkında istatistiki bilgi gibi pek çok özellik bulunmaktadır [rest of string was truncated]";.
	Purple (see page 21)	Looks up a localized string similar to Mor.
	Rate (see page 21)	Looks up a localized string similar to Degerlendir.
	Red (see page 21)	Looks up a localized string similar to Kirmizi.
	RemoveBackgroundImage (see page 22)	Looks up a localized string similar to Arka Plan Resmini Kaldır.
	ResetSettings (see page 22)	Looks up a localized string similar to Ayarları Sifirle.
	ResourceFlowDirection (see page 22)	Looks up a localized string similar to LeftToRight.
	ResourceLanguage (see page 22)	Looks up a localized string similar to tr-TR.
	ResourceManager (see page 22)	Returns the cached ResourceManager instance used by this class.
	Russian (see page 22)	Looks up a localized string similar to Russian.
	Sanskrit (see page 22)	Looks up a localized string similar to Sanskritçe.
	Save (see page 22)	Looks up a localized string similar to Kaydet.
	Search (see page 22)	Looks up a localized string similar to Ara.
	SearchCompleted (see page 23)	Looks up a localized string similar to Arama Tamamlandı.
	SearchFeature (see page 23)	Looks up a localized string similar to Arama Özellikleri.

	SearchFeatureNote (see page 23)	Looks up a localized string similar to Programda notun adi ve notun içeriği üzerinden sorgulama yapmaya imkan sağlayan bir özellik bulunmaktadır. Bu özellik sayesinde notlarınızın içinde kaybolmaktan kurtulmuş olursunuz. Sorgulama için arama yapacağınız metni yazdıktan sonra Enter tusuna basmanız yeterli olacaktır. Sorgu sonuçları yine aynı sayfada yer alacaktır. Burada önemli bir nokta ise, eğer sorgu sonuçları içerisinde daha önceden sifreledığınız bir not klasörü içerisindeki nota erişmeye kalkarsanız sorgulama ekranından sifre giris ekranına [rest of string was truncated]";.
	SearchResults (see page 23)	Looks up a localized string similar to Arama Sonuçları.
	SearchTrimFault (see page 23)	Looks up a localized string similar to Arama Alanını Doldurmak Zorundasınız..
	Select (see page 23)	Looks up a localized string similar to Seçiniz.
	SelectAll (see page 23)	Looks up a localized string similar to Tümünü Seç.
	SelectBackgroundColor (see page 23)	Looks up a localized string similar to Arka Plan Rengi Seçiniz.
	Selected (see page 24)	Looks up a localized string similar to Seçilen.
	SelectFontFamily (see page 24)	Looks up a localized string similar to Yazı Tipi Ailesi Seçiniz.
	SelectFontSize (see page 24)	Looks up a localized string similar to Yazı Tipi Boyu Seçiniz.
	SelectLanguage (see page 24)	Looks up a localized string similar to Lütfen Dili Seçiniz.
	SelectNoteFolder (see page 24)	Looks up a localized string similar to Lütfen Not Klasörünü Seçiniz.
	SelectOrderStyle (see page 24)	Looks up a localized string similar to Sıralama Türü Seçiniz.
	SelectOrderType (see page 24)	Looks up a localized string similar to Sıralama Ölçütünü Seçiniz.
	SelectPasswordState (see page 24)	Looks up a localized string similar to Sifre Durumunu Seçiniz.
	SendWithApp (see page 24)	Looks up a localized string similar to Awesome Note (see page 17) Uygulaması ile Gönderildi.
	SendWithMail (see page 25)	Looks up a localized string similar to E-Posta ile Gönder.
	SendWithSMS (see page 25)	Looks up a localized string similar to SMS ile Gönder.
	SetPassword (see page 25)	Looks up a localized string similar to Sifre Ayarla.
	Settings (see page 25)	Looks up a localized string similar to Ayarlar.
	Share (see page 25)	Looks up a localized string similar to Paylaş.
	SignIn (see page 25)	Looks up a localized string similar to Bağlan.
	SignOut (see page 25)	Looks up a localized string similar to Çıkış Yap.
	Spanish (see page 25)	Looks up a localized string similar to İspanyolca.
	SuccessfulAddNote (see page 25)	Looks up a localized string similar to Not Başarılı Bir Sekilde Olusturuldu.
	SuccessfulAddNoteFolder (see page 26)	Looks up a localized string similar to Not Klasörü Başarılı Bir Sekilde Olusturuldu.
	SuccessfulBackgroundColorChanged (see page 26)	Looks up a localized string similar to Arka Plan Rengi Başarıyla Degistirildi.
	SuccessfulBackgroundImageChanged (see page 26)	Looks up a localized string similar to Arka Plan Resmi Başarılı Bir Sekilde Degistirildi.
	SuccessfulBackgroundImageRemoved (see page 26)	Looks up a localized string similar to Arka Plan Resmi Başarılı Bir Sekilde Kaldirildi.
	SuccessfulChangeFolderName (see page 26)	Looks up a localized string similar to Not Klasör İsmi Başarılı Bir Sekilde Degistirildi.
	SuccessfulChangeNoteFolder (see page 26)	Looks up a localized string similar to Notunuzun Ait Oldugu Klasör Degistirildi.
	SuccessfulChangeNoteName (see page 26)	Looks up a localized string similar to Notun İsmi Başarılı Bir Sekilde Degistirildi.

	SuccessfulDeleteFolder (see page 26)	Looks up a localized string similar to Not Klasörü Basariyla Silindi.
	SuccessfulDeleteNote (see page 26)	Looks up a localized string similar to Not Basarili Bir Sekilde Silindi.
	SuccessfulFontFamilyChanged (see page 27)	Looks up a localized string similar to Yazi Tipi Ailesi Basarili Bir Sekilde Degistirildi.
	SuccessfulFontSizeChanged (see page 27)	Looks up a localized string similar to Yazi Tipi Boyutu Basarili Bir Sekilde Degistirildi.
	SuccessfulOrderStyleChanged (see page 27)	Looks up a localized string similar to Siralama Türü Basarili Bir Sekilde Degistirildi.
	SuccessfulOrderTypeChanged (see page 27)	Looks up a localized string similar to Siralama Kriteri Basarili Bir Sekilde Degistirildi.
	SuccessfulRemovePassword (see page 27)	Looks up a localized string similar to Sifre Basarili Bir Sekilde Kaldirildi.
	SuccessfulResetSettings (see page 27)	Looks up a localized string similar to Arka Plan Ayarlari Basariyla Sifirlandi.
	SuccessfulSendWithMail (see page 27)	Looks up a localized string similar to E-Posta Yoluyla Not Basarili Bir Sekilde Gönderildi.
	SuccessfulSendWithSMS (see page 27)	Looks up a localized string similar to SMS Yoluyla Not Basarili Bir Sekilde Gönderildi.
	SuccessfulSetPassword (see page 27)	Looks up a localized string similar to Sifre Basarili Bir Sekilde Olusturuldu.
	Sync (see page 28)	Looks up a localized string similar to Eszamanlama.
	SyncFeature (see page 28)	Looks up a localized string similar to Senkronizasyon Özellikleri.
	SyncFeatureNote (see page 28)	Looks up a localized string similar to Program vasitasiyla tüm notlarinizi OneDrive (see page 18) hesabiniz ile senkronize edebilirsiniz. Uygulama içerisinde Genel Ayarlar kismina Senkronizasyon bölümüne girdikten sonra Sign In yazan butona tıklayip, OneDrive (see page 18) hesabınıza giris yapmanız gerekmektedir. Giris esnasinda sordugu erisim iznini onaylamadiginiz zaman senkronizasyon islemini gerçeklestirmeyecektir. Sonrasinda ise Senkronizasyon için gerekli butona bastiginiz zaman program OneDrive (see page 18) hesabiniz içerisinde bir klasör olusturacak (örneğin: Awesome Note (see page 17) (21. [rest of string was truncated]";.
	SyncOnOneFile (see page 28)	Looks up a localized string similar to Hepsini Tek Dosyada Birlestir.
	Syncronizing (see page 28)	Looks up a localized string similar to Eszamanlaniliyor....
	SystemFault (see page 28)	Looks up a localized string similar to Sistemde Bir Hata Olustu. Uygulama Sonlanacak....
	Thai (see page 28)	Looks up a localized string similar to Tay Dili.
	Tips (see page 28)	Looks up a localized string similar to Programin İpuçlari.
	Turkish (see page 29)	Looks up a localized string similar to Türkçe.
	TwoFieldsNotSame (see page 29)	Looks up a localized string similar to Girilen Iki Sifre Alani Ayni Degil.
	Welcome (see page 29)	Looks up a localized string similar to Hosgeldiniz.
	WelcomeNote (see page 29)	Looks up a localized string similar to Merhaba. Not almayı seven bir kullanıcı olarak Bilgisayar ve Mobil cihazlarda pek çok not tutma programi kullandim. Ve faydalı olacağını düşündüğüm özellikleri birleştirip bu uygulamayı yazdim. Umarım uygulamadan memnun kalirsiniz..
	WordNumber (see page 29)	Looks up a localized string similar to Kelime Sayisi.
	Yellow (see page 29)	Looks up a localized string similar to Sari.
	Yes (see page 29)	Looks up a localized string similar to Evet.

1.1.1.1.1.1 AppResources.AppResources Constructor

C#

```
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
internal AppResources();
```

Description

This is AppResources, a member of class AppResources.

Body Source

```
1:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
2: internal AppResources() {
3: }
```

1.1.1.1.1.2 AppResources Properties

1.1.1.1.1.2.1 AppResources.About Property

Looks up a localized string similar to Hakkında.

C#

```
public static string About;
```

1.1.1.1.1.2.2 AppResources.AboutTheApp Property

Looks up a localized string similar to Uygulama Hakkında.

C#

```
public static string AboutTheApp;
```

1.1.1.1.1.2.3 AppResources.AboutTheAppText Property

Looks up a localized string similar to Umarım programi kullanırken keyif almissinizdir. Çünkü ben yazarken baya keyif aldım. Program hakkında taleplerinizi, bu sayfadaki İletişim butonuna tıklayarak veya direkt olarak coderserdar@outlook.com adresine mail atarak belirtirseniz ve uygulamayı oylarsanız çok sevinirim. Sizin sayenizde bu program daha çok gelişecek. İyi günler.

C#

```
public static string AboutTheAppText;
```

1.1.1.1.1.2.4 AppResources.AboutTheAwesomeNote Property

Looks up a localized string similar to Awesome Note (🔗 see page 17) Uygulaması Hakkında.

C#

```
public static string AboutTheAwesomeNote;
```

1.1.1.1.1.2.5 AppResources.Active Property

Looks up a localized string similar to Aktif.

C#

```
public static string Active;
```

1.1.1.1.1.2.6 AppResources.AddFolder Property

Looks up a localized string similar to Klasör Ekle.

C#

```
public static string AddFolder;
```

1.1.1.1.1.2.7 AppResources.AddItem Property

Looks up a localized string similar to Nesne Ekle.

C#

```
public static string AddItem;
```

1.1.1.1.1.2.8 AppResources.AddNote Property

Looks up a localized string similar to Not Ekle.

C#

```
public static string AddNote;
```

1.1.1.1.1.2.9 AppResources.AppFolder Property

Looks up a localized string similar to Not Klasörleri.

C#

```
public static string AppFolder;
```

1.1.1.1.1.2.10 AppResources.ApplicationTitle Property

Looks up a localized string similar to MY APPLICATION.

C#

```
public static string ApplicationTitle;
```

1.1.1.1.1.2.11 AppResources.AppTitle Property

Looks up a localized string similar to Awesome Note (see page 17).

C#

```
public static string AppTitle;
```

1.1.1.1.1.2.12 AppResources.Arabic Property

Looks up a localized string similar to Arabic.

C#

```
public static string Arabic;
```

1.1.1.1.1.2.13 AppResources.Ascending Property

Looks up a localized string similar to Artan Sirada.

C#

```
public static string Ascending;
```

1.1.1.1.1.2.14 AppResources.AtLeastFourDigits Property

Looks up a localized string similar to En Az Dört Karakter.

C#

```
public static string AtLeastFourDigits;
```

1.1.1.1.1.2.15 AppResources.Background Property

Looks up a localized string similar to Arka Plan.

C#

```
public static string Background;
```

1.1.1.1.1.2.16 AppResources.BackgroundColor Property

Looks up a localized string similar to Arka Plan Rengi.

C#

```
public static string BackgroundColor;
```

1.1.1.1.1.2.17 AppResources.BackgroundFeature Property

Looks up a localized string similar to Arka Plan Özellikleri.

C#

```
public static string BackgroundFeature;
```

1.1.1.1.1.2.18 AppResources.BackgroundFeatureNote Property

Looks up a localized string similar to Normalde kullandigim uygulamalarda görsellikten çok işlevsellige önem veririm. Yazarken de aynı şekilde yazdım bu programı ancak herkesin benim düşündüğüm gibi düşünmediğini de bildiğim için görselliği sade olan bu programda arka plan rengini ayarlanabilir bir şekilde yaptım. Programın Genel Ayarları kısmında Arka Plan bölümünde istenirse sadece renk seçimi yaparak, istenirse de mobil cihaz içerisindeki her hangi bir fotoğrafı arka plan olarak belirleyebilmek mümkündür. Arka plan rengi uygulamanın tamamı iç [rest of string was truncated];.

C#

```
public static string BackgroundFeatureNote;
```

1.1.1.1.1.2.19 AppResources.BackgroundImage Property

Looks up a localized string similar to Arka Plan Resmi.

C#

```
public static string BackgroundImage;
```

1.1.1.1.1.2.20 AppResources.BelongFolderName Property

Looks up a localized string similar to Ait Oldugu Klasörün Adı.

C#

```
public static string BelongFolderName;
```

1.1.1.1.1.2.21 AppResources.Black Property

Looks up a localized string similar to Siyah.

C#

```
public static string Black;
```

1.1.1.1.1.2.22 AppResources.Blue Property

Looks up a localized string similar to Mavi.

C#

```
public static string Blue;
```

1.1.1.1.1.2.23 AppResources.Brown Property

Looks up a localized string similar to Kahverengi.

C#

```
public static string Brown;
```


1.1.1.1.1.2.24 AppResources.Cancel Property

Looks up a localized string similar to İptal.

C#

```
public static string Cancel;
```

1.1.1.1.1.2.25 AppResources.ChangeFolder Property

Looks up a localized string similar to Klasör Degistir.

C#

```
public static string ChangeFolder;
```

1.1.1.1.1.2.26 AppResources.CharacterNumber Property

Looks up a localized string similar to Karakter Sayisi.

C#

```
public static string CharacterNumber;
```

1.1.1.1.1.2.27 AppResources.Chinese Property

Looks up a localized string similar to Chinese.

C#

```
public static string Chinese;
```

1.1.1.1.1.2.28 AppResources.ContactWithUs Property

Looks up a localized string similar to İletisime Geç.

C#

```
public static string ContactWithUs;
```

1.1.1.1.1.2.29 AppResources.CreationDate Property

Looks up a localized string similar to Olusturulma Tarihi.

C#

```
public static string CreationDate;
```

1.1.1.1.1.2.30 AppResources.Culture Property

Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

C#

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]  
public static global::System.Globalization.CultureInfo Culture;
```

1.1.1.1.1.2.31 AppResources.DeleteFolder Property

Looks up a localized string similar to Klasör Sil.

C#

```
public static string DeleteFolder;
```

1.1.1.1.1.2.32 AppResources.DeleteFolderQuestion Property

Looks up a localized string similar to Not Klasörünüz Tüm İçerigi İle Silinecek. Onaylıyor Musunuz?.

C#

```
public static string DeleteFolderQuestion;
```

1.1.1.1.1.2.33 AppResources.DeleteNote Property

Looks up a localized string similar to Notu Sil.

C#

```
public static string DeleteNote;
```

1.1.1.1.1.2.34 AppResources.DeleteNoteQuestion Property

Looks up a localized string similar to Notu Silmek Istediginizden Emin Misiniz?.

C#

```
public static string DeleteNoteQuestion;
```

1.1.1.1.1.2.35 AppResources.Descending Property

Looks up a localized string similar to Azalan Sirada.

C#

```
public static string Descending;
```

1.1.1.1.1.2.36 AppResources.English Property

Looks up a localized string similar to Ingilizce.

C#

```
public static string English;
```

1.1.1.1.1.2.37 AppResources.EnterFolderName Property

Looks up a localized string similar to Lütfen Not Klasörünün Adini Yaziniz.

C#

```
public static string EnterFolderName;
```

1.1.1.1.1.2.38 AppResources.EnterNoteName Property

Looks up a localized string similar to Lütfen Notun Adini Giriniz.

C#

```
public static string EnterNoteName;
```

1.1.1.1.1.2.39 AppResources.EnterPassword Property

Looks up a localized string similar to Sifre Girme.

C#

```
public static string EnterPassword;
```

1.1.1.1.1.2.40 AppResources.ExitApp Property

Looks up a localized string similar to Uygulamadan Çık.

C#

```
public static string ExitApp;
```

1.1.1.1.1.2.41 AppResources.ExitAppQuestion Property

Looks up a localized string similar to Uygulamadan Çıkmak Istediginizden Emin Misiniz?.

C#

```
public static string ExitAppQuestion;
```

1.1.1.1.1.2.42 AppResources.FolderExists Property

Looks up a localized string similar to Bu Isimde Bir Klasör Zaten Mevcut.

C#

```
public static string FolderExists;
```

1.1.1.1.1.2.43 AppResources.FolderName Property

Looks up a localized string similar to Klasör Adi.

C#

```
public static string FolderName;
```

1.1.1.1.1.2.44 AppResources.FolderOrder Property

Looks up a localized string similar to Klasör Sıralama.

C#

```
public static string FolderOrder;
```

1.1.1.1.1.2.45 AppResources.FolderSettings Property

Looks up a localized string similar to Klasör Ayarlari.

C#

```
public static string FolderSettings;
```

1.1.1.1.1.2.46 AppResources.Font Property

Looks up a localized string similar to Yazı Tipi.

C#

```
public static string Font;
```

1.1.1.1.1.2.47 AppResources.FontFamily Property

Looks up a localized string similar to Yazı Tipi Ailesi.

C#

```
public static string FontFamily;
```

1.1.1.1.1.2.48 AppResources.FontFeature Property

Looks up a localized string similar to Font (☐ see page 14) Özellikleri.

C#

```
public static string FontFeature;
```

1.1.1.1.1.2.49 AppResources.FontFeatureNote Property

Looks up a localized string similar to Program üzerinde notlar ile ilgili olarak, her klasör için bir font ailesi ve font boyutu ayarlaması yapabilirsiniz. Bunu seçtiğiniz not klasörü içerisinde Klasör Ayarlari butonuna bastıktan sonra çıkan ekranda Font (☐ see page 14) bölümüne gelerek yapabilirsiniz. Ekranda o an aktif olarak seçtiğiniz font ailesi ve font boyutu görünecektir. Eğer siz seçmezseniz program font ayarını otomatik olarak Verdana ve 26 punto olarak ayarlayacaktır..

C#

```
public static string FontFeatureNote;
```

1.1.1.1.1.2.50 AppResources.FontSize Property

Looks up a localized string similar to Yazı Tipi Boyutu.

C#

```
public static string FontSize;
```

1.1.1.1.1.2.51 AppResources.FourDigits Property

Looks up a localized string similar to Dört Karakter.

C#

```
public static string FourDigits;
```

1.1.1.1.1.2.52 AppResources.French Property

Looks up a localized string similar to French.

C#

```
public static string French;
```

1.1.1.1.1.2.53 AppResources.GeneralSettings Property

Looks up a localized string similar to Genel Ayarlar.

C#

```
public static string GeneralSettings;
```

1.1.1.1.1.2.54 AppResources.German Property

Looks up a localized string similar to Almanca.

C#

```
public static string German;
```

1.1.1.1.1.2.55 AppResources.Gray Property

Looks up a localized string similar to Gri.

C#

```
public static string Gray;
```

1.1.1.1.1.2.56 AppResources.Green Property

Looks up a localized string similar to Yesil.

C#

```
public static string Green;
```

1.1.1.1.1.2.57 AppResources.Irish Property

Looks up a localized string similar to Irish.

C#

```
public static string Irish;
```

1.1.1.1.1.2.58 AppResources.IsBold Property

Looks up a localized string similar to Yazı Tipi Kalın Olsun.

C#

```
public static string IsBold;
```

1.1.1.1.1.2.59 AppResources.IsItalic Property

Looks up a localized string similar to Yazı Tipi İtalik Olsun.

C#

```
public static string IsItalic;
```

1.1.1.1.1.2.60 AppResources.Italian Property

Looks up a localized string similar to Italian.

C#

```
public static string Italian;
```

1.1.1.1.1.2.61 AppResources.Language Property

Looks up a localized string similar to Dil.

C#

```
public static string Language;
```

1.1.1.1.1.2.62 AppResources.LanguageFeature Property

Looks up a localized string similar to Dil Özellikleri.

C#

```
public static string LanguageFeature;
```

1.1.1.1.1.2.63 AppResources.LanguageFeatureNote Property

Looks up a localized string similar to Program İngilizce, Türkçe, Almanca, İspanyolca, İtalyanca, Fransızca, Portekizce, Çince, Rusça, Arapça, Farsça gibi dillere tamamen destek vermektedir. Yani uygulamada dil gerekebilecek bütün menülerde ve sayfalarda kendi dilinizi görebileceksiniz..

C#

```
public static string LanguageFeatureNote;
```

1.1.1.1.1.2.64 AppResources.LanguageWarning Property

Looks up a localized string similar to Ayarların etkili olması için uygulamayı baştan başlatmanız gerekebilir....

C#

```
public static string LanguageWarning;
```

1.1.1.1.1.2.65 AppResources.Login Property

Looks up a localized string similar to Bağlan.

C#

```
public static string Login;
```

1.1.1.1.1.2.66 AppResources.ModificationDate Property

Looks up a localized string similar to Degistirilme Tarihi.

C#

```
public static string ModificationDate;
```

1.1.1.1.1.2.67 AppResources.Next Property

Looks up a localized string similar to Sonraki.

C#

```
public static string Next;
```

1.1.1.1.1.2.68 AppResources.NextNote Property

Looks up a localized string similar to Sonraki Not.

C#

```
public static string NextNote;
```

1.1.1.1.1.2.69 AppResources.No Property

Looks up a localized string similar to Hayir.

C#

```
public static string No;
```

1.1.1.1.1.2.70 AppResources.Note Property

Looks up a localized string similar to Not.

C#

```
public static string Note;
```

1.1.1.1.1.2.71 AppResources.NoteExists Property

Looks up a localized string similar to Bu Isimde Bir Not Zaten Mevcut.

C#

```
public static string NoteExists;
```

1.1.1.1.1.2.72 AppResources.NoteFeature Property

Looks up a localized string similar to Not Özellikleri.

C#

```
public static string NoteFeature;
```

1.1.1.1.1.2.73 AppResources.NoteFeatureNote Property

Looks up a localized string similar to Not içerisinde notun adı, içinde bulunduğu klasörün adı, notun oluşturulma tarihi ve notun kendisi bulunmaktadır. Not içerisinde klasördeki bir önceki nota ve bir sonraki nota gidebilmek mümkündür. Notun adına tıkladığı zaman notun adını değiştirme mümkündür. Not metninin tamamını seçebilmek, not hakkında detaylı bilgileri görüntülemek mümkündür. Seçilen bir notu SMS veya E-Posta yoluyla gönderebilmek de mümkündür. Notu silmek veya ait olduğu klasörü değiştirebilmek de mümkündür..

C#

```
public static string NoteFeatureNote;
```

1.1.1.1.1.2.74 AppResources.NoteFolderFeature Property

Looks up a localized string similar to Not Klasörü Özellikleri.

C#

```
public static string NoteFolderFeature;
```

1.1.1.1.1.2.75 AppResources.NoteFolderFeatureNote Property

Looks up a localized string similar to Not klasörü içerisinde, içindeki notların listesi bulunmaktadır. Buradan klasör içerisindeki her hangi bir nota gidebilmek mümkündür. Not klasörünü silebilmek mümkündür, ancak bu işlemi gerçekleştirdiğiniz zaman not klasörünü içerisindeki bütün notlarla birlikte silecektir. Bu yüzden klasör silmeden önce

Senkronizasyon yaparak notlarınızı yedeklemeniz önemle tavsiye edilir. Klasöre yeni bir not ekleyebilir veya ayarlarını değiştirebilirsiniz. Ayrıca klasör adına tıkladığınız zaman klasörün adını değiştirebilirsiniz [rest of string was truncated];

C#

```
public static string NoteFolderFeatureNote;
```

1.1.1.1.1.2.76 AppResources.NoteInfo Property

Looks up a localized string similar to Not Hakkında.

C#

```
public static string NoteInfo;
```

1.1.1.1.1.2.77 AppResources.NoteListOf Property

Looks up a localized string similar to Not Listesi.

C#

```
public static string NoteListOf;
```

1.1.1.1.1.2.78 AppResources.NoteName Property

Looks up a localized string similar to Not Adı.

C#

```
public static string NoteName;
```

1.1.1.1.1.2.79 AppResources.NoteOrder Property

Looks up a localized string similar to Not Sıralama.

C#

```
public static string NoteOrder;
```

1.1.1.1.1.2.80 AppResources.NoteSaved Property

Looks up a localized string similar to Not Kaydedildi.

C#

```
public static string NoteSaved;
```

1.1.1.1.1.2.81 AppResources.OK Property

Looks up a localized string similar to Tamam.

C#

```
public static string OK;
```

1.1.1.1.1.2.82 AppResources.OneDrive Property

Looks up a localized string similar to One Drive.

C#

```
public static string OneDrive;
```

1.1.1.1.1.2.83 AppResources.OneDriveSyncCompleted Property

Looks up a localized string similar to OneDrive (📁 see page 18) Senkronizasyonu Tamamlandı.

C#

```
public static string OneDriveSyncCompleted;
```

1.1.1.1.1.2.84 AppResources.Orange Property

Looks up a localized string similar to Turuncu.

C#

```
public static string Orange;
```

1.1.1.1.1.2.85 AppResources.OrderFeature Property

Looks up a localized string similar to Sıralama Özellikleri.

C#

```
public static string OrderFeature;
```

1.1.1.1.1.2.86 AppResources.OrderFeatureNote Property

Looks up a localized string similar to Program üzerinde hem not klasörlerini hem de klasör içerisindeki notları sıralamak mümkündür. Notları ve not klasörlerini; adına, oluşturma tarihine göre ve en son değiştirilme tarihine göre artan ve azalan şekilde sıralamak mümkündür. Not klasörlerinin sıralama ayarlarını değiştirmek için Genel Ayarlar kısmında Diğer Ayarlar bölümüne, klasör içerisindeki notların sıralama ayarlarını değiştirmek için ise Klasör Ayarları kısmındaki Diğer Ayarlar bölümüne girmek gerekmektedir. Ekranda hali hazırda seçilen a [rest of string was truncated]";.

C#

```
public static string OrderFeatureNote;
```

1.1.1.1.1.2.87 AppResources.OrderStyle Property

Looks up a localized string similar to Sıralama Türü.

C#

```
public static string OrderStyle;
```

1.1.1.1.1.2.88 AppResources.OtherSettings Property

Looks up a localized string similar to Diğer Ayarlar.

C#

```
public static string OtherSettings;
```

1.1.1.1.1.2.89 AppResources.Passive Property

Looks up a localized string similar to Pasif.

C#

```
public static string Passive;
```

1.1.1.1.1.2.90 AppResources.Password Property

Looks up a localized string similar to Sifre.

C#

```
public static string Password;
```

1.1.1.1.1.2.91 AppResources.PasswordFalse Property

Looks up a localized string similar to Sifre Yanlıs.

C#

```
public static string PasswordFalse;
```


1.1.1.1.1.2.92 AppResources.PasswordFeature Property

Looks up a localized string similar to Sifre Özelliği.

C#

```
public static string PasswordFeature;
```

1.1.1.1.1.2.93 AppResources.PasswordFeatureNote Property

Looks up a localized string similar to Seçilen her hangi bir not klasöründe Klasör Ayarları bölümünde Sifre kısmında o not klasörüne sifre atamasi yapabilmek mümkündür. Aynı bölümden mevcut sifre de kaldırilabilir. Girilen sifre de en az 4 karakterli olmalıdır. Sadece rakam olmalı gibi bir kisiltlama söz konusu degildir. Sifre atamasi yapildiktan sonra klasöre her giriste mevcut sifreyi soracaktır. Sifre girilirken kutucuklara sifre girildikten sonra Enter tusuna basilmasi yeterlidir. Bu özellik Windows Phone marketinde ilk kez bu uygulamada bulu [rest of string was truncated]";.

C#

```
public static string PasswordFeatureNote;
```

1.1.1.1.1.2.94 AppResources.PasswordProtection Property

Looks up a localized string similar to Sifre Korumasi.

C#

```
public static string PasswordProtection;
```

1.1.1.1.1.2.95 AppResources.PasswordState Property

Looks up a localized string similar to Sifre Durumu.

C#

```
public static string PasswordState;
```

1.1.1.1.1.2.96 AppResources.PasswordTrue Property

Looks up a localized string similar to Sifre Dogru.

C#

```
public static string PasswordTrue;
```

1.1.1.1.1.2.97 AppResources.Persian Property

Looks up a localized string similar to Persian.

C#

```
public static string Persian;
```

1.1.1.1.1.2.98 AppResources.PleaseEnterPassword Property

Looks up a localized string similar to Sifreyi Giriniz.

C#

```
public static string PleaseEnterPassword;
```

1.1.1.1.1.2.99 AppResources.PleaseReEnterPassword Property

Looks up a localized string similar to Sifreyi Tekrar Giriniz.

C#

```
public static string PleaseReEnterPassword;
```

1.1.1.1.2.100 AppResources.Portuguese Property

Looks up a localized string similar to Portuguese.

C#

```
public static string Portuguese;
```

1.1.1.1.2.101 AppResources.Previous Property

Looks up a localized string similar to Önceki.

C#

```
public static string Previous;
```

1.1.1.1.2.102 AppResources.PreviousNote Property

Looks up a localized string similar to Önceki Not.

C#

```
public static string PreviousNote;
```

1.1.1.1.2.103 AppResources.ProgramFeatures Property

Looks up a localized string similar to Programın Özellikleri.

C#

```
public static string ProgramFeatures;
```

1.1.1.1.2.104 AppResources.ProgramFeaturesNote Property

Looks up a localized string similar to Bu program ile not klasörü oluşturma, klasöre şifre atama, uygulamanın arka plan rengini değiştirme veya arka plana telefon üzerindeki fotoğraflardan birini yerleştirebilme, not klasörünün arka plan fotoğrafını değiştirebilme, içerik sorgulaması yapabilme, OneDrive (📁 see page 18) senkronizasyonu, herhangi bir notun ait olduğu klasörü kolaylıkla değiştirme, tutulan notları SMS veya E-Posta yoluyla gönderme, detaylı font ve sıralama ayarları, çoklu dil desteği, not hakkında istatistik bilgisi gibi pek çok özellik bulunmaktadır [rest of string was truncated];.

C#

```
public static string ProgramFeaturesNote;
```

1.1.1.1.2.105 AppResources.Purple Property

Looks up a localized string similar to Mor.

C#

```
public static string Purple;
```

1.1.1.1.2.106 AppResources.Rate Property

Looks up a localized string similar to Değerlendir.

C#

```
public static string Rate;
```

1.1.1.1.2.107 AppResources.Red Property

Looks up a localized string similar to Kırmızı.

C#

```
public static string Red;
```

1.1.1.1.1.2.108 AppResources.RemoveBackgroundImage Property

Looks up a localized string similar to Arka Plan Resmini Kaldır.

C#

```
public static string RemoveBackgroundImage;
```

1.1.1.1.1.2.109 AppResources.ResetSettings Property

Looks up a localized string similar to Ayarlari Sifirla.

C#

```
public static string ResetSettings;
```

1.1.1.1.1.2.110 AppResources.ResourceFlowDirection Property

Looks up a localized string similar to LeftToRight.

C#

```
public static string ResourceFlowDirection;
```

1.1.1.1.1.2.111 AppResources.ResourceLanguage Property

Looks up a localized string similar to tr-TR.

C#

```
public static string ResourceLanguage;
```

1.1.1.1.1.2.112 AppResources.ResourceManager Property

Returns the cached ResourceManager instance used by this class.

C#

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]  
public static global::System.Resources.ResourceManager ResourceManager;
```

1.1.1.1.1.2.113 AppResources.Russian Property

Looks up a localized string similar to Russian.

C#

```
public static string Russian;
```

1.1.1.1.1.2.114 AppResources.Sanskrit Property

Looks up a localized string similar to Sanskritçe.

C#

```
public static string Sanskrit;
```

1.1.1.1.1.2.115 AppResources.Save Property

Looks up a localized string similar to Kaydet.

C#

```
public static string Save;
```

1.1.1.1.1.2.116 AppResources.Search Property

Looks up a localized string similar to Ara.

C#

```
public static string Search;
```

1.1.1.1.1.2.117 AppResources.SearchCompleted Property

Looks up a localized string similar to Arama Tamamlandi.

C#

```
public static string SearchCompleted;
```

1.1.1.1.1.2.118 AppResources.SearchFeature Property

Looks up a localized string similar to Arama Özellikleri.

C#

```
public static string SearchFeature;
```

1.1.1.1.1.2.119 AppResources.SearchFeatureNote Property

Looks up a localized string similar to Programda notun adi ve notun içeriği üzerinden sorgulama yapmaya imkan sağlayan bir özellik bulunmaktadır. Bu özellik sayesinde notlarınızın içinde kaybolmaktan kurtulmuş olursunuz. Sorgulama için arama yapacağınız metni yazdıktan sonra Enter tusuna basmanız yeterli olacaktır. Sorgu sonuçları yine aynı sayfada yer alacaktır. Burada önemli bir nokta ise, eğer sorgu sonuçları içerisinde daha önceden şifrelediğiniz bir not klasörü içerisindeki nota erişmeye kalkarsanız sorgulama ekranından şifre giris ekranına [rest of string was truncated];.

C#

```
public static string SearchFeatureNote;
```

1.1.1.1.1.2.120 AppResources.SearchResults Property

Looks up a localized string similar to Arama Sonuçları.

C#

```
public static string SearchResults;
```

1.1.1.1.1.2.121 AppResources.SearchTrimFault Property

Looks up a localized string similar to Arama Alanını Doldurmak Zorundasınız..

C#

```
public static string SearchTrimFault;
```

1.1.1.1.1.2.122 AppResources.Select Property

Looks up a localized string similar to Seçiniz.

C#

```
public static string Select;
```

1.1.1.1.1.2.123 AppResources.SelectAll Property

Looks up a localized string similar to Tümünü Seç.

C#

```
public static string SelectAll;
```

1.1.1.1.1.2.124 AppResources.SelectBackgroundColor Property

Looks up a localized string similar to Arka Plan Rengi Seçiniz.

C#

```
public static string SelectBackgroundColor;
```

1.1.1.1.1.2.125 AppResources.Selected Property

Looks up a localized string similar to Seçilen.

C#

```
public static string Selected;
```

1.1.1.1.1.2.126 AppResources.SelectFontFamily Property

Looks up a localized string similar to Yazı Tipi Ailesi Seçiniz.

C#

```
public static string SelectFontFamily;
```

1.1.1.1.1.2.127 AppResources.SelectFontSize Property

Looks up a localized string similar to Yazı Tipi Boyu Seçiniz.

C#

```
public static string SelectFontSize;
```

1.1.1.1.1.2.128 AppResources.SelectLanguage Property

Looks up a localized string similar to Lütfen Dili Seçiniz.

C#

```
public static string SelectLanguage;
```

1.1.1.1.1.2.129 AppResources.SelectNoteFolder Property

Looks up a localized string similar to Lütfen Not Klasörünü Seçiniz.

C#

```
public static string SelectNoteFolder;
```

1.1.1.1.1.2.130 AppResources.SelectOrderStyle Property

Looks up a localized string similar to Sıralama Türü Seçiniz.

C#

```
public static string SelectOrderStyle;
```

1.1.1.1.1.2.131 AppResources.SelectOrderType Property

Looks up a localized string similar to Sıralama Ölçütünü Seçiniz.

C#

```
public static string SelectOrderType;
```

1.1.1.1.1.2.132 AppResources.SelectPasswordState Property

Looks up a localized string similar to Sifre Durumunu Seçiniz.

C#

```
public static string SelectPasswordState;
```

1.1.1.1.1.2.133 AppResources.SendWithApp Property

Looks up a localized string similar to Awesome Note (📧 see page 17) Uygulamasi ile Gönderildi.

C#

```
public static string SendWithApp;
```

1.1.1.1.1.2.134 AppResources.SendWithMail Property

Looks up a localized string similar to E-Posta Ile Gönder.

C#

```
public static string SendWithMail;
```

1.1.1.1.1.2.135 AppResources.SendWithSMS Property

Looks up a localized string similar to SMS Ile Gönder.

C#

```
public static string SendWithSMS;
```

1.1.1.1.1.2.136 AppResources.SetPassword Property

Looks up a localized string similar to Sifre Ayarla.

C#

```
public static string SetPassword;
```

1.1.1.1.1.2.137 AppResources.Settings Property

Looks up a localized string similar to Ayarlar.

C#

```
public static string Settings;
```

1.1.1.1.1.2.138 AppResources.Share Property

Looks up a localized string similar to Paylas.

C#

```
public static string Share;
```

1.1.1.1.1.2.139 AppResources.SignIn Property

Looks up a localized string similar to Baglan.

C#

```
public static string SignIn;
```

1.1.1.1.1.2.140 AppResources.SignOut Property

Looks up a localized string similar to Çikis Yap.

C#

```
public static string SignOut;
```

1.1.1.1.1.2.141 AppResources.Spanish Property

Looks up a localized string similar to Ispanyolca.

C#

```
public static string Spanish;
```

1.1.1.1.1.2.142 AppResources.SuccessfulAddNote Property

Looks up a localized string similar to Not Basarili Bir Sekilde Olusturuldu.

C#

```
public static string SuccessfulAddNote;
```

1.1.1.1.1.2.143 AppResources.SuccessfulAddNoteFolder Property

Looks up a localized string similar to Not Klasörü Basarili Bir Sekilde Olusturuldu.

C#

```
public static string SuccessfulAddNoteFolder;
```

1.1.1.1.1.2.144 AppResources.SuccessfulBackgroundColorChanged Property

Looks up a localized string similar to Arka Plan Rengi Basariyla Degistirildi.

C#

```
public static string SuccessfulBackgroundColorChanged;
```

1.1.1.1.1.2.145 AppResources.SuccessfulBackgroundImageChanged Property

Looks up a localized string similar to Arka Plan Resmi Basarili Bir Sekilde Degistirildi.

C#

```
public static string SuccessfulBackgroundImageChanged;
```

1.1.1.1.1.2.146 AppResources.SuccessfulBackgroundImageRemoved Property

Looks up a localized string similar to Arka Plan Resmi Basarili Bir Sekilde Kaldirildi.

C#

```
public static string SuccessfulBackgroundImageRemoved;
```

1.1.1.1.1.2.147 AppResources.SuccessfulChangeFolderName Property

Looks up a localized string similar to Not Klasör Ismi Basarili Bir Sekilde Degistirildi.

C#

```
public static string SuccessfulChangeFolderName;
```

1.1.1.1.1.2.148 AppResources.SuccessfulChangeNoteFolder Property

Looks up a localized string similar to Notunuzun Ait Oldugu Klasör Degistirildi.

C#

```
public static string SuccessfulChangeNoteFolder;
```

1.1.1.1.1.2.149 AppResources.SuccessfulChangeNoteName Property

Looks up a localized string similar to Notun Ismi Basarili Bir Sekilde Degistirildi.

C#

```
public static string SuccessfulChangeNoteName;
```

1.1.1.1.1.2.150 AppResources.SuccessfulDeleteFolder Property

Looks up a localized string similar to Not Klasörü Basariyla Silindi.

C#

```
public static string SuccessfulDeleteFolder;
```

1.1.1.1.1.2.151 AppResources.SuccessfulDeleteNote Property

Looks up a localized string similar to Not Basarili Bir Sekilde Silindi.

C#

```
public static string SuccessfulDeleteNote;
```

1.1.1.1.1.2.152 AppResources.SuccessfulFontFamilyChanged Property

Looks up a localized string similar to Yazı Tipi Ailesi Başarılı Bir Şekilde Kaydedildi.

C#

```
public static string SuccessfulFontFamilyChanged;
```

1.1.1.1.1.2.153 AppResources.SuccessfulFontSizeChanged Property

Looks up a localized string similar to Yazı Tipi Boyutu Başarılı Bir Şekilde Kaydedildi.

C#

```
public static string SuccessfulFontSizeChanged;
```

1.1.1.1.1.2.154 AppResources.SuccessfulOrderStyleChanged Property

Looks up a localized string similar to Sıralama Türü Başarılı Bir Şekilde Kaydedildi.

C#

```
public static string SuccessfulOrderStyleChanged;
```

1.1.1.1.1.2.155 AppResources.SuccessfulOrderTypeChanged Property

Looks up a localized string similar to Sıralama Kriteri Başarılı Bir Şekilde Kaydedildi.

C#

```
public static string SuccessfulOrderTypeChanged;
```

1.1.1.1.1.2.156 AppResources.SuccessfulRemovePassword Property

Looks up a localized string similar to Şifre Başarılı Bir Şekilde Kaldırıldı.

C#

```
public static string SuccessfulRemovePassword;
```

1.1.1.1.1.2.157 AppResources.SuccessfulResetSettings Property

Looks up a localized string similar to Arka Plan Ayarları Başarıyla Sıfırlandı.

C#

```
public static string SuccessfulResetSettings;
```

1.1.1.1.1.2.158 AppResources.SuccessfulSendWithMail Property

Looks up a localized string similar to E-Posta Yoluyla Not Başarılı Bir Şekilde Gönderildi.

C#

```
public static string SuccessfulSendWithMail;
```

1.1.1.1.1.2.159 AppResources.SuccessfulSendWithSMS Property

Looks up a localized string similar to SMS Yoluyla Not Başarılı Bir Şekilde Gönderildi.

C#

```
public static string SuccessfulSendWithSMS;
```

1.1.1.1.1.2.160 AppResources.SuccessfulSetPassword Property

Looks up a localized string similar to Şifre Başarılı Bir Şekilde Oluşturuldu.

C#

```
public static string SuccessfulSetPassword;
```


1.1.1.1.2.161 AppResources.Sync Property

Looks up a localized string similar to Eszamanlama.

C#

```
public static string Sync;
```

1.1.1.1.2.162 AppResources.SyncFeature Property

Looks up a localized string similar to Senkronizasyon Özellikleri.

C#

```
public static string SyncFeature;
```

1.1.1.1.2.163 AppResources.SyncFeatureNote Property

Looks up a localized string similar to Program vasitasiyla tüm notlarınızı OneDrive (🔗 see page 18) hesabınız ile senkronize edebilirsiniz. Uygulama içerisinde Genel Ayarlar kısmında Senkronizasyon bölümüne girdikten sonra Sign In yazan butona tıklayıp, OneDrive (🔗 see page 18) hesabınıza giriş yapmanız gerekmektedir. Giriş esnasında sordugu erişim iznini onaylamadığınız zaman senkronizasyon işlemini gerçekleştirmeyecektir. Sonrasında ise Senkronizasyon için gerekli butona bastığınız zaman program OneDrive (🔗 see page 18) hesabınız içerisinde bir klasör oluşturacak (örneğin: Awesome Note (🔗 see page 17) (21. [rest of string was truncated]);.

C#

```
public static string SyncFeatureNote;
```

1.1.1.1.2.164 AppResources.SyncOnOneFile Property

Looks up a localized string similar to Hepsini Tek Dosyada Birleştir.

C#

```
public static string SyncOnOneFile;
```

1.1.1.1.2.165 AppResources.Synchronizing Property

Looks up a localized string similar to Eszamanlanıyor....

C#

```
public static string Synchronizing;
```

1.1.1.1.2.166 AppResources.SystemFault Property

Looks up a localized string similar to Sistemde Bir Hata Olustu. Uygulama Sonlanacak....

C#

```
public static string SystemFault;
```

1.1.1.1.2.167 AppResources.Thai Property

Looks up a localized string similar to Tay Dili.

C#

```
public static string Thai;
```

1.1.1.1.2.168 AppResources.Tips Property

Looks up a localized string similar to Programın İpuçları.

C#

```
public static string Tips;
```

1.1.1.1.1.2.169 AppResources.Turkish Property

Looks up a localized string similar to Türkçe.

C#

```
public static string Turkish;
```

1.1.1.1.1.2.170 AppResources.TwoFieldsNotSame Property

Looks up a localized string similar to Girilen Iki Sifre Alani Ayni Degil.

C#

```
public static string TwoFieldsNotSame;
```

1.1.1.1.1.2.171 AppResources.Welcome Property

Looks up a localized string similar to Hosgeldiniz.

C#

```
public static string Welcome;
```

1.1.1.1.1.2.172 AppResources.WelcomeNote Property

Looks up a localized string similar to Merhaba. Not almayı seven bir kullanıcı olarak Bilgisayar ve Mobil cihazlarda pek çok not tutma programı kullandım. Ve faydalı olacağını düşündüğüm özellikleri birleştirip bu uygulamayı yazdım. Umarım uygulamadan memnun kalırsınız..

C#

```
public static string WelcomeNote;
```

1.1.1.1.1.2.173 AppResources.WordNumber Property

Looks up a localized string similar to Kelime Sayisi.

C#

```
public static string WordNumber;
```

1.1.1.1.1.2.174 AppResources.Yellow Property

Looks up a localized string similar to Sari.

C#

```
public static string Yellow;
```

1.1.1.1.1.2.175 AppResources.Yes Property

Looks up a localized string similar to Evet.


C#

```
public static string Yes;
```

1.1.2 Classes

The following table lists classes in this documentation.

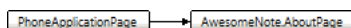
Classes

	Name	Description
	AboutPage (see page 30)	This is class AwesomeNote.AboutPage.

	App (see page 32)	This is class AwesomeNote.App.
	AppSettings (see page 33)	This is class AwesomeNote.AppSettings.
	AutoScrollBehavior (see page 35)	This is class AwesomeNote.AutoScrollBehavior.
	BackgroundColorSettings (see page 36)	This is class AwesomeNote.BackgroundColorSettings.
	ChangeNoteFolder (see page 37)	This is class AwesomeNote.ChangeNoteFolder.
	EnterPasswordPage (see page 40)	This is class AwesomeNote.EnterPasswordPage.
	FolderOrderSettings (see page 43)	This is class AwesomeNote.FolderOrderSettings.
	FolderOrderStyleSettings (see page 45)	This is class AwesomeNote.FolderOrderStyleSettings.
	FolderSettings (see page 46)	This is class AwesomeNote.FolderSettings.
	FontFamilySettings (see page 49)	This is class AwesomeNote.FontFamilySettings.
	FontSizeSettings (see page 52)	This is class AwesomeNote.FontSizeSettings.
	GeneralSettings (see page 54)	This is class AwesomeNote.GeneralSettings.
	Language (see page 59)	This is class AwesomeNote.Language.
	LanguageSettings (see page 60)	This is class AwesomeNote.LanguageSettings.
	LocalizedStrings (see page 62)	Provides access to string resources.
	MainPage (see page 62)	This is class AwesomeNote.MainPage.
	Note (see page 64)	This is class AwesomeNote.Note.
	NoteDetail (see page 66)	This is class AwesomeNote.NoteDetail.
	NoteFolder (see page 70)	This is class AwesomeNote.NoteFolder.
	NoteFolderDataContext (see page 73)	This is class AwesomeNote.NoteFolderDataContext.
	NoteList (see page 75)	This is class AwesomeNote.NoteList.
	NoteOrderSettings (see page 79)	This is class AwesomeNote.NoteOrderSettings.
	NoteOrderStyleSettings (see page 81)	This is class AwesomeNote.NoteOrderStyleSettings.
	Page1 (see page 83)	This is class AwesomeNote.Page1.
	PopupAddFolder (see page 86)	This is class AwesomeNote.PopupAddFolder.
	PopupAddNote (see page 86)	This is class AwesomeNote.PopupAddNote.
	PopupChangeFolder (see page 87)	This is class AwesomeNote.PopupChangeFolder.
	PopupChangeFolderName (see page 88)	This is class AwesomeNote.PopupChangeFolderName.
	PopupChangeNoteName (see page 88)	This is class AwesomeNote.PopupChangeNoteName.
	PopupEnterPassword (see page 89)	This is class AwesomeNote.PopupEnterPassword.
	PopupSetPassword (see page 90)	This is class AwesomeNote.PopupSetPassword.
	SearchPage (see page 90)	This is class AwesomeNote.SearchPage.
	SetPasswordPage (see page 92)	This is class AwesomeNote.SetPasswordPage.

1.1.2.1 AboutPage Class

Class Hierarchy



C#

```
public class AboutPage : PhoneApplicationPage;
```

File

AboutPage.xaml.cs (see page 95)

Description

This is class AwesomeNote.AboutPage.

Methods

	Name	Description
	AboutPage (see page 31)	This is AboutPage, a member of class AboutPage.

AboutPage Fields

	Name	Description
	messageBackGround (see page 32)	This is messageBackGround, a member of class AboutPage.

1.1.2.1.1 AboutPage.AboutPage Constructor**C#**

```
public AboutPage();
```

Description

This is AboutPage, a member of class AboutPage.

Body Source

```
1: public AboutPage()
2: {
3:     InitializeComponent();
4:     SetBackgroundColor();
5:
6:     ApplicationBar = new ApplicationBar();
7:
8:     ApplicationBarIconButton button2 = new ApplicationBarIconButton();
9:     button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
10:    button2.Text = AppResources.ContactWithUs;
11:    ApplicationBar.Buttons.Add(button2);
12:    button2.Click += new EventHandler(SendMailButton_Click);
13:
14:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
15:    button3.IconUri = new Uri("/Assets/Rate.png", UriKind.Relative);
16:    button3.Text = AppResources.Rate;
17:    ApplicationBar.Buttons.Add(button3);
18:    button3.Click += new EventHandler(RateButton_Click);
19:
20:    lblAboutTheApp.Text = AppResources.AboutTheApp;
21:    //txtAbout2.Text = AppResources.AboutTheAppText;
22:    //var paragraph = new Paragraph();
23:    //paragraph.Inlines.Add(AppResources.AboutTheAppText);
24:    //txtAbout.Blocks.Add(paragraph);
25:    txtAbout.Text = AppResources.AboutTheAppText;
26:    //txtAbout.IsEnabled = false;
27:    txtAbout.IsReadOnly = true;
28:    //this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
29: }
```

1.1.2.1.2 AboutPage Fields

1.1.2.1.2.1 AboutPage.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class AboutPage.

1.1.2.2 App Class

Class Hierarchy



C#

```
public class App : Application;
```

File

App.xaml.cs (🔗 see page 98)

Description

This is class AwesomeNote.App.

Methods

	Name	Description
🔗	App (🔗 see page 32)	Constructor for the Application object.

App Properties

	Name	Description
🔗	IsTrial (🔗 see page 33)	This is IsTrial, a member of class App.
🔗	RootFrame (🔗 see page 33)	Provides easy access to the root frame of the Phone Application.

1.1.2.2.1 App.App Constructor

Constructor for the Application object.

C#

```
public App();
```

Body Source

```

1: public App()
2: {
3:     // Global handler for uncaught exceptions.
4:     UnhandledException += Application_UnhandledException;
5:
6:     // Standard XAML initialization
7:     InitializeComponent();
8:
9:     ThemeManager.ToDarkTheme();
10:
11:    // Phone-specific initialization
12:    InitializePhoneApplication();
13:
14:    // Language display initialization
15:    InitializeLanguage();
16:
17:    // Show graphics profiling information while debugging.
  
```

```

18:         if (Debugger.IsAttached)
19:         {
20:             // Display the current frame rate counters.
21:             Application.Current.Host.Settings.EnableFrameRateCounter = true;
22:
23:             // Show the areas of the app that are being redrawn in each frame.
24:             //Application.Current.Host.Settings.EnableRedrawRegions = true;
25:
26:             // Enable non-production analysis visualization mode,
27:             // which shows areas of a page that are handed off to GPU with a colored
28:             //Application.Current.Host.Settings.EnableCacheVisualization = true;
29:
30:             // Prevent the screen from turning off while under the debugger by disabling
31:             // the application's idle detection.
32:             // Caution:- Use this under debug mode only. Application that disables user
33:             // idle detection will continue to run
34:             // and consume battery power when the user is not using the phone.
35:             PhoneApplicationService.Current.UserIdleDetectionMode =
36:             IdleDetectionMode.Disabled;
37:         }

```

1.1.2.2.2 App Properties

1.1.2.2.2.1 App.IsTrial Property

C#

```
public static bool IsTrial;
```

Description

This is IsTrial, a member of class App.

1.1.2.2.2.2 App.RootFrame Property

Provides easy access to the root frame of the Phone Application.

C#

```
public static PhoneApplicationFrame RootFrame;
```

Returns

The root frame of the Phone Application.

1.1.2.3 AppSettings Class

Class Hierarchy

```
AwesomeNote.AppSettings
```

C#

```
[Table]
public class AppSettings;
```







File

AppSettings.cs (🔗 see page 133)

Description

This is class AwesomeNote.AppSettings.

AppSettings Properties

	Name	Description
	AppBackgroundColor (see page 34)	This is AppBackgroundColor, a member of class AppSettings.
	AppBackgroundImage (see page 34)	This is AppBackgroundImage, a member of class AppSettings.
	AppLangName (see page 34)	This is AppLangName, a member of class AppSettings.
	AppSettingsId (see page 34)	This is AppSettingsId, a member of class AppSettings.
	FolderOrderBy (see page 35)	This is FolderOrderBy, a member of class AppSettings.
	FolderOrderStyle (see page 35)	This is FolderOrderStyle, a member of class AppSettings.

1.1.2.3.1 AppSettings Properties**1.1.2.3.1.1 AppSettings.AppBackgroundColor Property****C#**

```
[Column]
public string AppBackgroundColor;
```

Description

This is AppBackgroundColor, a member of class AppSettings.

1.1.2.3.1.2 AppSettings.AppBackgroundImage Property**C#**

```
[Column(DbType = "Image", UpdateCheck = UpdateCheck.Never)]
public byte AppBackgroundImage;
```

Description

This is AppBackgroundImage, a member of class AppSettings.

1.1.2.3.1.3 AppSettings.AppLangName Property**C#**

```
[Column]
public string AppLangName;
```

Description

This is AppLangName, a member of class AppSettings.

1.1.2.3.1.4 AppSettings.AppSettingsId Property**C#**

```
[Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
CanBeNull = false)]
public int AppSettingsId;
```

Description

This is AppSettingsId, a member of class AppSettings.

1.1.2.3.1.5 AppSettings.FolderOrderBy Property

C#

```
[Column]
public string FolderOrderBy;
```

Description

This is FolderOrderBy, a member of class AppSettings.

1.1.2.3.1.6 AppSettings.FolderOrderStyle Property

C#

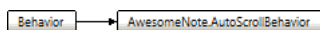
```
[Column]
public string FolderOrderStyle;
```

Description

This is FolderOrderStyle, a member of class AppSettings.

1.1.2.4 AutoScrollBehavior Class

Class Hierarchy



C#

```
public class AutoScrollBehavior : Behavior<ScrollView>;
```

File

AutoScrollBehavior.cs (see page 134)

Description

This is class AwesomeNote.AutoScrollBehavior.

AutoScrollBehavior Methods

	Name	Description
🔗	OnAttached (see page 35)	This is OnAttached, a member of class AutoScrollBehavior.
🔗	OnDetaching (see page 36)	This is OnDetaching, a member of class AutoScrollBehavior.

1.1.2.4.1 AutoScrollBehavior Methods

1.1.2.4.1.1 AutoScrollBehavior.OnAttached Method

C#

```
protected override void OnAttached();
```

Description

This is OnAttached, a member of class AutoScrollBehavior.

Body Source

```
1: protected override void OnAttached()
2: {
3:     base.OnAttached();
4:
5:     this._scrollView = base.AssociatedObject;
```



```
6:         this._scrollView.LayoutUpdated += new EventHandler(_scrollView_LayoutUpdated);
7:     }
```

1.1.2.4.1.2 AutoScrollBehavior.OnDetaching Method

C#

```
protected override void OnDetaching();
```

Description

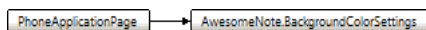
This is OnDetaching, a member of class AutoScrollBehavior.

Body Source

```
1: protected override void OnDetaching()
2: {
3:     base.OnDetaching();
4:
5:     if (this._scrollView != null)
6:         this._scrollView.LayoutUpdated -= new
EventHandler(_scrollView_LayoutUpdated);
7: }
```

1.1.2.5 BackgroundColorSettings Class

Class Hierarchy



C#

```
public class BackgroundColorSettings : PhoneApplicationPage;
```


File

BackgroundColorSettings.xaml.cs (see page 136)

Description

This is class AwesomeNote.BackgroundColorSettings.

Methods

	Name	Description
	BackgroundColorSettings (see page 36)	This is BackgroundColorSettings, a member of class BackgroundColorSettings.

BackgroundColorSettings Fields

	Name	Description
	messageBackGround (see page 37)	This is messageBackGround, a member of class BackgroundColorSettings.

1.1.2.5.1 BackgroundColorSettings.BackgroundColorSettings Constructor

C#

```
public BackgroundColorSettings();
```

Description

This is BackgroundColorSettings, a member of class BackgroundColorSettings.

Body Source

```
1: public BackgroundColorSettings()
2: {
```

```

3:     InitializeComponent();
4:
5:     lstBackgroundColor.Items.Clear();
6:     lstBackgroundColor.Items.Add(AppResources.Black);
7:     lstBackgroundColor.Items.Add(AppResources.Blue);
8:     lstBackgroundColor.Items.Add(AppResources.Brown);
9:     lstBackgroundColor.Items.Add(AppResources.Gray);
10:    lstBackgroundColor.Items.Add(AppResources.Green);
11:    lstBackgroundColor.Items.Add(AppResources.Orange);
12:    lstBackgroundColor.Items.Add(AppResources.Purple);
13:    lstBackgroundColor.Items.Add(AppResources.Red);
14:    lstBackgroundColor.Items.Add(AppResources.Yellow);
15:    lstBackgroundColor.SelectedIndex = -1;
16:
17:    lblBackgroundColor.Text = AppResources.SelectBackgroundColor;
18:
19: }

```

1.1.2.5.2 BackgroundColorSettings Fields

1.1.2.5.2.1 BackgroundColorSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class BackgroundColorSettings.

1.1.2.6 ChangeNoteFolder Class

Class Hierarchy



C#

```
public class ChangeNoteFolder : PhoneApplicationPage;
```

File

ChangeNoteFolder.xaml.cs ([🔗](#) see page 139)




Description

This is class AwesomeNote.ChangeNoteFolder.

Methods

	Name	Description
	ChangeNoteFolder (🔗 see page 38)	This is ChangeNoteFolder, a member of class ChangeNoteFolder.

ChangeNoteFolder Fields

	Name	Description
	messageBackGround (🔗 see page 38)	This is messageBackGround, a member of class ChangeNoteFolder.
	noteFolderId (🔗 see page 38)	This is noteFolderId, a member of class ChangeNoteFolder.
	notelId (🔗 see page 38)	This is notelId, a member of class ChangeNoteFolder.

ChangeNoteFolder Methods

	Name	Description
☞	OnFragmentNavigation (☞ see page 39)	This is OnFragmentNavigation, a member of class ChangeNoteFolder.
☞	OnNavigatedFrom (☞ see page 39)	This is OnNavigatedFrom, a member of class ChangeNoteFolder.
☞	OnNavigatedTo (☞ see page 39)	This is OnNavigatedTo, a member of class ChangeNoteFolder.

1.1.2.6.1 ChangeNoteFolder.ChangeNoteFolder Constructor**C#**

```
public ChangeNoteFolder();
```

Description

This is ChangeNoteFolder, a member of class ChangeNoteFolder.

Body Source

```
1: public ChangeNoteFolder()  
2: {  
3:     InitializeComponent();  
4:     SetBackgroundColor();  
5: }
```

1.1.2.6.2 ChangeNoteFolder Fields**1.1.2.6.2.1 ChangeNoteFolder.messageBackGround Field****C#**

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class ChangeNoteFolder.

1.1.2.6.2.2 ChangeNoteFolder.noteFolderId Field**C#**

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class ChangeNoteFolder.

1.1.2.6.2.3 ChangeNoteFolder.noteId Field**C#**

```
public int noteId;
```

Description

This is noteId, a member of class ChangeNoteFolder.

1.1.2.6.3 ChangeNoteFolder Methods

1.1.2.6.3.1 ChangeNoteFolder.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class ChangeNoteFolder.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         lstNoteFolders.Items.Clear();
10:        var note = context.Notes.Where(j => j.NoteId.Equals(noteId)).Single() as Note;
11:        noteFolderId = note.NoteFolderId;
12:        lblNoteName.Text = note.NoteName;
13:        lblChangeFolder.Text = AppResources.SelectNoteFolder;
14:        var noteFolders = context.NoteFolders;
15:        lstNoteFolders.ItemsSource = noteFolders;
16:        lstNoteFolders.DisplayMemberPath = "NoteFolderName";
17:    }
18: }
```

1.1.2.6.3.2 ChangeNoteFolder.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class ChangeNoteFolder.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.6.3.3 ChangeNoteFolder.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

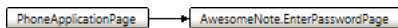
This is OnNavigatedTo, a member of class ChangeNoteFolder.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.7 EnterPasswordPage Class

Class Hierarchy



C#

```
public class EnterPasswordPage : PhoneApplicationPage;
```

File

EnterPasswordPage.xaml.cs ([see page 141](#))

Description

This is class AwesomeNote.EnterPasswordPage.

Methods

	Name	Description
	EnterPasswordPage (see page 40)	This is EnterPasswordPage, a member of class EnterPasswordPage.

EnterPasswordPage Fields

	Name	Description
	messageBackGround (see page 41)	This is messageBackGround, a member of class EnterPasswordPage.
	note (see page 41)	This is note, a member of class EnterPasswordPage.
	noteFolder (see page 41)	This is noteFolder, a member of class EnterPasswordPage.
	noteFolderId (see page 41)	This is noteFolderId, a member of class EnterPasswordPage.
	noteld (see page 41)	This is noteld, a member of class EnterPasswordPage.
	pageName (see page 41)	This is pageName, a member of class EnterPasswordPage.

EnterPasswordPage Methods

	Name	Description
	OnFragmentNavigation (see page 42)	This is OnFragmentNavigation, a member of class EnterPasswordPage.
	OnNavigatedFrom (see page 42)	This is OnNavigatedFrom, a member of class EnterPasswordPage.
	OnNavigatedTo (see page 43)	This is OnNavigatedTo, a member of class EnterPasswordPage.

1.1.2.7.1 EnterPasswordPage.EnterPasswordPage Constructor

C#

```
public EnterPasswordPage();
```

Description

This is EnterPasswordPage, a member of class EnterPasswordPage.

Body Source

```

1: public EnterPasswordPage()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:     //txtPassword.Focus();
6: }
```

1.1.2.7.2 EnterPasswordPage Fields

1.1.2.7.2.1 EnterPasswordPage.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class EnterPasswordPage.

1.1.2.7.2.2 EnterPasswordPage.note Field

C#

```
public Note note;
```

Description

This is note, a member of class EnterPasswordPage.

1.1.2.7.2.3 EnterPasswordPage.noteFolder Field

C#

```
public NoteFolder noteFolder;
```

Description

This is noteFolder, a member of class EnterPasswordPage.

1.1.2.7.2.4 EnterPasswordPage.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class EnterPasswordPage.

1.1.2.7.2.5 EnterPasswordPage.noteId Field

C#

```
public int noteId;
```

Description

This is noteld, a member of class EnterPasswordPage.

1.1.2.7.2.6 EnterPasswordPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class EnterPasswordPage.

1.1.2.7.3 EnterPasswordPage Methods

1.1.2.7.3.1 EnterPasswordPage.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class EnterPasswordPage.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:
6:     base.OnFragmentNavigation(e);
7:     if (pageName == "/MainPage.xaml")
8:     {
9:         noteFolderId = int.Parse(e.Fragment);
10:        using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
11:        {
12:            var noteFolders =
13:                context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
14:            noteFolder = noteFolders;
15:            lblFolderName.Text = noteFolders.NoteFolderName;
16:        }
17:    }
18:    else
19:    {
20:        noteId = int.Parse(e.Fragment);
21:        using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
22:        {
23:            var note = context.Notes.Where(j => j.NoteId.Equals(noteId)).Single() as
Note;
24:            noteFolderId = note.NoteFolderId;
25:            var noteFolders =
26:                context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
27:            noteFolder = noteFolders;
28:            lblFolderName.TextWrapping = TextWrapping.Wrap;
29:            //lblFolderName.Text = AppResources.SearchResults + " (" +
noteFolders.NoteFolderName + ")";
30:            lblFolderName.Text = noteFolders.NoteFolderName;
31:        }
32:    }
33:    lblEnterPassword.Text = AppResources.EnterPassword;
34:    txtLabel.Text = AppResources.PleaseEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
35:    //btnCancel.Content = AppResources.Cancel;
36:    //btnOK.Content = AppResources.OK;
37:    SetBackgroundColor();
38:
39: }
```

1.1.2.7.3.2 EnterPasswordPage.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class EnterPasswordPage.

Body Source

```

1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }

```

1.1.2.7.3.3 EnterPasswordPage.OnNavigatedTo Method**C#**

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class EnterPasswordPage.

Body Source

```

1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor bu bölüm
5:     var lastPage = NavigationService.BackStack.FirstOrDefault();
6:     pageName = lastPage.Source.ToString();
7:     txtPassword.Focus();
8: }

```

1.1.2.8 FolderOrderSettings Class**Class Hierarchy****C#**

```
public class FolderOrderSettings : PhoneApplicationPage;
```

File

FolderOrderSettings.xaml.cs (see page 145)

Description

This is class AwesomeNote.FolderOrderSettings.

Methods

	Name	Description
	FolderOrderSettings (see page 44)	This is FolderOrderSettings, a member of class FolderOrderSettings.

FolderOrderSettings Fields

	Name	Description
	messageBackGround (see page 44)	This is messageBackGround, a member of class FolderOrderSettings.

FolderOrderSettings Methods

	Name	Description
	OnNavigatedFrom (see page 44)	This is OnNavigatedFrom, a member of class FolderOrderSettings.
	OnNavigatedTo (see page 44)	This is OnNavigatedTo, a member of class FolderOrderSettings.

1.1.2.8.1 FolderOrderSettings.FolderOrderSettings Constructor

C#

```
public FolderOrderSettings();
```

Description

This is FolderOrderSettings, a member of class FolderOrderSettings.

Body Source

```
1: public FolderOrderSettings()  
2: {  
3:     InitializeComponent();  
4:     SetBackgroundColor();  
5:  
6:     lstFolderOrder.Items.Clear();  
7:     lstFolderOrder.Items.Add(AppResources.FolderName);  
8:     lstFolderOrder.Items.Add(AppResources.CreationDate);  
9:     lstFolderOrder.Items.Add(AppResources.ModificationDate);  
10:    lstFolderOrder.SelectedIndex = -1;  
11:  
12:    lblFolderOrder.Text = AppResources.SelectOrderType;  
13: }
```

1.1.2.8.2 FolderOrderSettings Fields

1.1.2.8.2.1 FolderOrderSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class FolderOrderSettings.

1.1.2.8.3 FolderOrderSettings Methods

1.1.2.8.3.1 FolderOrderSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class FolderOrderSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)  
2: {  
3:     base.OnNavigatedFrom(e);  
4: }
```

1.1.2.8.3.2 FolderOrderSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

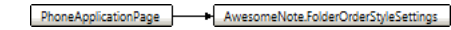
This is OnNavigatedTo, a member of class FolderOrderSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.9 FolderOrderStyleSettings Class

Class Hierarchy



C#

```
public class FolderOrderStyleSettings : PhoneApplicationPage;
```

File

FolderOrderStyleSettings.xaml.cs (see page 148)

Description

This is class AwesomeNote.FolderOrderStyleSettings.

Methods

	Name	Description
	FolderOrderStyleSettings (see page 45)	This is FolderOrderStyleSettings, a member of class FolderOrderStyleSettings.

FolderOrderStyleSettings Fields

	Name	Description
	messageBackGround (see page 46)	This is messageBackGround, a member of class FolderOrderStyleSettings.

FolderOrderStyleSettings Methods

	Name	Description
	OnNavigatedFrom (see page 46)	This is OnNavigatedFrom, a member of class FolderOrderStyleSettings.
	OnNavigatedTo (see page 46)	This is OnNavigatedTo, a member of class FolderOrderStyleSettings.

1.1.2.9.1 FolderOrderStyleSettings.FolderOrderStyleSettings Constructor

C#

```
public FolderOrderStyleSettings();
```

Description

This is FolderOrderStyleSettings, a member of class FolderOrderStyleSettings.

Body Source

```
1: public FolderOrderStyleSettings()
2: {
3:     InitializeComponent();
4:     SetBackgroundColor();
5:
6:     lstFolderOrderStyle.Items.Clear();
7:     lstFolderOrderStyle.Items.Add(AppResources.Ascending);
8:     lstFolderOrderStyle.Items.Add(AppResources.Descending);
9:     lstFolderOrderStyle.SelectedIndex = -1;
10:
11:     lblFolderOrderStyle.Text = AppResources.SelectOrderStyle;
```

```
12: }
```

1.1.2.9.2 FolderOrderStyleSettings Fields

1.1.2.9.2.1 FolderOrderStyleSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class FolderOrderStyleSettings.

1.1.2.9.3 FolderOrderStyleSettings Methods

1.1.2.9.3.1 FolderOrderStyleSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class FolderOrderStyleSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.9.3.2 FolderOrderStyleSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class FolderOrderStyleSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.10 FolderSettings Class

Class Hierarchy



C#

```
public class FolderSettings : PhoneApplicationPage;
```

File

FolderSettings.xaml.cs (see page 151)



Description

This is class AwesomeNote.FolderSettings.




Methods

	Name	Description
	FolderSettings (see page 47)	This is FolderSettings, a member of class FolderSettings.

FolderSettings Fields

	Name	Description
	messageBackGround (see page 47)	This is messageBackGround, a member of class FolderSettings.
	noteFolderId (see page 47)	This is noteFolderId, a member of class FolderSettings.

FolderSettings Methods

	Name	Description
	OnFragmentNavigation (see page 48)	This is OnFragmentNavigation, a member of class FolderSettings.
	OnNavigatedFrom (see page 48)	This is OnNavigatedFrom, a member of class FolderSettings.
	OnNavigatedTo (see page 49)	This is OnNavigatedTo, a member of class FolderSettings.

1.1.2.10.1 FolderSettings.FolderSettings Constructor

C#

```
public FolderSettings();
```

Description

This is FolderSettings, a member of class FolderSettings.

Body Source

```
1: public FolderSettings()  
2: {  
3:     InitializeComponent();  
4:     //SetBackgroundColor();  
5: }
```

1.1.2.10.2 FolderSettings Fields

1.1.2.10.2.1 FolderSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class FolderSettings.

1.1.2.10.2.2 FolderSettings.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class FolderSettings.

1.1.2.10.3 FolderSettings Methods

1.1.2.10.3.1 FolderSettings.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class FolderSettings.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolder =
10:             context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
11:         lblFontFamily.Text = AppResources.FontFamily + " (" + AppResources.Selected +
": " + noteFolder.FontFamily + ")";
12:         lblFontSize.Text = AppResources.FontSize + " (" + AppResources.Selected + ": "
+ noteFolder.FontSize + ")";
13:         if (noteFolder.NoteOrderBy == "NAME")
14:         {
15:             lblNoteOrder.Text = AppResources.NoteOrder + " (" + AppResources.Selected +
": " + AppResources.NoteName + ")";
16:         }
17:         if (noteFolder.NoteOrderBy == "CDATE")
18:         {
19:             lblNoteOrder.Text = AppResources.NoteOrder + " (" + AppResources.Selected +
": " + AppResources.CreationDate + ")";
20:         }
21:         if (noteFolder.NoteOrderBy == "MDATE")
22:         {
23:             lblNoteOrder.Text = AppResources.NoteOrder + " (" + AppResources.Selected +
": " + AppResources.ModificationDate + ")";
24:         }
25:         if (noteFolder.NoteOrderStyle == "A")
26:         {
27:             lblNoteOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
28:         }
29:         if (noteFolder.NoteOrderStyle == "D")
30:         {
31:             lblNoteOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
32:         }
33:     }
34:     SetBackgroundColor();
35: }
```

1.1.2.10.3.2 FolderSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class FolderSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.10.3.3 FolderSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class FolderSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.11 FontFamilySettings Class

Class Hierarchy



C#

```
public class FontFamilySettings : PhoneApplicationPage;
```

File

FontFamilySettings.xaml.cs (🔗 see page 157)

Description

This is class AwesomeNote.FontFamilySettings.

Methods

	Name	Description
🔗	FontFamilySettings (🔗 see page 50)	This is FontFamilySettings, a member of class FontFamilySettings.

FontFamilySettings Fields

	Name	Description
🔗	messageBackGround (🔗 see page 50)	This is messageBackGround, a member of class FontFamilySettings.
🔗	noteFolderId (🔗 see page 50)	This is noteFolderId, a member of class FontFamilySettings.

FontFamilySettings Methods

	Name	Description
🔗🔗	OnFragmentNavigation (🔗 see page 51)	This is OnFragmentNavigation, a member of class FontFamilySettings.
🔗🔗	OnNavigatedFrom (🔗 see page 51)	This is OnNavigatedFrom, a member of class FontFamilySettings.
🔗🔗	OnNavigatedTo (🔗 see page 51)	This is OnNavigatedTo, a member of class FontFamilySettings.

1.1.2.11.1 FontFamilySettings.FontFamilySettings Constructor

C#

```
public FontFamilySettings();
```

Description

This is FontFamilySettings, a member of class FontFamilySettings.

Body Source

```
1: public FontFamilySettings()  
2: {  
3:     InitializeComponent();  
4:     //SetBackgroundColor();  
5:  
6:     lstFontFamily.Items.Clear();  
7:     lstFontFamily.Items.Add("Arial");  
8:     lstFontFamily.Items.Add("Arial Black");  
9:     lstFontFamily.Items.Add("Baskerville Old Face");  
10:    lstFontFamily.Items.Add("Berlin Sans FB");  
11:    lstFontFamily.Items.Add("Bookman Old Style");  
12:    lstFontFamily.Items.Add("Calibri");  
13:    lstFontFamily.Items.Add("Cambria");  
14:    lstFontFamily.Items.Add("Candara");  
15:    lstFontFamily.Items.Add("Comic Sans MS");  
16:    lstFontFamily.Items.Add("Consolas");  
17:    lstFontFamily.Items.Add("Constantia");  
18:    lstFontFamily.Items.Add("Courier New");  
19:    lstFontFamily.Items.Add("DokChampa");  
20:    lstFontFamily.Items.Add("Ebrima");  
21:    lstFontFamily.Items.Add("Georgia");  
22:    lstFontFamily.Items.Add("Lucida Sans Unicode");  
23:    lstFontFamily.Items.Add("Meiryo UI");  
24:    lstFontFamily.Items.Add("Microsoft YaHei");  
25:    lstFontFamily.Items.Add("Malgun Gothic");  
26:    lstFontFamily.Items.Add("Segoe UI");  
27:    lstFontFamily.Items.Add("Segoe WP");  
28:    lstFontFamily.Items.Add("Tahoma");  
29:    lstFontFamily.Items.Add("Trebuchet MS");  
30:    lstFontFamily.Items.Add("Times New Roman");  
31:    lstFontFamily.Items.Add("Verdana");  
32:    lstFontFamily.SelectedIndex = -1;  
33: }
```

1.1.2.11.2 FontFamilySettings Fields

1.1.2.11.2.1 FontFamilySettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class FontFamilySettings.

1.1.2.11.2.2 FontFamilySettings.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class FontFamilySettings.

1.1.2.11.3 FontFamilySettings Methods

1.1.2.11.3.1 FontFamilySettings.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class FontFamilySettings.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
10:         lblFolderName.Text = noteFolders.NoteFolderName;
11:         lblFontFamily.Text = AppResources.SelectFontFamily;
12:     }
13:     SetBackgroundColor();
14: }
```

1.1.2.11.3.2 FontFamilySettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class FontFamilySettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.11.3.3 FontFamilySettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

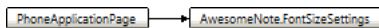
This is OnNavigatedTo, a member of class FontFamilySettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```


1.1.2.12 FontSizeSettings Class

Class Hierarchy



C#

```
public class FontSizeSettings : PhoneApplicationPage;
```

File

FontSizeSettings.xaml.cs ([see page 160](#))

Description

This is class AwesomeNote.FontSizeSettings.

Methods

	Name	Description
	FontSizeSettings (see page 52)	This is FontSizeSettings, a member of class FontSizeSettings.

FontSizeSettings Fields

	Name	Description
	messageBackGround (see page 53)	This is messageBackGround, a member of class FontSizeSettings.
	noteFolderId (see page 53)	This is noteFolderId, a member of class FontSizeSettings.

FontSizeSettings Methods

	Name	Description
	OnFragmentNavigation (see page 53)	This is OnFragmentNavigation, a member of class FontSizeSettings.
	OnNavigatedFrom (see page 54)	This is OnNavigatedFrom, a member of class FontSizeSettings.
	OnNavigatedTo (see page 54)	This is OnNavigatedTo, a member of class FontSizeSettings.

1.1.2.12.1 FontSizeSettings.FontSizeSettings Constructor

C#

```
public FontSizeSettings();
```

Description

This is FontSizeSettings, a member of class FontSizeSettings.

Body Source

```
1: public FontSizeSettings()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     lstFontSize.Items.Clear();
7:     lstFontSize.Items.Add("14");
8:     lstFontSize.Items.Add("18");
9:     lstFontSize.Items.Add("22");
10:    lstFontSize.Items.Add("26");
11:    lstFontSize.Items.Add("28");
12:    lstFontSize.Items.Add("30");
13:    lstFontSize.Items.Add("32");
14:    lstFontSize.Items.Add("34");
15:    lstFontSize.Items.Add("36");
```

```

16:     lstFontSize.Items.Add("38");
17:     lstFontSize.Items.Add("40");
18:     lstFontSize.Items.Add("42");
19:     lstFontSize.Items.Add("44");
20:     lstFontSize.Items.Add("64");
21:     lstFontSize.Items.Add("72");
22:     lstFontSize.SelectedIndex = -1;
23: }

```

1.1.2.12.2 FontSizeSettings Fields

1.1.2.12.2.1 FontSizeSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class FontSizeSettings.

1.1.2.12.2.2 FontSizeSettings.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class FontSizeSettings.

1.1.2.12.3 FontSizeSettings Methods

1.1.2.12.3.1 FontSizeSettings.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class FontSizeSettings.

Body Source

```

1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
10:         lblFolderName.Text = noteFolders.NoteFolderName;
11:         lblFontSize.Text = AppResources.SelectFontSize;
12:     }
13:     SetBackgroundColor();
14: }

```

1.1.2.12.3.2 FontSizeSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class FontSizeSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.12.3.3 FontSizeSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class FontSizeSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.13 GeneralSettings Class

Class Hierarchy



C#

```
public class GeneralSettings : PhoneApplicationPage;
```

File

GeneralSettings.xaml.cs ([🔗](#) see page 162)

Description

This is class AwesomeNote.GeneralSettings.

Methods

	Name	Description
🔗	GeneralSettings (🔗 see page 55)	This is GeneralSettings, a member of class GeneralSettings.

GeneralSettings Fields

	Name	Description
🔗	messageBackGround (🔗 see page 57)	This is messageBackGround, a member of class GeneralSettings.
🔗	signIn (🔗 see page 57)	This is signIn, a member of class GeneralSettings.

GeneralSettings Methods

	Name	Description
🔗	CreateDirectoryAsync (🔗 see page 58)	This is CreateDirectoryAsync, a member of class GeneralSettings.
🔗	DesignFileName (🔗 see page 58)	This is DesignFileName, a member of class GeneralSettings.
🔗🔗	OnNavigatedFrom (🔗 see page 59)	This is OnNavigatedFrom, a member of class GeneralSettings.
🔗🔗	OnNavigatedTo (🔗 see page 59)	This is OnNavigatedTo, a member of class GeneralSettings.

1.1.2.13.1 GeneralSettings.GeneralSettings Constructor**C#**

```
public GeneralSettings();
```

Description

This is GeneralSettings, a member of class GeneralSettings.

Body Source

```

1: public GeneralSettings()
2: {
3:     InitializeComponent();
4:     SetBackgroundColor();
5:     InitializePage();
6:     signIn = 0;
7:
8:     //btnOneDrive.Content = AppResources.SignIn;
9:
10:    using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
11:    {
12:        var appSettings = context.AppSettings.First() as AppSettings;
13:        if (appSettings.AppLangName == "EN")
14:        {
15:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.English + ")";
16:        }
17:        if (appSettings.AppLangName == "TR")
18:        {
19:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Turkish + ")";
20:        }
21:        if (appSettings.AppLangName == "DE")
22:        {
23:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.German + ")";
24:        }
25:        if (appSettings.AppLangName == "ES")
26:        {
27:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Spanish + ")";
28:        }
29:
30:        if (appSettings.AppLangName == "PT")
31:        {
32:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Portuguese + ")";
33:        }
34:        if (appSettings.AppLangName == "AR")
35:        {
36:            lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Arabic + ")";
37:        }
38:        if (appSettings.AppLangName == "FA")

```

```

39:         {
40:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Persian + ")";
41:         }
42:         if (appSettings.AppLangName == "IT")
43:         {
44:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Italian + ")";
45:         }
46:         if (appSettings.AppLangName == "FR")
47:         {
48:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.French + ")";
49:         }
50:         if (appSettings.AppLangName == "RU")
51:         {
52:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Russian + ")";
53:         }
54:         if (appSettings.AppLangName == "ZH")
55:         {
56:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Chinese + ")";
57:         }
58:         if (appSettings.AppLangName == "SA")
59:         {
60:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Sanskrit + ")";
61:         }
62:         if (appSettings.AppLangName == "TH")
63:         {
64:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Thai + ")";
65:         }
66:     }
67:     if (appSettings.FolderOrderBy == "NAME")
68:     {
69:         lblFolderOrder.Text = AppResources.FolderOrder + " (" +
AppResources.Selected + ": " + AppResources.FolderName + ")";
70:     }
71:     if (appSettings.FolderOrderBy == "CDATE")
72:     {
73:         lblFolderOrder.Text = AppResources.FolderOrder + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
74:     }
75:     if (appSettings.FolderOrderBy == "MDATE")
76:     {
77:         lblFolderOrder.Text = AppResources.FolderOrder + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
78:     }
79:     if (appSettings.FolderOrderStyle == "A")
80:     {
81:         lblFolderOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
82:     }
83:     if (appSettings.FolderOrderStyle == "D")
84:     {
85:         lblFolderOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
86:     }
87:     if (appSettings.AppBackgroundColor == "BLA")
88:     {
89:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + ")";
90:     }
91:     if (appSettings.AppBackgroundColor == "BLU")
92:     {
93:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + ")";

```

```

94:         }
95:         if (appSettings.AppBackgroundColor == "BRO")
96:         {
97:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + ")";
98:         }
99:         if (appSettings.AppBackgroundColor == "RED")
100:        {
101:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + ")";
102:        }
103:        if (appSettings.AppBackgroundColor == "GRE")
104:        {
105:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + ")";
106:        }
107:        if (appSettings.AppBackgroundColor == "YEL")
108:        {
109:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + ")";
110:        }
111:        if (appSettings.AppBackgroundColor == "GRA")
112:        {
113:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + ")";
114:        }
115:        if (appSettings.AppBackgroundColor == "ORA")
116:        {
117:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + ")";
118:        }
119:        if (appSettings.AppBackgroundColor == "PUR")
120:        {
121:            lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + ")";
122:        }
123:    }
124: }

```

1.1.2.13.2 GeneralSettings Fields

1.1.2.13.2.1 GeneralSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class GeneralSettings.

1.1.2.13.2.2 GeneralSettings.signIn Field

C#

```
public int signIn;
```

Description

This is signIn, a member of class GeneralSettings.

1.1.2.13.3 GeneralSettings Methods

1.1.2.13.3.1 GeneralSettings.CreateDirectoryAsync Method

C#

```
public asyncstatic Task<string> CreateDirectoryAsync(LiveConnectClient client, string
folderName, string parentFolder);
```

Description

This is CreateDirectoryAsync, a member of class GeneralSettings.

Body Source

```
1: public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
2: string folderName, string parentFolder)
3: {
4:     string folderId = null;
5:
6:     // Retrieves all the directories.
7:     var queryFolder = parentFolder + "/files?filter=folders,albums";
8:     var opResult = await client.GetAsync(queryFolder);
9:     dynamic result = opResult.Result;
10:
11:     foreach (dynamic folder in result.data)
12:     {
13:         // Checks if current folder has the passed name.
14:         if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
15:         {
16:             folderId = folder.id;
17:             break;
18:         }
19:     }
20:
21:     if (folderId == null)
22:     {
23:         // Directory hasn't been found, so creates it using the PostAsync
method.
24:         var folderData = new Dictionary<string, object>();
25:         folderData.Add("name", folderName);
26:         opResult = await client.PostAsync(parentFolder, folderData);
27:         result = opResult.Result;
28:
29:         // Retrieves the id of the created folder.
30:         folderId = result.id;
31:     }
32:
33:     return folderId;
34: }
```

1.1.2.13.3.2 GeneralSettings.DesignFileName Method

C#

```
public string DesignFileName(string fileName);
```

Description

This is DesignFileName, a member of class GeneralSettings.

Body Source

```
1: public string DesignFileName(string fileName)
2: {
3:     fileName = fileName.Replace(":", ".");
4:     fileName = fileName.Replace("?", ".");
5:     fileName = fileName.Replace("\\", ".");
6:     fileName = fileName.Replace("/", ".");
7:     fileName = fileName.Replace("<", ".");
8:     fileName = fileName.Replace(">", ".");
9:     fileName = fileName.Replace("|", ".");
```

```
10:     fileName = fileName.Replace("**", ".");
11:     return fileName;
12: }
```

1.1.2.13.3 GeneralSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class GeneralSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.13.4 GeneralSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class GeneralSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     //currentPage = ((PhoneApplicationFrame)Application.Current.RootVisual).Content as
PhoneApplicationPage;
5:     //currentPage.NavigationCacheMode = NavigationCacheMode.Enabled;
6: }
```

1.1.2.14 Language Class

Class Hierarchy

AwesomeNote.Language

C#

```
public class Language;
```

File

Language.cs (📄 see page 174)

Description

This is class AwesomeNote.Language.

Language Properties

	Name	Description
📄	LanguageId (📄 see page 60)	This is LanguageId, a member of class Language.
📄	LanguageName (📄 see page 60)	This is LanguageName, a member of class Language.

1.1.2.14.1 Language Properties

1.1.2.14.1.1 Language.LanguageId Property

C#

```
public int LanguageId;
```

Description

This is LanguageId, a member of class Language.

1.1.2.14.1.2 Language.LanguageName Property

C#

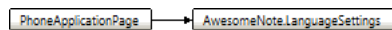
```
public string LanguageName;
```

Description

This is LanguageName, a member of class Language.

1.1.2.15 LanguageSettings Class

Class Hierarchy



C#

```
public class LanguageSettings : PhoneApplicationPage;
```

File

LanguageSettings.xaml.cs ([see page 174](#))

Description

This is class AwesomeNote.LanguageSettings.

Methods

	Name	Description
	LanguageSettings (see page 60)	This is LanguageSettings, a member of class LanguageSettings.

LanguageSettings Fields

	Name	Description
	messageBackGround (see page 61)	This is messageBackGround, a member of class LanguageSettings.

LanguageSettings Methods

	Name	Description
	OnNavigatedFrom (see page 61)	This is OnNavigatedFrom, a member of class LanguageSettings.
	OnNavigatedTo (see page 61)	This is OnNavigatedTo, a member of class LanguageSettings.

1.1.2.15.1 LanguageSettings.LanguageSettings Constructor

C#

```
public LanguageSettings();
```

Description

This is LanguageSettings, a member of class LanguageSettings.

Body Source

```
1: public LanguageSettings()  
2: {  
3:     InitializeComponent();  
4:     SetBackgroundColor();  
5:  
6:     lstLanguage.Items.Clear();  
7:     lstLanguage.Items.Add(AppResources.English);  
8:     lstLanguage.Items.Add(AppResources.Turkish);  
9:     lstLanguage.Items.Add(AppResources.German);  
10:    lstLanguage.Items.Add(AppResources.Spanish);  
11:    lstLanguage.Items.Add(AppResources.Russian);  
12:    lstLanguage.Items.Add(AppResources.Chinese);  
13:    lstLanguage.Items.Add(AppResources.Arabic);  
14:    lstLanguage.Items.Add(AppResources.Persian);  
15:    lstLanguage.Items.Add(AppResources.Italian);  
16:    lstLanguage.Items.Add(AppResources.French);  
17:    lstLanguage.Items.Add(AppResources.Portuguese);  
18:    lstLanguage.Items.Add(AppResources.Sanskrit);  
19:    lstLanguage.Items.Add(AppResources.Thai);  
20:    lstLanguage.SelectedIndex = -1;  
21:    lblLanguage.Text = AppResources.SelectLanguage;  
22: }
```

1.1.2.15.2 LanguageSettings Fields

1.1.2.15.2.1 LanguageSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class LanguageSettings.

1.1.2.15.3 LanguageSettings Methods

1.1.2.15.3.1 LanguageSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class LanguageSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)  
2: {  
3:     base.OnNavigatedFrom(e);  
4: }
```

1.1.2.15.3.2 LanguageSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class LanguageSettings.

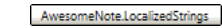
Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.16 LocalizedStrings Class

Provides access to string resources.

Class Hierarchy



C#

```
public class LocalizedStrings;
```

File

LocalizedStrings.cs (🔗 see page 178)

LocalizedStrings Properties

	Name	Description
	LocalizedResources (🔗 see page 62)	This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.16.1 LocalizedStrings Properties

1.1.2.16.1.1 LocalizedStrings.LocalizedResources Property

C#

```
public AppResources LocalizedResources;
```

Description

This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.17 MainPage Class

Class Hierarchy



C#

```
public class MainPage : PhoneApplicationPage;
```

File

MainPage.xaml.cs (🔗 see page 178)





Description

This is class AwesomeNote.MainPage.





Methods

	Name	Description
	MainPage (🔗 see page 63)	Constructor

MainPage Fields

	Name	Description
	messageBackGround ( see page 64)	This is messageBackGround, a member of class MainPage.
	popup ( see page 64)	This is popup, a member of class MainPage.

MainPage Methods

	Name	Description
	OnNavigatedFrom ( see page 64)	This is OnNavigatedFrom, a member of class MainPage.
	OnNavigatedTo ( see page 64)	This is OnNavigatedTo, a member of class MainPage.

1.1.2.17.1 MainPage.MainPage Constructor**C#**

```
public MainPage();
```

Description

Constructor

Body Source

```

1: public MainPage()
2: {
3:     InitializeComponent();
4:     SetBackgroundColor();
5:
6:     ApplicationBar = new ApplicationBar();
7:
8:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();
9:     button1.IconUri = new Uri("/Assets/AddFolder.png", UriKind.Relative);
10:    button1.Text = AppResources.AddFolder;
11:    ApplicationBar.Buttons.Add(button1);
12:    button1.Click += new EventHandler(AddFolderButton_Click);
13:
14:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
15:    button2.IconUri = new Uri("/Assets/Search.png", UriKind.Relative);
16:    button2.Text = AppResources.Search;
17:    ApplicationBar.Buttons.Add(button2);
18:    button2.Click += new EventHandler(SearchButton_Click);
19:
20:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
21:    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
22:    button3.Text = AppResources.Settings;
23:    ApplicationBar.Buttons.Add(button3);
24:    button3.Click += new EventHandler(SettingsButton_Click);
25:
26:    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
27:    button4.IconUri = new Uri("/Assets/About.png", UriKind.Relative);
28:    button4.Text = AppResources.About;
29:    ApplicationBar.Buttons.Add(button4);
30:    button4.Click += new EventHandler(AboutButton_Click);
31:
32:    popup = new Popup();
33:
34:    // Sample code to localize the ApplicationBar
35:    //BuildLocalizedApplicationBar();
36: }
```

1.1.2.17.2 MainPage Fields

1.1.2.17.2.1 MainPage.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class MainPage.

1.1.2.17.2.2 MainPage.popup Field

C#

```
public Popup popup;
```

Description

This is popup, a member of class MainPage.

1.1.2.17.3 MainPage Methods

1.1.2.17.3.1 MainPage.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class MainPage.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4:     //while (NavigationService.CanGoBack)
5:     //NavigationService.RemoveBackEntry();
6:
7: }
```

1.1.2.17.3.2 MainPage.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class MainPage.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     //while (NavigationService.CanGoBack)
5:     //NavigationService.RemoveBackEntry();
6:
7: }
```

1.1.2.18 Note Class

Class Hierarchy

```
AwesomeNote.Note
```

C#

```
[Table]
public class Note;
```









File

Note.cs (see page 184)

Description

This is class AwesomeNote.Note.

Note Properties

	Name	Description
	CreationDate (see page 65)	This is CreationDate, a member of class Note.
	ModificationDate (see page 65)	This is ModificationDate, a member of class Note.
	NameCreation (see page 65)	This is NameCreation, a member of class Note.
	NameDescription (see page 66)	This is NameDescription, a member of class Note.
	NoteDescription (see page 66)	burada notun kendisi NoteDescription oluyor
	NoteFolderId (see page 66)	This is NoteFolderId, a member of class Note.
	NotId (see page 66)	This is NotId, a member of class Note.
	NoteName (see page 66)	This is NoteName, a member of class Note.

1.1.2.18.1 Note Properties**1.1.2.18.1.1 Note.CreationDate Property****C#**

```
[Column]
public DateTime CreationDate;
```

Description

This is CreationDate, a member of class Note.

1.1.2.18.1.2 Note.ModificationDate Property**C#**

```
[Column]
public DateTime ModificationDate;
```

Description

This is ModificationDate, a member of class Note.

1.1.2.18.1.3 Note.NameCreation Property**C#**

```
[Column]
public string NameCreation;
```

Description

This is NameCreation, a member of class Note.

1.1.2.18.1.4 Note.NameDescription Property

C#

```
[Column]
public string NameDescription;
```

Description

This is NameDescription, a member of class Note.

1.1.2.18.1.5 Note.NoteDescription Property

C#

```
[Column]
public string NoteDescription;
```

Description

burada notun kendisi NoteDescription oluyor

1.1.2.18.1.6 Note.NoteFolderId Property

C#

```
[Column(DbType = "INT NOT NULL", CanBeNull = false)]
public int NoteFolderId;
```

Description

This is NoteFolderId, a member of class Note.

1.1.2.18.1.7 Note.NoteId Property

C#

```
[Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
CanBeNull = false)]
public int NoteId;
```

Description

This is NoteId, a member of class Note.

1.1.2.18.1.8 Note.NoteName Property

C#

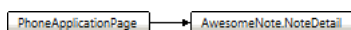
```
[Column]
public string NoteName;
```

Description

This is NoteName, a member of class Note.

1.1.2.19 NoteDetail Class

Class Hierarchy



C#

```
public class NoteDetail : PhoneApplicationPage;
```

File

NoteDetail.xaml.cs (🔗 see page 185)

Description

This is class AwesomeNote.NoteDetail.

Methods

	Name	Description
🔗	NoteDetail (🔗 see page 67)	This is NoteDetail, a member of class NoteDetail.

NoteDetail Fields

	Name	Description
🔗	flag (🔗 see page 68)	This is flag, a member of class NoteDetail.
🔗	messageBackGround (🔗 see page 68)	This is messageBackGround, a member of class NoteDetail.
🔗	noteFolderId (🔗 see page 68)	This is noteFolderId, a member of class NoteDetail.
🔗	notelId (🔗 see page 69)	This is notelId, a member of class NoteDetail.
🔗	pageName (🔗 see page 69)	This is pageName, a member of class NoteDetail.

NoteDetail Methods

	Name	Description
🔗	OnFragmentNavigation (🔗 see page 69)	This is OnFragmentNavigation, a member of class NoteDetail.
🔗	OnNavigatedFrom (🔗 see page 70)	This is OnNavigatedFrom, a member of class NoteDetail.
🔗	OnNavigatedTo (🔗 see page 70)	This is OnNavigatedTo, a member of class NoteDetail.

1.1.2.19.1 NoteDetail.NoteDetail Constructor**C#**

```
public NoteDetail();
```

Description

This is NoteDetail, a member of class NoteDetail.

Body Source

```
1: public NoteDetail()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     ApplicationBar = new ApplicationBar();
7:
8:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();
9:     button1.IconUri = new Uri("/Assets/PreviousNote.png", UriKind.Relative);
10:    button1.Text = AppResources.PreviousNote;
11:    ApplicationBar.Buttons.Add(button1);
12:    button1.Click += new EventHandler(PreviousNoteButton_Click);
13:
14:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
15:    button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
16:    button2.Text = AppResources.SendWithMail;
17:    ApplicationBar.Buttons.Add(button2);
18:    button2.Click += new EventHandler(SendMailButton_Click);
19:
20:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
21:    button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
22:    button3.Text = AppResources.SendWithSMS;
```



```

23:     ApplicationBar.Buttons.Add(button3);
24:     button3.Click += new EventHandler(SendSMSButton_Click);
25:
26:     ApplicationBarIconButton button4 = new ApplicationBarIconButton();
27:     button4.IconUri = new Uri("/Assets/NextNote.png", UriKind.Relative);
28:     button4.Text = AppResources.NextNote;
29:     ApplicationBar.Buttons.Add(button4);
30:     button4.Click += new EventHandler(NextNoteButton_Click);
31:
32:     ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
33:     menuItem1.Text = AppResources.SelectAll;
34:     ApplicationBar.MenuItems.Add(menuItem1);
35:     menuItem1.Click += new EventHandler(SelectAllMenuItem_Click);
36:
37:     ApplicationBarMenuItem menuItem2 = new ApplicationBarMenuItem();
38:     menuItem2.Text = AppResources.Share;
39:     ApplicationBar.MenuItems.Add(menuItem2);
40:     menuItem2.Click += new EventHandler(ShareMenuItem_Click);
41:
42:     ApplicationBarMenuItem menuItem3 = new ApplicationBarMenuItem();
43:     menuItem3.Text = AppResources.ChangeFolder;
44:     ApplicationBar.MenuItems.Add(menuItem3);
45:     menuItem3.Click += new EventHandler(ChangeFolderMenuItem_Click);
46:
47:     ApplicationBarMenuItem menuItem4 = new ApplicationBarMenuItem();
48:     menuItem4.Text = AppResources.DeleteNote;
49:     ApplicationBar.MenuItems.Add(menuItem4);
50:     menuItem4.Click += new EventHandler>DeleteMenuItem_Click);
51:
52:     ApplicationBarMenuItem menuItem5 = new ApplicationBarMenuItem();
53:     menuItem5.Text = AppResources.NoteInfo;
54:     ApplicationBar.MenuItems.Add(menuItem5);
55:     menuItem5.Click += new EventHandler(NoteInfoMenuItem_Click);
56:
57:
58:
59:     popup = new Popup();
60: }

```

1.1.2.19.2 NoteDetail Fields

1.1.2.19.2.1 NoteDetail.flag Field

C#

```
public bool flag;
```

Description

This is flag, a member of class NoteDetail.

1.1.2.19.2.2 NoteDetail.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class NoteDetail.

1.1.2.19.2.3 NoteDetail.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class NoteDetail.

1.1.2.19.2.4 NoteDetail.noteId Field**C#**

```
public int noteId;
```

Description

This is notelId, a member of class NoteDetail.

1.1.2.19.2.5 NoteDetail.pageName Field**C#**

```
public string pageName;
```

Description

This is pageName, a member of class NoteDetail.

1.1.2.19.3 NoteDetail Methods**1.1.2.19.3.1 NoteDetail.OnFragmentNavigation Method****C#**

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class NoteDetail.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Note Id: " + noteFolder.NoteFolderId);
5:     base.OnFragmentNavigation(e);
6:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
7:     {
8:         var note = context.Notes.Where(j => j.NoteId.Equals(e.Fragment)).Single() as
Note;
9:         lblNoteDetail.Text = note.NoteName;
10:        lblCreationDate.Text = AppResources.CreationDate + ": " +
note.CreationDate.ToString();
11:        noteId = note.NoteId;
12:        var noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
13:
14:        if (pageName == "/SearchPage.xaml")
15:        {
16:            lblNoteDetailApp.Text = AppResources.SearchResults;
17:        }
18:        else
19:        {
20:            lblNoteDetailApp.Text = noteFolder.NoteFolderName;
21:        }
22:        noteFolderId = noteFolder.NoteFolderId;
23:        FontFamily temp = new FontFamily(noteFolder.FontFamily);
24:        txtNote.FontFamily = temp;
```

```

25:         txtNote.FontSize = double.Parse(noteFolder.FontSize);
26:         txtNote.Text = note.NoteDescription;
27:         //lstNoteList.DisplayMemberPath = "NoteName";
28:
29:         SetBackgroundColor();
30:     }
31: }

```

1.1.2.19.3.2 NoteDetail.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class NoteDetail.

Body Source

```

1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }

```

1.1.2.19.3.3 NoteDetail.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class NoteDetail.

Body Source

```

1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     var lastPage = NavigationService.BackStack.FirstOrDefault();
5:     pageName = lastPage.Source.ToString();
6:
7:     //txtNote.Focus();
8:     //txtNote.Select(txtNote.Text.Length, 0);
9: }

```

1.1.2.20 NoteFolder Class

Class Hierarchy

AwesomeNote.NoteFolder

C#

```

[Table]
public class NoteFolder;

```


File














NoteFolder.cs (see page 197)

Description

This is class AwesomeNote.NoteFolder.

NoteFolder Properties

	Name	Description
	CreationDate (see page 71)	This is CreationDate, a member of class NoteFolder.

	FolderBackground (see page 71)	This is FolderBackground, a member of class NoteFolder.
	FontFamily (see page 71)	This is FontFamily, a member of class NoteFolder.
	FontSize (see page 71)	This is FontSize, a member of class NoteFolder.
	IsPasswordProtected (see page 72)	This is IsPasswordProtected, a member of class NoteFolder.
	ModificationDate (see page 72)	This is ModificationDate, a member of class NoteFolder.
	NameCount (see page 72)	This is NameCount, a member of class NoteFolder.
	NoteCount (see page 72)	This is NoteCount, a member of class NoteFolder.
	NoteFolderDescription (see page 72)	This is NoteFolderDescription, a member of class NoteFolder.
	NoteFolderId (see page 72)	This is NoteFolderId, a member of class NoteFolder.
	NoteFolderName (see page 73)	This is NoteFolderName, a member of class NoteFolder.
	NoteFolderPassword (see page 73)	This is NoteFolderPassword, a member of class NoteFolder.
	NoteOrderBy (see page 73)	This is NoteOrderBy, a member of class NoteFolder.
	NoteOrderStyle (see page 73)	This is NoteOrderStyle, a member of class NoteFolder.

1.1.2.20.1 NoteFolder Properties

1.1.2.20.1.1 NoteFolder.CreationDate Property

C#

```
[Column]
public DateTime CreationDate;
```

Description

This is CreationDate, a member of class NoteFolder.

1.1.2.20.1.2 NoteFolder.FolderBackground Property

C#

```
[Column (DbType = "Image", UpdateCheck = UpdateCheck.Never)]
public byte FolderBackground;
```

Description

This is FolderBackground, a member of class NoteFolder.

1.1.2.20.1.3 NoteFolder.FontFamily Property

C#

```
[Column]
public string FontFamily;
```

Description

This is FontFamily, a member of class NoteFolder.

1.1.2.20.1.4 NoteFolder.FontSize Property

C#

```
[Column]
public string FontSize;
```

Description

This is FontSize, a member of class NoteFolder.

1.1.2.20.1.5 NoteFolder.IsPasswordProtected Property**C#**

```
[Column]
public bool IsPasswordProtected;
```

Description

This is IsPasswordProtected, a member of class NoteFolder.

1.1.2.20.1.6 NoteFolder.ModificationDate Property**C#**

```
[Column]
public DateTime ModificationDate;
```

Description

This is ModificationDate, a member of class NoteFolder.

1.1.2.20.1.7 NoteFolder.NameCount Property**C#**

```
[Column]
public string NameCount;
```

Description

This is NameCount, a member of class NoteFolder.

1.1.2.20.1.8 NoteFolder.NoteCount Property**C#**

```
[Column]
public int NoteCount;
```

Description

This is NoteCount, a member of class NoteFolder.

1.1.2.20.1.9 NoteFolder.NoteFolderDescription Property**C#**

```
[Column]
public string NoteFolderDescription;
```

Description

This is NoteFolderDescription, a member of class NoteFolder.

1.1.2.20.1.10 NoteFolder.NoteFolderId Property**C#**

```
[Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
CanBeNull = false)]
public int NoteFolderId;
```

Description

This is NoteFolderId, a member of class NoteFolder.

1.1.2.20.1.11 NoteFolder.NoteFolderName Property**C#**

```
[Column]
public string NoteFolderName;
```

Description

This is NoteFolderName, a member of class NoteFolder.

1.1.2.20.1.12 NoteFolder.NoteFolderPassword Property**C#**

```
[Column]
public string NoteFolderPassword;
```

Description

This is NoteFolderPassword, a member of class NoteFolder.

1.1.2.20.1.13 NoteFolder.NoteOrderBy Property**C#**

```
[Column]
public string NoteOrderBy;
```

Description

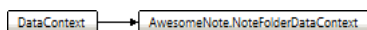
This is NoteOrderBy, a member of class NoteFolder.

1.1.2.20.1.14 NoteFolder.NoteOrderStyle Property**C#**

```
[Column]
public string NoteOrderStyle;
```

Description

This is NoteOrderStyle, a member of class NoteFolder.

1.1.2.21 NoteFolderDataContext Class**Class Hierarchy****C#**

```
public class NoteFolderDataContext : DataContext;
```


File

AwesomeNoteDataContext.cs (see page 135)





Description

This is class AwesomeNote.NoteFolderDataContext.

Methods

	Name	Description
	NoteFolderDataContext (see page 74)	This is NoteFolderDataContext, a member of class NoteFolderDataContext.

NoteFolderDataContext Fields

	Name	Description
	AppSettings (see page 74)	This is AppSettings, a member of class NoteFolderDataContext.
	ConnectionString (see page 74)	This is ConnectionString, a member of class NoteFolderDataContext.
	NoteFolders (see page 74)	This is NoteFolders, a member of class NoteFolderDataContext.
	Notes (see page 75)	This is Notes, a member of class NoteFolderDataContext.

1.1.2.21.1 NoteFolderDataContext.NoteFolderDataContext Constructor

C#

```
public NoteFolderDataContext(string connectionString);
```

Description

This is NoteFolderDataContext, a member of class NoteFolderDataContext.

Body Source

```
1: public NoteFolderDataContext(string connectionString)
2: : base(connectionString) { }
```

1.1.2.21.2 NoteFolderDataContext Fields

1.1.2.21.2.1 NoteFolderDataContext.AppSettings Field

C#

```
public Table<AppSettings> AppSettings;
```

Description

This is AppSettings, a member of class NoteFolderDataContext.

1.1.2.21.2.2 NoteFolderDataContext.ConnectionString Field

C#

```
public const string ConnectionString = @"Data Source=isostore:/NoteAndFolders.sdf";
```

Description

This is ConnectionString, a member of class NoteFolderDataContext.

1.1.2.21.2.3 NoteFolderDataContext.NoteFolders Field

C#

```
public Table<NoteFolder> NoteFolders;
```

Description

This is NoteFolders, a member of class NoteFolderDataContext.

1.1.2.21.2.4 NoteFolderDataContext.Notes Field

C#

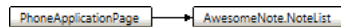
```
public Table<Note> Notes;
```

Description

This is Notes, a member of class NoteFolderDataContext.

1.1.2.22 NoteList Class

Class Hierarchy



C#

```
public class NoteList : PhoneApplicationPage;
```

File

NoteFolder.xaml.cs (see page 198)





Description

This is class AwesomeNote.NoteList.




Methods

	Name	Description
	NoteList (see page 75)	ListnoteListTemplate { set; get; }

NoteList Fields

	Name	Description
	messageBackGround (see page 76)	This is messageBackGround, a member of class NoteList.
	noteFolderId (see page 76)	This is noteFolderId, a member of class NoteList.
	noteld (see page 76)	This is noteld, a member of class NoteList.
	popup (see page 76)	This is popup, a member of class NoteList.

NoteList Methods

	Name	Description
	OnFragmentNavigation (see page 77)	not klasörünün altındaki notlar listelenirken ada göre- oluşturma zamanına göre vs. vs. artan azalan bir şekilde ayarlanıp öyle listeleniyor
	OnNavigatedFrom (see page 78)	This is OnNavigatedFrom, a member of class NoteList.
	OnNavigatedTo (see page 78)	This is OnNavigatedTo, a member of class NoteList.

1.1.2.22.1 NoteList.NoteList Constructor

C#

```
public NoteList();
```

Description

ListnoteListTemplate { set; get; }

Body Source

```
1: public NoteList()
2: {
```



```
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     ApplicationBar = new ApplicationBar();
7:
8:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();
9:     button1.IconUri = new Uri("/Assets/AddNote.png", UriKind.Relative);
10:    button1.Text = AppResources.AddNote;
11:    ApplicationBar.Buttons.Add(button1);
12:    button1.Click += new EventHandler(AddNoteButton_Click);
13:
14:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
15:    button2.IconUri = new Uri("/Assets/DeleteFolder.png", UriKind.Relative);
16:    button2.Text = AppResources.DeleteFolder;
17:    ApplicationBar.Buttons.Add(button2);
18:    button2.Click += new EventHandler>DeleteFolderButton_Click);
19:
20:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
21:    button3.IconUri = new Uri("/Assets/FolderSettings.png", UriKind.Relative);
22:    button3.Text = AppResources.FolderSettings;
23:    ApplicationBar.Buttons.Add(button3);
24:    button3.Click += new EventHandler(FolderSettingsButton_Click);
25:
26:    popup = new Popup();
27:    //lstNoteList.DataContext = noteListTemplate;
28: }
```

1.1.2.22.2 NoteList Fields

1.1.2.22.2.1 NoteList.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class NoteList.

1.1.2.22.2.2 NoteList.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class NoteList.

1.1.2.22.2.3 NoteList.noteId Field

C#

```
public int noteId;
```

Description

This is noteId, a member of class NoteList.

1.1.2.22.2.4 NoteList.popup Field

C#

```
public Popup popup;
```

Description

This is popup, a member of class NoteList.

1.1.2.22.3 NoteList Methods**1.1.2.22.3.1 NoteList.OnFragmentNavigation Method****C#**

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

not klasörünün altındaki notlar listelenirken ada göre- oluşturma zamanına göre vs. vs. artan azalan bir şekilde ayarlanıp böyle listeleniyor

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     List<Note> noteList = new List<Note>();
4:
5:     // displays "Fragment: Detail"
6:     //MessageBox.Show("Folder Id: " + e.Fragment);
7:     base.OnFragmentNavigation(e);
8:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
9:     {
10:         var noteFolder = context.NoteFolders.Where( j =>
j.NoteFolderId.Equals(e.Fragment)).Single() as NoteFolder;
11:         string orderStyle = noteFolder.NoteOrderStyle;
12:         switch (noteFolder.NoteOrderBy)
13:         {
14:             case "NAME":
15:                 if (orderStyle == "A")
16:                 {
17:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.NoteName).ToList() as List<Note>;
18:                 }
19:                 else
20:                 {
21:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.NoteName).ToList() as
List<Note>;
22:                 }
23:                 break;
24:             case "CDATE":
25:                 if (orderStyle == "A")
26:                 {
27:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.CreationDate).ToList() as List<Note>;
28:                 }
29:                 else
30:                 {
31:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.CreationDate).ToList() as
List<Note>;
32:                 }
33:                 break;
34:             case "MDATE":
35:                 if (orderStyle == "A")
36:                 {
37:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.ModificationDate).ToList() as List<Note>;
38:                 }
39:                 else
40:                 {
41:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.ModificationDate).ToList() as
List<Note>;
42:                 }
43:                 break;
44:             default:
45:                 break;
46:         }
47:     }
48: }
```

```

39:             else
40:             {
41:                 noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.ModificationDate).ToList() as
List<Note>;
42:             }
43:             break;
44:             default:
45:                 if (orderStyle == "A")
46:                 {
47:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.NoteName).ToList() as List<Note>;
48:                 }
49:                 else
50:                 {
51:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.NoteName).ToList() as
List<Note>;
52:                 }
53:                 break;
54:             }
55:
56:             lstNoteList.Items.Clear();
57:             noteFolderId = noteFolder.NoteFolderId;
58:             lblNoteFolder.Text = noteFolder.NoteFolderName;
59:             lblNoteList.Text = AppResources.NoteListOf + " (" + noteFolder.NoteFolderName +
")";
60:             lstNoteList.ItemsSource = noteList;
61:             lstNoteList.DisplayMemberPath = "NoteName";
62:             SetBackgroundColor();
63:             //lstNoteList.DisplayMemberPath = "NameCreation";
64:         }
65:     }

```

1.1.2.22.3.2 NoteList.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class NoteList.

Body Source

```

1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }

```

1.1.2.22.3.3 NoteList.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class NoteList.

Body Source

```

1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }

```

1.1.2.23 NoteOrderSettings Class

Class Hierarchy



C#

```
public class NoteOrderSettings : PhoneApplicationPage;
```

File

NoteOrderSettings.xaml.cs ([see page 207](#))

Description

This is class AwesomeNote.NoteOrderSettings.

Methods

	Name	Description
	NoteOrderSettings (see page 79)	This is NoteOrderSettings, a member of class NoteOrderSettings.

NoteOrderSettings Fields

	Name	Description
	messageBackGround (see page 80)	This is messageBackGround, a member of class NoteOrderSettings.
	noteFolderId (see page 80)	This is noteFolderId, a member of class NoteOrderSettings.

NoteOrderSettings Methods

	Name	Description
	OnFragmentNavigation (see page 80)	This is OnFragmentNavigation, a member of class NoteOrderSettings.
	OnNavigatedFrom (see page 80)	This is OnNavigatedFrom, a member of class NoteOrderSettings.
	OnNavigatedTo (see page 81)	This is OnNavigatedTo, a member of class NoteOrderSettings.

1.1.2.23.1 NoteOrderSettings.NoteOrderSettings Constructor

C#

```
public NoteOrderSettings();
```

Description

This is NoteOrderSettings, a member of class NoteOrderSettings.

Body Source

```

1: public NoteOrderSettings()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     lstNoteOrder.Items.Clear();
7:     lstNoteOrder.Items.Add(AppResources.NoteName);
8:     lstNoteOrder.Items.Add(AppResources.CreationDate);
9:     lstNoteOrder.Items.Add(AppResources.ModificationDate);
10:    lstNoteOrder.SelectedIndex = -1;
11: }
```

1.1.2.23.2 NoteOrderSettings Fields

1.1.2.23.2.1 NoteOrderSettings.messageBackGround Field

C#

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class NoteOrderSettings.

1.1.2.23.2.2 NoteOrderSettings.noteFolderId Field

C#

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class NoteOrderSettings.

1.1.2.23.3 NoteOrderSettings Methods

1.1.2.23.3.1 NoteOrderSettings.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class NoteOrderSettings.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
10:         lblFolderName.Text = noteFolders.NoteFolderName;
11:         lblNoteOrder.Text = AppResources.SelectOrderType;
12:     }
13:     SetBackgroundColor();
14: }
```

1.1.2.23.3.2 NoteOrderSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class NoteOrderSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.23.3 NoteOrderSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class NoteOrderSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.24 NoteOrderStyleSettings Class

Class Hierarchy



C#

```
public class NoteOrderStyleSettings : PhoneApplicationPage;
```

File

NoteOrderStyleSettings.xaml.cs (see page 210)

Description

This is class AwesomeNote.NoteOrderStyleSettings.

Methods

	Name	Description
	NoteOrderStyleSettings (see page 81)	This is NoteOrderStyleSettings, a member of class NoteOrderStyleSettings.

NoteOrderStyleSettings Fields

	Name	Description
	messageBackGround (see page 82)	This is messageBackGround, a member of class NoteOrderStyleSettings.
	noteFolderId (see page 82)	This is noteFolderId, a member of class NoteOrderStyleSettings.

NoteOrderStyleSettings Methods

	Name	Description
	OnFragmentNavigation (see page 82)	This is OnFragmentNavigation, a member of class NoteOrderStyleSettings.
	OnNavigatedFrom (see page 83)	This is OnNavigatedFrom, a member of class NoteOrderStyleSettings.
	OnNavigatedTo (see page 83)	This is OnNavigatedTo, a member of class NoteOrderStyleSettings.

1.1.2.24.1 NoteOrderStyleSettings.NoteOrderStyleSettings Constructor

C#

```
public NoteOrderStyleSettings();
```

Description

This is NoteOrderStyleSettings, a member of class NoteOrderStyleSettings.

Body Source

```

1: public NoteOrderStyleSettings()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     lstNoteOrderStyle.Items.Clear();
7:     lstNoteOrderStyle.Items.Add(AppResources.Ascending);
8:     lstNoteOrderStyle.Items.Add(AppResources.Descending);
9:     lstNoteOrderStyle.SelectedIndex = -1;
10:
11: }

```

1.1.2.24.2 NoteOrderStyleSettings Fields**1.1.2.24.2.1 NoteOrderStyleSettings.messageBackGround Field****C#**

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class NoteOrderStyleSettings.

1.1.2.24.2.2 NoteOrderStyleSettings.noteFolderId Field**C#**

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class NoteOrderStyleSettings.

1.1.2.24.3 NoteOrderStyleSettings Methods**1.1.2.24.3.1 NoteOrderStyleSettings.OnFragmentNavigation Method****C#**

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class NoteOrderStyleSettings.

Body Source

```

1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
10:         lblFolderName.Text = noteFolders.NoteFolderName;
11:         lblNoteOrderStyle.Text = AppResources.SelectOrderStyle;
12:     }
13:     SetBackgroundColor();
14: }

```

1.1.2.24.3.2 NoteOrderStyleSettings.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class NoteOrderStyleSettings.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.24.3.3 NoteOrderStyleSettings.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class NoteOrderStyleSettings.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.25 Page1 Class

Class Hierarchy



C#

```
public class Page1 : PhoneApplicationPage;
```

File

NoteInfo.xaml.cs (🔗 see page 205)

Description

This is class AwesomeNote.Page1.

Methods

	Name	Description
🔗	Page1 (🔗 see page 84)	This is Page1, a member of class Page1.

Page1 Fields

	Name	Description
🔗	messageBackGround (🔗 see page 84)	This is messageBackGround, a member of class Page1.
🔗	noteFolderId (🔗 see page 84)	This is noteFolderId, a member of class Page1.
🔗	notelId (🔗 see page 84)	This is notelId, a member of class Page1.

Page1 Methods

	Name	Description
☞	OnFragmentNavigation (☞ see page 85)	This is OnFragmentNavigation, a member of class Page1.
☞	OnNavigatedFrom (☞ see page 85)	This is OnNavigatedFrom, a member of class Page1.
☞	OnNavigatedTo (☞ see page 86)	This is OnNavigatedTo, a member of class Page1.

1.1.2.25.1 Page1.Page1 Constructor**C#**

```
public Page1();
```

Description

This is Page1, a member of class Page1.

Body Source

```
1: public Page1()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5: }
```

1.1.2.25.2 Page1 Fields**1.1.2.25.2.1 Page1.messageBackGround Field****C#**

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class Page1.

1.1.2.25.2.2 Page1.noteFolderId Field**C#**

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class Page1.

1.1.2.25.2.3 Page1.noteId Field**C#**

```
public int noteId;
```

Description

This is noteld, a member of class Page1.

1.1.2.25.3 Page1 Methods

1.1.2.25.3.1 Page1.OnFragmentNavigation Method

C#

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class Page1.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Note Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
7:     {
8:         var note = context.Notes.Where(j => j.NoteId.Equals(e.Fragment)).Single() as
Note;
9:         var noteFolder =
10:             context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
11:         lblNoteName.Text = note.NoteName;
12:         noteId = note.NoteId;
13:         noteFolderId = note.NoteFolderId;
14:         lblNoteInfo.Text = AppResources.About;
15:         StringBuilder sb = new StringBuilder();
16:         string[] wordNumbers;
17:         wordNumbers = note.NoteDescription.Split(' ');
18:
19:         // burada not dosyasi hakkinda istatistiki bilgiler hazirlaniyor
20:         sb.AppendLine(AppResources.BelongFolderName + ": " + noteFolder.NoteFolderName);
21:         sb.AppendLine(AppResources.CharacterNumber + ": " +
note.NoteDescription.Length);
22:         sb.AppendLine(AppResources.WordNumber + ": " + (wordNumbers.Length-1));
23:         sb.AppendLine(AppResources.CreationDate + ": " + note.CreationDate);
24:         sb.AppendLine(AppResources.ModificationDate + ": " + note.ModificationDate);
25:
26:         //var paragraph = new Paragraph();
27:         //paragraph.Inlines.Add(sb.ToString());
28:         //txtNoteInfo.Blocks.Add(paragraph);
29:         txtNoteInfo.Text = sb.ToString();
30:         //lstNoteList.DisplayMemberPath = "NoteName";
31:         SetBackgroundColor();
32:     }
33: }
```

1.1.2.25.3.2 Page1.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class Page1.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }
```

1.1.2.25.3.3 Page1.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

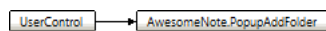
This is OnNavigatedTo, a member of class Page1.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.26 PopupAddFolder Class

Class Hierarchy



C#

```
public class PopupAddFolder : UserControl;
```


File

PopupAddFolder.xaml.cs (see page 213)

Description

This is class AwesomeNote.PopupAddFolder.

Methods

	Name	Description
	PopupAddFolder (see page 86)	This is PopupAddFolder, a member of class PopupAddFolder.

1.1.2.26.1 PopupAddFolder.PopupAddFolder Constructor

C#

```
public PopupAddFolder();
```

Description

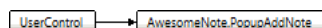
This is PopupAddFolder, a member of class PopupAddFolder.

Body Source

```
1: public PopupAddFolder()
2: {
3:     InitializeComponent();
4:     SetPopupBackgroundColor();
5: }
```

1.1.2.27 PopupAddNote Class

Class Hierarchy



C#

public class PopupAddNote : UserControl;


File

PopupAddNote.xaml.cs (see page 214)

Description

This is class AwesomeNote.PopupAddNote.

Methods

	Name	Description
	PopupAddNote (see page 87)	This is PopupAddNote, a member of class PopupAddNote.

1.1.2.27.1 PopupAddNote.PopupAddNote Constructor

C#

public PopupAddNote();

Description


This is PopupAddNote, a member of class PopupAddNote.

Body Source

1: public PopupAddNote()
2: {
3: InitializeComponent();
4: SetPopupBackgroundColor();
5: }

1.1.2.28 PopupChangeFolder Class

Class Hierarchy



C#

public class PopupChangeFolder : UserControl;


File

PopupChangeFolder.xaml.cs (see page 215)

Description

This is class AwesomeNote.PopupChangeFolder.

Methods

	Name	Description
	PopupChangeFolder (see page 87)	This is PopupChangeFolder, a member of class PopupChangeFolder.

1.1.2.28.1 PopupChangeFolder.PopupChangeFolder Constructor

C#

public PopupChangeFolder();

Description

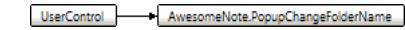
This is PopupChangeFolder, a member of class PopupChangeFolder.

Body Source

```
1: public PopupChangeFolder()  
2: {  
3:     InitializeComponent();  
4: }
```

1.1.2.29 PopupChangeFolderName Class

Class Hierarchy



C#

```
public class PopupChangeFolderName : UserControl;
```

File

PopupChangeFolderName.xaml.cs (see page 216)

Description

This is class AwesomeNote.PopupChangeFolderName.

Methods

	Name	Description
	PopupChangeFolderName (see page 88)	This is PopupChangeFolderName, a member of class PopupChangeFolderName.

1.1.2.29.1 PopupChangeFolderName.PopupChangeFolderName Constructor

C#

```
public PopupChangeFolderName();
```

Description

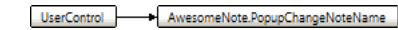
This is PopupChangeFolderName, a member of class PopupChangeFolderName.

Body Source

```
1: public PopupChangeFolderName()  
2: {  
3:     InitializeComponent();  
4:     SetPopupBackgroundColor();  
5: }
```

1.1.2.30 PopupChangeNoteName Class

Class Hierarchy



C#

```
public class PopupChangeNoteName : UserControl;
```



File

PopupChangeNoteName.xaml.cs (see page 217)

Description

This is class AwesomeNote.PopupChangeNoteName.

Methods

	Name	Description
	PopupChangeNoteName ( see page 89)	This is PopupChangeNoteName, a member of class PopupChangeNoteName.

1.1.2.30.1 PopupChangeNoteName.PopupChangeNoteName Constructor

C#

```
public PopupChangeNoteName ( ) ;
```

Description

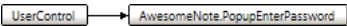
This is PopupChangeNoteName, a member of class PopupChangeNoteName.

Body Source

```
1: public PopupChangeNoteName ( )
2: {
3:     InitializeComponent ( ) ;
4:     SetPopupBackgroundColor ( ) ;
5: }
```

1.1.2.31 PopupEnterPassword Class


Class Hierarchy



C#

```
public class PopupEnterPassword : UserControl;
```



File

[PopupEnterPassword.xaml.cs](#) ( see page 218)

Description

This is class AwesomeNote.PopupEnterPassword.

Methods

	Name	Description
	PopupEnterPassword ( see page 89)	This is PopupEnterPassword, a member of class PopupEnterPassword.

1.1.2.31.1 PopupEnterPassword.PopupEnterPassword Constructor

C#

```
public PopupEnterPassword ( ) ;
```

Description

This is PopupEnterPassword, a member of class PopupEnterPassword.

Body Source

```
1: public PopupEnterPassword ( )
2: {
3:     InitializeComponent ( ) ;
4: }
```

1.1.2.32 PopupSetPassword Class

Class Hierarchy



C#

```
public class PopupSetPassword : UserControl;
```

File

PopupSetPassword.xaml.cs (🔗 see page 219)

Description

This is class AwesomeNote.PopupSetPassword.

Methods

	Name	Description
🔗	PopupSetPassword (🔗 see page 90)	This is PopupSetPassword, a member of class PopupSetPassword.

1.1.2.32.1 PopupSetPassword.PopupSetPassword Constructor

C#

```
public PopupSetPassword();
```

Description

This is PopupSetPassword, a member of class PopupSetPassword.

Body Source

```
1: public PopupSetPassword()
2: {
3:     InitializeComponent();
4: }
```

1.1.2.33 SearchPage Class

Class Hierarchy



C#

```
public class SearchPage : PhoneApplicationPage;
```

File

SearchPage.xaml.cs (🔗 see page 219)

Description

This is class AwesomeNote.SearchPage.



Methods

	Name	Description
🔗	SearchPage (🔗 see page 91)	This is SearchPage, a member of class SearchPage.

SearchPage Fields

	Name	Description
	messageBackGround (🔗 see page 91)	This is messageBackGround, a member of class SearchPage.

SearchPage Methods

	Name	Description
	OnNavigatedFrom (🔗 see page 91)	This is OnNavigatedFrom, a member of class SearchPage.
	OnNavigatedTo (🔗 see page 92)	This is OnNavigatedTo, a member of class SearchPage.

1.1.2.33.1 SearchPage.SearchPage Constructor**C#**

```
public SearchPage();
```

Description

This is SearchPage, a member of class SearchPage.

Body Source

```
1: public SearchPage()
2: {
3:     InitializeComponent();
4:     SetBackgroundColor();
5:
6:     txtSearchResult.Text = AppResources.SearchResults;
7:     lblSearch.Text = AppResources.Search;
8:     //btnSearch.Content = AppResources.Search;
9:     //lstSearch.SelectedIndex = -1;
10: }
```

1.1.2.33.2 SearchPage Fields**1.1.2.33.2.1 SearchPage.messageBackGround Field****C#**

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class SearchPage.

1.1.2.33.3 SearchPage Methods**1.1.2.33.3.1 SearchPage.OnNavigatedFrom Method****C#**

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class SearchPage.

Body Source

```
1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
```



```
4: }
```

1.1.2.33.3.2 SearchPage.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class SearchPage.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4: }
```

1.1.2.34 SetPasswordPage Class

Class Hierarchy



C#

```
public class SetPasswordPage : PhoneApplicationPage;
```

File

SetPasswordPage.xaml.cs ([see page 224](#))

Description

This is class AwesomeNote.SetPasswordPage.

Methods

	Name	Description
	SetPasswordPage (see page 92)	This is SetPasswordPage, a member of class SetPasswordPage.

SetPasswordPage Fields

	Name	Description
	messageBackGround (see page 93)	This is messageBackGround, a member of class SetPasswordPage.
	noteFolderId (see page 93)	This is noteFolderId, a member of class SetPasswordPage.

SetPasswordPage Methods

	Name	Description
	OnFragmentNavigation (see page 93)	This is OnFragmentNavigation, a member of class SetPasswordPage.
	OnNavigatedFrom (see page 94)	This is OnNavigatedFrom, a member of class SetPasswordPage.
	OnNavigatedTo (see page 94)	This is OnNavigatedTo, a member of class SetPasswordPage.

1.1.2.34.1 SetPasswordPage.SetPasswordPage Constructor

C#

```
public SetPasswordPage();
```

Description

This is SetPasswordPage, a member of class SetPasswordPage.

Body Source

```

1: public SetPasswordPage()
2: {
3:     InitializeComponent();
4:     //SetBackgroundColor();
5:
6:     //txtFirstPassword.Focus();
7: }

```

1.1.2.34.2 SetPasswordPage Fields**1.1.2.34.2.1 SetPasswordPage.messageBackGround Field****C#**

```
public SolidColorBrush messageBackGround;
```

Description

This is messageBackGround, a member of class SetPasswordPage.

1.1.2.34.2.2 SetPasswordPage.noteFolderId Field**C#**

```
public int noteFolderId;
```

Description

This is noteFolderId, a member of class SetPasswordPage.

1.1.2.34.3 SetPasswordPage Methods**1.1.2.34.3.1 SetPasswordPage.OnFragmentNavigation Method****C#**

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e);
```

Description

This is OnFragmentNavigation, a member of class SetPasswordPage.

Body Source

```

1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     // displays "Fragment: Detail"
4:     //MessageBox.Show("Folder Id: " + e.Fragment);
5:     base.OnFragmentNavigation(e);
6:     noteFolderId = int.Parse(e.Fragment);
7:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
8:     {
9:         var noteFolders =
10:             context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
11:         lblFolderName.Text = noteFolders.NoteFolderName;
12:         lblSetPassword.Text = AppResources.SetPassword;
13:         txtPass1.Text = AppResources.PleaseEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
14:         txtPass2.Text = AppResources.PleaseReEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
15:         //btnCancel.Content = AppResources.Cancel;
16:         //btnOK.Content = AppResources.OK;

```

```

17:     }
18:     SetBackgroundColor();
19: }

```

1.1.2.34.3.2 SetPasswordPage.OnNavigatedFrom Method

C#

```
protected override void OnNavigatedFrom(NavigationEventArgs e);
```

Description

This is OnNavigatedFrom, a member of class SetPasswordPage.

Body Source

```

1: protected override void OnNavigatedFrom(NavigationEventArgs e)
2: {
3:     base.OnNavigatedFrom(e);
4: }

```

1.1.2.34.3.3 SetPasswordPage.OnNavigatedTo Method

C#

```
protected override void OnNavigatedTo(NavigationEventArgs e);
```

Description

This is OnNavigatedTo, a member of class SetPasswordPage.

Body Source

```

1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     txtFirstPassword.Focus();
5: }

```

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AboutPage.xaml.cs (see page 95)	This is file AboutPage.xaml.cs.
App.xaml.cs (see page 98)	This is file App.xaml.cs.
AppResources.Designer.cs (see page 107)	This code was generated by a tool. Runtime Version:4.0.30319.34014 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
AppSettings.cs (see page 133)	This is file AppSettings.cs.
AssemblyInfo.cs (see page 133)	This is file AssemblyInfo.cs.
AutoScrollBehavior.cs (see page 134)	This is file AutoScrollBehavior.cs.
Awesome Note.csproj (see page 135)	This is file Awesome Note.csproj.
AwesomeNote.sln (see page 135)	This is file AwesomeNote.sln.
AwesomeNoteDataContext.cs (see page 135)	This is file AwesomeNoteDataContext.cs.
BackgroundColorSettings.xaml.cs (see page 136)	This is file BackgroundColorSettings.xaml.cs.
ChangeNoteFolder.xaml.cs (see page 139)	This is file ChangeNoteFolder.xaml.cs.

EnterPasswordPage.xaml.cs (see page 141)	This is file EnterPasswordPage.xaml.cs.
FolderOrderSettings.xaml.cs (see page 145)	This is file FolderOrderSettings.xaml.cs.
FolderOrderStyleSettings.xaml.cs (see page 148)	This is file FolderOrderStyleSettings.xaml.cs.
FolderSettings.xaml.cs (see page 151)	This is file FolderSettings.xaml.cs.
FontFamilySettings.xaml.cs (see page 157)	This is file FontFamilySettings.xaml.cs.
FontSizeSettings.xaml.cs (see page 160)	This is file FontSizeSettings.xaml.cs.
GeneralSettings.xaml.cs (see page 162)	This is file GeneralSettings.xaml.cs.
Language.cs (see page 174)	This is file Language.cs.
LanguageSettings.xaml.cs (see page 174)	This is file LanguageSettings.xaml.cs.
LocalizedStrings.cs (see page 178)	This is file LocalizedStrings.cs.
MainPage.xaml.cs (see page 178)	This is file MainPage.xaml.cs.
Note.cs (see page 184)	This is file Note.cs.
NoteDetail.xaml.cs (see page 185)	This is file NoteDetail.xaml.cs.
NoteFolder.cs (see page 197)	This is file NoteFolder.cs.
NoteFolder.xaml.cs (see page 198)	This is file NoteFolder.xaml.cs.
NoteInfo.xaml.cs (see page 205)	This is file NoteInfo.xaml.cs.
NoteOrderSettings.xaml.cs (see page 207)	This is file NoteOrderSettings.xaml.cs.
NoteOrderStyleSettings.xaml.cs (see page 210)	This is file NoteOrderStyleSettings.xaml.cs.
PopupAddFolder.xaml.cs (see page 213)	This is file PopupAddFolder.xaml.cs.
PopupAddNote.xaml.cs (see page 214)	This is file PopupAddNote.xaml.cs.
PopupChangeFolder.xaml.cs (see page 215)	This is file PopupChangeFolder.xaml.cs.
PopupChangeFolderName.xaml.cs (see page 216)	This is file PopupChangeFolderName.xaml.cs.
PopupChangeNoteName.xaml.cs (see page 217)	This is file PopupChangeNoteName.xaml.cs.
PopupEnterPassword.xaml.cs (see page 218)	This is file PopupEnterPassword.xaml.cs.
PopupSetPassword.xaml.cs (see page 219)	This is file PopupSetPassword.xaml.cs.
SearchPage.xaml.cs (see page 219)	This is file SearchPage.xaml.cs.
SetPasswordPage.xaml.cs (see page 224)	This is file SetPasswordPage.xaml.cs.

1.2.1 AboutPage.xaml.cs

This is file AboutPage.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Text;
7: using System.Windows;
8: using System.Windows.Controls;
9: using System.Windows.Documents;
10: using System.Windows.Media;
11: using System.Windows.Media.Imaging;
12: using System.Windows.Navigation;
13: using AwesomeNote.Resources;
14: using Microsoft.Phone.Controls;

```

```

15: using Microsoft.Phone.Shell;
16: using Microsoft.Phone.Tasks;
17: using Microsoft.Xna.Framework;
18:
19: namespace AwesomeNote
20: {
21:     public partial class AboutPage : PhoneApplicationPage
22:     {
23:         public SolidColorBrush messageBackGround;
24:         public AboutPage()
25:         {
26:             InitializeComponent();
27:             SetBackgroundColor();
28:
29:             ApplicationBar = new ApplicationBar();
30:
31:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
32:             button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
33:             button2.Text = AppResources.ContactWithUs;
34:             ApplicationBar.Buttons.Add(button2);
35:             button2.Click += new EventHandler(SendMailButton_Click);
36:
37:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
38:             button3.IconUri = new Uri("/Assets/Rate.png", UriKind.Relative);
39:             button3.Text = AppResources.Rate;
40:             ApplicationBar.Buttons.Add(button3);
41:             button3.Click += new EventHandler(RateButton_Click);
42:
43:             lblAboutTheApp.Text = AppResources.AboutTheApp;
44:             //txtAbout2.Text = AppResources.AboutTheAppText;
45:             //var paragraph = new Paragraph();
46:             //paragraph.Inlines.Add(AppResources.AboutTheAppText);
47:             //txtAbout.Blocks.Add(paragraph);
48:             txtAbout.Text = AppResources.AboutTheAppText;
49:             //txtAbout.IsEnabled = false;
50:             txtAbout.IsReadOnly = true;
51:             //this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
52:         }
53:
54:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
55:         {
56:             if (this.NavigationService.CanGoBack)
57:             {
58:                 this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
59:             }
60:         }
61:
62:         private void SendMailButton_Click(object sender, EventArgs e)
63:         {
64:             // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
65:             //EmailAddressChooserTask emailAddressChooserTask;
66:             //emailAddressChooserTask = new EmailAddressChooserTask();
67:             //emailAddressChooserTask.Completed += new
EventHandler<EmailResult>(emailAddressChooserTask_Completed);
68:             //emailAddressChooserTask.Show();
69:             StringBuilder sb = new StringBuilder();
70:             EmailComposeTask emailComposeTask = new EmailComposeTask();
71:
72:
73:             sb.AppendLine();
74:             sb.AppendLine();
75:             sb.AppendLine(AppResources.SendWithApp);
76:
77:             emailComposeTask.Subject = AppResources.AboutTheAwesomeNote;
78:             emailComposeTask.Body = sb.ToString();
79:             emailComposeTask.To = "coderserdar@outlook.com";
80:             emailComposeTask.Cc = "";

```

```

81:         emailComposeTask.Bcc = "";
82:
83:         emailComposeTask.Show();
84:         //MessageBox.Show(AppResources.SuccessfulSendWithMail);
85:     }
86:
87:     private void RateButton_Click(object sender, EventArgs e)
88:     {
89:         MarketplaceReviewTask marketplaceReviewTask = new MarketplaceReviewTask();
90:         marketplaceReviewTask.Show();
91:     }
92:
93:     private void SetBackgroundColor()
94:     {
95:         AppSettings appSettings = new AppSettings();
96:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
97:         {
98:             appSettings = context.AppSettings.First() as AppSettings;
99:         }
100:
101:         if (appSettings.AppBackgroundImage != null)
102:         {
103:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
104:             BitmapImage image = new BitmapImage();
105:             image.SetSource(stream);
106:             ImageBrush ib = new ImageBrush();
107:             ib.ImageSource = image;
108:             this.LayoutRoot.Background = ib;
109:         }
110:         else
111:         {
112:             switch (appSettings.AppBackgroundColor)
113:             {
114:                 case "BLA":
115:                     messageBackGround = new SolidColorBrush(Colors.Black);
116:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
117:                     break;
118:                 case "BLU":
119:                     messageBackGround = new SolidColorBrush(Colors.Blue);
120:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
121:                     break;
122:                 case "BRO":
123:                     messageBackGround = new SolidColorBrush(Colors.Brown);
124:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
125:                     break;
126:                 case "RED":
127:                     messageBackGround = new SolidColorBrush(Colors.Red);
128:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
129:                     break;
130:                 case "GRE":
131:                     messageBackGround = new SolidColorBrush(Colors.Green);
132:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
133:                     break;
134:                 case "GRA":
135:                     messageBackGround = new SolidColorBrush(Colors.Gray);
136:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
137:                     break;
138:                 case "YEL":
139:                     messageBackGround = new SolidColorBrush(Colors.Yellow);
140:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
141:                     break;
142:                 case "ORA":
143:                     messageBackGround = new SolidColorBrush(Colors.Orange);
144:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
145:                     break;
146:                 case "PUR":

```

```
147:         messageBackGround = new SolidColorBrush(Colors.Purple);
148:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
149:         break;
150:     default:
151:         messageBackGround = new SolidColorBrush(Colors.Black);
152:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
153:         break;
154:     }
155: }
156: }
157: }
158: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.2 App.xaml.cs

This is file App.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Diagnostics;
3: using System.Globalization;
4: using System.Linq;
5: using System.Resources;
6: using System.Threading;
7: using System.Windows;
8: using System.Windows.Markup;
9: using System.Windows.Navigation;
10: using Microsoft.Phone.Controls;
11: using Microsoft.Phone.Shell;
12: using AwesomeNote.Resources;
13: using System.Collections.Generic;
14: using Microsoft.Phone.Marketplace;
15:
16: namespace AwesomeNote
17: {
18:     public partial class App : Application
19:     {
20:         /// <summary>
21:         /// Provides easy access to the root frame of the Phone Application.
22:         /// </summary>
23:         /// <returns>The root frame of the Phone Application.</returns>
24:         public static PhoneApplicationFrame RootFrame { get; private set; }
25:
26:         private static LicenseInformation _licenseInfo = new LicenseInformation();
27:         public static bool IsTrial
28:         {
29:             get;
30:             // setting the IsTrial property from outside is not allowed
31:             private set;
32:         }
33:
34:         private void DetermineIsTrail()
35:         {
36:             #if TRIAL
37:                 // return true if debugging with trial enabled (DebugTrial configuration
is active)
38:                 IsTrial = true;
39:             #else
```

```

40:         var license = new Microsoft.Phone.Marketplace.LicenseInformation();
41:         IsTrial = license.IsTrial();
42:     #endif
43:     }
44:
45:     /// <summary>
46:     /// Constructor for the Application object.
47:     /// </summary>
48:     public App()
49:     {
50:         // Global handler for uncaught exceptions.
51:         UnhandledException += Application_UnhandledException;
52:
53:         // Standard XAML initialization
54:         InitializeComponent();
55:
56:         ThemeManager.ToDarkTheme();
57:
58:         // Phone-specific initialization
59:         InitializePhoneApplication();
60:
61:         // Language display initialization
62:         InitializeLanguage();
63:
64:         // Show graphics profiling information while debugging.
65:         if (Debugger.IsAttached)
66:         {
67:             // Display the current frame rate counters.
68:             Application.Current.Host.Settings.EnableFrameRateCounter = true;
69:
70:             // Show the areas of the app that are being redrawn in each frame.
71:             //Application.Current.Host.Settings.EnableRedrawRegions = true;
72:
73:             // Enable non-production analysis visualization mode,
74:             // which shows areas of a page that are handed off to GPU with a
colored overlay.
75:             //Application.Current.Host.Settings.EnableCacheVisualization = true;
76:
77:             // Prevent the screen from turning off while under the debugger by
disabling
78:             // the application's idle detection.
79:             // Caution:- Use this under debug mode only. Application that disables
user idle detection will continue to run
80:             // and consume battery power when the user is not using the phone.
81:             PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
82:         }
83:
84:     }
85:
86:     // Code to execute when the application is launching (eg, from Start)
87:     // This code will not execute when the application is reactivated
88:     private void Application_Launching(object sender, LaunchingEventArgs e)
89:     {
90:         //DetermineIsTrail();
91:     }
92:
93:     // Code to execute when the application is activated (brought to foreground)
94:     // This code will not execute when the application is first launched
95:     private void Application_Activated(object sender, ActivatedEventArgs e)
96:     {
97:         //DetermineIsTrail();
98:     }
99:
100:    // Code to execute when the application is deactivated (sent to background)
101:    // This code will not execute when the application is closing
102:    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
103:    {
104:    }

```



```

105:
106:     // Code to execute when the application is closing (eg, user hit Back)
107:     // This code will not execute when the application is deactivated
108:     private void Application_Closing(object sender, ClosingEventArgs e)
109:     {
110:     }
111:
112:     // Code to execute if a navigation fails
113:     private void RootFrame_NavigationFailed(object sender,
114:     NavigationFailedEventArgs e)
115:     {
116:         if (Debugger.IsAttached)
117:         {
118:             // A navigation has failed; break into the debugger
119:             Debugger.Break();
120:         }
121:
122:     // Code to execute on Unhandled Exceptions
123:     private void Application_UnhandledException(object sender,
124:     ApplicationUnhandledExceptionEventArgs e)
125:     {
126:         if (Debugger.IsAttached)
127:         {
128:             // An unhandled exception has occurred; break into the debugger
129:             Debugger.Break();
130:         }
131:
132:     #region Phone application initialization
133:
134:     // Avoid double-initialization
135:     private bool phoneApplicationInitialized = false;
136:
137:     // Do not add any additional code to this method
138:     private void InitializePhoneApplication()
139:     {
140:         if (phoneApplicationInitialized)
141:             return;
142:
143:         // Create the frame but don't set it as RootVisual yet; this allows the
144:         splash // screen to remain active until the application is ready to render.
145:         RootFrame = new PhoneApplicationFrame();
146:         RootFrame.Navigated += CompleteInitializePhoneApplication;
147:
148:         // Handle navigation failures
149:         RootFrame.NavigationFailed += RootFrame_NavigationFailed;
150:
151:         // Handle reset requests for clearing the backstack
152:         RootFrame.Navigated += CheckForResetNavigation;
153:
154:         // Ensure we don't initialize again
155:         phoneApplicationInitialized = true;
156:     }
157:
158:     // Do not add any additional code to this method
159:     private void CompleteInitializePhoneApplication(object sender,
160:     NavigationEventArgs e)
161:     {
162:         // Set the root visual to allow the application to render
163:         if (RootVisual != RootFrame)
164:             RootVisual = RootFrame;
165:
166:         // Remove this handler since it is no longer needed
167:         RootFrame.Navigated -= CompleteInitializePhoneApplication;
168:     }
169:
170:     private void CheckForResetNavigation(object sender, NavigationEventArgs e)

```

```

170:         {
171:             // If the app has received a 'reset' navigation, then we need to check
172:             // on the next navigation to see if the page stack should be reset
173:             if (e.NavigationMode == NavigationMode.Reset)
174:                 RootFrame.Navigated += ClearBackStackAfterReset;
175:         }
176:
177:         private void ClearBackStackAfterReset(object sender, NavigationEventArgs e)
178:         {
179:             // Unregister the event so it doesn't get called again
180:             RootFrame.Navigated -= ClearBackStackAfterReset;
181:
182:             // Only clear the stack for 'new' (forward) and 'refresh' navigations
183:             if (e.NavigationMode != NavigationMode.New && e.NavigationMode !=
NavigationMode.Refresh)
184:                 return;
185:
186:             // For UI consistency, clear the entire page stack
187:             while (RootFrame.RemoveBackEntry() != null)
188:             {
189:                 ; // do nothing
190:             }
191:         }
192:
193:         #endregion
194:
195:         // Initialize the app's font and flow direction as defined in its localized
resource strings.
196:         //
197:         // To ensure that the font of your application is aligned with its supported
languages and that the
198:         // FlowDirection for each of those languages follows its traditional
direction, ResourceLanguage
199:         // and ResourceFlowDirection should be initialized in each resx file to match
these values with that
200:         // file's culture. For example:
201:         //
202:         // AppResources.es-ES.resx
203:         //     ResourceLanguage's value should be "es-ES"
204:         //     ResourceFlowDirection's value should be "LeftToRight"
205:         //
206:         // AppResources.ar-SA.resx
207:         //     ResourceLanguage's value should be "ar-SA"
208:         //     ResourceFlowDirection's value should be "RightToLeft"
209:         //
210:         // For more info on localizing Windows Phone apps see
http://go.microsoft.com/fwlink/?LinkId=262072.
211:         //
212:         private void InitializeLanguage()
213:         {
214:             try
215:             {
216:                 // Set the font to match the display language defined by the
217:                 // ResourceLanguage resource string for each supported language.
218:                 //
219:                 // Fall back to the font of the neutral language if the Display
220:                 // language of the phone is not supported.
221:                 //
222:                 // If a compiler error is hit then ResourceLanguage is missing from
223:                 // the resource file.
224:                 RootFrame.Language =
XmlLanguage.GetLanguage(AppResources.ResourceLanguage);
225:
226:                 // Set the FlowDirection of all elements under the root frame based
227:                 // on the ResourceFlowDirection resource string for each
228:                 // supported language.
229:                 //
230:                 // If a compiler error is hit then ResourceFlowDirection is missing
from

```

```

231:                // the resource file.
232:                FlowDirection flow = (FlowDirection)Enum.Parse(typeof(FlowDirection),
AppResources.ResourceFlowDirection);
233:                RootFrame.FlowDirection = flow;
234:            }
235:            catch
236:            {
237:                // If an exception is caught here it is most likely due to either
238:                // ResourceLangauge not being correctly set to a supported language
239:                // code or ResourceFlowDirection is set to a value other than
LeftToRight
240:                // or RightToLeft.
241:
242:                if (Debugger.IsAttached)
243:                {
244:                    Debugger.Break();
245:                }
246:
247:                throw;
248:            }
249:        }
250:
251:        private void Application_Startup(object sender, StartupEventArgs e)
252:        {
253:            int noteFolderId;
254:            using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
255:            {
256:                if (!context.DatabaseExists())
257:                {
258:                    context.CreateDatabase();
259:                    DilAyariOlustur(context);
260:                }
261:                else
262:                {
263:                    using (var context2 = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
264:                    {
265:                        // burada dil ayarlarından ötürü bilgilendirici not
klasörlerini siliyoruz
266:                        var noteFolders = context2.NoteFolders.Where(j =>
j.NoteFolderDescription.Equals("AwesomeNote")).ToList() as List<NoteFolder>;
267:                        for (int i = 0; i < noteFolders.Count; i++)
268:                        {
269:                            var notes = context2.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolders[i].NoteFolderId)).ToList() as List<Note>;
270:                            context2.Notes.DeleteAllOnSubmit(notes);
271:                            context2.SubmitChanges();
272:                        }
273:                        context2.NoteFolders.DeleteAllOnSubmit(noteFolders);
274:                        context2.SubmitChanges();
275:
276:                        AppSettings lang =
277:                            context2.AppSettings.First() as AppSettings;
278:                        string culture = "";
279:                        switch (lang.AppLangName)
280:                        {
281:                            case "TR":
282:                                culture = "tr";
283:                                break;
284:                            case "EN":
285:                                culture = "en";
286:                                break;
287:                            case "DE":
288:                                culture = "de";
289:                                break;
290:                            case "ES":
291:                                culture = "es";
292:                                break;

```

```

293:         case "FR":
294:             culture = "fr";
295:             break;
296:         case "IT":
297:             culture = "it";
298:             break;
299:         case "AR":
300:             culture = "ar";
301:             break;
302:         case "FA":
303:             culture = "fa-IR";
304:             break;
305:         case "ZH":
306:             culture = "zh";
307:             break;
308:         case "PT":
309:             culture = "pt";
310:             break;
311:         case "RU":
312:             culture = "ru";
313:             break;
314:         case "SA":
315:             culture = "sa";
316:             break;
317:         case "TH":
318:             culture = "th";
319:             break;
320:         default:
321:             culture = "tr-TR";
322:             break;
323:     }
324:     CultureInfo newCulture = new CultureInfo(culture);
325:     Thread.CurrentThread.CurrentCulture = newCulture;
326:     Thread.CurrentThread.CurrentUICulture = newCulture;
327:     }
328: }
329:
330: // kullanıcının programla ilgili bilgilendirici notları kendi dilinde
331: // görebilmesi için burada ekliyoruz.
332: WelcomeNotKlasoruOlustur(context);
333: ProgramHakkindaBilgiKlasoruOlustur(context);
334: }
335:
336: private static void DilAyariOlustur(NoteFolderDataContext context)
337: {
338:     var appLangSettings = new AppSettings()
339:     {
340:         //AppLangId = 42,
341:         AppLangName = "EN",
342:         AppBackgroundColor = "BLA",
343:         FolderOrderBy = "CDATE",
344:         FolderOrderStyle = "A",
345:         AppBackgroundImage = null
346:     };
347:
348:     context.AppSettings.InsertOnSubmit(appLangSettings);
349:     context.SubmitChanges();
350:
351:     CultureInfo newCulture = new CultureInfo("en-US");
352:     Thread.CurrentThread.CurrentCulture = newCulture;
353:     Thread.CurrentThread.CurrentUICulture = newCulture;
354: }
355:
356: private static void WelcomeNotKlasoruOlustur(NoteFolderDataContext context)
357: {
358:     int noteFolderId;
359:     var noteFolder = new NoteFolder()
360:     {

```

```

361:         NoteFolderName = AppResources.Welcome,
362:         NoteFolderDescription = "AwesomeNote",
363:         NoteFolderPassword = "",
364:         IsPasswordProtected = false,
365:         FontFamily = "Verdana",
366:         FontSize = "26",
367:         NoteOrderBy = "NAME",
368:         NoteOrderStyle = "A",
369:         CreationDate = DateTime.Now,
370:         ModificationDate = DateTime.Now,
371:         NoteCount = 1,
372:         NameCount = AppResources.Welcome + " (1)",
373:         FolderBackground = null
374:         //FontWeight = "Normal",
375:         //FontStyle = "Normal"
376:     };
377:     context.NoteFolders.InsertOnSubmit(noteFolder);
378:     // commit the changes to the database
379:     context.SubmitChanges();
380:     noteFolderId = context.NoteFolders.First().NoteFolderId;
381:     var note = new Note()
382:     {
383:         CreationDate = DateTime.Now,
384:         ModificationDate = DateTime.Now,
385:         NoteFolderId = noteFolderId,
386:         NoteName = AppResources.Welcome,
387:         NameCreation = AppResources.Welcome + " (" + DateTime.Now.ToString() +
388:         ")",
389:         NoteDescription =
390:             AppResources.WelcomeNote,
391:         NameDescription = AppResources.Welcome + " " + AppResources.WelcomeNote
392:             //NameDescriptionWithoutNewline = "Note " + j.ToString() + " " +
393:             j.ToString() + ". note of folder and " + (i * 4 + j).ToString() + ". note of general"
394:     };
395:     // place the object in pending insert state
396:     context.Notes.InsertOnSubmit(note);
397:     // commit the changes to the database
398:     context.SubmitChanges();
399:     //return noteFolderId;
400:     }
401:     private static void ProgramHakkindaBilgiKlasoruOlustur(NoteFolderDataContext
context)
402:     {
403:         int noteFolderId;
404:         var noteFolder = new NoteFolder()
405:         {
406:             NoteFolderName = AppResources.Tips,
407:             NoteFolderDescription = "AwesomeNote",
408:             NoteFolderPassword = "",
409:             IsPasswordProtected = false,
410:             FontFamily = "Verdana",
411:             FontSize = "26",
412:             NoteOrderBy = "CDATE",
413:             NoteOrderStyle = "A",
414:             CreationDate = DateTime.Now,
415:             ModificationDate = DateTime.Now,
416:             NoteCount = 10,
417:             NameCount = AppResources.Tips + " (10)",
418:             FolderBackground = null
419:             //FontWeight = "Normal",
420:             //FontStyle = "Normal"
421:         };
422:         context.NoteFolders.InsertOnSubmit(noteFolder);
423:         // commit the changes to the database
424:         context.SubmitChanges();
425:         var noteFolder2 = context.NoteFolders.Where(j =>
j.NoteCount.Equals(10)).Single() as NoteFolder;
426:         noteFolderId = noteFolder2.NoteFolderId;

```

```
426:
427:
428:         var noteFeature = new Note()
429:         {
430:             CreationDate = DateTime.Now,
431:             ModificationDate = DateTime.Now,
432:             NoteFolderId = noteFolderId,
433:             NoteName = AppResources.ProgramFeatures,
434:             NameCreation = AppResources.ProgramFeatures + " (" +
DateTime.Now.ToString() + ")",
435:             NoteDescription =
436:                 AppResources.ProgramFeaturesNote,
437:             NameDescription = AppResources.ProgramFeatures + " " +
AppResources.ProgramFeaturesNote
438:         };
439:         context.Notes.InsertOnSubmit(noteFeature);
440:
441:         var notePassword = new Note()
442:         {
443:             CreationDate = DateTime.Now,
444:             ModificationDate = DateTime.Now,
445:             NoteFolderId = noteFolderId,
446:             NoteName = AppResources.PasswordFeature,
447:             NameCreation = AppResources.PasswordFeature + " (" +
DateTime.Now.ToString() + ")",
448:             NoteDescription =
449:                 AppResources.PasswordFeatureNote,
450:             NameDescription = AppResources.PasswordFeature + " " +
AppResources.PasswordFeatureNote
451:         };
452:         context.Notes.InsertOnSubmit(notePassword);
453:
454:         var noteLanguage = new Note()
455:         {
456:             CreationDate = DateTime.Now,
457:             ModificationDate = DateTime.Now,
458:             NoteFolderId = noteFolderId,
459:             NoteName = AppResources.LanguageFeature,
460:             NameCreation = AppResources.LanguageFeature + " (" +
DateTime.Now.ToString() + ")",
461:             NoteDescription =
462:                 AppResources.LanguageFeatureNote,
463:             NameDescription = AppResources.LanguageFeature + " " +
AppResources.LanguageFeatureNote
464:         };
465:         context.Notes.InsertOnSubmit(noteLanguage);
466:
467:         var noteBackground = new Note()
468:         {
469:             CreationDate = DateTime.Now,
470:             ModificationDate = DateTime.Now,
471:             NoteFolderId = noteFolderId,
472:             NoteName = AppResources.BackgroundFeature,
473:             NameCreation = AppResources.BackgroundFeature + " (" +
DateTime.Now.ToString() + ")",
474:             NoteDescription =
475:                 AppResources.BackgroundFeatureNote,
476:             NameDescription = AppResources.BackgroundFeature + " " +
AppResources.BackgroundFeatureNote
477:         };
478:         context.Notes.InsertOnSubmit(noteBackground);
479:
480:         var noteSearch = new Note()
481:         {
482:             CreationDate = DateTime.Now,
483:             ModificationDate = DateTime.Now,
484:             NoteFolderId = noteFolderId,
485:             NoteName = AppResources.SearchFeature,
486:             NameCreation = AppResources.SearchFeature + " (" +
```

```

DateTime.Now.ToString() + ")",
487:         NoteDescription =
488:             AppResources.SearchFeatureNote,
489:         NameDescription = AppResources.SearchFeature + " " +
AppResources.SearchFeatureNote
490:     };
491:     context.Notes.InsertOnSubmit(noteSearch);
492:
493:     var noteSync = new Note()
494:     {
495:         CreationDate = DateTime.Now,
496:         ModificationDate = DateTime.Now,
497:         NoteFolderId = noteFolderId,
498:         NoteName = AppResources.SyncFeature,
499:         NameCreation = AppResources.SyncFeature + " (" +
DateTime.Now.ToString() + ")",
500:         NoteDescription =
501:             AppResources.SyncFeatureNote,
502:         NameDescription = AppResources.SyncFeature + " " +
AppResources.SyncFeatureNote
503:     };
504:     context.Notes.InsertOnSubmit(noteSync);
505:
506:     var noteFont = new Note()
507:     {
508:         CreationDate = DateTime.Now,
509:         ModificationDate = DateTime.Now,
510:         NoteFolderId = noteFolderId,
511:         NoteName = AppResources.FontFeature,
512:         NameCreation = AppResources.FontFeature + " (" +
DateTime.Now.ToString() + ")",
513:         NoteDescription =
514:             AppResources.FontFeatureNote,
515:         NameDescription = AppResources.FontFeature + " " +
AppResources.FontFeatureNote
516:     };
517:     context.Notes.InsertOnSubmit(noteFont);
518:
519:     var noteOrder = new Note()
520:     {
521:         CreationDate = DateTime.Now,
522:         ModificationDate = DateTime.Now,
523:         NoteFolderId = noteFolderId,
524:         NoteName = AppResources.OrderFeature,
525:         NameCreation = AppResources.OrderFeature + " (" +
DateTime.Now.ToString() + ")",
526:         NoteDescription =
527:             AppResources.OrderFeatureNote,
528:         NameDescription = AppResources.OrderFeature + " " +
AppResources.OrderFeatureNote
529:     };
530:     context.Notes.InsertOnSubmit(noteOrder);
531:
532:     var noteNote = new Note()
533:     {
534:         CreationDate = DateTime.Now,
535:         ModificationDate = DateTime.Now,
536:         NoteFolderId = noteFolderId,
537:         NoteName = AppResources.NoteFeature,
538:         NameCreation = AppResources.NoteFeature + " (" +
DateTime.Now.ToString() + ")",
539:         NoteDescription =
540:             AppResources.NoteFeatureNote,
541:         NameDescription = AppResources.NoteFeature + " " +
AppResources.NoteFeatureNote
542:     };
543:     context.Notes.InsertOnSubmit(noteNote);
544:
545:     var noteNoteFolder = new Note()

```

```
546:         {
547:             CreationDate = DateTime.Now,
548:             ModificationDate = DateTime.Now,
549:             NoteFolderId = noteFolderId,
550:             NoteName = AppResources.NoteFolderFeature,
551:             NameCreation = AppResources.NoteFolderFeature + " (" +
DateTime.Now.ToString() + ")",
552:             NoteDescription =
553:                 AppResources.NoteFeatureNote,
554:             NameDescription = AppResources.NoteFolderFeature + " " +
AppResources.NoteFolderFeatureNote
555:         };
556:         context.Notes.InsertOnSubmit(noteNoteFolder);
557:         context.SubmitChanges();
558:     }
559: }
560: }
561: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.3 AppResources.Designer.cs

This code was generated by a tool. Runtime Version:4.0.30319.34014

Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

Body Source

```
1:  ?//-----
2:  // <auto-generated>
3:  //      This code was generated by a tool.
4:  //      Runtime Version:4.0.30319.34014
5:  //
6:  //      Changes to this file may cause incorrect behavior and will be lost if
7:  //      the code is regenerated.
8:  // </auto-generated>
9:  //-----
10:
11: namespace AwesomeNote.Resources {
12:     using System;
13:
14:
15:     /// <summary>
16:     ///     A strongly-typed resource class, for looking up localized strings, etc.
17:     /// </summary>
18:     // This class was auto-generated by the StronglyTypedResourceBuilder
19:     // class via a tool like ResGen or Visual Studio.
20:     // To add or remove a member, edit your .ResX file then rerun ResGen
21:     // with the /str option, or rebuild your VS project.
22:
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder",
"4.0.0.0")]
23:     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
24:     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
25:     public class AppResources {
26:
27:         private static global::System.Resources.ResourceManager resourceMan;
28:
29:         private static global::System.Globalization.CultureInfo resourceCulture;
30:     }
```



```

31:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
32:     internal AppResources() {
33:     }
34:
35:     /// <summary>
36:     ///     Returns the cached ResourceManager instance used by this class.
37:     /// </summary>
38:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
39:     public static global::System.Resources.ResourceManager ResourceManager {
40:     get {
41:         if (object.ReferenceEquals(resourceMan, null)) {
42:             global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("AwesomeNote.Resources.AppResources",
typeof(AppResources).Assembly);
43:             resourceMan = temp;
44:         }
45:         return resourceMan;
46:     }
47: }
48:
49:     /// <summary>
50:     ///     Overrides the current thread's CurrentUICulture property for all
51:     ///     resource lookups using this strongly typed resource class.
52:     /// </summary>
53:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
54:     public static global::System.Globalization.CultureInfo Culture {
55:     get {
56:         return resourceCulture;
57:     }
58:     set {
59:         resourceCulture = value;
60:     }
61: }
62:
63:     /// <summary>
64:     ///     Looks up a localized string similar to Hakkinda.
65:     /// </summary>
66:     public static string About {
67:     get {
68:         return ResourceManager.GetString("About", resourceCulture);
69:     }
70: }
71:
72:     /// <summary>
73:     ///     Looks up a localized string similar to Uygulama Hakkinda.
74:     /// </summary>
75:     public static string AboutTheApp {
76:     get {
77:         return ResourceManager.GetString("AboutTheApp", resourceCulture);
78:     }
79: }
80:
81:     /// <summary>
82:     ///     Looks up a localized string similar to Umarim programi kullanirken
keyif almissinizdir. Çünkü ben yazarken baya keyif aldım. Program hakkında taleplerinizi,
bu sayfadaki İletişim butonuna tıklayarak veya direkt olarak coderserdar@outlook.com
adresine mail atarak belirtirseniz ve uygulamayı oylarsanız çok sevinirim. Sizin sayenizde
bu program daha çok gelişecek. İyi günler.
83:     /// </summary>
84:     public static string AboutTheAppText {
85:     get {
86:         return ResourceManager.GetString("AboutTheAppText", resourceCulture);
87:     }

```

```
88:     }
89:
90:     /// <summary>
91:     /// Looks up a localized string similar to Awesome Note Uygulamasi Hakkinda.
92:     /// </summary>
93:     public static string AboutTheAwesomeNote {
94:         get {
95:             return ResourceManager.GetString("AboutTheAwesomeNote",
resourceCulture);
96:         }
97:     }
98:
99:     /// <summary>
100:    /// Looks up a localized string similar to Aktif.
101:    /// </summary>
102:    public static string Active {
103:        get {
104:            return ResourceManager.GetString("Active", resourceCulture);
105:        }
106:    }
107:
108:    /// <summary>
109:    /// Looks up a localized string similar to Klasör Ekle.
110:    /// </summary>
111:    public static string AddFolder {
112:        get {
113:            return ResourceManager.GetString("AddFolder", resourceCulture);
114:        }
115:    }
116:
117:    /// <summary>
118:    /// Looks up a localized string similar to Nesne Ekle.
119:    /// </summary>
120:    public static string AddItem {
121:        get {
122:            return ResourceManager.GetString("AddItem", resourceCulture);
123:        }
124:    }
125:
126:    /// <summary>
127:    /// Looks up a localized string similar to Not Ekle.
128:    /// </summary>
129:    public static string AddNote {
130:        get {
131:            return ResourceManager.GetString("AddNote", resourceCulture);
132:        }
133:    }
134:
135:    /// <summary>
136:    /// Looks up a localized string similar to Not Klasörleri.
137:    /// </summary>
138:    public static string AppFolder {
139:        get {
140:            return ResourceManager.GetString("AppFolder", resourceCulture);
141:        }
142:    }
143:
144:    /// <summary>
145:    /// Looks up a localized string similar to MY APPLICATION.
146:    /// </summary>
147:    public static string ApplicationTitle {
148:        get {
149:            return ResourceManager.GetString("ApplicationTitle", resourceCulture);
150:        }
151:    }
152:
153:    /// <summary>
154:    /// Looks up a localized string similar to Awesome Note.
155:    /// </summary>
```

```

156:         public static string AppTitle {
157:             get {
158:                 return ResourceManager.GetString("AppTitle", resourceCulture);
159:             }
160:         }
161:
162:         /// <summary>
163:         /// Looks up a localized string similar to Arabic.
164:         /// </summary>
165:         public static string Arabic {
166:             get {
167:                 return ResourceManager.GetString("Arabic", resourceCulture);
168:             }
169:         }
170:
171:         /// <summary>
172:         /// Looks up a localized string similar to Artan Sirada.
173:         /// </summary>
174:         public static string Ascending {
175:             get {
176:                 return ResourceManager.GetString("Ascending", resourceCulture);
177:             }
178:         }
179:
180:         /// <summary>
181:         /// Looks up a localized string similar to En Az Dört Karakter.
182:         /// </summary>
183:         public static string AtLeastFourDigits {
184:             get {
185:                 return ResourceManager.GetString("AtLeastFourDigits",
resourceCulture);
186:             }
187:         }
188:
189:         /// <summary>
190:         /// Looks up a localized string similar to Arka Plan.
191:         /// </summary>
192:         public static string Background {
193:             get {
194:                 return ResourceManager.GetString("Background", resourceCulture);
195:             }
196:         }
197:
198:         /// <summary>
199:         /// Looks up a localized string similar to Arka Plan Rengi.
200:         /// </summary>
201:         public static string BackgroundColor {
202:             get {
203:                 return ResourceManager.GetString("BackgroundColor", resourceCulture);
204:             }
205:         }
206:
207:         /// <summary>
208:         /// Looks up a localized string similar to Arka Plan Özellikleri.
209:         /// </summary>
210:         public static string BackgroundFeature {
211:             get {
212:                 return ResourceManager.GetString("BackgroundFeature",
resourceCulture);
213:             }
214:         }
215:
216:         /// <summary>
217:         /// Looks up a localized string similar to Normalde kullandığım
uygulamalarda görsellikten çok işlevsellige önem veririm. Yazarken de aynı şekilde yazdım
bu programı ancak herkesin benim düşündüğüm gibi düşünmediğini de bildiğim için görselliği
sade olan bu programda arka plan rengini ayarlanabilir bir şekilde yaptım. Programın Genel
Ayarları kısmında Arka Plan bölümünde istenirse sadece renk seçimi yaparak, istenirse de
mobil cihaz içerisindeki her hangi bir fotoğrafı arka plan olarak belirleyebilmek

```

```
mümkündür. Arka plan rengi uygulamanin tamamı iç [rest of string was truncated]&quot;;.
218:     /// </summary>
219:     public static string BackgroundFeatureNote {
220:         get {
221:             return ResourceManager.GetString("BackgroundFeatureNote",
resourceCulture);
222:         }
223:     }
224:
225:     /// <summary>
226:     /// Looks up a localized string similar to Arka Plan Resmi.
227:     /// </summary>
228:     public static string BackgroundImage {
229:         get {
230:             return ResourceManager.GetString("BackgroundImage", resourceCulture);
231:         }
232:     }
233:
234:     /// <summary>
235:     /// Looks up a localized string similar to Ait Oldugu Klasörün Adi.
236:     /// </summary>
237:     public static string BelongFolderName {
238:         get {
239:             return ResourceManager.GetString("BelongFolderName", resourceCulture);
240:         }
241:     }
242:
243:     /// <summary>
244:     /// Looks up a localized string similar to Siyah.
245:     /// </summary>
246:     public static string Black {
247:         get {
248:             return ResourceManager.GetString("Black", resourceCulture);
249:         }
250:     }
251:
252:     /// <summary>
253:     /// Looks up a localized string similar to Mavi.
254:     /// </summary>
255:     public static string Blue {
256:         get {
257:             return ResourceManager.GetString("Blue", resourceCulture);
258:         }
259:     }
260:
261:     /// <summary>
262:     /// Looks up a localized string similar to Kahverengi.
263:     /// </summary>
264:     public static string Brown {
265:         get {
266:             return ResourceManager.GetString("Brown", resourceCulture);
267:         }
268:     }
269:
270:     /// <summary>
271:     /// Looks up a localized string similar to Iptal.
272:     /// </summary>
273:     public static string Cancel {
274:         get {
275:             return ResourceManager.GetString("Cancel", resourceCulture);
276:         }
277:     }
278:
279:     /// <summary>
280:     /// Looks up a localized string similar to Klasör Degistir.
281:     /// </summary>
282:     public static string ChangeFolder {
283:         get {
284:             return ResourceManager.GetString("ChangeFolder", resourceCulture);
```

```

285:         }
286:     }
287:
288:     /// <summary>
289:     /// Looks up a localized string similar to Karakter Sayisi.
290:     /// </summary>
291:     public static string CharacterNumber {
292:         get {
293:             return ResourceManager.GetString("CharacterNumber", resourceCulture);
294:         }
295:     }
296:
297:     /// <summary>
298:     /// Looks up a localized string similar to Chinese.
299:     /// </summary>
300:     public static string Chinese {
301:         get {
302:             return ResourceManager.GetString("Chinese", resourceCulture);
303:         }
304:     }
305:
306:     /// <summary>
307:     /// Looks up a localized string similar to Iletisime Geç.
308:     /// </summary>
309:     public static string ContactWithUs {
310:         get {
311:             return ResourceManager.GetString("ContactWithUs", resourceCulture);
312:         }
313:     }
314:
315:     /// <summary>
316:     /// Looks up a localized string similar to Olusturulma Tarihi.
317:     /// </summary>
318:     public static string CreationDate {
319:         get {
320:             return ResourceManager.GetString("CreationDate", resourceCulture);
321:         }
322:     }
323:
324:     /// <summary>
325:     /// Looks up a localized string similar to Klasör Sil.
326:     /// </summary>
327:     public static string DeleteFolder {
328:         get {
329:             return ResourceManager.GetString("DeleteFolder", resourceCulture);
330:         }
331:     }
332:
333:     /// <summary>
334:     /// Looks up a localized string similar to Not Klasörünüz Tüm İçerigi Ile
335:     /// Silinecek. Onaylıyor Musunuz?..
336:     /// </summary>
337:     public static string DeleteFolderQuestion {
338:         get {
339:             return ResourceManager.GetString("DeleteFolderQuestion",
340:             resourceCulture);
341:         }
342:     }
343:
344:     /// <summary>
345:     /// Looks up a localized string similar to Notu Sil.
346:     /// </summary>
347:     public static string DeleteNote {
348:         get {
349:             return ResourceManager.GetString("DeleteNote", resourceCulture);
350:         }
351:     }
352:
353:     /// <summary>

```

```
352:         /// Looks up a localized string similar to Notu Silmek Istediginizden Emin
Misiniz?.
353:         /// </summary>
354:         public static string DeleteNoteQuestion {
355:             get {
356:                 return ResourceManager.GetString("DeleteNoteQuestion",
resourceCulture);
357:             }
358:         }
359:
360:         /// <summary>
361:         /// Looks up a localized string similar to Azalan Sirada.
362:         /// </summary>
363:         public static string Descending {
364:             get {
365:                 return ResourceManager.GetString("Descending", resourceCulture);
366:             }
367:         }
368:
369:         /// <summary>
370:         /// Looks up a localized string similar to Ingilizce.
371:         /// </summary>
372:         public static string English {
373:             get {
374:                 return ResourceManager.GetString("English", resourceCulture);
375:             }
376:         }
377:
378:         /// <summary>
379:         /// Looks up a localized string similar to Lütfen Not Klasörünün Adini
Yaziniz.
380:         /// </summary>
381:         public static string EnterFolderName {
382:             get {
383:                 return ResourceManager.GetString("EnterFolderName", resourceCulture);
384:             }
385:         }
386:
387:         /// <summary>
388:         /// Looks up a localized string similar to Lütfen Notun Adini Giriniz.
389:         /// </summary>
390:         public static string EnterNoteName {
391:             get {
392:                 return ResourceManager.GetString("EnterNoteName", resourceCulture);
393:             }
394:         }
395:
396:         /// <summary>
397:         /// Looks up a localized string similar to Sifre Girme.
398:         /// </summary>
399:         public static string EnterPassword {
400:             get {
401:                 return ResourceManager.GetString("EnterPassword", resourceCulture);
402:             }
403:         }
404:
405:         /// <summary>
406:         /// Looks up a localized string similar to Uygulamadan Çık.
407:         /// </summary>
408:         public static string ExitApp {
409:             get {
410:                 return ResourceManager.GetString("ExitApp", resourceCulture);
411:             }
412:         }
413:
414:         /// <summary>
415:         /// Looks up a localized string similar to Uygulamadan Çıkmak
Istediginizden Emin Misiniz?.
416:         /// </summary>
```

```
417:         public static string ExitAppQuestion {
418:             get {
419:                 return ResourceManager.GetString("ExitAppQuestion", resourceCulture);
420:             }
421:         }
422:
423:         /// <summary>
424:         /// Looks up a localized string similar to Bu Isimde Bir Klasör Zaten
Mevcut.
425:         /// </summary>
426:         public static string FolderExists {
427:             get {
428:                 return ResourceManager.GetString("FolderExists", resourceCulture);
429:             }
430:         }
431:
432:         /// <summary>
433:         /// Looks up a localized string similar to Klasör Adi.
434:         /// </summary>
435:         public static string FolderName {
436:             get {
437:                 return ResourceManager.GetString("FolderName", resourceCulture);
438:             }
439:         }
440:
441:         /// <summary>
442:         /// Looks up a localized string similar to Klasör Siralama.
443:         /// </summary>
444:         public static string FolderOrder {
445:             get {
446:                 return ResourceManager.GetString("FolderOrder", resourceCulture);
447:             }
448:         }
449:
450:         /// <summary>
451:         /// Looks up a localized string similar to Klasör Ayarlari.
452:         /// </summary>
453:         public static string FolderSettings {
454:             get {
455:                 return ResourceManager.GetString("FolderSettings", resourceCulture);
456:             }
457:         }
458:
459:         /// <summary>
460:         /// Looks up a localized string similar to Yazı Tipi.
461:         /// </summary>
462:         public static string Font {
463:             get {
464:                 return ResourceManager.GetString("Font", resourceCulture);
465:             }
466:         }
467:
468:         /// <summary>
469:         /// Looks up a localized string similar to Yazı Tipi Ailesi.
470:         /// </summary>
471:         public static string FontFamily {
472:             get {
473:                 return ResourceManager.GetString("FontFamily", resourceCulture);
474:             }
475:         }
476:
477:         /// <summary>
478:         /// Looks up a localized string similar to Font Özellikleri.
479:         /// </summary>
480:         public static string FontFeature {
481:             get {
482:                 return ResourceManager.GetString("FontFeature", resourceCulture);
483:             }
484:         }
```

```
485:
486:     /// <summary>
487:     /// Looks up a localized string similar to Program üzerinde notlar ile
    ilgili olarak, her klasör için bir font ailesi ve font boyutu ayarlamasi yapabilirsiniz.
    Bunu seçtiginiz not klasörü içerisinde Klasör Ayarlari butonuna bastiktan sonra çıkan
    ekranda Font bölümüne gelerek yapabilirsiniz. Ekranda o an aktif olarak seçtiginiz font
    ailesi ve font boyutu görünecektir. Eger siz seçmezseniz program font ayarini otomatik
    olaraak Verdana ve 26 punto olarak ayarlayacaktır..
488:     /// </summary>
489:     public static string FontFeatureNote {
490:         get {
491:             return ResourceManager.GetString("FontFeatureNote", resourceCulture);
492:         }
493:     }
494:
495:     /// <summary>
496:     /// Looks up a localized string similar to Yazı Tipi Boyutu.
497:     /// </summary>
498:     public static string FontSize {
499:         get {
500:             return ResourceManager.GetString("FontSize", resourceCulture);
501:         }
502:     }
503:
504:     /// <summary>
505:     /// Looks up a localized string similar to Dört Karakter.
506:     /// </summary>
507:     public static string FourDigits {
508:         get {
509:             return ResourceManager.GetString("FourDigits", resourceCulture);
510:         }
511:     }
512:
513:     /// <summary>
514:     /// Looks up a localized string similar to French.
515:     /// </summary>
516:     public static string French {
517:         get {
518:             return ResourceManager.GetString("French", resourceCulture);
519:         }
520:     }
521:
522:     /// <summary>
523:     /// Looks up a localized string similar to Genel Ayarlar.
524:     /// </summary>
525:     public static string GeneralSettings {
526:         get {
527:             return ResourceManager.GetString("GeneralSettings", resourceCulture);
528:         }
529:     }
530:
531:     /// <summary>
532:     /// Looks up a localized string similar to Almanca.
533:     /// </summary>
534:     public static string German {
535:         get {
536:             return ResourceManager.GetString("German", resourceCulture);
537:         }
538:     }
539:
540:     /// <summary>
541:     /// Looks up a localized string similar to Gri.
542:     /// </summary>
543:     public static string Gray {
544:         get {
545:             return ResourceManager.GetString("Gray", resourceCulture);
546:         }
547:     }
548:
```



```
549:         /// <summary>
550:         /// Looks up a localized string similar to Yesil.
551:         /// </summary>
552:         public static string Green {
553:             get {
554:                 return ResourceManager.GetString("Green", resourceCulture);
555:             }
556:         }
557:
558:         /// <summary>
559:         /// Looks up a localized string similar to Irish.
560:         /// </summary>
561:         public static string Irish {
562:             get {
563:                 return ResourceManager.GetString("Irish", resourceCulture);
564:             }
565:         }
566:
567:         /// <summary>
568:         /// Looks up a localized string similar to Yazı Tipi Kalın Olsun.
569:         /// </summary>
570:         public static string IsBold {
571:             get {
572:                 return ResourceManager.GetString("IsBold", resourceCulture);
573:             }
574:         }
575:
576:         /// <summary>
577:         /// Looks up a localized string similar to Yazı Tipi İtalik Olsun.
578:         /// </summary>
579:         public static string IsItalic {
580:             get {
581:                 return ResourceManager.GetString("IsItalic", resourceCulture);
582:             }
583:         }
584:
585:         /// <summary>
586:         /// Looks up a localized string similar to Italian.
587:         /// </summary>
588:         public static string Italian {
589:             get {
590:                 return ResourceManager.GetString("Italian", resourceCulture);
591:             }
592:         }
593:
594:         /// <summary>
595:         /// Looks up a localized string similar to Dil.
596:         /// </summary>
597:         public static string Language {
598:             get {
599:                 return ResourceManager.GetString("Language", resourceCulture);
600:             }
601:         }
602:
603:         /// <summary>
604:         /// Looks up a localized string similar to Dil Özellikleri.
605:         /// </summary>
606:         public static string LanguageFeature {
607:             get {
608:                 return ResourceManager.GetString("LanguageFeature", resourceCulture);
609:             }
610:         }
611:
612:         /// <summary>
613:         /// Looks up a localized string similar to Program İngilizce, Türkçe,
        Almanca, İspanyolca, İtalyanca, Fransızca, Portekizce, Çince, Rusça, Arapça, Farsça gibi
        dillere tamamen destek vermektedir. Yani uygulamada dil gerekebilecek bütün menülerde ve
        sayfalarda kendi dilinizi görebileceksiniz..
614:         /// </summary>
```

```
615:         public static string LanguageFeatureNote {
616:             get {
617:                 return ResourceManager.GetString("LanguageFeatureNote",
resourceCulture);
618:             }
619:         }
620:
621:         /// <summary>
622:         /// Looks up a localized string similar to Ayarlarin etkili olmasi için
uygulamayi bastan baslatmaniz gerekebilir....
623:         /// </summary>
624:         public static string LanguageWarning {
625:             get {
626:                 return ResourceManager.GetString("LanguageWarning", resourceCulture);
627:             }
628:         }
629:
630:         /// <summary>
631:         /// Looks up a localized string similar to Baglan.
632:         /// </summary>
633:         public static string Login {
634:             get {
635:                 return ResourceManager.GetString("Login", resourceCulture);
636:             }
637:         }
638:
639:         /// <summary>
640:         /// Looks up a localized string similar to Degistirilme Tarihi.
641:         /// </summary>
642:         public static string ModificationDate {
643:             get {
644:                 return ResourceManager.GetString("ModificationDate", resourceCulture);
645:             }
646:         }
647:
648:         /// <summary>
649:         /// Looks up a localized string similar to Sonraki.
650:         /// </summary>
651:         public static string Next {
652:             get {
653:                 return ResourceManager.GetString("Next", resourceCulture);
654:             }
655:         }
656:
657:         /// <summary>
658:         /// Looks up a localized string similar to Sonraki Not.
659:         /// </summary>
660:         public static string NextNote {
661:             get {
662:                 return ResourceManager.GetString("NextNote", resourceCulture);
663:             }
664:         }
665:
666:         /// <summary>
667:         /// Looks up a localized string similar to Hayir.
668:         /// </summary>
669:         public static string No {
670:             get {
671:                 return ResourceManager.GetString("No", resourceCulture);
672:             }
673:         }
674:
675:         /// <summary>
676:         /// Looks up a localized string similar to Not.
677:         /// </summary>
678:         public static string Note {
679:             get {
680:                 return ResourceManager.GetString("Note", resourceCulture);
681:             }
682:         }
```

```

682:         }
683:
684:         /// <summary>
685:         /// Looks up a localized string similar to Bu Isimde Bir Not Zaten Mevcut.
686:         /// </summary>
687:         public static string NoteExists {
688:             get {
689:                 return ResourceManager.GetString("NoteExists", resourceCulture);
690:             }
691:         }
692:
693:         /// <summary>
694:         /// Looks up a localized string similar to Not Özellikleri.
695:         /// </summary>
696:         public static string NoteFeature {
697:             get {
698:                 return ResourceManager.GetString("NoteFeature", resourceCulture);
699:             }
700:         }
701:
702:         /// <summary>
703:         /// Looks up a localized string similar to Not içerisinde notun adi, içinde
        bulunduğ u klasörün adi, notun olusturulma tarihi ve notun kendisi bulunmaktadır. Not
        içerisinde klasördeki bir önceki nota ve bir sonraki nota gidebilmek mümkündür. Notun adina
        tiklandigi zaman notun adini degistirme mümkündür. Not metninin tamamini seçebilmek, not
        hakkında detayli bilgileri görüntölemek mümkündür. Seçilen bir notu SMS veya E-Posta
        yoluyla gönderebilmek de mümkündür. Notu silmek veya ait olduğu klasörü degistirebilmek de
        mümkündür..
704:         /// </summary>
705:         public static string NoteFeatureNote {
706:             get {
707:                 return ResourceManager.GetString("NoteFeatureNote", resourceCulture);
708:             }
709:         }
710:
711:         /// <summary>
712:         /// Looks up a localized string similar to Not Klasörü Özellikleri.
713:         /// </summary>
714:         public static string NoteFolderFeature {
715:             get {
716:                 return ResourceManager.GetString("NoteFolderFeature",
resourceCulture);
717:             }
718:         }
719:
720:         /// <summary>
721:         /// Looks up a localized string similar to Not klasörü içerisinde, içindeki
        notların listesi bulunmaktadır. Buradan klasör içerisindeki her hangi bir nota gidebilmek
        mümkündür. Not klasörünü silebilmek mümkündür, ancak bu işlemi gerçekleştirdiginiz zaman
        not klasörünü içerisindeki bütün notlarla birlikte silecektir. Bu yüzden klasör silmeden
        önce Senkronizasyon yaparak notlarınızı yedeklemeniz önemle tavsiye edilir. Klasöre yeni
        bir not ekleyebilir veya ayarlarını degistirebilirsiniz. Ayrıca klasör adina tikladiginiz
        zaman klasörün adini degistirebi [rest of string was truncated]&quot;;;.
722:         /// </summary>
723:         public static string NoteFolderFeatureNote {
724:             get {
725:                 return ResourceManager.GetString("NoteFolderFeatureNote",
resourceCulture);
726:             }
727:         }
728:
729:         /// <summary>
730:         /// Looks up a localized string similar to Not Hakkında.
731:         /// </summary>
732:         public static string NoteInfo {
733:             get {
734:                 return ResourceManager.GetString("NoteInfo", resourceCulture);
735:             }
736:         }

```

```
737:
738:     /// <summary>
739:     /// Looks up a localized string similar to Not Listesi.
740:     /// </summary>
741:     public static string NoteListOf {
742:         get {
743:             return ResourceManager.GetString("NoteListOf", resourceCulture);
744:         }
745:     }
746:
747:     /// <summary>
748:     /// Looks up a localized string similar to Not Adi.
749:     /// </summary>
750:     public static string NoteName {
751:         get {
752:             return ResourceManager.GetString("NoteName", resourceCulture);
753:         }
754:     }
755:
756:     /// <summary>
757:     /// Looks up a localized string similar to Not Siralama.
758:     /// </summary>
759:     public static string NoteOrder {
760:         get {
761:             return ResourceManager.GetString("NoteOrder", resourceCulture);
762:         }
763:     }
764:
765:     /// <summary>
766:     /// Looks up a localized string similar to Not Kaydedildi.
767:     /// </summary>
768:     public static string NoteSaved {
769:         get {
770:             return ResourceManager.GetString("NoteSaved", resourceCulture);
771:         }
772:     }
773:
774:     /// <summary>
775:     /// Looks up a localized string similar to Tamam.
776:     /// </summary>
777:     public static string OK {
778:         get {
779:             return ResourceManager.GetString("OK", resourceCulture);
780:         }
781:     }
782:
783:     /// <summary>
784:     /// Looks up a localized string similar to One Drive.
785:     /// </summary>
786:     public static string OneDrive {
787:         get {
788:             return ResourceManager.GetString("OneDrive", resourceCulture);
789:         }
790:     }
791:
792:     /// <summary>
793:     /// Looks up a localized string similar to OneDrive Senkronizasyonu
794:     Tamamlandi.
795:     /// </summary>
796:     public static string OneDriveSyncCompleted {
797:         get {
798:             return ResourceManager.GetString("OneDriveSyncCompleted",
799:             resourceCulture);
800:         }
801:     }
802:
803:     /// <summary>
804:     /// Looks up a localized string similar to Turuncu.
805:     /// </summary>
```

```
804:         public static string Orange {
805:             get {
806:                 return ResourceManager.GetString("Orange", resourceCulture);
807:             }
808:         }
809:
810:         /// <summary>
811:         /// Looks up a localized string similar to Siralama Özellikleri.
812:         /// </summary>
813:         public static string OrderFeature {
814:             get {
815:                 return ResourceManager.GetString("OrderFeature", resourceCulture);
816:             }
817:         }
818:
819:         /// <summary>
820:         /// Looks up a localized string similar to Program üzerinde hem not
821:         /// klasörlerini hem de klasör içerisindeki notlari siralamak mümkündür. Notlari ve not
822:         /// klasörlerini; adina, olusturulma tarihine göre ve en son degistirilme tarihine göre artan
823:         /// ve azalan sekilde siralamak mümkündür. Not klasörlerinin siralama ayarlarini degistirmek
824:         /// için Genel Ayarlar kismina Diger Ayarlar bölümüne, klasör içerisindeki notlari siralama
825:         /// ayarlarini degistirmek için ise Klasör Ayarlari kismindeki Diger Ayarlar bölümüne girmek
826:         /// gerekmektedir. Ekranda hali hazirda seçilen a [rest of string was truncated]&quot;;.
827:         /// </summary>
828:         public static string OrderFeatureNote {
829:             get {
830:                 return ResourceManager.GetString("OrderFeatureNote", resourceCulture);
831:             }
832:         }
833:
834:         /// <summary>
835:         /// Looks up a localized string similar to Siralama Türü.
836:         /// </summary>
837:         public static string OrderStyle {
838:             get {
839:                 return ResourceManager.GetString("OrderStyle", resourceCulture);
840:             }
841:         }
842:
843:         /// <summary>
844:         /// Looks up a localized string similar to Diger Ayarlar.
845:         /// </summary>
846:         public static string OtherSettings {
847:             get {
848:                 return ResourceManager.GetString("OtherSettings", resourceCulture);
849:             }
850:         }
851:
852:         /// <summary>
853:         /// Looks up a localized string similar to Pasif.
854:         /// </summary>
855:         public static string Passive {
856:             get {
857:                 return ResourceManager.GetString("Passive", resourceCulture);
858:             }
859:         }
860:
861:         /// <summary>
862:         /// Looks up a localized string similar to Sifre.
863:         /// </summary>
864:         public static string Password {
865:             get {
866:                 return ResourceManager.GetString("Password", resourceCulture);
867:             }
868:         }
869:
870:         /// <summary>
871:         /// Looks up a localized string similar to Sifre Yanlis.
872:         /// </summary>
```

```

867:         public static string PasswordFalse {
868:             get {
869:                 return ResourceManager.GetString("PasswordFalse", resourceCulture);
870:             }
871:         }
872:
873:         /// <summary>
874:         /// Looks up a localized string similar to Sifre Özelligi.
875:         /// </summary>
876:         public static string PasswordFeature {
877:             get {
878:                 return ResourceManager.GetString("PasswordFeature", resourceCulture);
879:             }
880:         }
881:
882:         /// <summary>
883:         /// Looks up a localized string similar to Seçilen her hangi bir not
884:         /// klasöründe Klasör Ayarlari bölümünde Sifre kısmında o not klasörüne sifre ataması
885:         /// yapabilmek mümkündür. Aynı bölümden mevcut sifre de kaldırılabilir. Girilen sifre de en az
886:         /// 4 karakterli olmalıdır. Sadece rakam olmalı gibi bir kısıtlama söz konusu değildir. Sifre
887:         /// ataması yapıldıktan sonra klasöre her giriste mevcut sifreyi soracaktır. Sifre girilirken
888:         /// kutucuklara sifre girildikten sonra Enter tusuna basılması yeterlidir. Bu özellik Windows
889:         /// Phone marketinde ilk kez bu uygulamada bulu [rest of string was truncated]&quot;;.
890:         /// </summary>
891:         public static string PasswordFeatureNote {
892:             get {
893:                 return ResourceManager.GetString("PasswordFeatureNote",
894: resourceCulture);
895:             }
896:         }
897:
898:         /// <summary>
899:         /// Looks up a localized string similar to Sifre Korumasi.
900:         /// </summary>
901:         public static string PasswordProtection {
902:             get {
903:                 return ResourceManager.GetString("PasswordProtection",
904: resourceCulture);
905:             }
906:         }
907:
908:         /// <summary>
909:         /// Looks up a localized string similar to Sifre Durumu.
910:         /// </summary>
911:         public static string PasswordState {
912:             get {
913:                 return ResourceManager.GetString("PasswordState", resourceCulture);
914:             }
915:         }
916:
917:         /// <summary>
918:         /// Looks up a localized string similar to Sifre Dogru.
919:         /// </summary>
920:         public static string PasswordTrue {
921:             get {
922:                 return ResourceManager.GetString("PasswordTrue", resourceCulture);
923:             }
924:         }
925:
926:         /// <summary>
927:         /// Looks up a localized string similar to Persian.
928:         /// </summary>
929:         public static string Persian {
930:             get {
931:                 return ResourceManager.GetString("Persian", resourceCulture);
932:             }
933:         }
934:
935:         /// <summary>

```

```

928:         /// Looks up a localized string similar to Sifreyi Giriniz.
929:         /// </summary>
930:         public static string PleaseEnterPassword {
931:             get {
932:                 return ResourceManager.GetString("PleaseEnterPassword",
resourceCulture);
933:             }
934:         }
935:
936:         /// <summary>
937:         /// Looks up a localized string similar to Sifreyi Tekrar Giriniz.
938:         /// </summary>
939:         public static string PleaseReEnterPassword {
940:             get {
941:                 return ResourceManager.GetString("PleaseReEnterPassword",
resourceCulture);
942:             }
943:         }
944:
945:         /// <summary>
946:         /// Looks up a localized string similar to Portuguese.
947:         /// </summary>
948:         public static string Portuguese {
949:             get {
950:                 return ResourceManager.GetString("Portuguese", resourceCulture);
951:             }
952:         }
953:
954:         /// <summary>
955:         /// Looks up a localized string similar to Önceki.
956:         /// </summary>
957:         public static string Previous {
958:             get {
959:                 return ResourceManager.GetString("Previous", resourceCulture);
960:             }
961:         }
962:
963:         /// <summary>
964:         /// Looks up a localized string similar to Önceki Not.
965:         /// </summary>
966:         public static string PreviousNote {
967:             get {
968:                 return ResourceManager.GetString("PreviousNote", resourceCulture);
969:             }
970:         }
971:
972:         /// <summary>
973:         /// Looks up a localized string similar to Programın Özellikleri.
974:         /// </summary>
975:         public static string ProgramFeatures {
976:             get {
977:                 return ResourceManager.GetString("ProgramFeatures", resourceCulture);
978:             }
979:         }
980:
981:         /// <summary>
982:         /// Looks up a localized string similar to Bu program ile not klasörü
olusturma, klasöre sifre atama, uygulamanin arka plan rengini degistirme veya arka plana
telefon üzerindeki fotoğraflardan birini yerlestirebilme, not klasörünün arka plan
fotografini degistirebilme, içerik sorgulaması yapabilme, OneDrive senkronizasyonu,
herhangi bir notun ait olduğu klasörü kolaylıkla degistirme, tutulan notlari SMS veya
E-Posta yoluyla gönderme, detayli font ve siralama ayarlari, çoklu dil destegi, not
hakkında istatistiki bilgi gibi pek çok özellik bulunmaktadır [rest of string was
truncated]&quot;;.
983:         /// </summary>
984:         public static string ProgramFeaturesNote {
985:             get {
986:                 return ResourceManager.GetString("ProgramFeaturesNote",
resourceCulture);

```

```
987:         }
988:     }
989:
990:     /// <summary>
991:     /// Looks up a localized string similar to Mor.
992:     /// </summary>
993:     public static string Purple {
994:         get {
995:             return ResourceManager.GetString("Purple", resourceCulture);
996:         }
997:     }
998:
999:     /// <summary>
1000:    /// Looks up a localized string similar to Degerlendir.
1001:    /// </summary>
1002:    public static string Rate {
1003:        get {
1004:            return ResourceManager.GetString("Rate", resourceCulture);
1005:        }
1006:    }
1007:
1008:    /// <summary>
1009:    /// Looks up a localized string similar to Kirmizi.
1010:    /// </summary>
1011:    public static string Red {
1012:        get {
1013:            return ResourceManager.GetString("Red", resourceCulture);
1014:        }
1015:    }
1016:
1017:    /// <summary>
1018:    /// Looks up a localized string similar to Arka Plan Resmini Kaldır.
1019:    /// </summary>
1020:    public static string RemoveBackgroundImage {
1021:        get {
1022:            return ResourceManager.GetString("RemoveBackgroundImage",
resourceCulture);
1023:        }
1024:    }
1025:
1026:    /// <summary>
1027:    /// Looks up a localized string similar to Ayarlari Sifirla.
1028:    /// </summary>
1029:    public static string ResetSettings {
1030:        get {
1031:            return ResourceManager.GetString("ResetSettings", resourceCulture);
1032:        }
1033:    }
1034:
1035:    /// <summary>
1036:    /// Looks up a localized string similar to LeftToRight.
1037:    /// </summary>
1038:    public static string ResourceFlowDirection {
1039:        get {
1040:            return ResourceManager.GetString("ResourceFlowDirection",
resourceCulture);
1041:        }
1042:    }
1043:
1044:    /// <summary>
1045:    /// Looks up a localized string similar to tr-TR.
1046:    /// </summary>
1047:    public static string ResourceLanguage {
1048:        get {
1049:            return ResourceManager.GetString("ResourceLanguage", resourceCulture);
1050:        }
1051:    }
1052:
1053:    /// <summary>
```



```

1054:    /// Looks up a localized string similar to Russian.
1055:    /// </summary>
1056:    public static string Russian {
1057:        get {
1058:            return ResourceManager.GetString("Russian", resourceCulture);
1059:        }
1060:    }
1061:
1062:    /// <summary>
1063:    /// Looks up a localized string similar to Sanskritçe.
1064:    /// </summary>
1065:    public static string Sanskrit {
1066:        get {
1067:            return ResourceManager.GetString("Sanskrit", resourceCulture);
1068:        }
1069:    }
1070:
1071:    /// <summary>
1072:    /// Looks up a localized string similar to Kaydet.
1073:    /// </summary>
1074:    public static string Save {
1075:        get {
1076:            return ResourceManager.GetString("Save", resourceCulture);
1077:        }
1078:    }
1079:
1080:    /// <summary>
1081:    /// Looks up a localized string similar to Ara.
1082:    /// </summary>
1083:    public static string Search {
1084:        get {
1085:            return ResourceManager.GetString("Search", resourceCulture);
1086:        }
1087:    }
1088:
1089:    /// <summary>
1090:    /// Looks up a localized string similar to Arama Tamamlandi.
1091:    /// </summary>
1092:    public static string SearchCompleted {
1093:        get {
1094:            return ResourceManager.GetString("SearchCompleted", resourceCulture);
1095:        }
1096:    }
1097:
1098:    /// <summary>
1099:    /// Looks up a localized string similar to Arama Özellikleri.
1100:    /// </summary>
1101:    public static string SearchFeature {
1102:        get {
1103:            return ResourceManager.GetString("SearchFeature", resourceCulture);
1104:        }
1105:    }
1106:
1107:    /// <summary>
1108:    /// Looks up a localized string similar to Programda notun adi ve notun
1109:    /// içeriği üzerinden sorgulama yapmaya imkan saglayan bir özellik bulunmaktadır. Bu özellik
1110:    /// sayesinde notlarnızın içinde kaybolmaktan kurtulmuş olursunuz. Sorgulama için arama
1111:    /// yapacağınız metni yazdıktan sonra Enter tusuna basmanız yeterli olacaktır. Sorgu sonuçları
1112:    /// yine aynı sayfada yer alacaktır. Burada önemli bir nokta ise, eğer sorgu sonuçları
1113:    /// içerisinde daha önceden şifrelediğiniz bir not klasörü içerisindeki nota erismeye
1114:    /// kalkarsanız sorgulama ekranından şifre giriş ekranına [rest of string was truncated]";.
1115:    /// </summary>
1116:    public static string SearchFeatureNote {
1117:        get {
1118:            return ResourceManager.GetString("SearchFeatureNote",
1119:            resourceCulture);
1120:        }
1121:    }
1122:
1123:    }
1124:
1125:    }
1126:
1127:    }
1128:
1129:    }
1130:
1131:    }
1132:
1133:    }
1134:
1135:    }
1136:
1137:    }
1138:
1139:    }
1140:
1141:    }
1142:
1143:    }
1144:
1145:    }
1146:
1147:    }
1148:
1149:    }
1150:
1151:    }
1152:
1153:    }
1154:
1155:    }
1156:
1157:    }
1158:
1159:    }
1160:
1161:    }
1162:
1163:    }
1164:
1165:    }
1166:
1167:    }
1168:
1169:    }
1170:
1171:    }
1172:
1173:    }
1174:
1175:    }
1176:
1177:    }
1178:
1179:    }
1180:
1181:    }
1182:
1183:    }
1184:
1185:    }
1186:
1187:    }
1188:
1189:    }
1190:
1191:    }
1192:
1193:    }
1194:
1195:    }
1196:
1197:    }
1198:
1199:    }
1200:
1201:    }
1202:
1203:    }
1204:
1205:    }
1206:
1207:    }
1208:
1209:    }
1210:
1211:    }
1212:
1213:    }
1214:
1215:    }
1216:
1217:    }
1218:
1219:    }
1220:
1221:    }
1222:
1223:    }
1224:
1225:    }
1226:
1227:    }
1228:
1229:    }
1230:
1231:    }
1232:
1233:    }
1234:
1235:    }
1236:
1237:    }
1238:
1239:    }
1240:
1241:    }
1242:
1243:    }
1244:
1245:    }
1246:
1247:    }
1248:
1249:    }
1250:
1251:    }
1252:
1253:    }
1254:
1255:    }
1256:
1257:    }
1258:
1259:    }
1260:
1261:    }
1262:
1263:    }
1264:
1265:    }
1266:
1267:    }
1268:
1269:    }
1270:
1271:    }
1272:
1273:    }
1274:
1275:    }
1276:
1277:    }
1278:
1279:    }
1280:
1281:    }
1282:
1283:    }
1284:
1285:    }
1286:
1287:    }
1288:
1289:    }
1290:
1291:    }
1292:
1293:    }
1294:
1295:    }
1296:
1297:    }
1298:
1299:    }
1300:
1301:    }
1302:
1303:    }
1304:
1305:    }
1306:
1307:    }
1308:
1309:    }
1310:
1311:    }
1312:
1313:    }
1314:
1315:    }
1316:
1317:    }
1318:
1319:    }
1320:
1321:    }
1322:
1323:    }
1324:
1325:    }
1326:
1327:    }
1328:
1329:    }
1330:
1331:    }
1332:
1333:    }
1334:
1335:    }
1336:
1337:    }
1338:
1339:    }
1340:
1341:    }
1342:
1343:    }
1344:
1345:    }
1346:
1347:    }
1348:
1349:    }
1350:
1351:    }
1352:
1353:    }
1354:
1355:    }
1356:
1357:    }
1358:
1359:    }
1360:
1361:    }
1362:
1363:    }
1364:
1365:    }
1366:
1367:    }
1368:
1369:    }
1370:
1371:    }
1372:
1373:    }
1374:
1375:    }
1376:
1377:    }
1378:
1379:    }
1380:
1381:    }
1382:
1383:    }
1384:
1385:    }
1386:
1387:    }
1388:
1389:    }
1390:
1391:    }
1392:
1393:    }
1394:
1395:    }
1396:
1397:    }
1398:
1399:    }
1400:
1401:    }
1402:
1403:    }
1404:
1405:    }
1406:
1407:    }
1408:
1409:    }
1410:
1411:    }
1412:
1413:    }
1414:
1415:    }
1416:
1417:    }
1418:
1419:    }
1420:
1421:    }
1422:
1423:    }
1424:
1425:    }
1426:
1427:    }
1428:
1429:    }
1430:
1431:    }
1432:
1433:    }
1434:
1435:    }
1436:
1437:    }
1438:
1439:    }
1440:
1441:    }
1442:
1443:    }
1444:
1445:    }
1446:
1447:    }
1448:
1449:    }
1450:
1451:    }
1452:
1453:    }
1454:
1455:    }
1456:
1457:    }
1458:
1459:    }
1460:
1461:    }
1462:
1463:    }
1464:
1465:    }
1466:
1467:    }
1468:
1469:    }
1470:
1471:    }
1472:
1473:    }
1474:
1475:    }
1476:
1477:    }
1478:
1479:    }
1480:
1481:    }
1482:
1483:    }
1484:
1485:    }
1486:
1487:    }
1488:
1489:    }
1490:
1491:    }
1492:
1493:    }
1494:
1495:    }
1496:
1497:    }
1498:
1499:    }
1500:
1501:    }
1502:
1503:    }
1504:
1505:    }
1506:
1507:    }
1508:
1509:    }
1510:
1511:    }
1512:
1513:    }
1514:
1515:    }
1516:
1517:    }
1518:
1519:    }
1520:
1521:    }
1522:
1523:    }
1524:
1525:    }
1526:
1527:    }
1528:
1529:    }
1530:
1531:    }
1532:
1533:    }
1534:
1535:    }
1536:
1537:    }
1538:
1539:    }
1540:
1541:    }
1542:
1543:    }
1544:
1545:    }
1546:
1547:    }
1548:
1549:    }
1550:
1551:    }
1552:
1553:    }
1554:
1555:    }
1556:
1557:    }
1558:
1559:    }
1560:
1561:    }
1562:
1563:    }
1564:
1565:    }
1566:
1567:    }
1568:
1569:    }
1570:
1571:    }
1572:
1573:    }
1574:
1575:   
```

```
1116:         /// <summary>
1117:         /// Looks up a localized string similar to Arama Sonuçlari.
1118:         /// </summary>
1119:         public static string SearchResults {
1120:             get {
1121:                 return ResourceManager.GetString("SearchResults", resourceCulture);
1122:             }
1123:         }
1124:
1125:         /// <summary>
1126:         /// Looks up a localized string similar to Arama Alanini Doldurmak
Zorundasiniz..
1127:         /// </summary>
1128:         public static string SearchTrimFault {
1129:             get {
1130:                 return ResourceManager.GetString("SearchTrimFault", resourceCulture);
1131:             }
1132:         }
1133:
1134:         /// <summary>
1135:         /// Looks up a localized string similar to Seçiniz.
1136:         /// </summary>
1137:         public static string Select {
1138:             get {
1139:                 return ResourceManager.GetString("Select", resourceCulture);
1140:             }
1141:         }
1142:
1143:         /// <summary>
1144:         /// Looks up a localized string similar to Tümünü Seç.
1145:         /// </summary>
1146:         public static string SelectAll {
1147:             get {
1148:                 return ResourceManager.GetString("SelectAll", resourceCulture);
1149:             }
1150:         }
1151:
1152:         /// <summary>
1153:         /// Looks up a localized string similar to Arka Plan Rengi Seçiniz.
1154:         /// </summary>
1155:         public static string SelectBackgroundColor {
1156:             get {
1157:                 return ResourceManager.GetString("SelectBackgroundColor",
resourceCulture);
1158:             }
1159:         }
1160:
1161:         /// <summary>
1162:         /// Looks up a localized string similar to Seçilen.
1163:         /// </summary>
1164:         public static string Selected {
1165:             get {
1166:                 return ResourceManager.GetString("Selected", resourceCulture);
1167:             }
1168:         }
1169:
1170:         /// <summary>
1171:         /// Looks up a localized string similar to Yazı Tipi Ailesi Seçiniz.
1172:         /// </summary>
1173:         public static string SelectFontFamily {
1174:             get {
1175:                 return ResourceManager.GetString("SelectFontFamily", resourceCulture);
1176:             }
1177:         }
1178:
1179:         /// <summary>
1180:         /// Looks up a localized string similar to Yazı Tipi Boyu Seçiniz.
1181:         /// </summary>
1182:         public static string SelectFontSize {
```

```
1183:         get {
1184:             return ResourceManager.GetString("SelectFontSize", resourceCulture);
1185:         }
1186:     }
1187:
1188:     /// <summary>
1189:     /// Looks up a localized string similar to Lütfen Dili Seçiniz.
1190:     /// </summary>
1191:     public static string SelectLanguage {
1192:         get {
1193:             return ResourceManager.GetString("SelectLanguage", resourceCulture);
1194:         }
1195:     }
1196:
1197:     /// <summary>
1198:     /// Looks up a localized string similar to Lütfen Not Klasörünü Seçiniz.
1199:     /// </summary>
1200:     public static string SelectNoteFolder {
1201:         get {
1202:             return ResourceManager.GetString("SelectNoteFolder", resourceCulture);
1203:         }
1204:     }
1205:
1206:     /// <summary>
1207:     /// Looks up a localized string similar to Sıralama Türü Seçiniz.
1208:     /// </summary>
1209:     public static string SelectOrderStyle {
1210:         get {
1211:             return ResourceManager.GetString("SelectOrderStyle", resourceCulture);
1212:         }
1213:     }
1214:
1215:     /// <summary>
1216:     /// Looks up a localized string similar to Sıralama Ölçütünü Seçiniz.
1217:     /// </summary>
1218:     public static string SelectOrderType {
1219:         get {
1220:             return ResourceManager.GetString("SelectOrderType", resourceCulture);
1221:         }
1222:     }
1223:
1224:     /// <summary>
1225:     /// Looks up a localized string similar to Sifre Durumunu Seçiniz.
1226:     /// </summary>
1227:     public static string SelectPasswordState {
1228:         get {
1229:             return ResourceManager.GetString("SelectPasswordState",
resourceCulture);
1230:         }
1231:     }
1232:
1233:     /// <summary>
1234:     /// Looks up a localized string similar to Awesome Note Uygulaması ile
Gönderildi.
1235:     /// </summary>
1236:     public static string SendWithApp {
1237:         get {
1238:             return ResourceManager.GetString("SendWithApp", resourceCulture);
1239:         }
1240:     }
1241:
1242:     /// <summary>
1243:     /// Looks up a localized string similar to E-Posta ile Gönder.
1244:     /// </summary>
1245:     public static string SendWithMail {
1246:         get {
1247:             return ResourceManager.GetString("SendWithMail", resourceCulture);
1248:         }
1249:     }
```

```
1250:
1251:     /// <summary>
1252:     /// Looks up a localized string similar to SMS Ile Gönder.
1253:     /// </summary>
1254:     public static string SendWithSMS {
1255:         get {
1256:             return ResourceManager.GetString("SendWithSMS", resourceCulture);
1257:         }
1258:     }
1259:
1260:     /// <summary>
1261:     /// Looks up a localized string similar to Sifre Ayarla.
1262:     /// </summary>
1263:     public static string SetPassword {
1264:         get {
1265:             return ResourceManager.GetString("SetPassword", resourceCulture);
1266:         }
1267:     }
1268:
1269:     /// <summary>
1270:     /// Looks up a localized string similar to Ayarlar.
1271:     /// </summary>
1272:     public static string Settings {
1273:         get {
1274:             return ResourceManager.GetString("Settings", resourceCulture);
1275:         }
1276:     }
1277:
1278:     /// <summary>
1279:     /// Looks up a localized string similar to Paylas.
1280:     /// </summary>
1281:     public static string Share {
1282:         get {
1283:             return ResourceManager.GetString("Share", resourceCulture);
1284:         }
1285:     }
1286:
1287:     /// <summary>
1288:     /// Looks up a localized string similar to Baglan.
1289:     /// </summary>
1290:     public static string SignIn {
1291:         get {
1292:             return ResourceManager.GetString("SignIn", resourceCulture);
1293:         }
1294:     }
1295:
1296:     /// <summary>
1297:     /// Looks up a localized string similar to Çikis Yap.
1298:     /// </summary>
1299:     public static string SignOut {
1300:         get {
1301:             return ResourceManager.GetString("SignOut", resourceCulture);
1302:         }
1303:     }
1304:
1305:     /// <summary>
1306:     /// Looks up a localized string similar to Ispanyolca.
1307:     /// </summary>
1308:     public static string Spanish {
1309:         get {
1310:             return ResourceManager.GetString("Spanish", resourceCulture);
1311:         }
1312:     }
1313:
1314:     /// <summary>
1315:     /// Looks up a localized string similar to Not Basarili Bir Sekilde
Olusturuldu.
1316:     /// </summary>
1317:     public static string SuccessfulAddNote {
```

```
1318:         get {
1319:             return ResourceManager.GetString("SuccessfulAddNote",
resourceCulture);
1320:         }
1321:     }
1322:
1323:     /// <summary>
1324:     /// Looks up a localized string similar to Not Klasörü Basarili Bir Sekilde
Olusturuldu.
1325:     /// </summary>
1326:     public static string SuccessfulAddNoteFolder {
1327:         get {
1328:             return ResourceManager.GetString("SuccessfulAddNoteFolder",
resourceCulture);
1329:         }
1330:     }
1331:
1332:     /// <summary>
1333:     /// Looks up a localized string similar to Arka Plan Rengi Basariyla
Degistirildi.
1334:     /// </summary>
1335:     public static string SuccessfulBackgroundColorChanged {
1336:         get {
1337:             return ResourceManager.GetString("SuccessfulBackgroundColorChanged",
resourceCulture);
1338:         }
1339:     }
1340:
1341:     /// <summary>
1342:     /// Looks up a localized string similar to Arka Plan Resmi Basarili Bir
Sekilde Degistirildi.
1343:     /// </summary>
1344:     public static string SuccessfulBackgroundImageChanged {
1345:         get {
1346:             return ResourceManager.GetString("SuccessfulBackgroundImageChanged",
resourceCulture);
1347:         }
1348:     }
1349:
1350:     /// <summary>
1351:     /// Looks up a localized string similar to Arka Plan Resmi Basarili Bir
Sekilde Kaldirildi.
1352:     /// </summary>
1353:     public static string SuccessfulBackgroundImageRemoved {
1354:         get {
1355:             return ResourceManager.GetString("SuccessfulBackgroundImageRemoved",
resourceCulture);
1356:         }
1357:     }
1358:
1359:     /// <summary>
1360:     /// Looks up a localized string similar to Not Klasör Ismi Basarili Bir
Sekilde Degistirildi.
1361:     /// </summary>
1362:     public static string SuccessfulChangeFolderName {
1363:         get {
1364:             return ResourceManager.GetString("SuccessfulChangeFolderName",
resourceCulture);
1365:         }
1366:     }
1367:
1368:     /// <summary>
1369:     /// Looks up a localized string similar to Notunuzun Ait Oldugu Klasör
Degistirildi.
1370:     /// </summary>
1371:     public static string SuccessfulChangeNoteFolder {
1372:         get {
1373:             return ResourceManager.GetString("SuccessfulChangeNoteFolder",
resourceCulture);
```

```
1374:         }
1375:     }
1376:
1377:     /// <summary>
1378:     /// Looks up a localized string similar to Notun Ismi Basarili Bir Sekilde
Degistirildi.
1379:     /// </summary>
1380:     public static string SuccessfulChangeNoteName {
1381:     get {
1382:         return ResourceManager.GetString("SuccessfulChangeNoteName",
resourceCulture);
1383:     }
1384: }
1385:
1386:     /// <summary>
1387:     /// Looks up a localized string similar to Not Klasörü Basariyla Silindi.
1388:     /// </summary>
1389:     public static string SuccessfulDeleteFolder {
1390:     get {
1391:         return ResourceManager.GetString("SuccessfulDeleteFolder",
resourceCulture);
1392:     }
1393: }
1394:
1395:     /// <summary>
1396:     /// Looks up a localized string similar to Not Basarili Bir Sekilde Silindi.
1397:     /// </summary>
1398:     public static string SuccessfulDeleteNote {
1399:     get {
1400:         return ResourceManager.GetString("SuccessfulDeleteNote",
resourceCulture);
1401:     }
1402: }
1403:
1404:     /// <summary>
1405:     /// Looks up a localized string similar to Yazi Tipi Ailesi Basarili Bir
Sekilde Degistirildi.
1406:     /// </summary>
1407:     public static string SuccessfulFontFamilyChanged {
1408:     get {
1409:         return ResourceManager.GetString("SuccessfulFontFamilyChanged",
resourceCulture);
1410:     }
1411: }
1412:
1413:     /// <summary>
1414:     /// Looks up a localized string similar to Yazi Tipi Boyutu Basarili Bir
Sekilde Degistirildi.
1415:     /// </summary>
1416:     public static string SuccessfulFontSizeChanged {
1417:     get {
1418:         return ResourceManager.GetString("SuccessfulFontSizeChanged",
resourceCulture);
1419:     }
1420: }
1421:
1422:     /// <summary>
1423:     /// Looks up a localized string similar to Siralama Türü Basarili Bir
Sekilde Degistirildi.
1424:     /// </summary>
1425:     public static string SuccessfulOrderStyleChanged {
1426:     get {
1427:         return ResourceManager.GetString("SuccessfulOrderStyleChanged",
resourceCulture);
1428:     }
1429: }
1430:
1431:     /// <summary>
1432:     /// Looks up a localized string similar to Siralama Kriteri Basarili Bir
```

```
Sekilde Degistirildi.
1433:         /// </summary>
1434:         public static string SuccessfulOrderTypeChanged {
1435:             get {
1436:                 return ResourceManager.GetString("SuccessfulOrderTypeChanged",
resourceCulture);
1437:             }
1438:         }
1439:
1440:         /// <summary>
1441:         /// Looks up a localized string similar to Sifre Basarili Bir Sekilde
Kaldirildi.
1442:         /// </summary>
1443:         public static string SuccessfulRemovePassword {
1444:             get {
1445:                 return ResourceManager.GetString("SuccessfulRemovePassword",
resourceCulture);
1446:             }
1447:         }
1448:
1449:         /// <summary>
1450:         /// Looks up a localized string similar to Arka Plan Ayarlari Basariyla
Sifirlandi.
1451:         /// </summary>
1452:         public static string SuccessfulResetSettings {
1453:             get {
1454:                 return ResourceManager.GetString("SuccessfulResetSettings",
resourceCulture);
1455:             }
1456:         }
1457:
1458:         /// <summary>
1459:         /// Looks up a localized string similar to E-Posta Yoluyla Not Basarili Bir
Sekilde Gönderildi.
1460:         /// </summary>
1461:         public static string SuccessfulSendWithMail {
1462:             get {
1463:                 return ResourceManager.GetString("SuccessfulSendWithMail",
resourceCulture);
1464:             }
1465:         }
1466:
1467:         /// <summary>
1468:         /// Looks up a localized string similar to SMS Yoluyla Not Basarili Bir
Sekilde Gönderildi.
1469:         /// </summary>
1470:         public static string SuccessfulSendWithSMS {
1471:             get {
1472:                 return ResourceManager.GetString("SuccessfulSendWithSMS",
resourceCulture);
1473:             }
1474:         }
1475:
1476:         /// <summary>
1477:         /// Looks up a localized string similar to Sifre Basarili Bir Sekilde
Olusturuldu.
1478:         /// </summary>
1479:         public static string SuccessfulSetPassword {
1480:             get {
1481:                 return ResourceManager.GetString("SuccessfulSetPassword",
resourceCulture);
1482:             }
1483:         }
1484:
1485:         /// <summary>
1486:         /// Looks up a localized string similar to Eszamanlama.
1487:         /// </summary>
1488:         public static string Sync {
1489:             get {
```

```

1490:         return ResourceManager.GetString("Sync", resourceCulture);
1491:     }
1492: }
1493:
1494: /// <summary>
1495: /// Looks up a localized string similar to Senkronizasyon Özellikleri.
1496: /// </summary>
1497: public static string SyncFeature {
1498:     get {
1499:         return ResourceManager.GetString("SyncFeature", resourceCulture);
1500:     }
1501: }
1502:
1503: /// <summary>
1504: /// Looks up a localized string similar to Program vasitasiyla tüm
notlarinizi OneDrive hesabiniz ile senkronize edebilirsiniz. Uygulama içerisinde Genel
Ayarlar kısmında Senkronizasyon bölümüne girdikten sonra Sign In yazan butona tıklayıp,
OneDrive hesabınıza giriş yapmanız gerekmektedir. Giriş esnasında sorduğu erişim iznini
onaylamadığınız zaman senkronizasyon işlemini gerçekleştirmeyecektir. Sonrasında ise
Senkronizasyon için gerekli butona bastığınız zaman program OneDrive hesabınız içerisinde
bir klasör oluşturacak (örneğin: Awesome Note (21. [rest of string was truncated]&quot;;.
1505: /// </summary>
1506: public static string SyncFeatureNote {
1507:     get {
1508:         return ResourceManager.GetString("SyncFeatureNote", resourceCulture);
1509:     }
1510: }
1511:
1512: /// <summary>
1513: /// Looks up a localized string similar to Hepsini Tek Dosyada Birleştir.
1514: /// </summary>
1515: public static string SyncOnOneFile {
1516:     get {
1517:         return ResourceManager.GetString("SyncOnOneFile", resourceCulture);
1518:     }
1519: }
1520:
1521: /// <summary>
1522: /// Looks up a localized string similar to Eszamanlanıyor....
1523: /// </summary>
1524: public static string Synchronizing {
1525:     get {
1526:         return ResourceManager.GetString("Synchronizing", resourceCulture);
1527:     }
1528: }
1529:
1530: /// <summary>
1531: /// Looks up a localized string similar to Sistemde Bir Hata Olustu.
Uygulama Sonlanacak....
1532: /// </summary>
1533: public static string SystemFault {
1534:     get {
1535:         return ResourceManager.GetString("SystemFault", resourceCulture);
1536:     }
1537: }
1538:
1539: /// <summary>
1540: /// Looks up a localized string similar to Tay Dili.
1541: /// </summary>
1542: public static string Thai {
1543:     get {
1544:         return ResourceManager.GetString("Thai", resourceCulture);
1545:     }
1546: }
1547:
1548: /// <summary>
1549: /// Looks up a localized string similar to Programın Ipuçlari.
1550: /// </summary>
1551: public static string Tips {

```



```
1552:         get {
1553:             return ResourceManager.GetString("Tips", resourceCulture);
1554:         }
1555:     }
1556:
1557:     /// <summary>
1558:     /// Looks up a localized string similar to Türkçe.
1559:     /// </summary>
1560:     public static string Turkish {
1561:         get {
1562:             return ResourceManager.GetString("Turkish", resourceCulture);
1563:         }
1564:     }
1565:
1566:     /// <summary>
1567:     /// Looks up a localized string similar to Girilen İki Sifre Alani Aynı
1568:     /// </summary>
1569:     public static string TwoFieldsNotSame {
1570:         get {
1571:             return ResourceManager.GetString("TwoFieldsNotSame", resourceCulture);
1572:         }
1573:     }
1574:
1575:     /// <summary>
1576:     /// Looks up a localized string similar to Hosgeldiniz.
1577:     /// </summary>
1578:     public static string Welcome {
1579:         get {
1580:             return ResourceManager.GetString("Welcome", resourceCulture);
1581:         }
1582:     }
1583:
1584:     /// <summary>
1585:     /// Looks up a localized string similar to Merhaba. Not almayi seven bir
1586:     /// kullanıcı olarak Bilgisayar ve Mobil cihazlarda pek çok not tutma programi kullandim. Ve
1587:     /// faydali olacagini düşündüğüm özellikleri birlestirip bu uygulamayi yazdim. Umarim
1588:     /// uygulamadan memnun kalirsiniz..
1589:     /// </summary>
1590:     public static string WelcomeNote {
1591:         get {
1592:             return ResourceManager.GetString("WelcomeNote", resourceCulture);
1593:         }
1594:     }
1595:
1596:     /// <summary>
1597:     /// Looks up a localized string similar to Kelime Sayisi.
1598:     /// </summary>
1599:     public static string WordNumber {
1600:         get {
1601:             return ResourceManager.GetString("WordNumber", resourceCulture);
1602:         }
1603:     }
1604:
1605:     /// <summary>
1606:     /// Looks up a localized string similar to Sari.
1607:     /// </summary>
1608:     public static string Yellow {
1609:         get {
1610:             return ResourceManager.GetString("Yellow", resourceCulture);
1611:         }
1612:     }
1613:
1614:     /// <summary>
1615:     /// Looks up a localized string similar to Evet.
1616:     /// </summary>
1617:     public static string Yes {
1618:         get {
1619:             return ResourceManager.GetString("Yes", resourceCulture);
1620:         }
1621:     }
1622: }
```

```
1617:         }
1618:     }
1619: }
1620: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.4 AppSettings.cs

This is file AppSettings.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Data.Linq;
4: using System.Data.Linq.Mapping;
5: using System.Linq;
6: using System.Text;
7: using System.Threading.Tasks;
8:
9: namespace AwesomeNote
10: {
11:     [Table]
12:     public class AppSettings
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int AppSettingsId { get; set; }
19:
20:         [Column]
21:         public string AppLangName { get; set; }
22:
23:         [Column]
24:         public string AppBackgroundColor { get; set; }
25:
26:         [Column]
27:         public string FolderOrderBy { get; set; }
28:
29:         [Column]
30:         public string FolderOrderStyle { get; set; }
31:
32:         [Column(DbType = "Image", UpdateCheck = UpdateCheck.Never)]
33:         public byte[] AppBackgroundImage { get; set; }
34:     }
35: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.5 AssemblyInfo.cs

This is file AssemblyInfo.cs.

Body Source

```

1: ?using System.Reflection;
2: using System.Runtime.CompilerServices;
3: using System.Runtime.InteropServices;
4: using System.Resources;
5:
6: // General Information about an assembly is controlled through the following
7: // set of attributes. Change these attribute values to modify the information
8: // associated with an assembly.
9: [assembly: AssemblyTitle("Awesome Note")]
10: [assembly: AssemblyDescription("Best Note Taking App Ever")]
11: [assembly: AssemblyConfiguration("")]
12: [assembly: AssemblyCompany("CMS Software")]
13: [assembly: AssemblyProduct("")]
14: [assembly: AssemblyCopyright("Copyright © 2014")]
15: [assembly: AssemblyTrademark("")]
16: [assembly: AssemblyCulture("")]
17:
18: // Setting ComVisible to false makes the types in this assembly not visible
19: // to COM components. If you need to access a type in this assembly from
20: // COM, set the ComVisible attribute to true on that type.
21: [assembly: ComVisible(false)]
22:
23: // The following GUID is for the ID of the typelib if this project is exposed to COM
24: [assembly: Guid("bb41b552-c2a0-480b-a4f9-eb99531c1b11")]
25:
26: // Version information for an assembly consists of the following four values:
27: //
28: //      Major Version
29: //      Minor Version
30: //      Build Number
31: //      Revision
32: //
33: // You can specify all the values or you can default the Revision and Build Numbers
34: // by using the '*' as shown below:
35: [assembly: AssemblyVersion("1.0.3.0")]
36: [assembly: AssemblyFileVersion("1.0.3.0")]
37: [assembly: NeutralResourcesLanguageAttribute("tr-TR")]

```

1.2.6 AutoScrollBehavior.cs

This is file AutoScrollBehavior.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5: using System.Threading.Tasks;
6: using System.Windows.Controls;
7: using System.Windows.Interactivity;
8:
9: namespace AwesomeNote
10: {
11:     public class AutoScrollBehavior : Behavior<ScrollViewer>
12:     {
13:         private ScrollViewer _scrollViewer = null;
14:         private double _height = 0.0d;
15:
16:         protected override void OnAttached()
17:         {
18:             base.OnAttached();
19:

```

```
20:         this._scrollViewer = base.AssociatedObject;
21:         this._scrollViewer.LayoutUpdated += new
EventHandler(_scrollViewer_LayoutUpdated);
22:     }
23:
24:     private void _scrollViewer_LayoutUpdated(object sender, EventArgs e)
25:     {
26:         if (this._scrollViewer.ExtentHeight != _height)
27:         {
28:             this._scrollViewer.ScrollToVerticalOffset(this._scrollViewer.ExtentHeight);
29:             this._height = this._scrollViewer.ExtentHeight;
30:         }
31:     }
32:
33:     protected override void OnDetaching()
34:     {
35:         base.OnDetaching();
36:
37:         if (this._scrollViewer != null)
38:             this._scrollViewer.LayoutUpdated -= new
EventHandler(_scrollViewer_LayoutUpdated);
39:     }
40: }
41: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.7 Awesome Note.csproj

This is file Awesome Note.csproj.

1.2.8 AwesomeNote.sln

This is file AwesomeNote.sln.

1.2.9 AwesomeNoteDataContext.cs

This is file AwesomeNoteDataContext.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Data.Linq;
4: using System.Linq;
5: using System.Text;
6: using System.Threading.Tasks;
7:
8: namespace AwesomeNote
9: {
10:     public class NoteFolderDataContext : DataContext
11:     {
```

```

12:         public const string ConnectionString = @"Data
Source=isostore:/NoteAndFolders.sdf";
13:         public NoteFolderDataContext(string connectionString)
14:             : base(connectionString) { }
15:         public Table<NoteFolder> NoteFolders;
16:         public Table<Note> Notes;
17:         public Table<AppSettings> AppSettings;
18:     }
19: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.10 BackgroundColorSettings.xaml.cs

This is file BackgroundColorSettings.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class BackgroundColorSettings : PhoneApplicationPage
18:     {
19:         public SolidColorBrush messageBackGround;
20:         public BackgroundColorSettings()
21:         {
22:             InitializeComponent();
23:
24:             lstBackgroundColor.Items.Clear();
25:             lstBackgroundColor.Items.Add(AppResources.Black);
26:             lstBackgroundColor.Items.Add(AppResources.Blue);
27:             lstBackgroundColor.Items.Add(AppResources.Brown);
28:             lstBackgroundColor.Items.Add(AppResources.Gray);
29:             lstBackgroundColor.Items.Add(AppResources.Green);
30:             lstBackgroundColor.Items.Add(AppResources.Orange);
31:             lstBackgroundColor.Items.Add(AppResources.Purple);
32:             lstBackgroundColor.Items.Add(AppResources.Red);
33:             lstBackgroundColor.Items.Add(AppResources.Yellow);
34:             lstBackgroundColor.SelectedIndex = -1;
35:
36:             lblBackgroundColor.Text = AppResources.SelectBackgroundColor;
37:
38:         }
39:
40:         private void lstBackgroundColor_SelectionChanged(object sender,
SelectionChangedEventArgs e)
41:         {
42:             int index = lstBackgroundColor.SelectedIndex;
43:             string backgroundColor = "";

```

```

44:         if (index == 0)
45:         {
46:             backgroundColor = "BLA";
47:         }
48:         else if (index == 1)
49:         {
50:             backgroundColor = "BLU";
51:         }
52:         else if (index == 2)
53:         {
54:             backgroundColor = "BRO";
55:         }
56:         else if (index == 3)
57:         {
58:             backgroundColor = "GRA";
59:         }
60:         else if (index == 4)
61:         {
62:             backgroundColor = "GRE";
63:         }
64:         else if (index == 5)
65:         {
66:             backgroundColor = "ORA";
67:         }
68:         else if (index == 6)
69:         {
70:             backgroundColor = "PUR";
71:         }
72:         else if (index == 7)
73:         {
74:             backgroundColor = "RED";
75:         }
76:         else if (index == 8)
77:         {
78:             backgroundColor = "YEL";
79:         }
80:         else
81:         {
82:             backgroundColor = "BLA";
83:         }
84:
85:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
86:         {
87:             var appSettings = context.AppSettings;
88:             foreach (var appSetting in appSettings)
89:             {
90:                 appSetting.AppBackgroundColor = backgroundColor;
91:             }
92:             context.SubmitChanges();
93:             //CustomMessageBox messageBox = new CustomMessageBox()
94:             //{
95:             //    Caption = AppResources.BackgroundColor,
96:             //    Message = AppResources.SuccessfulBackgroundColorChanged,
97:             //    Background = messageBackGround
98:             //};
99:             //messageBox.Show();
100:             MessageBox.Show(AppResources.SuccessfulBackgroundColorChanged);
101:         }
102:         SetBackgroundColor();
103:         NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
104:     }
105:
106:     private void SetBackgroundColor()
107:     {
108:         AppSettings appSettings = new AppSettings();
109:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))

```

```

110:         {
111:             appSettings = context.AppSettings.First() as AppSettings;
112:         }
113:
114:         if (appSettings.AppBackgroundImage != null)
115:         {
116:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
117:             BitmapImage image = new BitmapImage();
118:             image.SetSource(stream);
119:             ImageBrush ib = new ImageBrush();
120:             ib.ImageSource = image;
121:             this.LayoutRoot.Background = ib;
122:         }
123:         else
124:         {
125:             switch (appSettings.AppBackgroundColor)
126:             {
127:                 case "BLA":
128:                     messageBackGround = new SolidColorBrush(Colors.Black);
129:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
130:                     break;
131:                 case "BLU":
132:                     messageBackGround = new SolidColorBrush(Colors.Blue);
133:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
134:                     break;
135:                 case "BRO":
136:                     messageBackGround = new SolidColorBrush(Colors.Brown);
137:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
138:                     break;
139:                 case "RED":
140:                     messageBackGround = new SolidColorBrush(Colors.Red);
141:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
142:                     break;
143:                 case "GRE":
144:                     messageBackGround = new SolidColorBrush(Colors.Green);
145:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
146:                     break;
147:                 case "GRA":
148:                     messageBackGround = new SolidColorBrush(Colors.Gray);
149:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
150:                     break;
151:                 case "YEL":
152:                     messageBackGround = new SolidColorBrush(Colors.Yellow);
153:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
154:                     break;
155:                 case "ORA":
156:                     messageBackGround = new SolidColorBrush(Colors.Orange);
157:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
158:                     break;
159:                 case "PUR":
160:                     messageBackGround = new SolidColorBrush(Colors.Purple);
161:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
162:                     break;
163:                 default:
164:                     messageBackGround = new SolidColorBrush(Colors.Black);
165:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
166:                     break;
167:             }
168:         }
169:     }
170: }
171: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.11 ChangeNoteFolder.xaml.cs

This is file ChangeNoteFolder.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Media;
8: using System.Windows.Navigation;
9: using AwesomeNote.Resources;
10: using Microsoft.Phone.Controls;
11: using Microsoft.Phone.Shell;
12:
13: namespace AwesomeNote
14: {
15:     public partial class ChangeNoteFolder : PhoneApplicationPage
16:     {
17:         public int noteId;
18:         public int noteFolderId;
19:         public SolidColorBrush messageBackGround;
20:
21:         public ChangeNoteFolder()
22:         {
23:             InitializeComponent();
24:             SetBackgroundColor();
25:         }
26:
27:         private void lstNoteFolder_SelectionChanged(object sender,
SelectionChangedEventArgs e)
28:         {
29:             NoteFolder temp = lstNoteFolders.SelectedItem as NoteFolder;
30:
31:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
32:             {
33:
34:                 var noteFolders1 = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(temp.NoteFolderId)).Select(j => j);
35:                 var noteFolders2 = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
36:                 var notes = context.Notes.Where(j => j.NoteId.Equals(noteId)).Select(j
=> j);
37:                 foreach (var note in notes)
38:                 {
39:                     note.NoteFolderId = temp.NoteFolderId;
40:                 }
41:
42:                 foreach (var noteFolder1 in noteFolders1)
43:                 {
44:                     noteFolder1.NoteCount = noteFolder1.NoteCount + 1;
45:                     noteFolder1.NameCount = noteFolder1.NoteFolderName + " (" +
noteFolder1.NoteCount.ToString() + ")";
46:                 }
47:             }
```



```

48:         foreach (var noteFolder2 in noteFolders2)
49:         {
50:             noteFolder2.NoteCount = noteFolder2.NoteCount - 1;
51:             noteFolder2.NameCount = noteFolder2.NoteFolderName + " (" +
noteFolder2.NoteCount.ToString() + ")";
52:         }
53:         context.SubmitChanges();
54:         //lstFolders.ItemsSource = context.NoteFolders;
55:         MessageBox.Show(AppResources.SuccessfulChangeNoteFolder);
56:     }
57:     this.NavigationService.Navigate(new Uri("/NoteDetail.xaml#" + noteId,
UriKind.Relative));
58:     }
59:
60:     protected override void OnNavigatedTo(NavigationEventArgs e)
61:     {
62:         base.OnNavigatedTo(e);
63:     }
64:
65:     protected override void OnNavigatedFrom(NavigationEventArgs e)
66:     {
67:         base.OnNavigatedFrom(e);
68:     }
69:
70:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
71:     {
72:         // displays "Fragment: Detail"
73:         //MessageBox.Show("Folder Id: " + e.Fragment);
74:         base.OnFragmentNavigation(e);
75:         noteId = int.Parse(e.Fragment);
76:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
77:         {
78:             lstNoteFolders.Items.Clear();
79:             var note = context.Notes.Where(j => j.NoteId.Equals(noteId)).Single()
as Note;
80:             noteFolderId = note.NoteFolderId;
81:             lblNoteName.Text = note.NoteName;
82:             lblChangeFolder.Text = AppResources.SelectNoteFolder;
83:             var noteFolders = context.NoteFolders;
84:             lstNoteFolders.ItemsSource = noteFolders;
85:             lstNoteFolders.DisplayMemberPath = "NoteFolderName";
86:         }
87:     }
88:
89:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
90:     {
91:         if (this.NavigationService.CanGoBack)
92:         {
93:             this.NavigationService.Navigate(new Uri("/NoteDetail.xaml#" + noteId,
UriKind.Relative));
94:         }
95:     }
96:
97:     private void SetBackgroundColor()
98:     {
99:         AppSettings appSettings = new AppSettings();
100:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
101:         {
102:             appSettings = context.AppSettings.First() as AppSettings;
103:         }
104:
105:         switch (appSettings.AppBackgroundColor)
106:         {
107:             case "BLA":
108:                 messageBackGround = new SolidColorBrush(Colors.Black);
109:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);

```

```
110:         break;
111:     case "BLU":
112:         messageBackGround = new SolidColorBrush(Colors.Blue);
113:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
114:         break;
115:     case "BRO":
116:         messageBackGround = new SolidColorBrush(Colors.Brown);
117:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
118:         break;
119:     case "RED":
120:         messageBackGround = new SolidColorBrush(Colors.Red);
121:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
122:         break;
123:     case "GRE":
124:         messageBackGround = new SolidColorBrush(Colors.Green);
125:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
126:         break;
127:     case "GRA":
128:         messageBackGround = new SolidColorBrush(Colors.Gray);
129:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
130:         break;
131:     case "YEL":
132:         messageBackGround = new SolidColorBrush(Colors.Yellow);
133:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
134:         break;
135:     case "ORA":
136:         messageBackGround = new SolidColorBrush(Colors.Orange);
137:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
138:         break;
139:     case "PUR":
140:         messageBackGround = new SolidColorBrush(Colors.Purple);
141:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
142:         break;
143:     default:
144:         messageBackGround = new SolidColorBrush(Colors.Black);
145:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
146:         break;
147:     }
148: }
149:
150:
151: }
152: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.12 EnterPasswordPage.xaml.cs

This is file EnterPasswordPage.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Input;
9: using System.Windows.Media;
10: using System.Windows.Media.Imaging;
```

```

11: using System.Windows.Navigation;
12: using AwesomeNote.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16: namespace AwesomeNote
17: {
18:     public partial class EnterPasswordPage : PhoneApplicationPage
19:     {
20:
21:         public int noteFolderId;
22:         public int noteId;
23:         public NoteFolder noteFolder;
24:         public string pageName;
25:         public Note note;
26:         public SolidColorBrush messageBackGround;
27:
28:         public EnterPasswordPage()
29:         {
30:             InitializeComponent();
31:             //SetBackgroundColor();
32:             //txtPassword.Focus();
33:         }
34:
35:         protected override void OnNavigatedTo(NavigationEventArgs e)
36:         {
37:             base.OnNavigatedTo(e);
38:             // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor
39:             var lastPage = NavigationService.BackStack.FirstOrDefault();
40:             pageName = lastPage.Source.ToString();
41:             txtPassword.Focus();
42:         }
43:
44:         protected override void OnNavigatedFrom(NavigationEventArgs e)
45:         {
46:             base.OnNavigatedFrom(e);
47:         }
48:
49:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
50:         {
51:             // displays "Fragment: Detail"
52:             //MessageBox.Show("Folder Id: " + e.Fragment);
53:
54:             base.OnFragmentNavigation(e);
55:             if (pageName == "/MainPage.xaml")
56:             {
57:                 noteFolderId = int.Parse(e.Fragment);
58:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
59:                 {
60:                     var noteFolders =
61:                         context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
62:                     noteFolder = noteFolders;
63:                     lblFolderName.Text = noteFolders.NoteFolderName;
64:                 }
65:             }
66:             else
67:             {
68:                 noteId = int.Parse(e.Fragment);
69:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
70:                 {
71:                     var note = context.Notes.Where(j =>
j.NoteId.Equals(noteId)).Single() as Note;
72:                     noteFolderId = note.NoteFolderId;
73:                     var noteFolders =
74:                         context.NoteFolders.Where(j =>

```

```

j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
75:         noteFolder = noteFolders;
76:         lblFolderName.TextWrapping = TextWrapping.Wrap;
77:         //lblFolderName.Text = AppResources.SearchResults + " (" +
noteFolders.NoteFolderName + ")";
78:         lblFolderName.Text = noteFolders.NoteFolderName;
79:     }
80: }
81: lblEnterPassword.Text = AppResources.EnterPassword;
82: txtLabel.Text = AppResources.PleaseEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
83: //btnCancel.Content = AppResources.Cancel;
84: //btnOK.Content = AppResources.OK;
85: SetBackgroundColor();
86:
87: }
88:
89: private void btnOK_Click(object sender, RoutedEventArgs e)
90: {
91:     string password;
92:     password = txtPassword.Password;
93:     // girilen sifre dogruysa
94:     if (password.Length >= 4 && password == noteFolder.NoteFolderPassword)
95:     {
96:         MessageBox.Show(AppResources.PasswordTrue);
97:         if (pageName == "/MainPage.xaml")
98:         {
99:             NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
noteFolderId, UriKind.Relative));
100:         }
101:         else
102:         {
103:             NavigationService.Navigate(new Uri("/NoteDetail.xaml#" + noteId,
UriKind.Relative));
104:         }
105:     }
106:     else
107:     {
108:         MessageBox.Show(AppResources.PasswordFalse);
109:         txtPassword.Password = "";
110:         txtPassword.Focus();
111:     }
112: }
113:
114: private void btnCancel_Click(object sender, RoutedEventArgs e)
115: {
116:     NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
117: }
118:
119: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
120: {
121:     if (pageName == "/MainPage.xaml")
122:     {
123:         NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
124:     }
125:     else
126:     {
127:         NavigationService.Navigate(new Uri("/SearchPage.xaml",
UriKind.Relative));
128:     }
129: }
130:
131: private void txtPassword_GotFocus(object sender, RoutedEventArgs e)
132: {
133:     this.svPassword.ScrollToVerticalOffset(this.txtPassword.ActualHeight);
134:     this.svPassword.UpdateLayout();
135: }

```

```

136:
137:     private void txtPassword_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
138:     {
139:         svPassword.ScrollToVerticalOffset(e.GetPosition(txtPassword).Y - 120);
140:     }
141:
142:     private void txtPassword_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
143:     {
144:         if (e.Key == Key.Enter)
145:         {
146:             string password;
147:             password = txtPassword.Password;
148:             // girilen sifre dogruysa
149:             if (password.Length >= 4 && password == noteFolder.NoteFolderPassword)
150:             {
151:                 MessageBox.Show(AppResources.PasswordTrue);
152:                 if (pageName == "/MainPage.xaml")
153:                 {
154:                     NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
noteFolderId, UriKind.Relative));
155:                 }
156:                 else
157:                 {
158:                     NavigationService.Navigate(new Uri("/NoteDetail.xaml#" +
noteId, UriKind.Relative));
159:                 }
160:             }
161:             else
162:             {
163:                 MessageBox.Show(AppResources.PasswordFalse);
164:                 txtPassword.Password = "";
165:                 txtPassword.Focus();
166:             }
167:         }
168:     }
169:
170:     private void SetBackgroundColor()
171:     {
172:         AppSettings appSettings = new AppSettings();
173:         NoteFolder noteFolder;
174:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
175:         {
176:             appSettings = context.AppSettings.First() as AppSettings;
177:             noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
178:         }
179:
180:         if (noteFolder.FolderBackground != null)
181:         {
182:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
183:             BitmapImage image = new BitmapImage();
184:             image.SetSource(stream);
185:             ImageBrush ib = new ImageBrush();
186:             ib.ImageSource = image;
187:             this.LayoutRoot.Background = ib;
188:         }
189:         else
190:         {
191:             switch (appSettings.AppBackgroundColor)
192:             {
193:                 case "BLA":
194:                     messageBackGround = new SolidColorBrush(Colors.Black);
195:                     //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
196:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
197:                     break;

```

```
198:         case "BLU":
199:             messageBackGround = new SolidColorBrush(Colors.Blue);
200:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
201:             break;
202:         case "BRO":
203:             messageBackGround = new SolidColorBrush(Colors.Brown);
204:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
205:             break;
206:         case "RED":
207:             messageBackGround = new SolidColorBrush(Colors.Red);
208:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
209:             break;
210:         case "GRE":
211:             messageBackGround = new SolidColorBrush(Colors.Green);
212:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
213:             break;
214:         case "GRA":
215:             messageBackGround = new SolidColorBrush(Colors.Gray);
216:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
217:             break;
218:         case "YEL":
219:             messageBackGround = new SolidColorBrush(Colors.Yellow);
220:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
221:             break;
222:         case "ORA":
223:             messageBackGround = new SolidColorBrush(Colors.Orange);
224:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
225:             break;
226:         case "PUR":
227:             messageBackGround = new SolidColorBrush(Colors.Purple);
228:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
229:             break;
230:         default:
231:             messageBackGround = new SolidColorBrush(Colors.Black);
232:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
233:             break;
234:     }
235: }
236: }
237: }
238: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.13 FolderOrderSettings.xaml.cs

This is file FolderOrderSettings.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
```

```

10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class FolderOrderSettings : PhoneApplicationPage
18:     {
19:         public SolidColorBrush messageBackGround;
20:         public FolderOrderSettings()
21:         {
22:             InitializeComponent();
23:             SetBackgroundColor();
24:
25:             lstFolderOrder.Items.Clear();
26:             lstFolderOrder.Items.Add(AppResources.FolderName);
27:             lstFolderOrder.Items.Add(AppResources.CreationDate);
28:             lstFolderOrder.Items.Add(AppResources.ModificationDate);
29:             lstFolderOrder.SelectedIndex = -1;
30:
31:             lblFolderOrder.Text = AppResources.SelectOrderType;
32:         }
33:
34:         private void lstFolderOrder_SelectionChanged(object sender,
35: SelectionChangedEventArgs e)
36:         {
37:             int index = lstFolderOrder.SelectedIndex;
38:             string orderType = "";
39:             if (index == 0)
40:             {
41:                 orderType = "NAME";
42:             }
43:             else if (index == 1)
44:             {
45:                 orderType = "CDATE";
46:             }
47:             else if (index == 2)
48:             {
49:                 orderType = "MDATE";
50:             }
51:             else
52:             {
53:                 orderType = "NAME";
54:             }
55:
56:             using (var context = new
57: NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
58:             {
59:                 var appSettings = context.AppSettings;
60:                 foreach (var appSetting in appSettings)
61:                 {
62:                     appSetting.FolderOrderBy = orderType;
63:                 }
64:                 context.SubmitChanges();
65:                 //CustomMessageBox messageBox = new CustomMessageBox()
66:                 //{
67:                 //    Caption = AppResources.NoteOrder,
68:                 //    Message = AppResources.SuccessfulOrderTypeChanged,
69:                 //    Background = messageBackGround,
70:                 //    LeftButtonContent = AppResources.OK
71:                 //};
72:                 //messageBox.Dismissed += (s1, e1) =>
73:                 //{
74:                 //    switch (e1.Result)
75:                 //    {
76:                 //        case CustomMessageBoxResult.LeftButton:
77:                 //            // Do something.
78:                 //            break;
79:                 //    }
80:                 //}
81:             }
82:         }
83:     }
84: }

```

```

77:         //         case CustomMessageBoxResult.RightButton:
78:         //             // Do something.
79:         //             break;
80:         //         case CustomMessageBoxResult.None:
81:         //             // Do something.
82:         //             break;
83:         //         default:
84:         //             break;
85:         //     }
86:     //};
87:     //messageBox.Show();
88:     MessageBox.Show(AppResources.SuccessfulOrderTypeChanged);
89: }
90: NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
91: }
92:
93: protected override void OnNavigatedTo(NavigationEventArgs e)
94: {
95:     base.OnNavigatedTo(e);
96: }
97:
98: protected override void OnNavigatedFrom(NavigationEventArgs e)
99: {
100:     base.OnNavigatedFrom(e);
101: }
102:
103: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
104: {
105:     if (this.NavigationService.CanGoBack)
106:     {
107:         this.NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
108:     }
109: }
110:
111: private void SetBackgroundColor()
112: {
113:     AppSettings appSettings = new AppSettings();
114:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
115:     {
116:         appSettings = context.AppSettings.First() as AppSettings;
117:     }
118:
119:     if (appSettings.AppBackgroundImage != null)
120:     {
121:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
122:         BitmapImage image = new BitmapImage();
123:         image.SetSource(stream);
124:         ImageBrush ib = new ImageBrush();
125:         ib.ImageSource = image;
126:         this.LayoutRoot.Background = ib;
127:     }
128:     else
129:     {
130:         switch (appSettings.AppBackgroundColor)
131:         {
132:             case "BLA":
133:                 messageBackGround = new SolidColorBrush(Colors.Black);
134:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
135:                 break;
136:             case "BLU":
137:                 messageBackGround = new SolidColorBrush(Colors.Blue);
138:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
139:                 break;
140:             case "BRO":
141:                 messageBackGround = new SolidColorBrush(Colors.Brown);

```



```
142:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
143:         break;
144:     case "RED":
145:         messageBackGround = new SolidColorBrush(Colors.Red);
146:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
147:         break;
148:     case "GRE":
149:         messageBackGround = new SolidColorBrush(Colors.Green);
150:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
151:         break;
152:     case "GRA":
153:         messageBackGround = new SolidColorBrush(Colors.Gray);
154:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
155:         break;
156:     case "YEL":
157:         messageBackGround = new SolidColorBrush(Colors.Yellow);
158:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
159:         break;
160:     case "ORA":
161:         messageBackGround = new SolidColorBrush(Colors.Orange);
162:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
163:         break;
164:     case "PUR":
165:         messageBackGround = new SolidColorBrush(Colors.Purple);
166:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
167:         break;
168:     default:
169:         messageBackGround = new SolidColorBrush(Colors.Black);
170:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
171:         break;
172:     }
173: }
174: }
175: }
176: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.14 FolderOrderStyleSettings.xaml.cs

This is file FolderOrderStyleSettings.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
```

```

16: {
17:     public partial class FolderOrderStyleSettings : PhoneApplicationPage
18:     {
19:         public SolidColorBrush messageBackGround;
20:         public FolderOrderStyleSettings()
21:         {
22:             InitializeComponent();
23:             SetBackgroundColor();
24:
25:             lstFolderOrderStyle.Items.Clear();
26:             lstFolderOrderStyle.Items.Add(AppResources.Ascending);
27:             lstFolderOrderStyle.Items.Add(AppResources.Descending);
28:             lstFolderOrderStyle.SelectedIndex = -1;
29:
30:             lblFolderOrderStyle.Text = AppResources.SelectOrderStyle;
31:         }
32:
33:         private void lstFolderOrderStyle_SelectionChanged(object sender,
SelectionChangedEventArgs e)
34:         {
35:             int index = lstFolderOrderStyle.SelectedIndex;
36:             string orderStyle = "";
37:             if (index == 0)
38:             {
39:                 orderStyle = "A";
40:             }
41:             else if (index == 1)
42:             {
43:                 orderStyle = "D";
44:             }
45:             else
46:             {
47:                 orderStyle = "A";
48:             }
49:
50:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
51:             {
52:                 var appSettings = context.AppSettings;
53:                 foreach (var appSetting in appSettings)
54:                 {
55:                     appSetting.FolderOrderStyle = orderStyle;
56:                 }
57:                 context.SubmitChanges();
58:                 MessageBox.Show(AppResources.SuccessfulOrderStyleChanged);
59:             }
60:             NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
61:         }
62:
63:         protected override void OnNavigatedTo(NavigationEventArgs e)
64:         {
65:             base.OnNavigatedTo(e);
66:         }
67:
68:         protected override void OnNavigatedFrom(NavigationEventArgs e)
69:         {
70:             base.OnNavigatedFrom(e);
71:         }
72:
73:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
74:         {
75:             if (this.NavigationService.CanGoBack)
76:             {
77:                 this.NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
78:             }
79:         }

```

```

80:
81:     private void SetBackgroundColor()
82:     {
83:         AppSettings appSettings = new AppSettings();
84:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
85:         {
86:             appSettings = context.AppSettings.First() as AppSettings;
87:         }
88:
89:         if (appSettings.AppBackgroundImage != null)
90:         {
91:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
92:             BitmapImage image = new BitmapImage();
93:             image.SetSource(stream);
94:             ImageBrush ib = new ImageBrush();
95:             ib.ImageSource = image;
96:             this.LayoutRoot.Background = ib;
97:         }
98:         else
99:         {
100:             switch (appSettings.AppBackgroundColor)
101:             {
102:                 case "BLA":
103:                     messageBackGround = new SolidColorBrush(Colors.Black);
104:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
105:                     break;
106:                 case "BLU":
107:                     messageBackGround = new SolidColorBrush(Colors.Blue);
108:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
109:                     break;
110:                 case "BRO":
111:                     messageBackGround = new SolidColorBrush(Colors.Brown);
112:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
113:                     break;
114:                 case "RED":
115:                     messageBackGround = new SolidColorBrush(Colors.Red);
116:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
117:                     break;
118:                 case "GRE":
119:                     messageBackGround = new SolidColorBrush(Colors.Green);
120:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
121:                     break;
122:                 case "GRA":
123:                     messageBackGround = new SolidColorBrush(Colors.Gray);
124:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
125:                     break;
126:                 case "YEL":
127:                     messageBackGround = new SolidColorBrush(Colors.Yellow);
128:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
129:                     break;
130:                 case "ORA":
131:                     messageBackGround = new SolidColorBrush(Colors.Orange);
132:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
133:                     break;
134:                 case "PUR":
135:                     messageBackGround = new SolidColorBrush(Colors.Purple);
136:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
137:                     break;
138:                 default:
139:                     messageBackGround = new SolidColorBrush(Colors.Black);
140:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
141:                     break;
142:             }
143:         }
144:     }

```

```

145:     }
146: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.15 FolderSettings.xaml.cs

This is file FolderSettings.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Controls.Primitives;
9: using System.Windows.Media;
10: using System.Windows.Media.Imaging;
11: using System.Windows.Navigation;
12: using AwesomeNote.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15: using Microsoft.Phone.Tasks;
16:
17: namespace AwesomeNote
18: {
19:     public partial class FolderSettings : PhoneApplicationPage
20:     {
21:
22:         public int noteFolderId;
23:         public SolidColorBrush messageBackGround;
24:
25:         public FolderSettings()
26:         {
27:             InitializeComponent();
28:             //SetBackgroundColor();
29:         }
30:
31:         protected override void OnNavigatedTo(NavigationEventArgs e)
32:         {
33:             base.OnNavigatedTo(e);
34:         }
35:
36:         protected override void OnNavigatedFrom(NavigationEventArgs e)
37:         {
38:             base.OnNavigatedFrom(e);
39:         }
40:
41:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
42:         {
43:             // displays "Fragment: Detail"
44:             //MessageBox.Show("Folder Id: " + e.Fragment);
45:             base.OnFragmentNavigation(e);
46:             noteFolderId = int.Parse(e.Fragment);
47:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
48:             {
49:                 var noteFolder =
50:                     context.NoteFolders.Where(j =>

```

```

j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
51:         lblFontFamily.Text = AppResources.FontFamily + " (" +
AppResources.Selected + ": " + noteFolder.FontFamily + ")";
52:         lblFontSize.Text = AppResources.FontSize + " (" +
AppResources.Selected + ": " + noteFolder.FontSize + ")";
53:         if (noteFolder.NoteOrderBy == "NAME")
54:         {
55:             lblNoteOrder.Text = AppResources.NoteOrder + " (" +
AppResources.Selected + ": " + AppResources.NoteName + ")";
56:         }
57:         if (noteFolder.NoteOrderBy == "CDATE")
58:         {
59:             lblNoteOrder.Text = AppResources.NoteOrder + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
60:         }
61:         if (noteFolder.NoteOrderBy == "MDATE")
62:         {
63:             lblNoteOrder.Text = AppResources.NoteOrder + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
64:         }
65:         if (noteFolder.NoteOrderStyle == "A")
66:         {
67:             lblNoteOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
68:         }
69:         if (noteFolder.NoteOrderStyle == "D")
70:         {
71:             lblNoteOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
72:         }
73:     }
74:     }
75:     SetBackgroundColor();
76: }
77:
78: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
79: {
80:     piFont.Header = AppResources.Font;
81:     piOtherSettings.Header = AppResources.OtherSettings;
82:     piPassword.Header = AppResources.Password;
83:     piBackground.Header = AppResources.Background;
84:
85:     btnFontFamily.Content = AppResources.Select;
86:     btnFontSize.Content = AppResources.Select;
87:     btnNoteOrder.Content = AppResources.Select;
88:     btnNoteOrderStyle.Content = AppResources.Select;
89:
90:
91:     // önce listbox içindeki degerlerin tamamı silinir
92:     lpPassword.Items.Clear();
93:
94:     // sifre pivot sayfasında olması gereken düzenlemeler yapiliyor
95:     lblPassword.Text = AppResources.PasswordState;
96:     lpPassword.Items.Add(AppResources.SelectPasswordState);
97:     lpPassword.Items.Add(AppResources.Active);
98:     lpPassword.Items.Add(AppResources.Passive);
99:
100:     lblBackgroundImage.Text = AppResources.BackgroundImage;
101:     btnBackgroundImage.Content = AppResources.Select;
102:     btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
103:
104:     //chbBold.Content = AppResources.IsBold;
105:     //chbItalic.Content = AppResources.IsItalic;
106:
107: }
108:
109: private void lpPassword_SelectionChanged(object sender,
SelectionChangedEventArgs e)
110: {

```

```

111:         int index;
112:
113:         if (lpPassword.Items.Count > 2)
114:         {
115:             index = lpPassword.SelectedIndex;
116:             // eger sifre özelligi aktif hale gelmisse
117:             if (index == 1)
118:             {
119:
120:                 NavigationService.Navigate(new Uri("/SetPasswordPage.xaml#" +
noteFolderId, UriKind.Relative));
121:
122:                 //Popup popup = new Popup();
123:                 //popup.Height = 440;
124:                 //popup.Width = 480;
125:                 //popup.VerticalOffset = 20;
126:                 //PopupSetPassword control = new PopupSetPassword();
127:                 //control.txtPass1.Text = AppResources.PleaseEnterPassword + " ("
+ AppResources.FourDigits + ")";
128:                 //control.txtPass2.Text = AppResources.PleaseReEnterPassword + "
(" + AppResources.FourDigits + ")";
129:                 //control.btnCancel.Content = AppResources.Cancel;
130:                 //control.btnOK.Content = AppResources.OK;
131:                 //popup.Child = control;
132:                 //popup.IsOpen = true;
133:
134:                 //control.btnOK.Click += (s, args) =>
135:                 //{
136:                 //    sifre dört karakterli ise
137:                 //    ve iki sifre alani da ayni ise
138:                 //    if ((control.txtFirstPassword.Text ==
control.txtSecondPassword.Text) &&
139:                 //        (control.txtFirstPassword.Text.Length == 4))
140:                 //    {
141:                 //        using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
142:                 //        {
143:                 //            var noteFolders =
144:                 //                context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
145:                 //            foreach (var noteFolder in noteFolders)
146:                 //            {
147:                 //                noteFolder.IsPasswordProtected = true;
148:                 //                noteFolder.NoteFolderPassword =
control.txtFirstPassword.Text;
149:                 //                noteFolder.ModificationDate = DateTime.Now;
150:                 //            }
151:                 //            //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
152:                 //            //            .Single()
153:                 //            //            .NoteFolderPassword =
control.txtFirstPassword.Text;
154:                 //            //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
155:                 //            //            .Single()
156:                 //            //            .IsPasswordProtected = true;
157:                 //            context.SubmitChanges();
158:                 //            MessageBox.Show(AppResources.SuccessfulSetPassword);
159:                 //            popup.IsOpen = false;
160:                 //        }
161:                 //    }
162:                 //    // sifreler ayni degilse veya 4 karakter degilse
163:                 //    else
164:                 //    {
165:                 //        MessageBox.Show(AppResources.TwoFieldsNotSame);
166:                 //        control.txtFirstPassword.Text = "";
167:                 //        control.txtSecondPassword.Text = "";
168:                 //        control.txtFirstPassword.Focus();
169:                 //    }

```

```

170:
171:         ///};
172:         //control.btnCancel.Click += (s, args) =>
173:         //{
174:         //    popup.IsOpen = false;
175:         //};
176:     }
177:     // eger sifre kaldirilacaksa
178:     else if (index == 2)
179:     {
180:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
181:         {
182:             var noteFolders =
183:                 context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
184:             foreach (var noteFolder in noteFolders)
185:             {
186:                 noteFolder.IsPasswordProtected = false;
187:                 noteFolder.NoteFolderPassword = "";
188:                 noteFolder.ModificationDate = DateTime.Now;
189:             }
190:             context.SubmitChanges();
191:             MessageBox.Show(AppResources.SuccessfulRemovePassword);
192:         }
193:     }
194:     else
195:     {
196:
197:     }
198: }
199: }
200:
201: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
202: {
203:     if (this.NavigationService.CanGoBack)
204:     {
205:         this.NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
noteFolderId, UriKind.Relative));
206:     }
207: }
208:
209: private void SetBackgroundColor()
210: {
211:     AppSettings appSettings = new AppSettings();
212:     NoteFolder noteFolder;
213:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
214:     {
215:         appSettings = context.AppSettings.First() as AppSettings;
216:         noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
217:     }
218:
219:     if (noteFolder.FolderBackground != null)
220:     {
221:         MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
222:         BitmapImage image = new BitmapImage();
223:         image.SetSource(stream);
224:         ImageBrush ib = new ImageBrush();
225:         ib.ImageSource = image;
226:         this.LayoutRoot.Background = ib;
227:     }
228:     else
229:     {
230:         switch (appSettings.AppBackgroundColor)
231:         {
232:             case "BLA":

```

```

233:         messageBackGround = new SolidColorBrush(Colors.Black);
234:         //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
235:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
236:         break;
237:     case "BLU":
238:         messageBackGround = new SolidColorBrush(Colors.Blue);
239:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
240:         break;
241:     case "BRO":
242:         messageBackGround = new SolidColorBrush(Colors.Brown);
243:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
244:         break;
245:     case "RED":
246:         messageBackGround = new SolidColorBrush(Colors.Red);
247:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
248:         break;
249:     case "GRE":
250:         messageBackGround = new SolidColorBrush(Colors.Green);
251:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
252:         break;
253:     case "GRA":
254:         messageBackGround = new SolidColorBrush(Colors.Gray);
255:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
256:         break;
257:     case "YEL":
258:         messageBackGround = new SolidColorBrush(Colors.Yellow);
259:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
260:         break;
261:     case "ORA":
262:         messageBackGround = new SolidColorBrush(Colors.Orange);
263:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
264:         break;
265:     case "PUR":
266:         messageBackGround = new SolidColorBrush(Colors.Purple);
267:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
268:         break;
269:     default:
270:         messageBackGround = new SolidColorBrush(Colors.Black);
271:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
272:         break;
273:     }
274: }
275: }
276:
277: private void btnFontFamily_Click(object sender, RoutedEventArgs e)
278: {
279:     NavigationService.Navigate(new Uri("/FontFamilySettings.xaml#" +
noteFolderId, UriKind.Relative));
280: }
281:
282: private void btnFontSize_Click(object sender, RoutedEventArgs e)
283: {
284:     NavigationService.Navigate(new Uri("/FontSizeSettings.xaml#" +
noteFolderId, UriKind.Relative));
285: }
286:
287: private void btnNoteOrder_Click(object sender, RoutedEventArgs e)
288: {
289:     NavigationService.Navigate(new Uri("/NoteOrderSettings.xaml#" +
noteFolderId, UriKind.Relative));
290: }
291:
292: private void btnNoteOrderStyle_Click(object sender, RoutedEventArgs e)
293: {
294:     NavigationService.Navigate(new Uri("/NoteOrderStyleSettings.xaml#" +

```



```

noteFolderId, UriKind.Relative));
295:     }
296:
297:     private void btnBackgroundImage_Click(object sender, RoutedEventArgs e)
298:     {
299:         PhotoChooserTask objPhotoChooser = new PhotoChooserTask();
300:         objPhotoChooser.Completed += new
EventHandler<PhotoResult>(PhotoChooseCall);
301:         objPhotoChooser.Show();
302:     }
303:
304:     private void PhotoChooseCall(object sender, PhotoResult e)
305:     {
306:         switch (e.TaskResult)
307:         {
308:             case TaskResult.OK:
309:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
310:                 {
311:                     var noteFolders =
312:                         context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
313:                     foreach (var noteFolder in noteFolders)
314:                     {
315:                         noteFolder.FolderBackground = new
byte[e.ChosenPhoto.Length];
316:                         e.ChosenPhoto.Position = 0;
317:                         e.ChosenPhoto.Read(noteFolder.FolderBackground, 0,
noteFolder.FolderBackground.Length);
318:                         //noteFolder.NoteFolderPassword = "";
319:                         noteFolder.ModificationDate = DateTime.Now;
320:                     }
321:                     context.SubmitChanges();
322:                     MessageBox.Show(AppResources.SuccessfulBackgroundImageChanged);
323:                 }
324:                 break;
325:             case TaskResult.Cancel:
326:                 //MessageBox.Show("Cancelled");
327:                 break;
328:             case TaskResult.None:
329:                 //MessageBox.Show("Nothing Entered");
330:                 break;
331:         }
332:         SetBackgroundColor();
333:     }
334:
335:     private void btnRemoveBackgroundImage_Click(object sender, RoutedEventArgs e)
336:     {
337:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
338:         {
339:             var noteFolders =
340:                 context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
341:             foreach (var noteFolder in noteFolders)
342:             {
343:                 noteFolder.FolderBackground = null;
344:                 //noteFolder.NoteFolderPassword = "";
345:                 noteFolder.ModificationDate = DateTime.Now;
346:             }
347:             context.SubmitChanges();
348:             MessageBox.Show(AppResources.SuccessfulBackgroundImageRemoved);
349:         }
350:     }
351: }
352: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.16 FontFamilySettings.xaml.cs

This is file FontFamilySettings.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class FontFamilySettings : PhoneApplicationPage
18:     {
19:         public int noteFolderId;
20:         public SolidColorBrush messageBackGround;
21:         public FontFamilySettings()
22:         {
23:             InitializeComponent();
24:             //SetBackgroundColor();
25:
26:             lstFontFamily.Items.Clear();
27:             lstFontFamily.Items.Add("Arial");
28:             lstFontFamily.Items.Add("Arial Black");
29:             lstFontFamily.Items.Add("Baskerville Old Face");
30:             lstFontFamily.Items.Add("Berlin Sans FB");
31:             lstFontFamily.Items.Add("Bookman Old Style");
32:             lstFontFamily.Items.Add("Calibri");
33:             lstFontFamily.Items.Add("Cambria");
34:             lstFontFamily.Items.Add("Candara");
35:             lstFontFamily.Items.Add("Comic Sans MS");
36:             lstFontFamily.Items.Add("Consolas");
37:             lstFontFamily.Items.Add("Constantia");
38:             lstFontFamily.Items.Add("Courier New");
39:             lstFontFamily.Items.Add("DokChampa");
40:             lstFontFamily.Items.Add("Ebrima");
41:             lstFontFamily.Items.Add("Georgia");
42:             lstFontFamily.Items.Add("Lucida Sans Unicode");
43:             lstFontFamily.Items.Add("Meiryo UI");
44:             lstFontFamily.Items.Add("Microsoft YaHei");
45:             lstFontFamily.Items.Add("Malgun Gothic");
46:             lstFontFamily.Items.Add("Segoe UI");
47:             lstFontFamily.Items.Add("Segoe WP");
48:             lstFontFamily.Items.Add("Tahoma");
49:             lstFontFamily.Items.Add("Trebuchet MS");
50:             lstFontFamily.Items.Add("Times New Roman");
51:             lstFontFamily.Items.Add("Verdana");
52:             lstFontFamily.SelectedIndex = -1;
53:         }

```

```

54:
55:     private void lstFontFamily_SelectionChanged(object sender,
SelectionChangedEventArgs e)
56:     {
57:         if (lstFontFamily.SelectedIndex != -1)
58:         {
59:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
60:             {
61:                 var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
62:                 foreach (var noteFolder in noteFolders)
63:                 {
64:                     noteFolder.FontFamily = lstFontFamily.SelectedItem.ToString();
65:                 }
66:                 context.SubmitChanges();
67:                 MessageBox.Show(AppResources.SuccessfulFontFamilyChanged);
68:             }
69:         }
70:         NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
71:     }
72:
73:     protected override void OnNavigatedTo(NavigationEventArgs e)
74:     {
75:         base.OnNavigatedTo(e);
76:     }
77:
78:     protected override void OnNavigatedFrom(NavigationEventArgs e)
79:     {
80:         base.OnNavigatedFrom(e);
81:     }
82:
83:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
84:     {
85:         // displays "Fragment: Detail"
86:         //MessageBox.Show("Folder Id: " + e.Fragment);
87:         base.OnFragmentNavigation(e);
88:         noteFolderId = int.Parse(e.Fragment);
89:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
90:         {
91:             var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
92:             lblFolderName.Text = noteFolders.NoteFolderName;
93:             lblFontFamily.Text = AppResources.SelectFontFamily;
94:         }
95:         SetBackgroundColor();
96:     }
97:
98:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
99:     {
100:         if (this.NavigationService.CanGoBack)
101:         {
102:             this.NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
103:         }
104:     }
105:
106:     private void SetBackgroundColor()
107:     {
108:         AppSettings appSettings = new AppSettings();
109:         NoteFolder noteFolder;
110:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
111:         {
112:             appSettings = context.AppSettings.First() as AppSettings;
113:             noteFolder = context.NoteFolders.Where(j =>

```

```

j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
114:         }
115:
116:         if (noteFolder.FolderBackground != null)
117:         {
118:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
119:             BitmapImage image = new BitmapImage();
120:             image.SetSource(stream);
121:             ImageBrush ib = new ImageBrush();
122:             ib.ImageSource = image;
123:             this.LayoutRoot.Background = ib;
124:         }
125:     else
126:     {
127:         switch (appSettings.AppBackgroundColor)
128:         {
129:             case "BLA":
130:                 messageBackGround = new SolidColorBrush(Colors.Black);
131:                 //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
132:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
133:                 break;
134:             case "BLU":
135:                 messageBackGround = new SolidColorBrush(Colors.Blue);
136:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
137:                 break;
138:             case "BRO":
139:                 messageBackGround = new SolidColorBrush(Colors.Brown);
140:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
141:                 break;
142:             case "RED":
143:                 messageBackGround = new SolidColorBrush(Colors.Red);
144:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
145:                 break;
146:             case "GRE":
147:                 messageBackGround = new SolidColorBrush(Colors.Green);
148:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
149:                 break;
150:             case "GRA":
151:                 messageBackGround = new SolidColorBrush(Colors.Gray);
152:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
153:                 break;
154:             case "YEL":
155:                 messageBackGround = new SolidColorBrush(Colors.Yellow);
156:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
157:                 break;
158:             case "ORA":
159:                 messageBackGround = new SolidColorBrush(Colors.Orange);
160:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
161:                 break;
162:             case "PUR":
163:                 messageBackGround = new SolidColorBrush(Colors.Purple);
164:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
165:                 break;
166:             default:
167:                 messageBackGround = new SolidColorBrush(Colors.Black);
168:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
169:                 break;
170:         }
171:     }
172: }
173: }
174: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.17 FontSizeSettings.xaml.cs

This is file FontSizeSettings.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class FontSizeSettings : PhoneApplicationPage
18:     {
19:         public int noteFolderId;
20:         public SolidColorBrush messageBackGround;
21:         public FontSizeSettings()
22:         {
23:             InitializeComponent();
24:             //SetBackgroundColor();
25:
26:             lstFontSize.Items.Clear();
27:             lstFontSize.Items.Add("14");
28:             lstFontSize.Items.Add("18");
29:             lstFontSize.Items.Add("22");
30:             lstFontSize.Items.Add("26");
31:             lstFontSize.Items.Add("28");
32:             lstFontSize.Items.Add("30");
33:             lstFontSize.Items.Add("32");
34:             lstFontSize.Items.Add("34");
35:             lstFontSize.Items.Add("36");
36:             lstFontSize.Items.Add("38");
37:             lstFontSize.Items.Add("40");
38:             lstFontSize.Items.Add("42");
39:             lstFontSize.Items.Add("44");
40:             lstFontSize.Items.Add("64");
41:             lstFontSize.Items.Add("72");
42:             lstFontSize.SelectedIndex = -1;
43:         }
44:
45:         private void lstFontSize_SelectionChanged(object sender,
SelectionChangedEventArgs e)
46:         {
47:             if (lstFontSize.SelectedIndex != -1)
48:             {
49:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
50:                 {
51:                     var noteFolders = context.NoteFolders.Where(j =>

```

```

j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
52:         foreach (var noteFolder in noteFolders)
53:         {
54:             noteFolder.FontSize = lstFontSize.SelectedItem.ToString();
55:         }
56:         context.SubmitChanges();
57:         MessageBox.Show(AppResources.SuccessfulFontSizeChanged);
58:     }
59: }
60:     NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
61: }
62:
63:     protected override void OnNavigatedTo(NavigationEventArgs e)
64:     {
65:         base.OnNavigatedTo(e);
66:     }
67:
68:     protected override void OnNavigatedFrom(NavigationEventArgs e)
69:     {
70:         base.OnNavigatedFrom(e);
71:     }
72:
73:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
74:     {
75:         // displays "Fragment: Detail"
76:         //MessageBox.Show("Folder Id: " + e.Fragment);
77:         base.OnFragmentNavigation(e);
78:         noteFolderId = int.Parse(e.Fragment);
79:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
80:         {
81:             var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
82:             lblFolderName.Text = noteFolders.NoteFolderName;
83:             lblFontSize.Text = AppResources.SelectFontSize;
84:         }
85:         SetBackgroundColor();
86:     }
87:
88:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
89:     {
90:         if (this.NavigationService.CanGoBack)
91:         {
92:             this.NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
93:         }
94:     }
95:
96:     private void SetBackgroundColor()
97:     {
98:         AppSettings appSettings = new AppSettings();
99:         NoteFolder noteFolder;
100:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
101:         {
102:             appSettings = context.AppSettings.First() as AppSettings;
103:             noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
104:         }
105:
106:         if (noteFolder.FolderBackground != null)
107:         {
108:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
109:             BitmapImage image = new BitmapImage();
110:             image.SetSource(stream);
111:             ImageBrush ib = new ImageBrush();
112:             ib.ImageSource = image;

```

```
113:         this.LayoutRoot.Background = ib;
114:     }
115:     else
116:     {
117:         switch (appSettings.AppBackgroundColor)
118:         {
119:             case "BLA":
120:                 messageBackGround = new SolidColorBrush(Colors.Black);
121:                 //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
122:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
123:                 break;
124:             case "BLU":
125:                 messageBackGround = new SolidColorBrush(Colors.Blue);
126:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
127:                 break;
128:             case "BRO":
129:                 messageBackGround = new SolidColorBrush(Colors.Brown);
130:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
131:                 break;
132:             case "RED":
133:                 messageBackGround = new SolidColorBrush(Colors.Red);
134:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
135:                 break;
136:             case "GRE":
137:                 messageBackGround = new SolidColorBrush(Colors.Green);
138:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
139:                 break;
140:             case "GRA":
141:                 messageBackGround = new SolidColorBrush(Colors.Gray);
142:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
143:                 break;
144:             case "YEL":
145:                 messageBackGround = new SolidColorBrush(Colors.Yellow);
146:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
147:                 break;
148:             case "ORA":
149:                 messageBackGround = new SolidColorBrush(Colors.Orange);
150:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
151:                 break;
152:             case "PUR":
153:                 messageBackGround = new SolidColorBrush(Colors.Purple);
154:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
155:                 break;
156:             default:
157:                 messageBackGround = new SolidColorBrush(Colors.Black);
158:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
159:                 break;
160:         }
161:     }
162: }
163: }
164: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.18 GeneralSettings.xaml.cs

This is file GeneralSettings.xaml.cs.

Body Source

```

1: ?using System;
2: using System.Collections.Generic;
3: using System.Globalization;
4: using System.IO;
5: using System.IO.IsolatedStorage;
6: using System.Linq;
7: using System.Net;
8: using System.Text;
9: using System.Threading;
10: using System.Threading.Tasks;
11: using System.Windows;
12: using System.Windows.Controls;
13: using System.Windows.Media;
14: using System.Windows.Media.Imaging;
15: using System.Windows.Navigation;
16: using AwesomeNote.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21:
22: namespace AwesomeNote
23: {
24:     public partial class GeneralSettings : PhoneApplicationPage
25:     {
26:         /// <summary>
27:         ///     Defines the scopes the application needs.
28:         /// </summary>
29:         private static readonly string[] scopes = new string[] { "wl.signin",
"wl.basic", "wl.offline_access", "wl.skydrive", "wl.skydrive_update" };
30:
31:         /// <summary>
32:         ///     Stores the LiveAuthClient instance.
33:         /// </summary>
34:         private LiveAuthClient authClient;
35:
36:         /// <summary>
37:         ///     Stores the LiveConnectClient instance.
38:         /// </summary>
39:         private LiveConnectClient liveClient;
40:
41:         public int signIn;
42:
43:         private PhoneApplicationPage currentPage;
44:
45:         public SolidColorBrush messageBackGround;
46:
47:         public GeneralSettings()
48:         {
49:             InitializeComponent();
50:             SetBackgroundColor();
51:             InitializePage();
52:             signIn = 0;
53:
54:             //btnOneDrive.Content = AppResources.SignIn;
55:
56:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
57:             {
58:                 var appSettings = context.AppSettings.First() as AppSettings;
59:                 if (appSettings.AppLangName == "EN")
60:                 {
61:                     lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.English + ")";
62:                 }
63:                 if (appSettings.AppLangName == "TR")

```



```

64:         {
65:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Turkish + ")";
66:         }
67:         if (appSettings.AppLangName == "DE")
68:         {
69:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.German + ")";
70:         }
71:         if (appSettings.AppLangName == "ES")
72:         {
73:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Spanish + ")";
74:         }
75:
76:         if (appSettings.AppLangName == "PT")
77:         {
78:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Portuguese + ")";
79:         }
80:         if (appSettings.AppLangName == "AR")
81:         {
82:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Arabic + ")";
83:         }
84:         if (appSettings.AppLangName == "FA")
85:         {
86:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Persian + ")";
87:         }
88:         if (appSettings.AppLangName == "IT")
89:         {
90:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Italian + ")";
91:         }
92:         if (appSettings.AppLangName == "FR")
93:         {
94:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.French + ")";
95:         }
96:         if (appSettings.AppLangName == "RU")
97:         {
98:             lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Russian + ")";
99:         }
100:        if (appSettings.AppLangName == "ZH")
101:        {
102:            lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Chinese + ")";
103:        }
104:        if (appSettings.AppLangName == "SA")
105:        {
106:            lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Sanskrit + ")";
107:        }
108:        if (appSettings.AppLangName == "TH")
109:        {
110:            lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ":" + AppResources.Thai + ")";
111:        }
112:
113:        if (appSettings.FolderOrderBy == "NAME")
114:        {
115:            lblFolderOrder.Text = AppResources.FolderOrder + " (" +
AppResources.Selected + ":" + AppResources.FolderName + ")";
116:        }
117:        if (appSettings.FolderOrderBy == "CDATE")
118:        {
119:            lblFolderOrder.Text = AppResources.FolderOrder + " (" +

```

```

AppResources.Selected + ": " + AppResources.CreationDate + "));
120:     }
121:     if (appSettings.FolderOrderBy == "MDATE")
122:     {
123:         lblFolderOrder.Text = AppResources.FolderOrder + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + "));";
124:     }
125:     if (appSettings.FolderOrderStyle == "A")
126:     {
127:         lblFolderOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + "));";
128:     }
129:     if (appSettings.FolderOrderStyle == "D")
130:     {
131:         lblFolderOrderStyle.Text = AppResources.OrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + "));";
132:     }
133:     if (appSettings.AppBackgroundColor == "BLA")
134:     {
135:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + "));";
136:     }
137:     if (appSettings.AppBackgroundColor == "BLU")
138:     {
139:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + "));";
140:     }
141:     if (appSettings.AppBackgroundColor == "BRO")
142:     {
143:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + "));";
144:     }
145:     if (appSettings.AppBackgroundColor == "RED")
146:     {
147:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + "));";
148:     }
149:     if (appSettings.AppBackgroundColor == "GRE")
150:     {
151:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + "));";
152:     }
153:     if (appSettings.AppBackgroundColor == "YEL")
154:     {
155:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + "));";
156:     }
157:     if (appSettings.AppBackgroundColor == "GRA")
158:     {
159:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + "));";
160:     }
161:     if (appSettings.AppBackgroundColor == "ORA")
162:     {
163:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + "));";
164:     }
165:     if (appSettings.AppBackgroundColor == "PUR")
166:     {
167:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + "));";
168:     }
169:     }
170: }
171:
172: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
173: {
174:     if (this.NavigationService.CanGoBack)

```

```

175:         {
176:             this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
177:         }
178:     }
179:
180:     public string DesignFileName(string fileName)
181:     {
182:         fileName = fileName.Replace(":", ".");
183:         fileName = fileName.Replace("?", ".");
184:         fileName = fileName.Replace("\\", ".");
185:         fileName = fileName.Replace("/", ".");
186:         fileName = fileName.Replace("<", ".");
187:         fileName = fileName.Replace(">", ".");
188:         fileName = fileName.Replace("|", ".");
189:         fileName = fileName.Replace("*", ".");
190:         return fileName;
191:     }
192:
193:     protected override void OnNavigatedTo(NavigationEventArgs e)
194:     {
195:         base.OnNavigatedTo(e);
196:         //currentPage =
((PhoneApplicationFrame)Application.Current.RootVisual).Content as PhoneApplicationPage;
197:         //currentPage.NavigationCacheMode = NavigationCacheMode.Enabled;
198:     }
199:
200:     protected override void OnNavigatedFrom(NavigationEventArgs e)
201:     {
202:         base.OnNavigatedFrom(e);
203:     }
204:
205:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
206:     {
207:         piLanguage.Header = AppResources.Language;
208:         piSync.Header = AppResources.Sync;
209:         piOtherSettings.Header = AppResources.OtherSettings;
210:         piBackground.Header = AppResources.Background;
211:
212:         //lblOneDrive.Text = AppResources.OneDrive;
213:
214:         btnFolderOrder.Content = AppResources.Select;
215:         btnFolderOrderStyle.Content = AppResources.Select;
216:         btnLanguage.Content = AppResources.Select;
217:         btnBackgroundColor.Content = AppResources.Select;
218:         //btnOneDrive.Content = AppResources.Login;
219:         //btnOneDrive.SignInText = AppResources.SignIn;
220:         //btnOneDrive.SignOutText = AppResources.SignOut;
221:         btnOneDriveSync.Content = AppResources.Sync;
222:         lblOneDrive.Text = AppResources.OneDrive;
223:         txtSynchronizing.Text = AppResources.Synchronizing;
224:
225:         pbSync.Visibility = Visibility.Collapsed;
226:         txtSynchronizing.Visibility = Visibility.Collapsed;
227:         txtSynchronizing.BorderBrush = this.LayoutRoot.Background;
228:
229:         btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
230:         lblBackgroundImage.Text = AppResources.BackgroundImage;
231:         btnBackgroundImage.Content = AppResources.Select;
232:         btnResetSettings.Content = AppResources.ResetSettings;
233:
234:         btnOneDriveSync.IsEnabled = false;
235:         cbSync.Content = AppResources.SyncOnOneFile;
236:         cbSync.IsEnabled = false;
237:         btnOneDrive.Content = "Sign In";
238:
239:     }
240:
241:     private void btnFolderOrder_Click(object sender, RoutedEventArgs e)

```

```

242:         {
243:             NavigationService.Navigate(new Uri("/FolderOrderSettings.xaml",
UriKind.Relative));
244:         }
245:
246:         private void btnFolderOrderStyle_Click(object sender, RoutedEventArgs e)
247:         {
248:             NavigationService.Navigate(new Uri("/FolderOrderStyleSettings.xaml",
UriKind.Relative));
249:         }
250:
251:         private void btnLanguage_Click(object sender, RoutedEventArgs e)
252:         {
253:             NavigationService.Navigate(new Uri("/LanguageSettings.xaml",
UriKind.Relative));
254:         }
255:
256:         private void SetBackgroundColor()
257:         {
258:             AppSettings appSettings = new AppSettings();
259:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
260:             {
261:                 appSettings = context.AppSettings.First() as AppSettings;
262:             }
263:
264:             if (appSettings.AppBackgroundImage != null)
265:             {
266:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
267:                 BitmapImage image = new BitmapImage();
268:                 image.SetSource(stream);
269:                 ImageBrush ib = new ImageBrush();
270:                 ib.ImageSource = image;
271:                 this.LayoutRoot.Background = ib;
272:             }
273:             else
274:             {
275:                 switch (appSettings.AppBackgroundColor)
276:                 {
277:                     case "BLA":
278:                         messageBackGround = new SolidColorBrush(Colors.Black);
279:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
280:                         break;
281:                     case "BLU":
282:                         messageBackGround = new SolidColorBrush(Colors.Blue);
283:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
284:                         break;
285:                     case "BRO":
286:                         messageBackGround = new SolidColorBrush(Colors.Brown);
287:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
288:                         break;
289:                     case "RED":
290:                         messageBackGround = new SolidColorBrush(Colors.Red);
291:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
292:                         break;
293:                     case "GRE":
294:                         messageBackGround = new SolidColorBrush(Colors.Green);
295:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
296:                         break;
297:                     case "GRA":
298:                         messageBackGround = new SolidColorBrush(Colors.Gray);
299:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
300:                         break;
301:                     case "YEL":
302:                         messageBackGround = new SolidColorBrush(Colors.Yellow);
303:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
304:                         break;
305:                     case "ORA":

```

```

306:         messageBackGround = new SolidColorBrush(Colors.Orange);
307:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
308:         break;
309:         case "PUR":
310:             messageBackGround = new SolidColorBrush(Colors.Purple);
311:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
312:             break;
313:         default:
314:             messageBackGround = new SolidColorBrush(Colors.Black);
315:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
316:             break;
317:     }
318: }
319: }
320:
321: private void btnBackgroundImage_Click(object sender, RoutedEventArgs e)
322: {
323:     PhotoChooserTask objPhotoChooser = new PhotoChooserTask();
324:     objPhotoChooser.Completed += new
EventHandler<PhotoResult>(PhotoChooseCall);
325:     objPhotoChooser.Show();
326: }
327:
328: private void PhotoChooseCall(object sender, PhotoResult e)
329: {
330:     switch (e.TaskResult)
331:     {
332:         case TaskResult.OK:
333:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
334:             {
335:                 var appSettings = context.AppSettings;
336:                 foreach (var appSetting in appSettings)
337:                 {
338:                     appSetting.AppBackgroundImage = new byte[e.ChosenPhoto.Length];
339:                     e.ChosenPhoto.Position = 0;
340:                     e.ChosenPhoto.Read(appSetting.AppBackgroundImage, 0,
appSetting.AppBackgroundImage.Length);
341:                     //noteFolder.NoteFolderPassword = "";
342:                 }
343:                 context.SubmitChanges();
344:                 MessageBox.Show(AppResources.SuccessfulBackgroundImageChanged);
345:             }
346:             break;
347:         case TaskResult.Cancel:
348:             //MessageBox.Show("Cancelled");
349:             break;
350:         case TaskResult.None:
351:             //MessageBox.Show("Nothing Entered");
352:             break;
353:     }
354:     SetBackgroundColor();
355: }
356:
357: private void btnRemoveBackgroundImage_Click(object sender, RoutedEventArgs e)
358: {
359:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
360:     {
361:         var appSettings = context.AppSettings;
362:         foreach (var appSetting in appSettings)
363:         {
364:             appSetting.AppBackgroundImage = null;
365:         }
366:         context.SubmitChanges();
367:         MessageBox.Show(AppResources.SuccessfulBackgroundImageRemoved);
368:     }

```

```

369:         }
370:
371:         private void btnBackgroundColor_Click(object sender, RoutedEventArgs e)
372:         {
373:             NavigationService.Navigate(new Uri("/BackgroundColorSettings.xaml",
UriKind.Relative));
374:         }
375:
376:         private async void btnOneDrive_Click(object sender, RoutedEventArgs e)
377:         {
378:             try
379:             {
380:                 if (this.btnOneDrive.Content.ToString() == "Sign In" ||
this.btnOneDrive.Content.ToString() == "Sign in")
381:                 {
382:                     LiveLoginResult loginResult = await
this.authClient.LoginAsync(scopes);
383:                     if (loginResult.Status == LiveConnectSessionStatus.Connected)
384:                     {
385:                         //this.btnOneDrive.Content = AppResources.SignOut;
386:                         this.btnOneDrive.Content = "Sign Out";
387:
388:                         this.liveClient = new LiveConnectClient(loginResult.Session);
389:                         this.GetMe();
390:                         btnOneDriveSync.IsEnabled = true;
391:                         cbSync.IsEnabled = true;
392:                     }
393:                 }
394:             }
395:             else
396:             {
397:                 this.authClient.Logout();
398:                 //this.btnOneDrive.Content = AppResources.SignIn;
399:                 this.btnOneDrive.Content = "Sign Out";
400:                 btnOneDriveSync.IsEnabled = true;
401:                 cbSync.IsEnabled = true;
402:                 //this.tbResponse.Text = "";
403:             }
404:         }
405:         catch (LiveAuthException authExp)
406:         {
407:             //this.tbResponse.Text = authExp.ToString();
408:         }
409:     }
410:
411:     private async void InitializePage()
412:     {
413:         try
414:         {
415:             // bu benim uygulamama ait bir client id
416:             this.authClient = new LiveAuthClient("000000004C120D50");
417:             LiveLoginResult loginResult = await
this.authClient.InitializeAsync(scopes);
418:             btnOneDrive.Content = "Sign In";
419:             if (loginResult.Status == LiveConnectSessionStatus.Connected)
420:             {
421:                 //this.btnOneDrive.Content = AppResources.SignOut;
422:                 this.btnOneDrive.Content = "Sign Out";
423:
424:                 this.liveClient = new LiveConnectClient(loginResult.Session);
425:                 //this.GetMe();
426:             }
427:         }
428:         catch (LiveAuthException authExp)
429:         {
430:             //this.tbResponse.Text = authExp.ToString();
431:         }
432:     }
433:

```

```

434:         private async void GetMe()
435:         {
436:             try
437:             {
438:                 LiveOperationResult operationResult = await
this.liveClient.GetAsync("me");
439:
440:                 dynamic properties = operationResult.Result;
441:                 //this.tbResponse.Text = properties.first_name + " " +
properties.last_name;
442:             }
443:             catch (LiveConnectException e)
444:             {
445:                 //this.tbResponse.Text = e.ToString();
446:             }
447:         }
448:
449:         private void btnOneDrive_SessionChanged(object sender,
Microsoft.Live.Controls.LiveConnectSessionChangedEventArgs e)
450:         {
451:             if (e != null && e.Status == LiveConnectSessionStatus.Connected)
452:             {
453:                 //the session status is connected so we need to set this session
status to client
454:                 this.liveClient = new LiveConnectClient(e.Session);
455:             }
456:             else
457:             {
458:                 this.liveClient = null;
459:             }
460:         }
461:
462:         private async void btnOneDriveSync_Click(object sender, RoutedEventArgs e)
463:         {
464:             IsolatedStorageFile myIsolatedStorage = null;
465:             StringBuilder sb = null;
466:
467:             string folderName;
468:             try
469:             {
470:                 //var folderData = new Dictionary<string, object>();
471:                 folderName = "Awesome Note (" + DateTime.Now + ")";
472:                 //folderName = folderName.Replace(":", ".");
473:                 //folderName = folderName.Replace("/", ".");
474:
475:                 folderName = DesignFileName(folderName);
476:
477:                 //folderData.Add("name", folderName);
478:                 //LiveConnectClient liveClient = new LiveConnectClient(this.session);
479:                 //LiveOperationResult operationResult =
480:                 //    await this.liveClient.PostAsync("me/skydrive", folderData);
481:                 //dynamic result = operationResult.Result;
482:
483:                 string skyDriveFolder = await CreateDirectoryAsync(liveClient,
folderName, "me/skydrive");
484:                 if (cbSync.IsChecked == false)
485:                 {
486:                     btnOneDrive.IsEnabled = false;
487:                     pbSync.Visibility = Visibility.Visible;
488:                     txtSynchronizing.Visibility = Visibility.Visible;
489:
490:                     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
491:                     {
492:                         //var noteFolders = context.NoteFolders.ToList() as
List<NoteFolder>;
493:                         var notes = context.Notes.ToList() as List<Note>;
494:
495:                         for (int i = 0; i < notes.Count; i++)

```



```

496:         {
497:             var noteFolder =
498:                 context.NoteFolders.Where(j =>
499:                     j.NoteFolderId.Equals(notes[i].NoteFolderId)).Single() as
500:                     NoteFolder;
501:
502:             string fileName = Guid.NewGuid() + ". " +
notes[i].NoteName + " (" + noteFolder.NoteFolderName +
503:                 ").txt";
504:
505:             fileName = DesignFileName(fileName);
506:
507:             //StringBuilder sb = new StringBuilder();
508:             //sb.AppendLine(AppResources.NoteName + ": " +
notes[i].NoteName);
509:             //sb.AppendLine(AppResources.FolderName + ": " +
noteFolder.NoteFolderName);
510:             //sb.AppendLine(AppResources.Password + ": " +
noteFolder.IsPasswordProtected);
511:             //sb.AppendLine(AppResources.CreationDate + ": " +
notes[i].CreationDate);
512:             //sb.AppendLine(AppResources.ModificationDate + ": " +
notes[i].ModificationDate);
513:             //sb.AppendLine(AppResources.Note + ": " +
notes[i].NoteDescription);
514:
515:
516:             myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
517:             //if (myIsolatedStorage.FileExists(fileName))
518:             //{
519:             //myIsolatedStorage.DeleteFile(fileName);
520:             //} //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
521:             using (StreamWriter writeFile = new StreamWriter
522:                 (new IsolatedStorageFileStream(fileName, FileMode.Create,
FileAccess.Write, myIsolatedStorage)))
523:             {
524:                 writeFile.WriteLine(AppResources.NoteName + ": " +
notes[i].NoteName);
525:                 writeFile.WriteLine(AppResources.FolderName + ": " +
noteFolder.NoteFolderName);
526:                 writeFile.WriteLine(AppResources.PasswordProtection +
": " + (noteFolder.IsPasswordProtected == true ? AppResources.Yes : AppResources.No));
527:                 writeFile.WriteLine(AppResources.CreationDate + ": " +
notes[i].CreationDate);
528:                 writeFile.WriteLine(AppResources.ModificationDate + ":
" + notes[i].ModificationDate);
529:                 writeFile.WriteLine(AppResources.Note + ": " +
notes[i].NoteDescription);
530:                 writeFile.Close();
531:             }
532:             IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.Open, FileAccess.Read);
533:             var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
534:             pbSync.Value = (i + 1) * (100) / notes.Count;
535:             //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
536:         }
537:     }
538:     //this.infoTextBlock.Text = string.Join(" ", "Created folder:",
result.name, "ID:", result.id);
539:     MessageBox.Show(AppResources.OneDriveSyncCompleted);
540: }
541: else
542: {
543:     using (var context = new

```



```

NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
544:         {
545:             //var noteFolders = context.NoteFolders.ToList() as
List<NoteFolder>;
546:             var notes = context.Notes.OrderBy(j =>
j.CreationDate).ToList() as List<Note>;
547:
548:             var noteFirst = notes.First();
549:             var noteLast = notes.Last();
550:
551:             string fileName = Guid.NewGuid() + ". Awesome Note (" +
noteFirst.CreationDate.ToShortDateString() + " - " +
noteLast.CreationDate.ToShortDateString() + ").txt";
552:             fileName = DesignFileName(fileName);
553:
554:             myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
555:             sb = new StringBuilder();
556:             for (int i = 0; i < notes.Count; i++)
557:             {
558:                 var noteFolder =
                    context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(notes[i].NoteFolderId)).Single() as
559:                 NoteFolder;
560:
561:
562:                 sb.AppendLine();
563:                 sb.AppendLine(AppResources.NoteName + ": " +
notes[i].NoteName);
564:                 sb.AppendLine(AppResources.FolderName + ": " +
noteFolder.NoteFolderName);
565:                 sb.AppendLine(AppResources.Password + ": " +
noteFolder.IsPasswordProtected);
566:                 sb.AppendLine(AppResources.CreationDate + ": " +
notes[i].CreationDate);
567:                 sb.AppendLine(AppResources.ModificationDate + ": " +
notes[i].ModificationDate);
568:                 sb.AppendLine(AppResources.Note + ": " +
notes[i].NoteDescription);
569:                 sb.AppendLine();
570:
571:                 //if (myIsolatedStorage.FileExists(fileName))
572:                 //{
573:                 //    myIsolatedStorage.DeleteFile(fileName);
574:                 //} //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
575:
576:                 //pbSync.Value = (i + 1) * (100) / notes.Count;
577:                 //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
578:             }
579:
580:             using (StreamWriter writeFile = new StreamWriter
581:                 (new IsolatedStorageFileStream(fileName, FileMode.Create,
FileAccess.Write, myIsolatedStorage)))
582:             {
583:                 writeFile.Write(sb.ToString());
584:                 writeFile.Close();
585:             }
586:
587:             IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.Open, FileAccess.Read);
588:             var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
589:             //this.infoTextBlock.Text = string.Join(" ", "Created folder:",
result.name, "ID:", result.id);
590:             //this.infoTextBlock.Text = string.Join(" ", "Created folder:",
result.name, "ID:", result.id);
591:             MessageBox.Show(AppResources.OneDriveSyncCompleted);
592:         }
593:

```

```

594:         pbSync.Visibility = Visibility.Collapsed;
595:         txtSynchronizing.Visibility = Visibility.Collapsed;
596:         pbSync.Value = 0;
597:         btnOneDrive.IsEnabled = true;
598:     }
599:     catch (Exception exception)
600:     {
601:         //this.infoTextBlock.Text = "Error creating folder: " +
exception.Message;
602:         MessageBox.Show(AppResources.SystemFault);
603:     }
604: }
605:
606:     public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
607: string folderName, string parentFolder)
608:     {
609:         string folderId = null;
610:
611:         // Retrieves all the directories.
612:         var queryFolder = parentFolder + "/files?filter=folders,albums";
613:         var opResult = await client.GetAsync(queryFolder);
614:         dynamic result = opResult.Result;
615:
616:         foreach (dynamic folder in result.data)
617:         {
618:             // Checks if current folder has the passed name.
619:             if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
620:             {
621:                 folderId = folder.id;
622:                 break;
623:             }
624:         }
625:
626:         if (folderId == null)
627:         {
628:             // Directory hasn't been found, so creates it using the PostAsync
method.
629:             var folderData = new Dictionary<string, object>();
630:             folderData.Add("name", folderName);
631:             opResult = await client.PostAsync(parentFolder, folderData);
632:             result = opResult.Result;
633:
634:             // Retrieves the id of the created folder.
635:             folderId = result.id;
636:         }
637:
638:         return folderId;
639:     }
640:
641:     private void btnResetSettings_Click(object sender, RoutedEventArgs e)
642:     {
643:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
644:         {
645:             var appSettings = context.AppSettings;
646:             foreach (var appSetting in appSettings)
647:             {
648:                 appSetting.AppBackgroundImage = null;
649:                 appSetting.AppBackgroundColor = "BLA";
650:             }
651:             context.SubmitChanges();
652:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
653:             MessageBox.Show(AppResources.SuccessfulResetSettings);
654:         }
655:     }
656:
657: }
658:
659: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.19 Language.cs

This is file Language.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5: using System.Threading.Tasks;
6:
7: namespace AwesomeNote
8: {
9:     public class Language
10:    {
11:        public int LanguageId { get; set; }
12:        public string LanguageName { get; set; }
13:    }
14: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.20 LanguageSettings.xaml.cs

This is file LanguageSettings.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Globalization;
4: using System.IO;
5: using System.Linq;
6: using System.Net;
7: using System.Threading;
8: using System.Windows;
9: using System.Windows.Controls;
10: using System.Windows.Media;
11: using System.Windows.Media.Imaging;
12: using System.Windows.Navigation;
13: using AwesomeNote.Resources;
14: using Microsoft.Phone.Controls;
15: using Microsoft.Phone.Shell;
16:
17: namespace AwesomeNote
18: {
19:     public partial class LanguageSettings : PhoneApplicationPage
20:     {
21:         public SolidColorBrush messageBackGround;
22:         public LanguageSettings()
23:         {
```

```
24:         InitializeComponent();
25:         SetBackgroundColor();
26:
27:         lstLanguage.Items.Clear();
28:         lstLanguage.Items.Add(AppResources.English);
29:         lstLanguage.Items.Add(AppResources.Turkish);
30:         lstLanguage.Items.Add(AppResources.German);
31:         lstLanguage.Items.Add(AppResources.Spanish);
32:         lstLanguage.Items.Add(AppResources.Russian);
33:         lstLanguage.Items.Add(AppResources.Chinese);
34:         lstLanguage.Items.Add(AppResources.Arabic);
35:         lstLanguage.Items.Add(AppResources.Persian);
36:         lstLanguage.Items.Add(AppResources.Italian);
37:         lstLanguage.Items.Add(AppResources.French);
38:         lstLanguage.Items.Add(AppResources.Portuguese);
39:         lstLanguage.Items.Add(AppResources.Sanskrit);
40:         lstLanguage.Items.Add(AppResources.Thai);
41:         lstLanguage.SelectedIndex = -1;
42:         lblLanguage.Text = AppResources.SelectLanguage;
43:     }
44:
45:     private void lstLanguage_SelectionChanged(object sender,
SelectionChangedEventArgs e)
46:     {
47:         int index = lstLanguage.SelectedIndex;
48:         string culture = "";
49:         string lang = "";
50:         if (index == 0)
51:         {
52:             culture = "en";
53:             lang = "EN";
54:         }
55:         else if (index == 1)
56:         {
57:             culture = "tr";
58:             lang = "TR";
59:         }
60:         else if (index == 2)
61:         {
62:             culture = "de";
63:             lang = "DE";
64:         }
65:         else if (index == 3)
66:         {
67:             culture = "es";
68:             lang = "ES";
69:         }
70:         else if (index == 4)
71:         {
72:             culture = "ru";
73:             lang = "RU";
74:         }
75:         else if (index == 5)
76:         {
77:             culture = "zh";
78:             lang = "AR";
79:         }
80:         else if (index == 6)
81:         {
82:             culture = "ar";
83:             lang = "AR";
84:         }
85:         else if (index == 7)
86:         {
87:             culture = "fa-IR";
88:             lang = "FA";
89:         }
90:         else if (index == 8)
91:         {
```

```

92:         culture = "it";
93:         lang = "IT";
94:     }
95:     else if (index == 9)
96:     {
97:         culture = "fr";
98:         lang = "FR";
99:     }
100:    else if (index == 10)
101:    {
102:        culture = "pt";
103:        lang = "PT";
104:    }
105:    else if (index == 11)
106:    {
107:        culture = "sa";
108:        lang = "SA";
109:    }
110:    else if (index == 12)
111:    {
112:        culture = "th";
113:        lang = "TH";
114:    }
115:    else
116:    {
117:        culture = "en";
118:        lang = "EN";
119:    }
120:
121:    using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
122:    {
123:        var appSettings = context.AppSettings;
124:        foreach (var appSetting in appSettings)
125:        {
126:            appSetting.AppLangName = lang;
127:        }
128:        context.SubmitChanges();
129:    }
130:
131:    CultureInfo newCulture = new CultureInfo(culture);
132:    Thread.CurrentThread.CurrentCulture = newCulture;
133:    Thread.CurrentThread.CurrentUICulture = newCulture;
134:    MessageBox.Show(AppResources.LanguageWarning);
135:    NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
136:    }
137:
138:    protected override void OnNavigatedTo(NavigationEventArgs e)
139:    {
140:        base.OnNavigatedTo(e);
141:    }
142:
143:    protected override void OnNavigatedFrom(NavigationEventArgs e)
144:    {
145:        base.OnNavigatedFrom(e);
146:    }
147:
148:    private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
149:    {
150:        if (this.NavigationService.CanGoBack)
151:        {
152:            this.NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
153:        }
154:    }
155:
156:    private void SetBackgroundColor()

```

```

157:         {
158:             AppSettings appSettings = new AppSettings();
159:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
160:             {
161:                 appSettings = context.AppSettings.First() as AppSettings;
162:             }
163:
164:             if (appSettings.AppBackgroundImage != null)
165:             {
166:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
167:                 BitmapImage image = new BitmapImage();
168:                 image.SetSource(stream);
169:                 ImageBrush ib = new ImageBrush();
170:                 ib.ImageSource = image;
171:                 this.LayoutRoot.Background = ib;
172:             }
173:             else
174:             {
175:                 switch (appSettings.AppBackgroundColor)
176:                 {
177:                     case "BLA":
178:                         messageBackGround = new SolidColorBrush(Colors.Black);
179:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
180:                         break;
181:                     case "BLU":
182:                         messageBackGround = new SolidColorBrush(Colors.Blue);
183:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
184:                         break;
185:                     case "BRO":
186:                         messageBackGround = new SolidColorBrush(Colors.Brown);
187:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
188:                         break;
189:                     case "RED":
190:                         messageBackGround = new SolidColorBrush(Colors.Red);
191:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
192:                         break;
193:                     case "GRE":
194:                         messageBackGround = new SolidColorBrush(Colors.Green);
195:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
196:                         break;
197:                     case "GRA":
198:                         messageBackGround = new SolidColorBrush(Colors.Gray);
199:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
200:                         break;
201:                     case "YEL":
202:                         messageBackGround = new SolidColorBrush(Colors.Yellow);
203:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
204:                         break;
205:                     case "ORA":
206:                         messageBackGround = new SolidColorBrush(Colors.Orange);
207:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
208:                         break;
209:                     case "PUR":
210:                         messageBackGround = new SolidColorBrush(Colors.Purple);
211:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
212:                         break;
213:                     default:
214:                         messageBackGround = new SolidColorBrush(Colors.Black);
215:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
216:                         break;
217:                 }
218:             }
219:         }
220:     }
221: }

```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.21 LocalizedStrings.cs

This is file LocalizedStrings.cs.

Body Source

```
1: ?using AwesomeNote.Resources;
2:
3: namespace AwesomeNote
4: {
5:     /// <summary>
6:     /// Provides access to string resources.
7:     /// </summary>
8:     public class LocalizedStrings
9:     {
10:         private static AppResources _localizedResources = new AppResources();
11:
12:         public AppResources LocalizedResources { get { return _localizedResources; } }
13:     }
14: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.22 MainPage.xaml.cs

This is file MainPage.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Controls.Primitives;
9: using System.Windows.Media;
10: using System.Windows.Media.Imaging;
11: using System.Windows.Navigation;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Controls.Primitives;
14: using Microsoft.Phone.Shell;
15: using AwesomeNote.Resources;
16:
17: namespace AwesomeNote
18: {
19:     public partial class MainPage : PhoneApplicationPage
20:     {
21:
22:         public Popup popup;
23:         public SolidColorBrush messageBackGround;
```

```

24:         // Constructor
25:         public MainPage()
26:         {
27:             InitializeComponent();
28:             SetBackgroundColor();
29:
30:             ApplicationBar = new ApplicationBar();
31:
32:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
33:             button1.IconUri = new Uri("/Assets/AddFolder.png", UriKind.Relative);
34:             button1.Text = AppResources.AddFolder;
35:             ApplicationBar.Buttons.Add(button1);
36:             button1.Click += new EventHandler(AddFolderButton_Click);
37:
38:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
39:             button2.IconUri = new Uri("/Assets/Search.png", UriKind.Relative);
40:             button2.Text = AppResources.Search;
41:             ApplicationBar.Buttons.Add(button2);
42:             button2.Click += new EventHandler(SearchButton_Click);
43:
44:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
45:             button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
46:             button3.Text = AppResources.Settings;
47:             ApplicationBar.Buttons.Add(button3);
48:             button3.Click += new EventHandler(SettingsButton_Click);
49:
50:             ApplicationBarIconButton button4 = new ApplicationBarIconButton();
51:             button4.IconUri = new Uri("/Assets/About.png", UriKind.Relative);
52:             button4.Text = AppResources.About;
53:             ApplicationBar.Buttons.Add(button4);
54:             button4.Click += new EventHandler(AboutButton_Click);
55:
56:             popup = new Popup();
57:
58:             // Sample code to localize the ApplicationBar
59:             //BuildLocalizedApplicationBar();
60:         }
61:
62:         private void SetBackgroundColor()
63:         {
64:             AppSettings appSettings = new AppSettings();
65:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
66:             {
67:                 appSettings = context.AppSettings.First() as AppSettings;
68:             }
69:
70:             if (appSettings.AppBackgroundImage != null)
71:             {
72:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
73:                 BitmapImage image = new BitmapImage();
74:                 image.SetSource(stream);
75:                 ImageBrush ib = new ImageBrush();
76:                 ib.ImageSource = image;
77:                 this.LayoutRoot.Background = ib;
78:             }
79:             else
80:             {
81:                 switch (appSettings.AppBackgroundColor)
82:                 {
83:                     case "BLA":
84:                         messageBackGround = new SolidColorBrush(Colors.Black);
85:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
86:                         break;
87:                     case "BLU":
88:                         messageBackGround = new SolidColorBrush(Colors.Blue);
89:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
90:                         break;
91:                     case "BRO":

```



```

92:         messageBackGround = new SolidColorBrush(Colors.Brown);
93:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
94:         break;
95:     case "RED":
96:         messageBackGround = new SolidColorBrush(Colors.Red);
97:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
98:         break;
99:     case "GRE":
100:        messageBackGround = new SolidColorBrush(Colors.Green);
101:        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
102:        break;
103:    case "GRA":
104:        messageBackGround = new SolidColorBrush(Colors.Gray);
105:        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
106:        break;
107:    case "YEL":
108:        messageBackGround = new SolidColorBrush(Colors.Yellow);
109:        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
110:        break;
111:    case "ORA":
112:        messageBackGround = new SolidColorBrush(Colors.Orange);
113:        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
114:        break;
115:    case "PUR":
116:        messageBackGround = new SolidColorBrush(Colors.Purple);
117:        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
118:        break;
119:    default:
120:        messageBackGround = new SolidColorBrush(Colors.Black);
121:        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
122:        break;
123:    }
124: }
125: }
126: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
127: {
128:     //txtMainApp.Text = AppResources.AppTitle;
129:     //txtMainFolder.Text = AppResources.AppFolder;
130:
131:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
132:     {
133:         var appSettings = context.AppSettings.First() as AppSettings;
134:         string orderStyle = appSettings.FolderOrderStyle;
135:         lstFolders.Items.Clear();
136:         switch (appSettings.FolderOrderBy)
137:         {
138:             case "NAME":
139:                 if (orderStyle == "A")
140:                 {
141:                     lstFolders.ItemsSource = context.NoteFolders.OrderBy(j =>
j.NoteFolderName).ToList();
142:                 }
143:                 else
144:                 {
145:                     lstFolders.ItemsSource =
context.NoteFolders.OrderByDescending(j => j.NoteFolderName).ToList();
146:                 }
147:                 break;
148:             case "CDATE":
149:                 if (orderStyle == "A")
150:                 {
151:                     lstFolders.ItemsSource = context.NoteFolders.OrderBy(j =>
j.CreationDate).ToList();
152:                 }
153:                 else

```

```

154:                {
155:                    lstFolders.ItemsSource =
context.NoteFolders.OrderByDescending(j => j.CreationDate).ToList();
156:                }
157:                break;
158:                case "MDATE":
159:                    if (orderStyle == "A")
160:                    {
161:                        lstFolders.ItemsSource = context.NoteFolders.OrderBy(j =>
j.ModificationDate).ToList();
162:                    }
163:                    else
164:                    {
165:                        lstFolders.ItemsSource =
context.NoteFolders.OrderByDescending(j => j.ModificationDate).ToList();
166:                    }
167:                    break;
168:                default:
169:                    if (orderStyle == "A")
170:                    {
171:                        lstFolders.ItemsSource = context.NoteFolders.OrderBy(j =>
j.NoteFolderName).ToList();
172:                    }
173:                    else
174:                    {
175:                        lstFolders.ItemsSource =
context.NoteFolders.OrderByDescending(j => j.NoteFolderName).ToList();
176:                    }
177:                    break;
178:                }
179:                lstFolders.DisplayMemberPath = "NameCount";
180:            }
181:        }
182:
183:        private void lstFolders_SelectionChanged(object sender,
SelectionChangedEventArgs e)
184:        {
185:            var noteFolder = (NoteFolder)lstFolders.SelectedItem;
186:            int folderId = noteFolder.NoteFolderId;
187:
188:            // eger açılmak istenen not klasörü şifreli ise
189:            if (noteFolder.NoteFolderPassword.Length > 0)
190:            {
191:
192:                NavigationService.Navigate(new Uri("/EnterPasswordPage.xaml#" +
folderId, UriKind.Relative));
193:                //Popup popup = new Popup();
194:                //popup.Height = 300;
195:                //popup.Width = 400;
196:                //popup.VerticalOffset = 20;
197:                //PopupEnterPassword control = new PopupEnterPassword();
198:                //control.txtLabel.Text = AppResources.PleaseEnterPassword + " (" +
AppResources.FourDigits + ")" ;
199:                //control.btnCancel.Content = AppResources.Cancel;
200:                //control.btnOK.Content = AppResources.OK;
201:                //popup.Child = control;
202:                //popup.IsOpen = true;
203:                //control.txtPassword.Focus();
204:
205:                //control.btnOK.Click += (s, args) =>
206:                //{
207:                //    string password;
208:                //    popup.IsOpen = false;
209:                //    password = control.txtPassword.Text;
210:                //    // girilen şifre doğruysa
211:                //    if (password.Length == 4 && password ==
noteFolder.NoteFolderPassword)
212:                //    {
213:                //        MessageBox.Show(AppResources.PasswordTrue);

```

```

214:         //      NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
folderId, UriKind.Relative));
215:         //      }
216:         //      else
217:         //      {
218:         //          MessageBox.Show(AppResources.PasswordFalse);
219:         //          control.txtPassword.Text = "";
220:         //          control.txtPassword.Focus();
221:         //      }
222:         //};
223:         //control.btnCancel.Click += (s, args) =>
224:         //{
225:         //    popup.IsOpen = false;
226:         //};
227:     }
228:     // eger açılmak istenen not klasörü şifresiz ise
229:     else
230:     {
231:         NavigationService.Navigate(new Uri("/NoteFolder.xaml#" + folderId,
UriKind.Relative));
232:     }
233: }
234:
235: protected override void OnNavigatedTo(NavigationEventArgs e)
236: {
237:     base.OnNavigatedTo(e);
238:     //while (NavigationService.CanGoBack)
239:     //NavigationService.RemoveBackEntry();
240: }
241:
242:
243: protected override void OnNavigatedFrom(NavigationEventArgs e)
244: {
245:     base.OnNavigatedFrom(e);
246:     //while (NavigationService.CanGoBack)
247:     //NavigationService.RemoveBackEntry();
248: }
249:
250:
251: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
252: {
253:     if (popup.IsOpen)
254:     {
255:         popup.IsOpen = false;
256:     }
257:     if (MessageBox.Show(AppResources.ExitAppQuestion,
AppResources.ExitApp, MessageBoxButton.OKCancel)
!= MessageBoxResult.OK)
258:     {
259:         e.Cancel = true;
260:     }
261:     else
262:     {
263:         Application.Current.Terminate();
264:     }
265: }
266:
267:
268:
269:
270: private void ApplicationBarMenuItem_Click(object sender, EventArgs e)
271: {
272: }
273:
274:
275: private void AddFolderButton_Click(object sender, EventArgs e)
276: {
277:     int noteFolderId;
278:
279:     popup = new Popup();

```

```

280:         popup.Height = 300;
281:         popup.Width = 400;
282:         popup.VerticalOffset = 20;
283:         PopupAddFolder control = new PopupAddFolder();
284:         control.txtLabel.Text = AppResources.EnterFolderName;
285:         control.btnCancel.Content = AppResources.Cancel;
286:         control.btnOK.Content = AppResources.OK;
287:         popup.Child = control;
288:         popup.IsOpen = true;
289:         control.txtFolderName.Focus();
290:
291:         control.btnOK.Click += (s, args) =>
292:         {
293:             bool folder = false;
294:             string folderName;
295:             popup.IsOpen = false;
296:             folderName = control.txtFolderName.Text;
297:
298:             int length = control.txtFolderName.Text.Length;
299:             string space = control.txtFolderName.Text.Substring(length -
Math.Min(1, length));
300:             if (space == " ")
301:             {
302:                 folderName = control.txtFolderName.Text.Remove(length - 1, 1);
303:             }
304:             else
305:             {
306:                 folderName = control.txtFolderName.Text;
307:             }
308:
309:             // aynı isimde bir klasörün daha önceden oluşturulup oluşturulmadigini
310:             // kontrol eden bir kod bölümü
311:             using (var contextFolder = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
312:             {
313:                 folder =
314:                     contextFolder.NoteFolders.Any(j =>
j.NoteFolderName.Equals(folderName));
315:             }
316:             if (folder == true)
317:             {
318:                 MessageBox.Show(AppResources.FolderExists);
319:             }
320:             // eger bu isimde bir klasör oluşturulmamissa
321:             // oluşturulmasi için gerekli kodlar asagidadir
322:             else
323:             {
324:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
325:                 {
326:                     //noteFolderId = context.NoteFolders.Count() + 1;
327:                     NoteFolder noteFolder = new NoteFolder();
328:                     noteFolder.NoteFolderName = folderName;
329:                     //noteFolder.NoteFolderId = noteFolderId;
330:                     noteFolder.IsPasswordProtected = false;
331:                     noteFolder.NoteCount = 0;
332:                     noteFolder.NoteFolderPassword = "";
333:                     noteFolder.FontFamily = "Verdana";
334:                     noteFolder.FontSize = "26";
335:                     noteFolder.CreationDate = DateTime.Now;
336:                     noteFolder.ModificationDate = DateTime.Now;
337:                     noteFolder.NoteOrderBy = "NAME";
338:                     noteFolder.NoteOrderStyle = "A";
339:                     noteFolder.NameCount = noteFolder.NoteFolderName + " (" +
noteFolder.NoteCount.ToString() + ")";
340:                     noteFolder.FolderBackground = null;
341:
342:                     context.NoteFolders.InsertOnSubmit(noteFolder);
343:                     context.SubmitChanges();

```

```
344:
345:             lstFolders.ItemsSource = context.NoteFolders;
346:             MessageBox.Show(AppResources.SuccessfulAddNoteFolder);
347:             NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
noteFolder.NoteFolderId, UriKind.Relative));
348:         }
349:     }
350: }
351: };
352: control.btnCancel.Click += (s, args) =>
353: {
354:     popup.IsOpen = false;
355: };
356:
357: //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
358: }
359:
360: private void SettingsButton_Click(object sender, EventArgs e)
361: {
362:     NavigationService.Navigate(new Uri("/GeneralSettings.xaml",
UriKind.Relative));
363: }
364:
365: private void SearchButton_Click(object sender, EventArgs e)
366: {
367:     NavigationService.Navigate(new Uri("/SearchPage.xaml", UriKind.Relative));
368: }
369:
370: private void AboutButton_Click(object sender, EventArgs e)
371: {
372:     NavigationService.Navigate(new Uri("/AboutPage.xaml", UriKind.Relative));
373: }
374:
375: // Sample code for building a localized ApplicationBar
376: //private void BuildLocalizedApplicationBar()
377: //{
378: //    // Set the page's ApplicationBar to a new instance of ApplicationBar.
379: //    ApplicationBar = new ApplicationBar();
380:
381: //    // Create a new button and set the text value to the localized string
from AppResources.
382: //    ApplicationBarIconButton appBarButton = new ApplicationBarIconButton(new
Uri("/Assets/AppBar/appbar.add.rest.png", UriKind.Relative));
383: //    appBarButton.Text = AppResources.AppBarButtonText;
384: //    ApplicationBar.Buttons.Add(appBarButton);
385:
386: //    // Create a new menu item with the localized string from AppResources.
387: //    ApplicationBarMenuItem appBarMenuItem = new
ApplicationBarMenuItem(AppResources.AppBarMenuItemText);
388: //    ApplicationBar.MenuItems.Add(appBarMenuItem);
389: //}
390: }
391: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.23 Note.cs

This is file Note.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5: using System.Threading.Tasks;
6: using System.Data.Linq.Mapping;
7:
8: namespace AwesomeNote
9: {
10:     [Table]
11:     public class Note
12:     {
13:         [Column(IsPrimaryKey = true,
14:             IsDbGenerated = true,
15:             DbType = "INT NOT NULL Identity",
16:             CanBeNull = false)]
17:         public int NoteId { get; set; }
18:
19:         [Column(DbType = "INT NOT NULL",
20:             CanBeNull = false)]
21:         public int NoteFolderId { get; set; }
22:
23:         [Column]
24:         public string NoteName { get; set; }
25:
26:         [Column]
27:         // burada notun kendisi NoteDescription oluyor
28:         public string NoteDescription { get; set; }
29:
30:         [Column]
31:         public DateTime CreationDate { get; set; }
32:
33:         [Column]
34:         public DateTime ModificationDate { get; set; }
35:
36:         [Column]
37:         public string NameCreation { get; set; }
38:
39:         [Column]
40:         public string NameDescription { get; set; }
41:
42:     }
43: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.24 NoteDetail.xaml.cs

This is file NoteDetail.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Text;
7: using System.Windows;
8: using System.Windows.Controls;
```

```
9: using System.Windows.Controls.Primitives;
10: using System.Windows.Input;
11: using System.Windows.Media;
12: using System.Windows.Media.Imaging;
13: using System.Windows.Navigation;
14: using Microsoft.Phone.Tasks;
15: using AwesomeNote.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeNote
20: {
21:     public partial class NoteDetail : PhoneApplicationPage
22:     {
23:
24:         double InputHeight = 0.0;
25:         Popup popup;
26:         public SolidColorBrush messageBackGround;
27:         public bool flag;
28:
29:         public NoteDetail()
30:         {
31:             InitializeComponent();
32:             //SetBackgroundColor();
33:
34:             ApplicationBar = new ApplicationBar();
35:
36:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
37:             button1.IconUri = new Uri("/Assets/PreviousNote.png", UriKind.Relative);
38:             button1.Text = AppResources.PreviousNote;
39:             ApplicationBar.Buttons.Add(button1);
40:             button1.Click += new EventHandler(PreviousNoteButton_Click);
41:
42:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
43:             button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
44:             button2.Text = AppResources.SendWithMail;
45:             ApplicationBar.Buttons.Add(button2);
46:             button2.Click += new EventHandler(SendMailButton_Click);
47:
48:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
49:             button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
50:             button3.Text = AppResources.SendWithSMS;
51:             ApplicationBar.Buttons.Add(button3);
52:             button3.Click += new EventHandler(SendSMSButton_Click);
53:
54:             ApplicationBarIconButton button4 = new ApplicationBarIconButton();
55:             button4.IconUri = new Uri("/Assets/NextNote.png", UriKind.Relative);
56:             button4.Text = AppResources.NextNote;
57:             ApplicationBar.Buttons.Add(button4);
58:             button4.Click += new EventHandler(NextNoteButton_Click);
59:
60:             ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
61:             menuItem1.Text = AppResources.SelectAll;
62:             ApplicationBar.MenuItems.Add(menuItem1);
63:             menuItem1.Click += new EventHandler(SelectAllMenuItem_Click);
64:
65:             ApplicationBarMenuItem menuItem2 = new ApplicationBarMenuItem();
66:             menuItem2.Text = AppResources.Share;
67:             ApplicationBar.MenuItems.Add(menuItem2);
68:             menuItem2.Click += new EventHandler(ShareMenuItem_Click);
69:
70:             ApplicationBarMenuItem menuItem3 = new ApplicationBarMenuItem();
71:             menuItem3.Text = AppResources.ChangeFolder;
72:             ApplicationBar.MenuItems.Add(menuItem3);
73:             menuItem3.Click += new EventHandler(ChangeFolderMenuItem_Click);
74:
75:             ApplicationBarMenuItem menuItem4 = new ApplicationBarMenuItem();
76:             menuItem4.Text = AppResources.DeleteNote;
77:             ApplicationBar.MenuItems.Add(menuItem4);
```

```

78:         menuItem4.Click += new EventHandler(DeleteMenuItem_Click);
79:
80:         ApplicationBarItem menuItem5 = new ApplicationBarItem();
81:         menuItem5.Text = AppResources.NoteInfo;
82:         ApplicationBar.MenuItems.Add(menuItem5);
83:         menuItem5.Click += new EventHandler(NoteInfoMenuItem_Click);
84:
85:
86:
87:         popup = new Popup();
88:     }
89:
90:     public int noteFolderId;
91:     public int noteId;
92:     public string pageName;
93:
94:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
95:     {
96:         ///txtNote.Focus();
97:         ///txtNote.Select(txtNote.Text.Length, 1);
98:         ///determine if HD device
99:         //var deviceWidth = this.ActualWidth;
100:        //var isHdDevice = (deviceWidth > 500 ? true : false);
101:
102:        ///the keyboard height differs between HD devices and regular ones
103:        //if (isHdDevice)
104:        //    keyboardHeight = 540;
105:        //else
106:        //    keyboardHeight = 336;
107:
108:        ///make the keyboard placeholder's height as high as
109:        ///the anticipated keyboard height
110:        ///this will be used to offset other controls on the page into the
viewable area
111:        //pnlKeyboardPlaceholder.Height = keyboardHeight;
112:    }
113:    protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
114:    {
115:        // displays "Fragment: Detail"
116:        //MessageBox.Show("Note Id: " + noteFolder.NoteFolderId);
117:        base.OnFragmentNavigation(e);
118:        using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
119:        {
120:            var note = context.Notes.Where(j =>
j.NoteId.Equals(e.Fragment)).Single() as Note;
121:            lblNoteDetail.Text = note.NoteName;
122:            lblCreationDate.Text = AppResources.CreationDate + ": " +
note.CreationDate.ToString();
123:            noteId = note.NoteId;
124:            var noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
125:
126:            if (pageName == "/SearchPage.xaml")
127:            {
128:                lblNoteDetailApp.Text = AppResources.SearchResults;
129:            }
130:            else
131:            {
132:                lblNoteDetailApp.Text = noteFolder.NoteFolderName;
133:            }
134:            noteFolderId = noteFolder.NoteFolderId;
135:            FontFamily temp = new FontFamily(noteFolder.FontFamily);
136:            txtNote.FontFamily = temp;
137:            txtNote.FontSize = double.Parse(noteFolder.FontSize);
138:            txtNote.Text = note.NoteDescription;
139:            //lstNoteList.DisplayMemberPath = "NoteName";
140:
141:            SetBackgroundColor();

```



```

142:         }
143:     }
144:
145:     protected override void OnNavigatedTo(NavigationEventArgs e)
146:     {
147:         base.OnNavigatedTo(e);
148:         var lastPage = NavigationService.BackStack.FirstOrDefault();
149:         pageName = lastPage.Source.ToString();
150:
151:         //txtNote.Focus();
152:         //txtNote.Select(txtNote.Text.Length, 0);
153:     }
154:
155:     protected override void OnNavigatedFrom(NavigationEventArgs e)
156:     {
157:         base.OnNavigatedFrom(e);
158:     }
159:
160:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
161:     {
162:         if (popup.IsOpen)
163:         {
164:             popup.IsOpen = false;
165:         }
166:
167:         if (this.NavigationService.CanGoBack)
168:         {
169:             SaveMenuItem_Click(this, new EventArgs());
170:             if (pageName == "/SearchPage.xaml")
171:             {
172:                 //this.NavigationService.Navigate(new Uri("/SearchPage.xaml" ,
UriKind.Relative));
173:             }
174:             else
175:             {
176:                 this.NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
noteFolderId, UriKind.Relative));
177:             }
178:         }
179:     }
180:
181:     private void ApplicationBarMenuItem_Click(object sender, EventArgs e)
182:     {
183:     }
184:
185:
186:     private void SendSMSButton_Click(object sender, EventArgs e)
187:     {
188:         SmsComposeTask smsComposeTask = new SmsComposeTask();
189:
190:         smsComposeTask.To = "";
191:         smsComposeTask.Body = lblNoteDetail.Text + ": " + txtNote.Text;
192:
193:         smsComposeTask.Show();
194:         //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
195:     }
196:
197:     private void NextNoteButton_Click(object sender, EventArgs e)
198:     {
199:         SaveMenuItem_Click(this, new EventArgs());
200:
201:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
202:         {
203:             int gidecekSayfa = 0;
204:             var note = context.Notes.Where(j => j.NoteId.Equals(noteId)).Single()
as Note;
205:             var noteFolder = context.NoteFolders.Where(j =>

```

```

j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
206:         string orderStyle = noteFolder.NoteOrderStyle;
207:         List<Note> noteList = new List<Note>();
208:         switch (noteFolder.NoteOrderBy)
209:         {
210:             case "NAME":
211:                 if (orderStyle == "A")
212:                 {
213:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.NoteName).ToList() as
List<Note>;
214:                 }
215:                 else
216:                 {
217:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j => j.NoteName).ToList()
as List<Note>;
218:                 }
219:                 break;
220:             case "CDATE":
221:                 if (orderStyle == "A")
222:                 {
223:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.CreationDate).ToList() as
List<Note>;
224:                 }
225:                 else
226:                 {
227:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j =>
j.CreationDate).ToList() as List<Note>;
228:                 }
229:                 break;
230:             case "MDATE":
231:                 if (orderStyle == "A")
232:                 {
233:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.ModificationDate).ToList()
as List<Note>;
234:                 }
235:                 else
236:                 {
237:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j =>
j.ModificationDate).ToList() as List<Note>;
238:                 }
239:                 break;
240:             default:
241:                 if (orderStyle == "A")
242:                 {
243:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.NoteName).ToList() as
List<Note>;
244:                 }
245:                 else
246:                 {
247:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j => j.NoteName).ToList()
as List<Note>;
248:                 }
249:                 break;
250:         }
251:         if (note != noteList.Last())
252:         {
253:             for (int i = 0; i < noteList.Count; i++)
254:             {
255:                 if (note == noteList[i])
256:                 {
257:

```

```

258:             gidecekSayfa = i + 1;
259:             break;
260:         }
261:     }
262:     lblNoteDetail.Text = noteList[gidecekSayfa].NoteName;
263:     txtNote.Text = noteList[gidecekSayfa].NoteDescription;
264:     noteId = noteList[gidecekSayfa].NoteId;
265:     //txtNote.Focus();
266:     //txtNote.Select(txtNote.Text.Length,1);
267: }
268: else
269: {
270:
271: }
272:
273: }
274: //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
275: }
276:
277: private void PreviousNoteButton_Click(object sender, EventArgs e)
278: {
279:     SaveMenuItem_Click(this, new EventArgs());
280:
281:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
282:     {
283:         var note = context.Notes.Where(j => j.NoteId.Equals(noteId)).Single()
as Note;
284:         var noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
285:         string orderStyle = noteFolder.NoteOrderStyle;
286:         int gidecekSayfa = 0;
287:         List<Note> noteList = new List<Note>();
288:         switch (noteFolder.NoteOrderBy)
289:         {
290:             case "NAME":
291:                 if (orderStyle == "A")
292:                 {
293:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.NoteName).ToList() as
List<Note>;
294:                 }
295:                 else
296:                 {
297:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j => j.NoteName).ToList()
as List<Note>;
298:                 }
299:                 break;
300:             case "CDATE":
301:                 if (orderStyle == "A")
302:                 {
303:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.CreationDate).ToList() as
List<Note>;
304:                 }
305:                 else
306:                 {
307:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j =>
j.CreationDate).ToList() as List<Note>;
308:                 }
309:                 break;
310:             case "MDATE":
311:                 if (orderStyle == "A")
312:                 {
313:                     noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.ModificationDate).ToList()
as List<Note>;

```

```

314:         }
315:         else
316:         {
317:             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j =>
j.ModificationDate).ToList() as List<Note>;
318:         }
319:         break;
320:         default:
321:             if (orderStyle == "A")
322:             {
323:                 noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderBy(j => j.NoteName).ToList() as
List<Note>;
324:             }
325:             else
326:             {
327:                 noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolder.NoteFolderId)).OrderByDescending(j => j.NoteName).ToList()
as List<Note>;
328:             }
329:             break;
330:         }
331:
332:         if (note != noteList.First())
333:         {
334:             for (int i = 0; i < noteList.Count; i++)
335:             {
336:                 if (note == noteList[i])
337:                 {
338:                     gidecekSayfa = i - 1;
339:                     break;
340:                 }
341:             }
342:             lblNoteDetail.Text = noteList[gidecekSayfa].NoteName;
343:             txtNote.Text = noteList[gidecekSayfa].NoteDescription;
344:             noteId = noteList[gidecekSayfa].NoteId;
345:             //txtNote.Focus();
346:             //txtNote.Select(txtNote.Text.Length, 1);
347:         }
348:         else
349:         {
350:
351:         }
352:
353:     }
354:     //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
355: }
356:
357: private void ShareMenuItem_Click(object sender, EventArgs e)
358: {
359:     ShareStatusTask shareStatusTask = new ShareStatusTask();
360:
361:     StringBuilder sb = new StringBuilder();
362:
363:     sb.AppendLine(lblNoteDetail.Text);
364:     sb.AppendLine();
365:     sb.AppendLine(txtNote.Text);
366:     sb.AppendLine();
367:     sb.AppendLine(AppResources.SendWithApp);
368:
369:     shareStatusTask.Status = sb.ToString();
370:
371:     shareStatusTask.Show();
372: }
373:
374: private void SendMailButton_Click(object sender, EventArgs e)
375: {
376:     // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor

```

```

377:         //EmailAddressChooserTask emailAddressChooserTask;
378:         //emailAddressChooserTask = new EmailAddressChooserTask();
379:         //emailAddressChooserTask.Completed += new
EventHandler<EmailResult>(emailAddressChooserTask_Completed);
380:         //emailAddressChooserTask.Show();
381:         StringBuilder sb = new StringBuilder();
382:         EmailComposeTask emailComposeTask = new EmailComposeTask();
383:
384:         sb.AppendLine(txtNote.Text);
385:         sb.AppendLine();
386:         sb.AppendLine(AppResources.SendWithApp);
387:
388:         emailComposeTask.Subject = lblNoteDetail.Text;
389:         emailComposeTask.Body = sb.ToString();
390:         emailComposeTask.To = "";
391:         emailComposeTask.Cc = "";
392:         emailComposeTask.Bcc = "";
393:
394:         emailComposeTask.Show();
395:         //MessageBox.Show(AppResources.SuccessfulSendWithMail);
396:     }
397:
398:     // burada da seçilen e_posta adresi üzerinden gönderim yapılacak
399:     void emailAddressChooserTask_Completed(object sender, EmailResult e)
400:     {
401:         if (e.TaskResult == TaskResult.OK)
402:         {
403:             //MessageBox.Show("The email for " + e.DisplayName + " is " + e.Email);
404:
405:             //Code to send a new email message using the retrieved email address.
406:             EmailComposeTask emailComposeTask = new EmailComposeTask();
407:             emailComposeTask.To = e.Email;
408:             emailComposeTask.Subject = e.DisplayName + ", here is an email from my
app!";
409:             emailComposeTask.Show();
410:         }
411:     }
412:
413:     private void SelectAllMenuItem_Click(object sender, EventArgs e)
414:     {
415:         flag = true;
416:         txtNote.Focus();
417:         txtNote.SelectAll();
418:     }
419:
420:     private void SaveMenuItem_Click(object sender, EventArgs e)
421:     {
422:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
423:         {
424:             var notes = context.Notes.Where(j => j.NoteId.Equals(noteId)).Select(j
=> j);
425:             foreach (var note in notes)
426:             {
427:                 note.NoteDescription = txtNote.Text;
428:                 note.ModificationDate = DateTime.Now;
429:                 note.NameDescription = note.NoteName + " " + note.NoteDescription;
430:                 //note.NameDescriptionWithoutNewline =
note.NameDescription.Replace(Environment.NewLine, " ");
431:             }
432:             context.SubmitChanges();
433:         }
434:         //MessageBox.Show(AppResources.NoteSaved);
435:     }
436:
437:
438:     // notu silmek için kullanılan bir metod bu
439:     private void DeleteMenuItem_Click(object sender, EventArgs e)
440:     {

```

```

441:         if (MessageBox.Show(AppResources.DeleteNoteQuestion,
442:             AppResources.DeleteNote, MessageBoxButton.OKCancel)
443:             != MessageBoxResult.OK)
444:         {
445:         }
446:     }
447:     else
448:     {
449:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
450:         {
451:             var note = context.Notes.Where(j =>
j.NoteId.Equals(noteId)).Single() as Note;
452:             context.Notes.DeleteOnSubmit(note);
453:             context.SubmitChanges();
454:         }
455:         MessageBox.Show(AppResources.SuccessfulDeleteNote);
456:         NavigationService.Navigate(new Uri("/NoteFolder.xaml#" + noteFolderId,
UriKind.Relative));
457:     }
458:     //MessageBox.Show(AppResources.NoteSaved);
459: }
460:
461: private void NoteInfoMenuItem_Click(object sender, EventArgs e)
462: {
463:     NavigationService.Navigate(new Uri("/NoteInfo.xaml#" + noteId,
UriKind.Relative));
464:     //MessageBox.Show(AppResources.NoteSaved);
465: }
466:
467: private void ChangeFolderMenuItem_Click(object sender, EventArgs e)
468: {
469:     NavigationService.Navigate(new Uri("/ChangeNoteFolder.xaml#" + noteId,
UriKind.Relative));
470: }
471:
472: // notun adini degistirmek için kullanılan bir metod
473: private void lblNoteDetail_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
474: {
475:     popup = new Popup();
476:     popup.Height = 300;
477:     popup.Width = 400;
478:     popup.VerticalOffset = 20;
479:     PopupChangeNoteName control = new PopupChangeNoteName();
480:     control.txtLabel.Text = AppResources.EnterNoteName;
481:     control.btnCancel.Content = AppResources.Cancel;
482:     control.btnOK.Content = AppResources.OK;
483:     popup.Child = control;
484:     popup.IsOpen = true;
485:     control.txtNoteName.Text = lblNoteDetail.Text;
486:     control.txtNoteName.Focus();
487:     control.txtNoteName.Select(0, control.txtNoteName.Text.Length);
488:
489:     control.btnOK.Click += (s, args) =>
490:     {
491:         bool isCreated;
492:         string noteName;
493:         popup.IsOpen = false;
494:         noteName = control.txtNoteName.Text;
495:
496:         int length = control.txtNoteName.Text.Length;
497:         string space = control.txtNoteName.Text.Substring(length - Math.Min(1,
length));
498:         if (space == " ")
499:         {
500:             noteName = control.txtNoteName.Text.Remove(length - 1, 1);
501:         }
502:         else

```

```

503:         {
504:             noteName = control.txtNoteName.Text;
505:         }
506:
507:         if (noteName != lblNoteDetail.Text)
508:         {
509:             // ayni isimde bir notun daha önceden olusturulup olusturulmadigini
510:             // kontrol eden bir kod bölümü
511:             using (var contextFolder = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
512:             {
513:                 isCreated =
514:                     contextFolder.Notes.Any(j => j.NoteName.Equals(noteName));
515:             }
516:             if (isCreated == true)
517:             {
518:                 MessageBox.Show(AppResources.NoteExists);
519:             }
520:             // eger bu isimde bir not olusturulmamissa
521:             // olusturulmasi için gerekli kodlar asagidadir
522:             else
523:             {
524:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
525:                 {
526:                     var notes = context.Notes.Where(j =>
j.NoteId.Equals(noteId)).Select(j => j);
527:                     foreach (var note in notes)
528:                     {
529:                         note.NoteName = noteName;
530:                         note.ModificationDate = DateTime.Now;
531:                         note.NameCreation = note.NoteName + " (" +
note.CreationDate.ToString() + ")";
532:                         note.NameDescription = note.NoteName + " " +
note.NoteDescription;
533:                         //note.NameDescriptionWithoutNewline =
note.NameDescription.Replace(Environment.NewLine, " ");
534:                     }
535:                     context.SubmitChanges();
536:                     //lstFolders.ItemsSource = context.NoteFolders;
537:                     MessageBox.Show(AppResources.SuccessfulChangeNoteName);
538:                     Note note2 = context.Notes.Where(j =>
j.NoteName.Equals(noteName)).Single() as Note;
539:                     NavigationService.Navigate(new Uri("/NoteDetail.xaml#" +
note2.NoteId, UriKind.Relative));
540:                 }
541:             }
542:         }
543:     };
544:     control.btnCancel.Click += (s, args) =>
545:     {
546:         popup.IsOpen = false;
547:     };
548: }
549:
550:
551: private void txtNote_TextInputStart(object sender,
System.Windows.Input.TextCompositionEventArgs e)
552: {
553:     this.svNote.UpdateLayout();
554:     this.svNote.ScrollToVerticalOffset(this.txtNote.ActualHeight);
555:     this.svNote.UpdateLayout();
556: }
557:
558: private void svNote_GotFocus(object sender, RoutedEventArgs e)
559: {
560:     //this.svNote.Height = 590;
561:     this.svNote.ScrollToVerticalOffset(this.txtNote.ActualHeight);
562:     this.svNote.UpdateLayout();

```

```

563:         //App.RootFrame.RenderTransform = new CompositeTransform();
564:     }
565:
566:
567:     private void svNote_LostFocus(object sender, RoutedEventArgs e)
568:     {
569:         //this.svNote.Height = 600;
570:     }
571:
572:     private void txtNote_Tap(object sender, System.Windows.Input.GestureEventArgs
573: e)
574:     {
575:         txtNote.Focus();
576:         //txtNote.Select(txtNote.Text.Length, 1);
577:         svNote.ScrollToVerticalOffset(e.GetPosition(txtNote).Y - 80);
578:     }
579:
580:     private void txtNote_TextChanged(object sender, TextChangedEventArgs e)
581:     {
582:         Dispatcher.BeginInvoke(() =>
583:         {
584:             double CurrentInputHeight = txtNote.ActualHeight;
585:             if (CurrentInputHeight > InputHeight)
586:             {
587:                 svNote.ScrollToVerticalOffset(svNote.VerticalOffset +
588: CurrentInputHeight - InputHeight);
589:             }
590:             InputHeight = CurrentInputHeight;
591:         });
592:     }
593:
594:     private void txtNote_GotFocus(object sender, RoutedEventArgs e)
595:     {
596:
597:         //reset any page movement cause by keyboard opening
598:         App.RootFrame.RenderTransform = new CompositeTransform();
599:     }
600:
601:     private void SetBackgroundColor()
602:     {
603:         AppSettings appSettings = new AppSettings();
604:         NoteFolder noteFolder;
605:         using (var context = new
606: NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
607:         {
608:             appSettings = context.AppSettings.First() as AppSettings;
609:             noteFolder = context.NoteFolders.Where(j =>
610: j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
611:         }
612:         if (noteFolder.FolderBackground != null)
613:         {
614:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
615:             BitmapImage image = new BitmapImage();
616:             image.SetSource(stream);
617:             ImageBrush ib = new ImageBrush();
618:             ib.ImageSource = image;
619:             this.LayoutRoot.Background = ib;
620:         }
621:         else
622:         {
623:             switch (appSettings.AppBackgroundColor)
624:             {
625:                 case "BLA":
626:                     messageBackGround = new SolidColorBrush(Colors.Black);
627:                     //this.LayoutRoot.Background = new ImageBrush(new
628: BitmapImage());

```



```
627:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
628:         break;
629:     case "BLU":
630:         messageBackGround = new SolidColorBrush(Colors.Blue);
631:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
632:         break;
633:     case "BRO":
634:         messageBackGround = new SolidColorBrush(Colors.Brown);
635:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
636:         break;
637:     case "RED":
638:         messageBackGround = new SolidColorBrush(Colors.Red);
639:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
640:         break;
641:     case "GRE":
642:         messageBackGround = new SolidColorBrush(Colors.Green);
643:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
644:         break;
645:     case "GRA":
646:         messageBackGround = new SolidColorBrush(Colors.Gray);
647:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
648:         break;
649:     case "YEL":
650:         messageBackGround = new SolidColorBrush(Colors.Yellow);
651:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
652:         break;
653:     case "ORA":
654:         messageBackGround = new SolidColorBrush(Colors.Orange);
655:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
656:         break;
657:     case "PUR":
658:         messageBackGround = new SolidColorBrush(Colors.Purple);
659:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
660:         break;
661:     default:
662:         messageBackGround = new SolidColorBrush(Colors.Black);
663:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
664:         break;
665:     }
666: }
667: }
668:
669: private void txtNote_KeyDown(object sender, System.Windows.Input.KeyEventArgs
e)
670: {
671:     if (e.Key == Key.Enter)
672:     {
673:         svNote.ScrollToVerticalOffset(txtNote.ActualHeight);
674:     }
675: }
676:
677: private void txtNote_LostFocus(object sender, RoutedEventArgs e)
678: {
679:     if (!flag) return;
680:     txtNote.Focus();
681:     flag = false;
682:     this.pnlKeyboardPlaceholder.Visibility = Visibility.Collapsed;
683: }
684:
685: }
686: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.25 NoteFolder.cs

This is file NoteFolder.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Text;
5: using System.Threading.Tasks;
6: using System.Data.Linq.Mapping;
7: using System.Windows.Media.Imaging;
8:
9: namespace AwesomeNote
10: {
11:     [Table]
12:     public class NoteFolder
13:     {
14:         [Column(IsPrimaryKey = true,
15: IsDbGenerated = true,
16: DbType = "INT NOT NULL Identity",
17: CanBeNull = false)]
18:         public int NoteFolderId { get; set; }
19:         [Column]
20:         public string NoteFolderName { get; set; }
21:         [Column]
22:         public string NoteFolderDescription { get; set; }
23:         [Column]
24:         public bool IsPasswordProtected { get; set; }
25:         [Column]
26:         public string NoteFolderPassword { get; set; }
27:         [Column]
28:         public string FontFamily { get; set; }
29:         [Column]
30:         public string FontSize { get; set; }
31:         [Column]
32:         public string NoteOrderBy { get; set; }
33:         [Column]
34:         public string NoteOrderStyle { get; set; }
35:         [Column]
36:         public DateTime CreationDate { get; set; }
37:
38:         [Column]
39:         public DateTime ModificationDate { get; set; }
40:
41:         [Column]
42:         public int NoteCount { get; set; }
43:
44:         [Column]
45:         public string NameCount { get; set; }
46:
47:         [Column (DbType = "Image", UpdateCheck = UpdateCheck.Never)]
48:         public byte[] FolderBackground { get; set; }
49:         //[Column]
50:         //public string FontStyle { get; set; }
51:         //[Column]
52:         //public string FontWeight { get; set; }
53:     }
```

```
54: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.26 NoteFolder.xaml.cs

This is file NoteFolder.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Controls.Primitives;
9: using System.Windows.Media;
10: using System.Windows.Media.Imaging;
11: using System.Windows.Navigation;
12: using AwesomeNote.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16: namespace AwesomeNote
17: {
18:     public partial class NoteList : PhoneApplicationPage
19:     {
20:
21:         //List<Note> noteListTemplate { set; get; }
22:         public NoteList()
23:         {
24:             InitializeComponent();
25:             //SetBackgroundColor();
26:
27:             ApplicationBar = new ApplicationBar();
28:
29:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
30:             button1.IconUri = new Uri("/Assets/AddNote.png", UriKind.Relative);
31:             button1.Text = AppResources.AddNote;
32:             ApplicationBar.Buttons.Add(button1);
33:             button1.Click += new EventHandler(AddNoteButton_Click);
34:
35:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
36:             button2.IconUri = new Uri("/Assets/DeleteFolder.png", UriKind.Relative);
37:             button2.Text = AppResources.DeleteFolder;
38:             ApplicationBar.Buttons.Add(button2);
39:             button2.Click += new EventHandler>DeleteFolderButton_Click);
40:
41:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
42:             button3.IconUri = new Uri("/Assets/FolderSettings.png", UriKind.Relative);
43:             button3.Text = AppResources.FolderSettings;
44:             ApplicationBar.Buttons.Add(button3);
45:             button3.Click += new EventHandler(FolderSettingsButton_Click);
46:
47:             popup = new Popup();
48:             //lstNoteList.DataContext = noteListTemplate;
49:         }
50:
51:         public int noteFolderId;
52:         public int noteId;
```

```

53:         public Popup popup;
54:         public SolidColorBrush messageBackGround;
55:
56:         private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
57:         {
58:
59:         }
60:
61:         // not klasörünün altındaki notlar listelenirken
62:         // ada göre- oluşturma zamanına göre vs. vs.
63:         // artan azalan bir şekilde ayarlanıp öyle listeleniyor
64:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
65:         {
66:             List<Note> noteList = new List<Note>();
67:
68:             // displays "Fragment: Detail"
69:             //MessageBox.Show("Folder Id: " + e.Fragment);
70:             base.OnFragmentNavigation(e);
71:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
72:             {
73:                 var noteFolder = context.NoteFolders.Where( j =>
j.NoteFolderId.Equals(e.Fragment)).Single() as NoteFolder;
74:                 string orderStyle = noteFolder.NoteOrderStyle;
75:                 switch (noteFolder.NoteOrderBy)
76:                 {
77:                     case "NAME":
78:                         if (orderStyle == "A")
79:                         {
80:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.NoteName).ToList() as List<Note>;
81:                         }
82:                         else
83:                         {
84:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.NoteName).ToList() as
List<Note>;
85:                         }
86:                         break;
87:                     case "CDATE":
88:                         if (orderStyle == "A")
89:                         {
90:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.CreationDate).ToList() as List<Note>;
91:                         }
92:                         else
93:                         {
94:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.CreationDate).ToList() as
List<Note>;
95:                         }
96:                         break;
97:                     case "MDATE":
98:                         if (orderStyle == "A")
99:                         {
100:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.ModificationDate).ToList() as List<Note>;
101:                         }
102:                         else
103:                         {
104:                             noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.ModificationDate).ToList() as
List<Note>;
105:                         }
106:                         break;
107:                     default:
108:                         if (orderStyle == "A")
109:                         {
110:                             noteList = context.Notes.Where(j =>

```

```

j.NoteFolderId.Equals(e.Fragment)).OrderBy(j => j.NoteName).ToList() as List<Note>;
111:                }
112:            else
113:            {
114:                noteList = context.Notes.Where(j =>
j.NoteFolderId.Equals(e.Fragment)).OrderByDescending(j => j.NoteName).ToList() as
List<Note>;
115:            }
116:            break;
117:        }
118:
119:        lstNoteList.Items.Clear();
120:        noteFolderId = noteFolder.NoteFolderId;
121:        lblNoteFolder.Text = noteFolder.NoteFolderName;
122:        lblNoteList.Text = AppResources.NoteListOf + " (" +
noteFolder.NoteFolderName + ")";
123:        lstNoteList.ItemsSource = noteList;
124:        lstNoteList.DisplayMemberPath = "NoteName";
125:        SetBackgroundColor();
126:        //lstNoteList.DisplayMemberPath = "NameCreation";
127:    }
128:    }
129:
130:    private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
131:    {
132:
133:        if (popup.IsOpen)
134:        {
135:            popup.IsOpen = false;
136:        }
137:        if (this.NavigationService.CanGoBack)
138:        {
139:            this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
140:        }
141:    }
142:
143:    private void lstNoteList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
144:    {
145:        var note = (Note)lstNoteList.SelectedItem;
146:        noteId = note.NoteId;
147:        noteFolderId = note.NoteFolderId;
148:        NavigationService.Navigate(new Uri("/NoteDetail.xaml#" + noteId,
UriKind.Relative));
149:    }
150:
151:    protected override void OnNavigatedTo(NavigationEventArgs e)
152:    {
153:        base.OnNavigatedTo(e);
154:    }
155:
156:    protected override void OnNavigatedFrom(NavigationEventArgs e)
157:    {
158:        base.OnNavigatedFrom(e);
159:    }
160:
161:    private void AddNoteButton_Click(object sender, EventArgs e)
162:    {
163:        popup = new Popup();
164:        popup.Height = 300;
165:        popup.Width = 400;
166:        popup.VerticalOffset = 20;
167:        PopupAddNote control = new PopupAddNote();
168:        control.txtLabel.Text = AppResources.EnterNoteName;
169:        control.btnCancel.Content = AppResources.Cancel;
170:        control.btnOK.Content = AppResources.OK;
171:        popup.Child = control;

```

```

172:         popup.IsOpen = true;
173:         control.txtNoteName.Focus();
174:
175:         control.btnOK.Click += (s, args) =>
176:         {
177:             bool isCreated;
178:             string noteName;
179:             popup.IsOpen = false;
180:             noteName = control.txtNoteName.Text;
181:
182:             int length = control.txtNoteName.Text.Length;
183:             string space = control.txtNoteName.Text.Substring(length - Math.Min(1,
length));
184:             if (space == " ")
185:             {
186:                 noteName = control.txtNoteName.Text.Remove(length - 1, 1);
187:             }
188:             else
189:             {
190:                 noteName = control.txtNoteName.Text;
191:             }
192:
193:             // aynı isimde bir klasörün daha önceden oluşturulup oluşturulmadığını
194:             // kontrol eden bir kod bölümü
195:             using (var contextFolder = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
196:             {
197:                 isCreated =
198:                     contextFolder.Notes.Any(j => j.NoteName.Equals(noteName));
199:             }
200:             if (isCreated == true)
201:             {
202:                 MessageBox.Show(AppResources.NoteExists);
203:             }
204:             // eğer bu isimde bir klasör oluşturulmamışsa
205:             // oluşturulması için gerekli kodlar aşağıdadır
206:             else
207:             {
208:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
209:                 {
210:                     Note note = new Note();
211:                     note.NoteName = noteName;
212:                     note.NoteFolderId = noteFolderId;
213:                     note.NoteDescription = "";
214:                     note.CreationDate = DateTime.Now;
215:                     note.ModificationDate = DateTime.Now;
216:                     note.NameCreation = note.NoteName + " (" +
note.CreationDate.ToString() + ")";
217:                     note.NameDescription = note.NoteName + " " +
note.NoteDescription;
218:                     //note.NameDescriptionWithoutNewline =
note.NameDescription.Replace(Environment.NewLine, " ");
219:                     //note.IsPasswordProtected = false;
220:
221:                     context.Notes.InsertOnSubmit(note);
222:                     context.SubmitChanges();
223:
224:                     var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
225:                     foreach (var noteFolder in noteFolders)
226:                     {
227:                         noteFolder.ModificationDate = DateTime.Now;
228:                         noteFolder.NoteCount = noteFolder.NoteCount + 1;
229:                         noteFolder.NameCount = noteFolder.NoteFolderName + " (" +
noteFolder.NoteCount + ")";
230:                     }
231:                     context.SubmitChanges();
232:

```

```

233:         lstNoteList.ItemsSource = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteFolderId));
234:         MessageBox.Show(AppResources.SuccessfulAddNote);
235:         Note note2 = context.Notes.Where(j =>
j.NoteName.Equals(noteName)).Single() as Note;
236:         NavigationService.Navigate(new Uri("/NoteDetail.xaml#" +
note2.NoteId, UriKind.Relative));
237:     }
238: }
239: };
240: control.btnCancel.Click += (s, args) =>
241: {
242:     popup.IsOpen = false;
243: };
244:
245: //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
246: }
247:
248: private void FolderSettingsButton_Click(object sender, EventArgs e)
249: {
250:     NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
251: }
252:
253: private void lblNoteFolder_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
254: {
255:     popup = new Popup();
256:     popup.Height = 300;
257:     popup.Width = 400;
258:     popup.VerticalOffset = 20;
259:     PopupChangeFolderName control = new PopupChangeFolderName();
260:     control.txtLabel.Text = AppResources.EnterNoteName;
261:     control.btnCancel.Content = AppResources.Cancel;
262:     control.btnOK.Content = AppResources.OK;
263:     popup.Child = control;
264:     popup.IsOpen = true;
265:     control.txtFolderName.Text = lblNoteFolder.Text;
266:     control.txtFolderName.Focus();
267:     control.txtFolderName.Select(0, control.txtFolderName.Text.Length);
268:
269:     control.btnOK.Click += (s, args) =>
270:     {
271:         bool isCreated;
272:         string folderName;
273:         popup.IsOpen = false;
274:         folderName = control.txtFolderName.Text;
275:
276:         int length = control.txtFolderName.Text.Length;
277:         string space = control.txtFolderName.Text.Substring(length -
Math.Min(1, length));
278:         if (space == " ")
279:         {
280:             folderName = control.txtFolderName.Text.Remove(length - 1, 1);
281:         }
282:         else
283:         {
284:             folderName = control.txtFolderName.Text;
285:         }
286:
287:
288:         if (folderName != lblNoteFolder.Text)
289:         {
290:             // ayni isimde bir klasörün daha önceden olusturulup
olusturulmadigini
291:             // kontrol eden bir kod bölümü
292:             using (var contextFolder = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
293:             {

```

```

294:             isCreated =
295:                 contextFolder.NoteFolders.Any(j =>
j.NoteFolderName.Equals(folderName));
296:         }
297:         if (isCreated == true)
298:         {
299:             MessageBox.Show(AppResources.FolderExists);
300:         }
301:         // eger bu isimde bir klasör olusturulmamissa
302:         // olusturulmasi için gerekli kodlar asagidadir
303:         else
304:         {
305:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
306:             {
307:                 var noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
308:                 foreach (var folder in noteFolder)
309:                 {
310:                     folder.NoteFolderName = folderName;
311:                     folder.ModificationDate = DateTime.Now;
312:                     folder.NameCount = folder.NoteFolderName + " (" +
folder.NoteCount.ToString() + ")";
313:                 }
314:                 context.SubmitChanges();
315:                 //lstFolders.ItemsSource = context.NoteFolders;
316:                 MessageBox.Show(AppResources.SuccessfulChangeFolderName);
317:                 NoteFolder note2 = context.NoteFolders.Where(j =>
j.NoteFolderName.Equals(folderName)).Single() as NoteFolder;
318:                 NavigationService.Navigate(new Uri("/NoteFolder.xaml#" +
note2.NoteFolderId, UriKind.Relative));
319:             }
320:         }
321:     }
322: };
323: control.btnCancel.Click += (s, args) =>
324: {
325:     popup.IsOpen = false;
326: };
327: }
328:
329: private void DeleteFolderButton_Click(object sender, EventArgs e)
330: {
331:     if (MessageBox.Show(AppResources.DeleteFolderQuestion,
AppResources.DeleteFolder, MessageBoxButton.OKCancel)
332:         != MessageBoxResult.OK)
333:     {
334:     }
335: }
336: }
337: else
338: {
339:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
340:     {
341:         var note = context.Notes.Where(j =>
j.NoteFolderId.Equals(noteId)).ToList() as List<Note>;
342:         context.Notes.DeleteAllOnSubmit(note);
343:
344:         var noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
345:         context.NoteFolders.DeleteOnSubmit(noteFolder);
346:
347:         context.SubmitChanges();
348:     }
349:     MessageBox.Show(AppResources.SuccessfulDeleteFolder);
350:     NavigationService.Navigate(new Uri("/MainPage.xaml" ,
UriKind.Relative));
351: }
352: //MessageBox.Show(AppResources.NoteSaved);

```



```

353:     }
354:
355:     private void SetBackgroundColor()
356:     {
357:         AppSettings appSettings = new AppSettings();
358:         NoteFolder noteFolder;
359:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
360:         {
361:             appSettings = context.AppSettings.First() as AppSettings;
362:             noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
363:         }
364:
365:         if (noteFolder.FolderBackground != null)
366:         {
367:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
368:             BitmapImage image = new BitmapImage();
369:             image.SetSource(stream);
370:             ImageBrush ib = new ImageBrush();
371:             ib.ImageSource = image;
372:             this.LayoutRoot.Background = ib;
373:         }
374:         else
375:         {
376:             switch (appSettings.AppBackgroundColor)
377:             {
378:                 case "BLA":
379:                     messageBackGround = new SolidColorBrush(Colors.Black);
380:                     //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
381:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
382:                     break;
383:                 case "BLU":
384:                     messageBackGround = new SolidColorBrush(Colors.Blue);
385:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
386:                     break;
387:                 case "BRO":
388:                     messageBackGround = new SolidColorBrush(Colors.Brown);
389:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
390:                     break;
391:                 case "RED":
392:                     messageBackGround = new SolidColorBrush(Colors.Red);
393:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
394:                     break;
395:                 case "GRE":
396:                     messageBackGround = new SolidColorBrush(Colors.Green);
397:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
398:                     break;
399:                 case "GRA":
400:                     messageBackGround = new SolidColorBrush(Colors.Gray);
401:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
402:                     break;
403:                 case "YEL":
404:                     messageBackGround = new SolidColorBrush(Colors.Yellow);
405:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
406:                     break;
407:                 case "ORA":
408:                     messageBackGround = new SolidColorBrush(Colors.Orange);
409:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
410:                     break;
411:                 case "PUR":
412:                     messageBackGround = new SolidColorBrush(Colors.Purple);
413:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
414:                     break;
415:                 default:

```

```
416:         messageBackGround = new SolidColorBrush(Colors.Black);
417:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
418:         break;
419:     }
420: }
421: }
422: }
423: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.27 NotelInfo.xaml.cs

This is file NotelInfo.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Text;
7: using System.Windows;
8: using System.Windows.Controls;
9: using System.Windows.Documents;
10: using System.Windows.Media;
11: using System.Windows.Media.Imaging;
12: using System.Windows.Navigation;
13: using AwesomeNote.Resources;
14: using Microsoft.Phone.Controls;
15: using Microsoft.Phone.Shell;
16:
17: namespace AwesomeNote
18: {
19:     public partial class Page1 : PhoneApplicationPage
20:     {
21:         public SolidColorBrush messageBackGround;
22:         public Page1()
23:         {
24:             InitializeComponent();
25:             //SetBackgroundColor();
26:         }
27:
28:         public int noteId;
29:         public int noteFolderId;
30:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
31:         {
32:             // displays "Fragment: Detail"
33:             //MessageBox.Show("Note Id: " + e.Fragment);
34:             base.OnFragmentNavigation(e);
35:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
36:             {
37:                 var note = context.Notes.Where(j =>
j.NoteId.Equals(e.Fragment)).Single() as Note;
38:                 var noteFolder =
39:                     context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(note.NoteFolderId)).Single() as NoteFolder;
40:                 lblNoteName.Text = note.NoteName;
41:                 noteId = note.NoteId;
42:                 noteFolderId = note.NoteFolderId;
```

```

43:         lblNoteInfo.Text = AppResources.About;
44:         StringBuilder sb = new StringBuilder();
45:         string[] wordNumbers;
46:         wordNumbers = note.NoteDescription.Split(' ');
47:
48:         // burada not dosyasi hakkında istatistiki bilgiler hazirlaniyor
49:         sb.AppendLine(AppResources.BelongFolderName + ": " +
noteFolder.NoteFolderName);
50:         sb.AppendLine(AppResources.CharacterNumber + ": " +
note.NoteDescription.Length);
51:         sb.AppendLine(AppResources.WordNumber + ": " + (wordNumbers.Length-1));
52:         sb.AppendLine(AppResources.CreationDate + ": " + note.CreationDate);
53:         sb.AppendLine(AppResources.ModificationDate + ": " +
note.ModificationDate);
54:
55:         //var paragraph = new Paragraph();
56:         //paragraph.Inlines.Add(sb.ToString());
57:         //txtNoteInfo.Blocks.Add(paragraph);
58:         txtNoteInfo.Text = sb.ToString();
59:         //lstNoteList.DisplayMemberPath = "NoteName";
60:         SetBackgroundColor();
61:     }
62: }
63:
64: protected override void OnNavigatedTo(NavigationEventArgs e)
65: {
66:     base.OnNavigatedTo(e);
67: }
68:
69: protected override void OnNavigatedFrom(NavigationEventArgs e)
70: {
71:     base.OnNavigatedFrom(e);
72: }
73:
74: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
75: {
76:     if (this.NavigationService.CanGoBack)
77:     {
78:         this.NavigationService.Navigate(new Uri("/NoteDetail.xaml#" + noteId,
UriKind.Relative));
79:     }
80: }
81:
82: private void SetBackgroundColor()
83: {
84:     AppSettings appSettings = new AppSettings();
85:     NoteFolder noteFolder;
86:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
87:     {
88:         appSettings = context.AppSettings.First() as AppSettings;
89:         noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
90:     }
91:
92:     if (noteFolder.FolderBackground != null)
93:     {
94:         MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
95:         BitmapImage image = new BitmapImage();
96:         image.SetSource(stream);
97:         ImageBrush ib = new ImageBrush();
98:         ib.ImageSource = image;
99:         this.LayoutRoot.Background = ib;
100:     }
101:     else
102:     {
103:         switch (appSettings.AppBackgroundColor)
104:         {

```

```
105:         case "BLA":
106:             messageBackGround = new SolidColorBrush(Colors.Black);
107:             //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
108:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
109:             break;
110:         case "BLU":
111:             messageBackGround = new SolidColorBrush(Colors.Blue);
112:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
113:             break;
114:         case "BRO":
115:             messageBackGround = new SolidColorBrush(Colors.Brown);
116:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
117:             break;
118:         case "RED":
119:             messageBackGround = new SolidColorBrush(Colors.Red);
120:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
121:             break;
122:         case "GRE":
123:             messageBackGround = new SolidColorBrush(Colors.Green);
124:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
125:             break;
126:         case "GRA":
127:             messageBackGround = new SolidColorBrush(Colors.Gray);
128:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
129:             break;
130:         case "YEL":
131:             messageBackGround = new SolidColorBrush(Colors.Yellow);
132:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
133:             break;
134:         case "ORA":
135:             messageBackGround = new SolidColorBrush(Colors.Orange);
136:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
137:             break;
138:         case "PUR":
139:             messageBackGround = new SolidColorBrush(Colors.Purple);
140:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
141:             break;
142:         default:
143:             messageBackGround = new SolidColorBrush(Colors.Black);
144:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
145:             break;
146:     }
147: }
148: }
149: }
150: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.28 NoteOrderSettings.xaml.cs

This is file NoteOrderSettings.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
```

```

4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class NoteOrderSettings : PhoneApplicationPage
18:     {
19:         public int noteFolderId;
20:         public SolidColorBrush messageBackGround;
21:         public NoteOrderSettings()
22:         {
23:             InitializeComponent();
24:             //SetBackgroundColor();
25:
26:             lstNoteOrder.Items.Clear();
27:             lstNoteOrder.Items.Add(AppResources.NoteName);
28:             lstNoteOrder.Items.Add(AppResources.CreationDate);
29:             lstNoteOrder.Items.Add(AppResources.ModificationDate);
30:             lstNoteOrder.SelectedIndex = -1;
31:         }
32:
33:         private void lstNoteOrder_SelectionChanged(object sender,
SelectionChangedEventArgs e)
34:         {
35:             int index = lstNoteOrder.SelectedIndex;
36:             string orderType = "";
37:             if (index == 0)
38:             {
39:                 orderType = "NAME";
40:             }
41:             else if (index == 1)
42:             {
43:                 orderType = "CDATE";
44:             }
45:             else if (index == 2)
46:             {
47:                 orderType = "MDATE";
48:             }
49:             else
50:             {
51:                 orderType = "NAME";
52:             }
53:
54:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
55:             {
56:                 var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
57:                 foreach (var noteFolder in noteFolders)
58:                 {
59:                     noteFolder.NoteOrderBy = orderType;
60:                 }
61:                 context.SubmitChanges();
62:                 MessageBox.Show(AppResources.SuccessfulOrderTypeChanged);
63:             }
64:             NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
65:         }
66:
67:         protected override void OnNavigatedTo(NavigationEventArgs e)
68:         {

```

```

69:         base.OnNavigatedTo(e);
70:     }
71:
72:     protected override void OnNavigatedFrom(NavigationEventArgs e)
73:     {
74:         base.OnNavigatedFrom(e);
75:     }
76:
77:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
78:     {
79:         // displays "Fragment: Detail"
80:         //MessageBox.Show("Folder Id: " + e.Fragment);
81:         base.OnFragmentNavigation(e);
82:         noteFolderId = int.Parse(e.Fragment);
83:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
84:         {
85:             var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
86:             lblFolderName.Text = noteFolders.NoteFolderName;
87:             lblNoteOrder.Text = AppResources.SelectOrderType;
88:         }
89:         SetBackgroundColor();
90:     }
91:
92:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
93:     {
94:         if (this.NavigationService.CanGoBack)
95:         {
96:             this.NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
97:         }
98:     }
99:
100:    private void SetBackgroundColor()
101:    {
102:        AppSettings appSettings = new AppSettings();
103:        NoteFolder noteFolder;
104:        using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
105:        {
106:            appSettings = context.AppSettings.First() as AppSettings;
107:            noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
108:        }
109:
110:        if (noteFolder.FolderBackground != null)
111:        {
112:            MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
113:            BitmapImage image = new BitmapImage();
114:            image.SetSource(stream);
115:            ImageBrush ib = new ImageBrush();
116:            ib.ImageSource = image;
117:            this.LayoutRoot.Background = ib;
118:        }
119:        else
120:        {
121:            switch (appSettings.AppBackgroundColor)
122:            {
123:                case "BLA":
124:                    messageBackGround = new SolidColorBrush(Colors.Black);
125:                    //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
126:                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
127:                    break;
128:                case "BLU":
129:                    messageBackGround = new SolidColorBrush(Colors.Blue);
130:                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);

```

```
131:         break;
132:     case "BRO":
133:         messageBackGround = new SolidColorBrush(Colors.Brown);
134:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
135:         break;
136:     case "RED":
137:         messageBackGround = new SolidColorBrush(Colors.Red);
138:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
139:         break;
140:     case "GRE":
141:         messageBackGround = new SolidColorBrush(Colors.Green);
142:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
143:         break;
144:     case "GRA":
145:         messageBackGround = new SolidColorBrush(Colors.Gray);
146:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
147:         break;
148:     case "YEL":
149:         messageBackGround = new SolidColorBrush(Colors.Yellow);
150:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
151:         break;
152:     case "ORA":
153:         messageBackGround = new SolidColorBrush(Colors.Orange);
154:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
155:         break;
156:     case "PUR":
157:         messageBackGround = new SolidColorBrush(Colors.Purple);
158:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
159:         break;
160:     default:
161:         messageBackGround = new SolidColorBrush(Colors.Black);
162:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
163:         break;
164:     }
165: }
166: }
167: }
168: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.29 NoteOrderStyleSettings.xaml.cs

This is file NoteOrderStyleSettings.xaml.cs.

Body Source

```
1: using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Media;
9: using System.Windows.Media.Imaging;
10: using System.Windows.Navigation;
11: using AwesomeNote.Resources;
12: using Microsoft.Phone.Controls;
```

```

13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeNote
16: {
17:     public partial class NoteOrderStyleSettings : PhoneApplicationPage
18:     {
19:         public int noteFolderId;
20:         public SolidColorBrush messageBackGround;
21:         public NoteOrderStyleSettings()
22:         {
23:             InitializeComponent();
24:             //SetBackgroundColor();
25:
26:             lstNoteOrderStyle.Items.Clear();
27:             lstNoteOrderStyle.Items.Add(AppResources.Ascending);
28:             lstNoteOrderStyle.Items.Add(AppResources.Descending);
29:             lstNoteOrderStyle.SelectedIndex = -1;
30:
31:         }
32:
33:         private void lstNoteOrderStyle_SelectionChanged(object sender,
SelectionChangedEventArgs e)
34:         {
35:             int index = lstNoteOrderStyle.SelectedIndex;
36:             string orderStyle = "";
37:             if (index == 0)
38:             {
39:                 orderStyle = "A";
40:             }
41:             else if (index == 1)
42:             {
43:                 orderStyle = "D";
44:             }
45:             else
46:             {
47:                 orderStyle = "A";
48:             }
49:
50:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
51:             {
52:                 var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
53:                 foreach (var noteFolder in noteFolders)
54:                 {
55:                     noteFolder.NoteOrderStyle = orderStyle;
56:                 }
57:                 context.SubmitChanges();
58:                 MessageBox.Show(AppResources.SuccessfulOrderStyleChanged);
59:             }
60:             NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
61:         }
62:
63:         protected override void OnNavigatedTo(NavigationEventArgs e)
64:         {
65:             base.OnNavigatedTo(e);
66:         }
67:
68:         protected override void OnNavigatedFrom(NavigationEventArgs e)
69:         {
70:             base.OnNavigatedFrom(e);
71:         }
72:
73:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
74:         {
75:             // displays "Fragment: Detail"
76:             //MessageBox.Show("Folder Id: " + e.Fragment);
77:             base.OnFragmentNavigation(e);

```



```

78:         noteFolderId = int.Parse(e.Fragment);
79:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
80:         {
81:             var noteFolders = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
82:             lblFolderName.Text = noteFolders.NoteFolderName;
83:             lblNoteOrderStyle.Text = AppResources.SelectOrderStyle;
84:         }
85:         SetBackgroundColor();
86:     }
87:
88:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
89:     {
90:         if (this.NavigationService.CanGoBack)
91:         {
92:             this.NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
93:         }
94:     }
95:
96:     private void SetBackgroundColor()
97:     {
98:         AppSettings appSettings = new AppSettings();
99:         NoteFolder noteFolder;
100:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
101:         {
102:             appSettings = context.AppSettings.First() as AppSettings;
103:             noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
104:         }
105:
106:         if (noteFolder.FolderBackground != null)
107:         {
108:             MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
109:             BitmapImage image = new BitmapImage();
110:             image.SetSource(stream);
111:             ImageBrush ib = new ImageBrush();
112:             ib.ImageSource = image;
113:             this.LayoutRoot.Background = ib;
114:         }
115:         else
116:         {
117:             switch (appSettings.AppBackgroundColor)
118:             {
119:                 case "BLA":
120:                     messageBackGround = new SolidColorBrush(Colors.Black);
121:                     //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
122:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
123:                     break;
124:                 case "BLU":
125:                     messageBackGround = new SolidColorBrush(Colors.Blue);
126:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
127:                     break;
128:                 case "BRO":
129:                     messageBackGround = new SolidColorBrush(Colors.Brown);
130:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
131:                     break;
132:                 case "RED":
133:                     messageBackGround = new SolidColorBrush(Colors.Red);
134:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
135:                     break;
136:                 case "GRE":
137:                     messageBackGround = new SolidColorBrush(Colors.Green);
138:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
139:                     break;

```

```
140:         case "GRA":
141:             messageBackGround = new SolidColorBrush(Colors.Gray);
142:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
143:             break;
144:         case "YEL":
145:             messageBackGround = new SolidColorBrush(Colors.Yellow);
146:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
147:             break;
148:         case "ORA":
149:             messageBackGround = new SolidColorBrush(Colors.Orange);
150:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
151:             break;
152:         case "PUR":
153:             messageBackGround = new SolidColorBrush(Colors.Purple);
154:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
155:             break;
156:         default:
157:             messageBackGround = new SolidColorBrush(Colors.Black);
158:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
159:             break;
160:     }
161: }
162: }
163: }
164: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.30 PopupAddFolder.xaml.cs

This is file PopupAddFolder.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Media;
8: using System.Windows.Navigation;
9: using Microsoft.Phone.Controls;
10: using Microsoft.Phone.Shell;
11:
12: namespace AwesomeNote
13: {
14:     public partial class PopupAddFolder : UserControl
15:     {
16:         public PopupAddFolder()
17:         {
18:             InitializeComponent();
19:             SetPopupBackgroundColor();
20:         }
21:
22:         private void SetPopupBackgroundColor()
23:         {
24:             AppSettings appSettings = new AppSettings();
25:             using (var context = new
```

```
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
26:         {
27:             appSettings = context.AppSettings.First() as AppSettings;
28:         }
29:
30:         switch (appSettings.AppBackgroundColor)
31:         {
32:             case "BLA":
33:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
34:                 break;
35:             case "BLU":
36:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
37:                 break;
38:             case "BRO":
39:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
40:                 break;
41:             case "RED":
42:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
43:                 break;
44:             case "GRE":
45:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
46:                 break;
47:             case "GRA":
48:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
49:                 break;
50:             case "YEL":
51:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
52:                 break;
53:             case "ORA":
54:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
55:                 break;
56:             case "PUR":
57:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
58:                 break;
59:             default:
60:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
61:                 break;
62:         }
63:     }
64: }
65: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.31 PopupAddNote.xaml.cs

This is file PopupAddNote.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Media;
8: using System.Windows.Navigation;
9: using Microsoft.Phone.Controls;
10: using Microsoft.Phone.Shell;
11:
12: namespace AwesomeNote
```

```
13: {
14:     public partial class PopupAddNote : UserControl
15:     {
16:         public PopupAddNote()
17:         {
18:             InitializeComponent();
19:             SetPopupBackgroundColor();
20:         }
21:
22:         private void SetPopupBackgroundColor()
23:         {
24:             AppSettings appSettings = new AppSettings();
25:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
26:             {
27:                 appSettings = context.AppSettings.First() as AppSettings;
28:             }
29:
30:             switch (appSettings.AppBackgroundColor)
31:             {
32:                 case "BLA":
33:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
34:                     break;
35:                 case "BLU":
36:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
37:                     break;
38:                 case "BRO":
39:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
40:                     break;
41:                 case "RED":
42:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
43:                     break;
44:                 case "GRE":
45:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
46:                     break;
47:                 case "GRA":
48:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
49:                     break;
50:                 case "YEL":
51:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
52:                     break;
53:                 case "ORA":
54:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
55:                     break;
56:                 case "PUR":
57:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
58:                     break;
59:                 default:
60:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
61:                     break;
62:             }
63:         }
64:     }
65: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.32 PopupChangeFolder.xaml.cs

This is file PopupChangeFolder.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Navigation;
8: using Microsoft.Phone.Controls;
9: using Microsoft.Phone.Shell;
10:
11: namespace AwesomeNote
12: {
13:     public partial class PopupChangeFolder : UserControl
14:     {
15:         public PopupChangeFolder()
16:         {
17:             InitializeComponent();
18:         }
19:     }
20: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.33 PopupChangeFolderName.xaml.cs

This is file PopupChangeFolderName.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Media;
8: using System.Windows.Navigation;
9: using Microsoft.Phone.Controls;
10: using Microsoft.Phone.Shell;
11:
12: namespace AwesomeNote
13: {
14:     public partial class PopupChangeFolderName : UserControl
15:     {
16:         public PopupChangeFolderName()
17:         {
18:             InitializeComponent();
19:             SetPopupBackgroundColor();
20:         }
21:
22:         private void SetPopupBackgroundColor()
23:         {
24:             AppSettings appSettings = new AppSettings();
25:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
26:             {
27:                 appSettings = context.AppSettings.First() as AppSettings;
28:             }
29:
30:             switch (appSettings.AppBackgroundColor)
```

```
31:         {
32:             case "BLA":
33:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
34:                 break;
35:             case "BLU":
36:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
37:                 break;
38:             case "BRO":
39:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
40:                 break;
41:             case "RED":
42:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
43:                 break;
44:             case "GRE":
45:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
46:                 break;
47:             case "GRA":
48:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
49:                 break;
50:             case "YEL":
51:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
52:                 break;
53:             case "ORA":
54:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
55:                 break;
56:             case "PUR":
57:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
58:                 break;
59:             default:
60:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
61:                 break;
62:         }
63:     }
64: }
65: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.34 PopupChangeNoteName.xaml.cs

This is file PopupChangeNoteName.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Media;
8: using System.Windows.Navigation;
9: using Microsoft.Phone.Controls;
10: using Microsoft.Phone.Shell;
11:
12: namespace AwesomeNote
13: {
14:     public partial class PopupChangeNoteName : UserControl
15:     {
16:         public PopupChangeNoteName()
17:         {
18:             InitializeComponent();
```

```
19:         SetPopupBackgroundColor();
20:     }
21:
22:     private void SetPopupBackgroundColor()
23:     {
24:         AppSettings appSettings = new AppSettings();
25:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
26:         {
27:             appSettings = context.AppSettings.First() as AppSettings;
28:         }
29:
30:         switch (appSettings.AppBackgroundColor)
31:         {
32:             case "BLA":
33:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
34:                 break;
35:             case "BLU":
36:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
37:                 break;
38:             case "BRO":
39:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
40:                 break;
41:             case "RED":
42:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
43:                 break;
44:             case "GRE":
45:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
46:                 break;
47:             case "GRA":
48:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
49:                 break;
50:             case "YEL":
51:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
52:                 break;
53:             case "ORA":
54:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
55:                 break;
56:             case "PUR":
57:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
58:                 break;
59:             default:
60:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
61:                 break;
62:         }
63:     }
64: }
65: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.35 PopupEnterPassword.xaml.cs

This is file PopupEnterPassword.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
```

```
6: using System.Windows.Controls;
7: using System.Windows.Navigation;
8: using Microsoft.Phone.Controls;
9: using Microsoft.Phone.Shell;
10:
11: namespace AwesomeNote
12: {
13:     public partial class PopupEnterPassword : UserControl
14:     {
15:         public PopupEnterPassword()
16:         {
17:             InitializeComponent();
18:         }
19:     }
20: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.36 PopupSetPassword.xaml.cs

This is file PopupSetPassword.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.Linq;
4: using System.Net;
5: using System.Windows;
6: using System.Windows.Controls;
7: using System.Windows.Navigation;
8: using Microsoft.Phone.Controls;
9: using Microsoft.Phone.Shell;
10:
11: namespace AwesomeNote
12: {
13:     public partial class PopupSetPassword : UserControl
14:     {
15:         public PopupSetPassword()
16:         {
17:             InitializeComponent();
18:         }
19:     }
20: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.37 SearchPage.xaml.cs

This is file SearchPage.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
```



```

3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Threading;
7: using System.Windows;
8: using System.Windows.Controls;
9: using System.Windows.Controls.Primitives;
10: using System.Windows.Input;
11: using System.Windows.Media;
12: using System.Windows.Media.Imaging;
13: using System.Windows.Navigation;
14: using AwesomeNote.Resources;
15: using Microsoft.Phone.Controls;
16: using Microsoft.Phone.Shell;
17:
18: namespace AwesomeNote
19: {
20:     public partial class SearchPage : PhoneApplicationPage
21:     {
22:         public SolidColorBrush messageBackGround;
23:         public SearchPage()
24:         {
25:             InitializeComponent();
26:             SetBackgroundColor();
27:
28:             txtSearchResult.Text = AppResources.SearchResults;
29:             lblSearch.Text = AppResources.Search;
30:             //btnSearch.Content = AppResources.Search;
31:             //lstSearch.SelectedIndex = -1;
32:         }
33:
34:         private void btnSearch_Click(object sender, RoutedEventArgs e)
35:         {
36:             if (txtSearch.Text.TrimStart().Length < 1)
37:             {
38:                 MessageBox.Show(AppResources.SearchTrimFault);
39:             }
40:             else
41:             {
42:                 lstSearch.Items.Clear();
43:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
44:                 {
45:                     var noteList =
46:                         context.Notes.Where(j =>
j.NameDescription.ToLower().Contains(txtSearch.Text.ToLower())).ToList() as List<Note>;
47:                     //var noteList = context.Notes.ToList() as List<Note>;
48:
49:                     if (noteList != null)
50:                     {
51:                         txtSearchResult.Text = AppResources.SearchResults + " (" +
noteList.Count() + ")";
52:                     }
53:
54:                     //lstSearch.ItemsSource = noteList;
55:                     for (int i = 0; i < noteList.Count; i++)
56:                     {
57:                         //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Index
Of(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
58:                         //{
59:                             lstSearch.Items.Add(noteList[i] as Note);
60:                         //}
61:                     }
62:                     //lstSearch.ItemTemplate.
63:                     //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate"
+ ")";
64:                     lstSearch.DisplayMemberPath = "NameCreation";

```

```

65:         MessageBox.Show(AppResources.SearchCompleted);
66:     }
67: }
68: }
69:
70:     private void lstSearch_SelectionChanged(object sender,
SelectionChangedEventArgs e)
71:     {
72:         try
73:         {
74:             if (lstSearch.SelectedIndex != -1)
75:             {
76:                 Note selectedNote = lstSearch.SelectedItem as Note;
77:                 using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
78:                 {
79:                     var noteFolder =
80:                         context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(selectedNote.NoteFolderId)).Single() as
81:                         NoteFolder;
82:                     if (noteFolder.IsPasswordProtected == true)
83:                     {
84:
85:                         // burada parametre olarak notun kendi id bilgisi
gönderiliyor ki
86:                         // ayni sayfadan döndüğünde sorunsuz bir sekilde sorgu
sonucu not dosyasina yönlendirme yapilsin
87:                         NavigationService.Navigate(new
Uri("/EnterPasswordPage.xaml#" + selectedNote.NoteId, UriKind.Relative));
88:
89:                         //Popup popup = new Popup();
90:                         //popup.Height = 300;
91:                         //popup.Width = 400;
92:                         //popup.VerticalOffset = 20;
93:                         //PopupEnterPassword control = new PopupEnterPassword();
94:                         //control.txtLabel.Text = AppResources.PleaseEnterPassword
+ " (" + AppResources.FourDigits + ")";
95:                         //control.btnCancel.Content = AppResources.Cancel;
96:                         //control.btnOK.Content = AppResources.OK;
97:                         //popup.Child = control;
98:                         //popup.IsOpen = true;
99:                         //control.txtPassword.Focus();
100:
101:                         //control.btnOK.Click += (s, args) =>
102:                         //{
103:                             //     string password;
104:                             //     popup.IsOpen = false;
105:                             //     password = control.txtPassword.Text;
106:                             //     // girilen sifre dogruysa
107:                             //     if (password.Length == 4 && password ==
noteFolder.NoteFolderPassword)
108:                             //     {
109:                                 //         MessageBox.Show(AppResources.PasswordTrue);
110:                                 //         NavigationService.Navigate(new
Uri("/NoteDetail.xaml#" + selectedNote.NoteId,
111:                                 //         UriKind.Relative));
112:                             //     }
113:                             //     else
114:                             //     {
115:                                 //         MessageBox.Show(AppResources.PasswordFalse);
116:                                 //         control.txtPassword.Text = "";
117:                                 //         control.txtPassword.Focus();
118:                             //     }
119:                         //};
120:                         //control.btnCancel.Click += (s, args) =>
121:                         //{
122:                             //     popup.IsOpen = false;
123:                         //};
124:                     }

```

```

125:                 else
126:                 {
127:                     NavigationService.Navigate(new Uri("/NoteDetail.xaml#" +
selectedNote.NoteId, UriKind.Relative));
128:                 }
129:             }
130:             lstSearch.SelectedIndex = -1;
131:         }
132:     }
133: }
134: catch (Exception)
135: {
136:     MessageBox.Show(AppResources.SystemFault);
137: }
138: }
139:
140: protected override void OnNavigatedTo(NavigationEventArgs e)
141: {
142:     base.OnNavigatedTo(e);
143: }
144: protected override void OnNavigatedFrom(NavigationEventArgs e)
145: {
146:     base.OnNavigatedFrom(e);
147: }
148:
149: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
150: {
151:     if (this.NavigationService.CanGoBack)
152:     {
153:         this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
154:     }
155: }
156:
157: private void txtSearch_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
158: {
159:     if (e.Key == Key.Enter)
160:     {
161:         if (txtSearch.Text.TrimStart().Length < 1)
162:         {
163:             MessageBox.Show(AppResources.SearchTrimFault);
164:         }
165:         else
166:         {
167:             lstSearch.Items.Clear();
168:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
169:             {
170:                 var noteList =
171:                     context.Notes.Where(j =>
j.NameDescription.ToLower().Contains(txtSearch.Text.ToLower())).ToList() as List<Note>;
172:                 //var noteList = context.Notes.ToList() as List<Note>;
173:
174:                 if (noteList != null)
175:                 {
176:                     txtSearchResult.Text = AppResources.SearchResults + " (" +
noteList.Count() + ")";
177:                 }
178:
179:                 //lstSearch.ItemsSource = noteList;
180:                 for (int i = 0; i < noteList.Count; i++)
181:                 {
182:                     //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Index
Of(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
183:                     //{

```

```

184:             lstSearch.Items.Add(noteList[i] as Note);
185:             //}
186:         }
187:         //lstSearch.ItemTemplate.
188:         //lstSearch.DisplayMemberPath = "NoteName" + " (" +
"CreationDate" + ")";
189:         lstSearch.DisplayMemberPath = "NameCreation";
190:         MessageBox.Show(AppResources.SearchCompleted);
191:         lstSearch.SelectedIndex = -1;
192:     }
193: }
194: }
195: }
196:
197: private void SetBackgroundColor()
198: {
199:     AppSettings appSettings = new AppSettings();
200:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
201:     {
202:         appSettings = context.AppSettings.First() as AppSettings;
203:     }
204:
205:     if (appSettings.AppBackgroundImage != null)
206:     {
207:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
208:         BitmapImage image = new BitmapImage();
209:         image.SetSource(stream);
210:         ImageBrush ib = new ImageBrush();
211:         ib.ImageSource = image;
212:         this.LayoutRoot.Background = ib;
213:     }
214:     else
215:     {
216:         switch (appSettings.AppBackgroundColor)
217:         {
218:             case "BLA":
219:                 messageBackGround = new SolidColorBrush(Colors.Black);
220:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
221:                 break;
222:             case "BLU":
223:                 messageBackGround = new SolidColorBrush(Colors.Blue);
224:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
225:                 break;
226:             case "BRO":
227:                 messageBackGround = new SolidColorBrush(Colors.Brown);
228:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
229:                 break;
230:             case "RED":
231:                 messageBackGround = new SolidColorBrush(Colors.Red);
232:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
233:                 break;
234:             case "GRE":
235:                 messageBackGround = new SolidColorBrush(Colors.Green);
236:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
237:                 break;
238:             case "GRA":
239:                 messageBackGround = new SolidColorBrush(Colors.Gray);
240:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
241:                 break;
242:             case "YEL":
243:                 messageBackGround = new SolidColorBrush(Colors.Yellow);
244:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
245:                 break;
246:             case "ORA":
247:                 messageBackGround = new SolidColorBrush(Colors.Orange);
248:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);

```

```
249:         break;
250:         case "PUR":
251:             messageBackGround = new SolidColorBrush(Colors.Purple);
252:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
253:             break;
254:         default:
255:             messageBackGround = new SolidColorBrush(Colors.Black);
256:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
257:             break;
258:     }
259: }
260: }
261:
262: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
263: {
264:     txtSearch.Focus();
265: }
266: }
267: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

1.2.38 SetPasswordPage.xaml.cs

This is file SetPasswordPage.xaml.cs.

Body Source

```
1: ?using System;
2: using System.Collections.Generic;
3: using System.IO;
4: using System.Linq;
5: using System.Net;
6: using System.Windows;
7: using System.Windows.Controls;
8: using System.Windows.Input;
9: using System.Windows.Media;
10: using System.Windows.Media.Imaging;
11: using System.Windows.Navigation;
12: using AwesomeNote.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16: namespace AwesomeNote
17: {
18:     public partial class SetPasswordPage : PhoneApplicationPage
19:     {
20:         public int noteFolderId;
21:         public SolidColorBrush messageBackGround;
22:
23:         public SetPasswordPage()
24:         {
25:             InitializeComponent();
26:             //SetBackgroundColor();
27:
28:             //txtFirstPassword.Focus();
29:         }
30:
31:
32:         protected override void OnNavigatedTo(NavigationEventArgs e)
33:         {
```

```

34:         base.OnNavigatedTo(e);
35:         txtFirstPassword.Focus();
36:     }
37:
38:     protected override void OnNavigatedFrom(NavigationEventArgs e)
39:     {
40:         base.OnNavigatedFrom(e);
41:     }
42:
43:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
44:     {
45:         // displays "Fragment: Detail"
46:         //MessageBox.Show("Folder Id: " + e.Fragment);
47:         base.OnFragmentNavigation(e);
48:         noteFolderId = int.Parse(e.Fragment);
49:         using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
50:         {
51:             var noteFolders =
52:                 context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
53:             lblFolderName.Text = noteFolders.NoteFolderName;
54:             lblSetPassword.Text = AppResources.SetPassword;
55:             txtPass1.Text = AppResources.PleaseEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
56:             txtPass2.Text = AppResources.PleaseReEnterPassword + " (" +
AppResources.AtLeastFourDigits + ")";
57:             //btnCancel.Content = AppResources.Cancel;
58:             //btnOK.Content = AppResources.OK;
59:         }
60:         SetBackgroundColor();
61:     }
62:
63:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
64:     {
65:         if (this.NavigationService.CanGoBack)
66:         {
67:             this.NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
68:         }
69:     }
70:
71:     private void btnCancel_Click(object sender, RoutedEventArgs e)
72:     {
73:         NavigationService.Navigate(new Uri("/FolderSettings.xaml#" + noteFolderId,
UriKind.Relative));
74:     }
75:
76:     private void btnOK_Click(object sender, RoutedEventArgs e)
77:     {
78:         if ((txtFirstPassword.Password == txtSecondPassword.Password) &&
79:             (txtFirstPassword.Password.Length >= 4))
80:         {
81:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
82:             {
83:                 var noteFolders =
84:                     context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
85:                 foreach (var noteFolder in noteFolders)
86:                 {
87:                     noteFolder.IsPasswordProtected = true;
88:                     noteFolder.NoteFolderPassword = txtFirstPassword.Password;
89:                     noteFolder.ModificationDate = DateTime.Now;
90:                 }
91:                 //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
92:                 //    .Single()

```

```

93:         // .NoteFolderPassword = txtFirstPassword.Text;
94:         //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
95:         // .Single()
96:         // .IsPasswordProtected = true;
97:         context.SubmitChanges();
98:         MessageBox.Show(AppResources.SuccessfulSetPassword);
99:         NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
100:     }
101: }
102:     // sifreler ayni degilse veya 4 karakter degilse
103:     else
104:     {
105:         MessageBox.Show(AppResources.TwoFieldsNotSame);
106:         txtFirstPassword.Password = "";
107:         txtSecondPassword.Password = "";
108:         txtFirstPassword.Focus();
109:     }
110: }
111:
112: private void txtFirstPassword_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
113: {
114:     if (e.Key == Key.Enter)
115:         txtSecondPassword.Focus();
116: }
117:
118: private void txtFirstPassword_GotFocus(object sender, RoutedEventArgs e)
119: {
120:     this.svPassword.ScrollToVerticalOffset(this.txtFirstPassword.ActualHeight);
121:     this.svPassword.UpdateLayout();
122: }
123:
124: private void txtFirstPassword_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
125: {
126:     svPassword.ScrollToVerticalOffset(e.GetPosition(txtFirstPassword).Y - 120);
127: }
128:
129: private void txtSecondPassword_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
130: {
131:     svPassword.ScrollToVerticalOffset(e.GetPosition(txtSecondPassword).Y -
120);
132: }
133:
134: private void txtSecondPassword_GotFocus(object sender, RoutedEventArgs e)
135: {
136:     this.svPassword.ScrollToVerticalOffset(this.txtSecondPassword.ActualHeight);
137:     this.svPassword.UpdateLayout();
138: }
139:
140: private void txtSecondPassword_KeyDown(object sender, KeyEventArgs e)
141: {
142:     if (e.Key == Key.Enter)
143:     {
144:         if ((txtFirstPassword.Password == txtSecondPassword.Password) &&
145:             (txtFirstPassword.Password.Length >= 4))
146:         {
147:             using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
148:             {
149:                 var noteFolders =
150:                     context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Select(j => j);
151:                 foreach (var noteFolder in noteFolders)
152:                 {

```

```

153:             noteFolder.IsPasswordProtected = true;
154:             noteFolder.NoteFolderPassword = txtFirstPassword.Password;
155:             noteFolder.ModificationDate = DateTime.Now;
156:         }
157:         //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
158:         //    .Single()
159:         //    .NoteFolderPassword = txtFirstPassword.Text;
160:         //context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId))
161:         //    .Single()
162:         //    .IsPasswordProtected = true;
163:         context.SubmitChanges();
164:         MessageBox.Show(AppResources.SuccessfulSetPassword);
165:         NavigationService.Navigate(new Uri("/FolderSettings.xaml#" +
noteFolderId, UriKind.Relative));
166:     }
167: }
168: // sifreler ayni degilse veya 4 karakter degilse
169: else
170: {
171:     MessageBox.Show(AppResources.TwoFieldsNotSame);
172:     txtFirstPassword.Password = "";
173:     txtSecondPassword.Password = "";
174:     txtFirstPassword.Focus();
175: }
176: }
177: }
178:
179: private void SetBackgroundColor()
180: {
181:     AppSettings appSettings = new AppSettings();
182:     NoteFolder noteFolder;
183:     using (var context = new
NoteFolderDataContext(NoteFolderDataContext.ConnectionString))
184:     {
185:         appSettings = context.AppSettings.First() as AppSettings;
186:         noteFolder = context.NoteFolders.Where(j =>
j.NoteFolderId.Equals(noteFolderId)).Single() as NoteFolder;
187:     }
188:
189:     if (noteFolder.FolderBackground != null)
190:     {
191:         MemoryStream stream = new MemoryStream(noteFolder.FolderBackground);
192:         BitmapImage image = new BitmapImage();
193:         image.SetSource(stream);
194:         ImageBrush ib = new ImageBrush();
195:         ib.ImageSource = image;
196:         this.LayoutRoot.Background = ib;
197:     }
198:     else
199:     {
200:         switch (appSettings.AppBackgroundColor)
201:         {
202:             case "BLA":
203:                 messageBackGround = new SolidColorBrush(Colors.Black);
204:                 //this.LayoutRoot.Background = new ImageBrush(new
BitmapImage());
205:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
206:                 break;
207:             case "BLU":
208:                 messageBackGround = new SolidColorBrush(Colors.Blue);
209:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
210:                 break;
211:             case "BRO":
212:                 messageBackGround = new SolidColorBrush(Colors.Brown);
213:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
214:                 break;
215:             case "RED":

```



```
216:         messageBackGround = new SolidColorBrush(Colors.Red);
217:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
218:         break;
219:     case "GRE":
220:         messageBackGround = new SolidColorBrush(Colors.Green);
221:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
222:         break;
223:     case "GRA":
224:         messageBackGround = new SolidColorBrush(Colors.Gray);
225:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
226:         break;
227:     case "YEL":
228:         messageBackGround = new SolidColorBrush(Colors.Yellow);
229:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
230:         break;
231:     case "ORA":
232:         messageBackGround = new SolidColorBrush(Colors.Orange);
233:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
234:         break;
235:     case "PUR":
236:         messageBackGround = new SolidColorBrush(Colors.Purple);
237:         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
238:         break;
239:     default:
240:         messageBackGround = new SolidColorBrush(Colors.Black);
241:         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
242:         break;
243:     }
244: }
245: }
246: }
247: }
```

Namespaces

Name	Description
AwesomeNote (see page 1)	This is namespace AwesomeNote.

Index

A

AboutPage class 30

 about AboutPage class 30

 AboutPage 31

 messageBackGround 32

AboutPage.xaml.cs 95

App class 32

 about App class 32

 App 32

 IsTrial 33

 RootFrame 33

App.xaml.cs 98

AppResources class 2

 About 9

 about AppResources class 2

 AboutTheApp 9

 AboutTheAppText 9

 AboutTheAwesomeNote 9

 Active 9

 AddFolder 9

 AddItem 10

 AddNote 10

 AppFolder 10

 ApplicationTitle 10

 AppResources 9

 AppTitle 10

 Arabic 10

 Ascending 10

 AtLeastFourDigits 10

 Background 10

 BackgroundColor 11

 BackgroundFeature 11

 BackgroundFeatureNote 11

 BackgroundImage 11

 BelongFolderName 11

 Black 11

 Blue 11

 Brown 11

 Cancel 12

 ChangeFolder 12

 CharacterNumber 12

 Chinese 12

 ContactWithUs 12

 CreationDate 12

 Culture 12

 DeleteFolder 12

 DeleteFolderQuestion 12

 DeleteNote 13

 DeleteNoteQuestion 13

 Descending 13

 English 13

 EnterFolderName 13

 EnterNoteName 13

 EnterPassword 13

 ExitApp 13

 ExitAppQuestion 13

 FolderExists 14

 FolderName 14

 FolderOrder 14

 FolderSettings 14

 Font 14

 FontFamily 14

 FontFeature 14

 FontFeatureNote 14

 FontSize 15

 FourDigits 15

 French 15

 GeneralSettings 15

 German 15

 Gray 15

 Green 15

 Irish 15

 IsBold 15

 IsItalic 16

 Italian 16

 Language 16

 LanguageFeature 16

 LanguageFeatureNote 16

 LanguageWarning 16

Login 16	Rate 21
ModificationDate 16	Red 21
Next 16	RemoveBackgroundImage 22
NextNote 17	ResetSettings 22
No 17	ResourceFlowDirection 22
Note 17	ResourceLanguage 22
NoteExists 17	ResourceManager 22
NoteFeature 17	Russian 22
NoteFeatureNote 17	Sanskrit 22
NoteFolderFeature 17	Save 22
NoteFolderFeatureNote 17	Search 22
NoteInfo 18	SearchCompleted 23
NoteListOf 18	SearchFeature 23
NoteName 18	SearchFeatureNote 23
NoteOrder 18	SearchResults 23
NoteSaved 18	SearchTrimFault 23
OK 18	Select 23
OneDrive 18	SelectAll 23
OneDriveSyncCompleted 18	SelectBackgroundColor 23
Orange 19	Selected 24
OrderFeature 19	SelectFontFamily 24
OrderFeatureNote 19	SelectFontSize 24
OrderStyle 19	SelectLanguage 24
OtherSettings 19	SelectNoteFolder 24
Passive 19	SelectOrderStyle 24
Password 19	SelectOrderType 24
PasswordFalse 19	SelectPasswordState 24
PasswordFeature 20	SendWithApp 24
PasswordFeatureNote 20	SendWithMail 25
PasswordProtection 20	SendWithSMS 25
PasswordState 20	SetPassword 25
PasswordTrue 20	Settings 25
Persian 20	Share 25
PleaseEnterPassword 20	SignIn 25
PleaseReEnterPassword 20	SignOut 25
Portuguese 21	Spanish 25
Previous 21	SuccessfulAddNote 25
PreviousNote 21	SuccessfulAddNoteFolder 26
ProgramFeatures 21	SuccessfulBackgroundColorChanged 26
ProgramFeaturesNote 21	SuccessfulBackgroundImageChanged 26
Purple 21	SuccessfulBackgroundImageRemoved 26

SuccessfulChangeFolderName 26	about AutoScrollBehavior class 35
SuccessfulChangeNoteFolder 26	OnAttached 35
SuccessfulChangeNoteName 26	OnDetaching 36
SuccessfulDeleteFolder 26	AutoScrollBehavior.cs 134
SuccessfulDeleteNote 26	Awesome Note.csproj 135
SuccessfulFontFamilyChanged 27	AwesomeNote 1
SuccessfulFontSizeChanged 27	AwesomeNote namespace 1
SuccessfulOrderStyleChanged 27	Classes 29
SuccessfulOrderTypeChanged 27	AwesomeNote.AboutPage 30
SuccessfulRemovePassword 27	AwesomeNote.AboutPage.AboutPage 31
SuccessfulResetSettings 27	AwesomeNote.AboutPage.messageBackGround 32
SuccessfulSendWithMail 27	AwesomeNote.App 32
SuccessfulSendWithSMS 27	AwesomeNote.App.App 32
SuccessfulSetPassword 27	AwesomeNote.App.IsTrial 33
Sync 28	AwesomeNote.App.RootFrame 33
SyncFeature 28	AwesomeNote.App.Settings 33
SyncFeatureNote 28	AwesomeNote.App.Settings.App.BackgroundColor 34
SyncOnOneFile 28	AwesomeNote.App.Settings.App.BackgroundImage 34
Synchronizing 28	AwesomeNote.App.Settings.App.LangName 34
SystemFault 28	AwesomeNote.App.Settings.App.SettingsId 34
Thai 28	AwesomeNote.App.Settings.FolderOrderBy 35
Tips 28	AwesomeNote.App.Settings.FolderOrderStyle 35
Turkish 29	AwesomeNote.AutoScrollBehavior 35
TwoFieldsNotSame 29	AwesomeNote.AutoScrollBehavior.OnAttached 35
Welcome 29	AwesomeNote.AutoScrollBehavior.OnDetaching 36
WelcomeNote 29	AwesomeNote.BackgroundColorSettings 36
WordNumber 29	AwesomeNote.BackgroundColorSettings.BackgroundColorSettings 36
Yellow 29	AwesomeNote.BackgroundColorSettings.messageBackGround 37
Yes 29	AwesomeNote.ChangeNoteFolder 37
AppResources.Designer.cs 107	AwesomeNote.ChangeNoteFolder.ChangeNoteFolder 38
AppSettings class 33	AwesomeNote.ChangeNoteFolder.messageBackGround 38
about AppSettings class 33	AwesomeNote.ChangeNoteFolder.noteFolderId 38
AppBackgroundColor 34	AwesomeNote.ChangeNoteFolder.noteId 38
AppBackgroundImage 34	AwesomeNote.ChangeNoteFolder.OnFragmentNavigation 39
AppLangName 34	AwesomeNote.ChangeNoteFolder.OnNavigatedFrom 39
AppSettingsId 34	AwesomeNote.ChangeNoteFolder.OnNavigatedTo 39
FolderOrderBy 35	AwesomeNote.EnterPasswordPage 40
FolderOrderStyle 35	AwesomeNote.EnterPasswordPage.EnterPasswordPage 40
AppSettings.cs 133	
AssemblyInfo.cs 133	
AutoScrollBehavior class 35	

AwesomeNote.EnterPasswordPage.messageBackGround	41	AwesomeNote.FontSizeSettings.OnFragmentNavigation	53
AwesomeNote.EnterPasswordPage.note	41	AwesomeNote.FontSizeSettings.OnNavigatedFrom	54
AwesomeNote.EnterPasswordPage.noteFolder	41	AwesomeNote.FontSizeSettings.OnNavigatedTo	54
AwesomeNote.EnterPasswordPage.noteFolderId	41	AwesomeNote.GeneralSettings	54
AwesomeNote.EnterPasswordPage.noteId	41	AwesomeNote.GeneralSettings.CreateDirectoryAsync	58
AwesomeNote.EnterPasswordPage.OnFragmentNavigation	42	AwesomeNote.GeneralSettings.DesignFileName	58
AwesomeNote.EnterPasswordPage.OnNavigatedFrom	42	AwesomeNote.GeneralSettings.GeneralSettings	55
AwesomeNote.EnterPasswordPage.OnNavigatedTo	43	AwesomeNote.GeneralSettings.messageBackGround	57
AwesomeNote.EnterPasswordPage.pageName	41	AwesomeNote.GeneralSettings.OnNavigatedFrom	59
AwesomeNote.FolderOrderSettings	43	AwesomeNote.GeneralSettings.OnNavigatedTo	59
AwesomeNote.FolderOrderSettings.FolderOrderSettings	44	AwesomeNote.GeneralSettings.signIn	57
AwesomeNote.FolderOrderSettings.messageBackGround	44	AwesomeNote.Language	59
AwesomeNote.FolderOrderSettings.OnNavigatedFrom	44	AwesomeNote.Language.LanguageId	60
AwesomeNote.FolderOrderSettings.OnNavigatedTo	44	AwesomeNote.Language.LanguageName	60
AwesomeNote.FolderOrderStyleSettings	45	AwesomeNote.LanguageSettings	60
AwesomeNote.FolderOrderStyleSettings.FolderOrderStyleSettings	45	AwesomeNote.LanguageSettings.LanguageSettings	60
AwesomeNote.FolderOrderStyleSettings.messageBackGround	46	AwesomeNote.LanguageSettings.messageBackGround	61
AwesomeNote.FolderOrderStyleSettings.OnNavigatedFrom	46	AwesomeNote.LanguageSettings.OnNavigatedFrom	61
AwesomeNote.FolderOrderStyleSettings.OnNavigatedTo	46	AwesomeNote.LanguageSettings.OnNavigatedTo	61
AwesomeNote.FolderSettings	46	AwesomeNote.LocalizedStrings	62
AwesomeNote.FolderSettings.FolderSettings	47	AwesomeNote.LocalizedStrings.LocalizedResources	62
AwesomeNote.FolderSettings.messageBackGround	47	AwesomeNote.MainPage	62
AwesomeNote.FolderSettings.noteFolderId	47	AwesomeNote.MainPage.MainPage	63
AwesomeNote.FolderSettings.OnFragmentNavigation	48	AwesomeNote.MainPage.messageBackGround	64
AwesomeNote.FolderSettings.OnNavigatedFrom	48	AwesomeNote.MainPage.OnNavigatedFrom	64
AwesomeNote.FolderSettings.OnNavigatedTo	49	AwesomeNote.MainPage.OnNavigatedTo	64
AwesomeNote.FontFamilySettings	49	AwesomeNote.MainPage.popup	64
AwesomeNote.FontFamilySettings.FontFamilySettings	50	AwesomeNote.Note	64
AwesomeNote.FontFamilySettings.messageBackGround	50	AwesomeNote.Note.CreationDate	65
AwesomeNote.FontFamilySettings.noteFolderId	50	AwesomeNote.Note.ModificationDate	65
AwesomeNote.FontFamilySettings.OnFragmentNavigation	51	AwesomeNote.Note.NameCreation	65
AwesomeNote.FontFamilySettings.OnNavigatedFrom	51	AwesomeNote.Note.NameDescription	66
AwesomeNote.FontFamilySettings.OnNavigatedTo	51	AwesomeNote.Note.NoteDescription	66
AwesomeNote.FontSizeSettings	52	AwesomeNote.Note.NoteFolderId	66
AwesomeNote.FontSizeSettings.FontSizeSettings	52	AwesomeNote.Note.NoteId	66
AwesomeNote.FontSizeSettings.messageBackGround	53	AwesomeNote.Note.NoteName	66
AwesomeNote.FontSizeSettings.noteFolderId	53	AwesomeNote.NoteDetail	66
		AwesomeNote.NoteDetail.flag	68
		AwesomeNote.NoteDetail.messageBackGround	68
		AwesomeNote.NoteDetail.NoteDetail	67
		AwesomeNote.NoteDetail.noteFolderId	68

AwesomeNote.NoteDetail.noteId 69	AwesomeNote.NoteOrderSettings.OnFragmentNavigation 80
AwesomeNote.NoteDetail.OnFragmentNavigation 69	AwesomeNote.NoteOrderSettings.OnNavigatedFrom 80
AwesomeNote.NoteDetail.OnNavigatedFrom 70	AwesomeNote.NoteOrderSettings.OnNavigatedTo 81
AwesomeNote.NoteDetail.OnNavigatedTo 70	AwesomeNote.NoteOrderStyleSettings 81
AwesomeNote.NoteDetail.pageName 69	AwesomeNote.NoteOrderStyleSettings.messageBackGround 82
AwesomeNote.NoteFolder 70	AwesomeNote.NoteOrderStyleSettings.noteFolderId 82
AwesomeNote.NoteFolder.CreationDate 71	AwesomeNote.NoteOrderStyleSettings.NoteOrderStyleSettings 81
AwesomeNote.NoteFolder.FolderBackground 71	AwesomeNote.NoteOrderStyleSettings.OnFragmentNavigation 82
AwesomeNote.NoteFolder.FontFamily 71	AwesomeNote.NoteOrderStyleSettings.OnNavigatedFrom 83
AwesomeNote.NoteFolder.FontSize 71	AwesomeNote.NoteOrderStyleSettings.OnNavigatedTo 83
AwesomeNote.NoteFolder.IsPasswordProtected 72	AwesomeNote.Page1 83
AwesomeNote.NoteFolder.ModificationDate 72	AwesomeNote.Page1.messageBackGround 84
AwesomeNote.NoteFolder.NameCount 72	AwesomeNote.Page1.noteFolderId 84
AwesomeNote.NoteFolder.NoteCount 72	AwesomeNote.Page1.noteId 84
AwesomeNote.NoteFolder.NoteFolderDescription 72	AwesomeNote.Page1.OnFragmentNavigation 85
AwesomeNote.NoteFolder.NoteFolderId 72	AwesomeNote.Page1.OnNavigatedFrom 85
AwesomeNote.NoteFolder.NoteFolderName 73	AwesomeNote.Page1.OnNavigatedTo 86
AwesomeNote.NoteFolder.NoteFolderPassword 73	AwesomeNote.Page1.Page1 84
AwesomeNote.NoteFolder.NoteOrderBy 73	AwesomeNote.PopupAddFolder 86
AwesomeNote.NoteFolder.NoteOrderStyle 73	AwesomeNote.PopupAddFolder.PopupAddFolder 86
AwesomeNote.NoteFolderDataContext 73	AwesomeNote.PopupAddNote 86
AwesomeNote.NoteFolderDataContext.AppSettings 74	AwesomeNote.PopupAddNote.PopupAddNote 87
AwesomeNote.NoteFolderDataContext.ConnectionString 74	AwesomeNote.PopupChangeFolder 87
AwesomeNote.NoteFolderDataContext.NoteFolderDataContext 74	AwesomeNote.PopupChangeFolder.PopupChangeFolder 87
AwesomeNote.NoteFolderDataContext.NoteFolders 74	AwesomeNote.PopupChangeFolderName 88
AwesomeNote.NoteFolderDataContext.Notes 75	AwesomeNote.PopupChangeFolderName.PopupChangeFolderName 88
AwesomeNote.NoteList 75	AwesomeNote.PopupChangeNoteName 88
AwesomeNote.NoteList.messageBackGround 76	AwesomeNote.PopupChangeNoteName.PopupChangeNoteName 89
AwesomeNote.NoteList.noteFolderId 76	AwesomeNote.PopupEnterPassword 89
AwesomeNote.NoteList.noteId 76	AwesomeNote.PopupEnterPassword.PopupEnterPassword 89
AwesomeNote.NoteList.NoteList 75	AwesomeNote.PopupSetPassword 90
AwesomeNote.NoteList.OnFragmentNavigation 77	AwesomeNote.PopupSetPassword.PopupSetPassword 90
AwesomeNote.NoteList.OnNavigatedFrom 78	AwesomeNote.Resources 2
AwesomeNote.NoteList.OnNavigatedTo 78	AwesomeNote.Resources namespace 2
AwesomeNote.NoteList.popup 76	
AwesomeNote.NoteOrderSettings 79	
AwesomeNote.NoteOrderSettings.messageBackGround 80	
AwesomeNote.NoteOrderSettings.noteFolderId 80	
AwesomeNote.NoteOrderSettings.NoteOrderSettings 79	

Classes 2

AwesomeNote.Resources.AppResources 2	AwesomeNote.Resources.AppResources.DeleteFolderQuestion 12
AwesomeNote.Resources.AppResources.About 9	AwesomeNote.Resources.AppResources.DeleteNote 13
AwesomeNote.Resources.AppResources.AboutTheApp 9	AwesomeNote.Resources.AppResources.DeleteNoteQuestion 13
AwesomeNote.Resources.AppResources.AboutTheAppText 9	AwesomeNote.Resources.AppResources.Descending 13
AwesomeNote.Resources.AppResources.AboutTheAwesomeNote 9	AwesomeNote.Resources.AppResources.English 13
AwesomeNote.Resources.AppResources.Active 9	AwesomeNote.Resources.AppResources.EnterFolderName 13
AwesomeNote.Resources.AppResources.AddFolder 9	AwesomeNote.Resources.AppResources.EnterNoteName 13
AwesomeNote.Resources.AppResources.AddItem 10	AwesomeNote.Resources.AppResources.EnterPassword 13
AwesomeNote.Resources.AppResources.AddNote 10	AwesomeNote.Resources.AppResources.ExitApp 13
AwesomeNote.Resources.AppResources.AppFolder 10	AwesomeNote.Resources.AppResources.ExitAppQuestion 13
AwesomeNote.Resources.AppResources.ApplicationTitle 10	AwesomeNote.Resources.AppResources.FolderExists 14
AwesomeNote.Resources.AppResources.AppResources 9	AwesomeNote.Resources.AppResources.FolderName 14
AwesomeNote.Resources.AppResources.AppTitle 10	AwesomeNote.Resources.AppResources.FolderOrder 14
AwesomeNote.Resources.AppResources.Arabic 10	AwesomeNote.Resources.AppResources.FolderSettings 14
AwesomeNote.Resources.AppResources.Ascending 10	AwesomeNote.Resources.AppResources.Font 14
AwesomeNote.Resources.AppResources.AtLeastFourDigits 10	AwesomeNote.Resources.AppResources.FontFamily 14
AwesomeNote.Resources.AppResources.Background 10	AwesomeNote.Resources.AppResources.FontFeature 14
AwesomeNote.Resources.AppResources.BackgroundColor 11	AwesomeNote.Resources.AppResources.FontFeatureNote 14
AwesomeNote.Resources.AppResources.BackgroundFeature 11	AwesomeNote.Resources.AppResources.FontSize 15
AwesomeNote.Resources.AppResources.BackgroundFeatureNote 11	AwesomeNote.Resources.AppResources.FourDigits 15
AwesomeNote.Resources.AppResources.BackgroundImage 11	AwesomeNote.Resources.AppResources.French 15
AwesomeNote.Resources.AppResources.BelongFolderName 11	AwesomeNote.Resources.AppResources.GeneralSettings 15
AwesomeNote.Resources.AppResources.Black 11	AwesomeNote.Resources.AppResources.German 15
AwesomeNote.Resources.AppResources.Blue 11	AwesomeNote.Resources.AppResources.Gray 15
AwesomeNote.Resources.AppResources.Brown 11	AwesomeNote.Resources.AppResources.Green 15
AwesomeNote.Resources.AppResources.Cancel 12	AwesomeNote.Resources.AppResources.Irish 15
AwesomeNote.Resources.AppResources.ChangeFolder 12	AwesomeNote.Resources.AppResources.IsBold 15
AwesomeNote.Resources.AppResources.CharacterNumber 12	AwesomeNote.Resources.AppResources.IsItalic 16
AwesomeNote.Resources.AppResources.Chinese 12	AwesomeNote.Resources.AppResources.Italian 16
AwesomeNote.Resources.AppResources.ContactWithUs 12	AwesomeNote.Resources.AppResources.Language 16
AwesomeNote.Resources.AppResources.CreationDate 12	AwesomeNote.Resources.AppResources.LanguageFeature 16
AwesomeNote.Resources.AppResources.Culture 12	AwesomeNote.Resources.AppResources.LanguageFeatureNote 16
AwesomeNote.Resources.AppResources.DeleteFolder 12	AwesomeNote.Resources.AppResources.LanguageWarning 16
	AwesomeNote.Resources.AppResources.Login 16
	AwesomeNote.Resources.AppResources.ModificationDate 16

AwesomeNote.Resources.AppResources.Next 16	AwesomeNote.Resources.AppResources.PleaseReEnterPass word 20
AwesomeNote.Resources.AppResources.NextNote 17	
AwesomeNote.Resources.AppResources.No 17	AwesomeNote.Resources.AppResources.Portuguese 21
AwesomeNote.Resources.AppResources.Note 17	AwesomeNote.Resources.AppResources.Previous 21
AwesomeNote.Resources.AppResources.NoteExists 17	AwesomeNote.Resources.AppResources.PreviousNote 21
AwesomeNote.Resources.AppResources.NoteFeature 17	AwesomeNote.Resources.AppResources.ProgramFeatures 21
AwesomeNote.Resources.AppResources.NoteFeatureNote 17	AwesomeNote.Resources.AppResources.ProgramFeaturesN ote 21
AwesomeNote.Resources.AppResources.NoteFolderFeature 17	
AwesomeNote.Resources.AppResources.NoteFolderFeature Note 17	AwesomeNote.Resources.AppResources.Purple 21
	AwesomeNote.Resources.AppResources.Rate 21
AwesomeNote.Resources.AppResources.NoteInfo 18	AwesomeNote.Resources.AppResources.Red 21
AwesomeNote.Resources.AppResources.NoteListOf 18	AwesomeNote.Resources.AppResources.RemoveBackgroun dImage 22
AwesomeNote.Resources.AppResources.NoteName 18	
AwesomeNote.Resources.AppResources.NoteOrder 18	AwesomeNote.Resources.AppResources.ResetSettings 22
AwesomeNote.Resources.AppResources.NoteSaved 18	AwesomeNote.Resources.AppResources.ResourceFlowDirect ion 22
AwesomeNote.Resources.AppResources.OK 18	
AwesomeNote.Resources.AppResources.OneDrive 18	AwesomeNote.Resources.AppResources.ResourceLanguage 22
AwesomeNote.Resources.AppResources.OneDriveSyncCom pleted 18	AwesomeNote.Resources.AppResources.ResourceManager 22
	AwesomeNote.Resources.AppResources.Russian 22
AwesomeNote.Resources.AppResources.Orange 19	AwesomeNote.Resources.AppResources.Sanskrit 22
AwesomeNote.Resources.AppResources.OrderFeature 19	AwesomeNote.Resources.AppResources.Save 22
AwesomeNote.Resources.AppResources.OrderFeatureNote 19	AwesomeNote.Resources.AppResources.Search 22
AwesomeNote.Resources.AppResources.OrderStyle 19	AwesomeNote.Resources.AppResources.SearchCompleted 23
AwesomeNote.Resources.AppResources.OtherSettings 19	AwesomeNote.Resources.AppResources.SearchFeature 23
AwesomeNote.Resources.AppResources.Passive 19	AwesomeNote.Resources.AppResources.SearchFeatureNote 23
AwesomeNote.Resources.AppResources.Password 19	AwesomeNote.Resources.AppResources.SearchResults 23
AwesomeNote.Resources.AppResources.PasswordFalse 19	AwesomeNote.Resources.AppResources.SearchTrimFault 23
AwesomeNote.Resources.AppResources.PasswordFeature 20	AwesomeNote.Resources.AppResources.Select 23
AwesomeNote.Resources.AppResources.PasswordFeatureN ote 20	AwesomeNote.Resources.AppResources.SelectAll 23
AwesomeNote.Resources.AppResources.PasswordProtection 20	AwesomeNote.Resources.AppResources.SelectBackgroundC olor 23
	AwesomeNote.Resources.AppResources.Selected 24
AwesomeNote.Resources.AppResources.PasswordState 20	AwesomeNote.Resources.AppResources.SelectFontFamily 24
AwesomeNote.Resources.AppResources.PasswordTrue 20	AwesomeNote.Resources.AppResources.SelectFontSize 24
AwesomeNote.Resources.AppResources.Persian 20	AwesomeNote.Resources.AppResources.SelectLanguage 24
AwesomeNote.Resources.AppResources.PleaseEnterPasswo rd 20	

AwesomeNote.Resources.AppResources.SelectNoteFolder 24	27	AwesomeNote.Resources.AppResources.SuccessfulOrderSty leChanged 27
AwesomeNote.Resources.AppResources.SelectOrderStyle 24		
AwesomeNote.Resources.AppResources.SelectOrderType 24		AwesomeNote.Resources.AppResources.SuccessfulOrderTy peChanged 27
AwesomeNote.Resources.AppResources.SelectPasswordStat e 24		
AwesomeNote.Resources.AppResources.SendWithApp 24		AwesomeNote.Resources.AppResources.SuccessfulRemove Password 27
AwesomeNote.Resources.AppResources.SendWithMail 25		
AwesomeNote.Resources.AppResources.SendWithSMS 25		AwesomeNote.Resources.AppResources.SuccessfulResetSet tings 27
AwesomeNote.Resources.AppResources.SetPassword 25		
AwesomeNote.Resources.AppResources.Settings 25		AwesomeNote.Resources.AppResources.SuccessfulSendWit hMail 27
AwesomeNote.Resources.AppResources.Share 25		
AwesomeNote.Resources.AppResources.SignIn 25		AwesomeNote.Resources.AppResources.SuccessfulSendWit hSMS 27
AwesomeNote.Resources.AppResources.SignOut 25		
AwesomeNote.Resources.AppResources.Spanish 25		
AwesomeNote.Resources.AppResources.SuccessfulAddNote 25		AwesomeNote.Resources.AppResources.SuccessfulSetPass word 27
AwesomeNote.Resources.AppResources.SuccessfulAddNote Folder 26		AwesomeNote.Resources.AppResources.Sync 28
AwesomeNote.Resources.AppResources.SuccessfulBackgro undColorChanged 26		AwesomeNote.Resources.AppResources.SyncFeature 28
AwesomeNote.Resources.AppResources.SuccessfulBackgro undImageChanged 26		AwesomeNote.Resources.AppResources.SyncFeatureNote 28
AwesomeNote.Resources.AppResources.SuccessfulBackgro undImageRemoved 26		AwesomeNote.Resources.AppResources.SyncOnOneFile 28
AwesomeNote.Resources.AppResources.SuccessfulChangeF olderName 26		AwesomeNote.Resources.AppResources.Synchronizing 28
AwesomeNote.Resources.AppResources.SuccessfulChange NoteFolder 26		AwesomeNote.Resources.AppResources.SystemFault 28
AwesomeNote.Resources.AppResources.SuccessfulChange NoteName 26		AwesomeNote.Resources.AppResources.Thai 28
AwesomeNote.Resources.AppResources.SuccessfulDeleteFo lder 26		AwesomeNote.Resources.AppResources.Tips 28
AwesomeNote.Resources.AppResources.SuccessfulDeleteNo te 26		AwesomeNote.Resources.AppResources.Turkish 29
AwesomeNote.Resources.AppResources.SuccessfulFontFam ilyChanged 27		AwesomeNote.Resources.AppResources.TwoFieldsNotSame 29
AwesomeNote.Resources.AppResources.SuccessfulFontSize Changed		AwesomeNote.Resources.AppResources.Welcome 29
		AwesomeNote.Resources.AppResources.WelcomeNote 29
		AwesomeNote.Resources.AppResources.WordNumber 29
		AwesomeNote.Resources.AppResources.Yellow 29
		AwesomeNote.Resources.AppResources.Yes 29
		AwesomeNote.SearchPage 90
		AwesomeNote.SearchPage.messageBackGround 91
		AwesomeNote.SearchPage.OnNavigatedFrom 91
		AwesomeNote.SearchPage.OnNavigatedTo 92
		AwesomeNote.SearchPage.SearchPage 91
		AwesomeNote.SetPasswordPage 92
		AwesomeNote.SetPasswordPage.messageBackGround 93

AwesomeNote.SetPasswordPage.noteFolderId 93
 AwesomeNote.SetPasswordPage.OnFragmentNavigation 93
 AwesomeNote.SetPasswordPage.OnNavigatedFrom 94
 AwesomeNote.SetPasswordPage.OnNavigatedTo 94
 AwesomeNote.SetPasswordPage.SetPasswordPage 92
 AwesomeNote.sln 135
 AwesomeNoteDataContext.cs 135

B

BackgroundColorSettings class 36
 about BackgroundColorSettings class 36
 BackgroundColorSettings 36
 messageBackGround 37
 BackgroundColorSettings.xaml.cs 136

C

ChangeNoteFolder class 37
 about ChangeNoteFolder class 37
 ChangeNoteFolder 38
 messageBackGround 38
 noteFolderId 38
 notelId 38
 OnFragmentNavigation 39
 OnNavigatedFrom 39
 OnNavigatedTo 39
 ChangeNoteFolder.xaml.cs 139

E

EnterPasswordPage class 40
 about EnterPasswordPage class 40
 EnterPasswordPage 40
 messageBackGround 41
 note 41
 noteFolder 41
 noteFolderId 41
 notelId 41
 OnFragmentNavigation 42
 OnNavigatedFrom 42
 OnNavigatedTo 43
 pageName 41

EnterPasswordPage.xaml.cs 141

F

Files 94
 FolderOrderSettings class 43
 about FolderOrderSettings class 43
 FolderOrderSettings 44
 messageBackGround 44
 OnNavigatedFrom 44
 OnNavigatedTo 44
 FolderOrderSettings.xaml.cs 145
 FolderOrderStyleSettings class 45
 about FolderOrderStyleSettings class 45
 FolderOrderStyleSettings 45
 messageBackGround 46
 OnNavigatedFrom 46
 OnNavigatedTo 46
 FolderOrderStyleSettings.xaml.cs 148
 FolderSettings class 46
 about FolderSettings class 46
 FolderSettings 47
 messageBackGround 47
 noteFolderId 47
 OnFragmentNavigation 48
 OnNavigatedFrom 48
 OnNavigatedTo 49
 FolderSettings.xaml.cs 151
 FontFamilySettings class 49
 about FontFamilySettings class 49
 FontFamilySettings 50
 messageBackGround 50
 noteFolderId 50
 OnFragmentNavigation 51
 OnNavigatedFrom 51
 OnNavigatedTo 51
 FontFamilySettings.xaml.cs 157
 FontSizeSettings class 52
 about FontSizeSettings class 52
 FontSizeSettings 52
 messageBackGround 53

noteFolderId 53
 OnFragmentNavigation 53
 OnNavigatedFrom 54
 OnNavigatedTo 54
 FontSizeSettings.xaml.cs 160

G

GeneralSettings class 54
 about GeneralSettings class 54
 CreateDirectoryAsync 58
 DesignFileName 58
 GeneralSettings 55
 messageBackGround 57
 OnNavigatedFrom 59
 OnNavigatedTo 59
 signIn 57
 GeneralSettings.xaml.cs 162

L

Language class 59
 about Language class 59
 LanguageId 60
 LanguageName 60
 Language.cs 174
 LanguageSettings class 60
 about LanguageSettings class 60
 LanguageSettings 60
 messageBackGround 61
 OnNavigatedFrom 61
 OnNavigatedTo 61
 LanguageSettings.xaml.cs 174
 LocalizedStrings class 62
 about LocalizedStrings class 62
 LocalizedResources 62
 LocalizedStrings.cs 178

M

MainPage class 62
 about MainPage class 62
 MainPage 63

messageBackGround 64
 OnNavigatedFrom 64
 OnNavigatedTo 64
 popup 64
 MainPage.xaml.cs 178

N

Note class 64
 about Note class 64
 CreationDate 65
 ModificationDate 65
 NameCreation 65
 NameDescription 66
 NoteDescription 66
 NoteFolderId 66
 NoteId 66
 NoteName 66
 Note.cs 184
 NoteDetail class 66
 about NoteDetail class 66
 flag 68
 messageBackGround 68
 NoteDetail 67
 noteFolderId 68
 noteId 69
 OnFragmentNavigation 69
 OnNavigatedFrom 70
 OnNavigatedTo 70
 pageName 69
 NoteDetail.xaml.cs 185
 NoteFolder class 70
 about NoteFolder class 70
 CreationDate 71
 FolderBackground 71
 FontFamily 71
 FontSize 71
 IsPasswordProtected 72
 ModificationDate 72
 NameCount 72
 NoteCount 72

- NoteFolderDescription 72
- NoteFolderId 72
- NoteFolderName 73
- NoteFolderPassword 73
- NoteOrderBy 73
- NoteOrderStyle 73
- NoteFolder.cs 197
- NoteFolder.xaml.cs 198
- NoteFolderDataContext class 73
 - about NoteFolderDataContext class 73
- AppSettings 74
- ConnectionString 74
- NoteFolderDataContext 74
- NoteFolders 74
- Notes 75
- NoteInfo.xaml.cs 205
- NoteList class 75
 - about NoteList class 75
 - messageBackGround 76
 - noteFolderId 76
 - noteId 76
 - NoteList 75
 - OnFragmentNavigation 77
 - OnNavigatedFrom 78
 - OnNavigatedTo 78
 - popup 76
- NoteOrderSettings class 79
 - about NoteOrderSettings class 79
 - messageBackGround 80
 - noteFolderId 80
 - NoteOrderSettings 79
 - OnFragmentNavigation 80
 - OnNavigatedFrom 80
 - OnNavigatedTo 81
- NoteOrderSettings.xaml.cs 207
- NoteOrderStyleSettings class 81
 - about NoteOrderStyleSettings class 81
 - messageBackGround 82
 - noteFolderId 82
 - NoteOrderStyleSettings 81
 - OnFragmentNavigation 82

- OnNavigatedFrom 83
- OnNavigatedTo 83
- NoteOrderStyleSettings.xaml.cs 210

P

- Page1 class 83
 - about Page1 class 83
 - messageBackGround 84
 - noteFolderId 84
 - noteId 84
 - OnFragmentNavigation 85
 - OnNavigatedFrom 85
 - OnNavigatedTo 86
 - Page1 84
- PopupAddFolder class 86
 - about PopupAddFolder class 86
 - PopupAddFolder 86
- PopupAddFolder.xaml.cs 213
- PopupAddNote class 86
 - about PopupAddNote class 86
 - PopupAddNote 87
- PopupAddNote.xaml.cs 214
- PopupChangeFolder class 87
 - about PopupChangeFolder class 87
 - PopupChangeFolder 87
- PopupChangeFolder.xaml.cs 215
- PopupChangeFolderName class 88
 - about PopupChangeFolderName class 88
 - PopupChangeFolderName 88
- PopupChangeFolderName.xaml.cs 216
- PopupChangeNoteName class 88
 - about PopupChangeNoteName class 88
 - PopupChangeNoteName 89
- PopupChangeNoteName.xaml.cs 217
- PopupEnterPassword class 89
 - about PopupEnterPassword class 89
 - PopupEnterPassword 89
- PopupEnterPassword.xaml.cs 218
- PopupSetPassword class 90
 - about PopupSetPassword class 90

PopupSetPassword 90

PopupSetPassword.xaml.cs 219

S

SearchPage class 90

about SearchPage class 90

messageBackGround 91

OnNavigatedFrom 91

OnNavigatedTo 92

SearchPage 91

SearchPage.xaml.cs 219

SetPasswordPage class 92

about SetPasswordPage class 92

messageBackGround 93

noteFolderId 93

OnFragmentNavigation 93

OnNavigatedFrom 94

OnNavigatedTo 94

SetPasswordPage 92

SetPasswordPage.xaml.cs 224