

Awesome Music

Dinlediginiz albümleri RateYourMusic sitesindeki gibi
puanladiginiz bir Windows Phone uygulaması

Table of Contents

Symbol Reference	1
AwesomeMusic Namespace	1
AwesomeMusic.Resources Namespace	2
Classes	2
AppResources Class	2
Classes	23
AboutPage Class	24
AboutPage>AboutPage Constructor	24
AddCategoryPage Class	25
AddCategoryPage.AddCategoryPage Constructor	25
AddCategoryPage Fields	26
AddCategoryPage Methods	26
Album Class	27
Album Properties	28
AlbumArtist Class	30
AlbumArtist Properties	30
AlbumPage Class	31
AlbumPage.AlbumPage Constructor	32
AlbumPage Fields	33
AlbumPage Methods	34
App Class	36
App.App Constructor	37
App Fields	37
App Properties	38
AppSettings Class	38
AppSettings Properties	39
Artist Class	40
Artist Properties	41
ArtistPage Class	42
ArtistPage.ArtistPage Constructor	43
ArtistPage Fields	44
ArtistPage Methods	44
ArtistSettingsPage Class	47
ArtistSettingsPage.ArtistSettingsPage Constructor	48
ArtistSettingsPage Fields	48
ArtistSettingsPage Methods	49

AwesomeMusicDataContext Class	51
AwesomeMusicDataContext.AwesomeMusicDataContext Constructor	51
AwesomeMusicDataContext Fields	51
BackgroundColorSettingsPage Class	52
BackgroundColorSettingsPage.BackgroundColorSettingsPage Constructor	53
BackgroundColorSettingsPage Fields	54
BackgroundColorSettingsPage Methods	54
Category Class	55
Category Properties	55
CategoryArtist Class	57
CategoryArtist Properties	57
CategoryPage Class	58
CategoryPage.CategoryPage Constructor	58
CategoryPage Fields	59
CategoryPage Methods	59
CategorySettingsPage Class	62
CategorySettingsPage.CategorySettingsPage Constructor	62
CategorySettingsPage Fields	63
CategorySettingsPage Methods	63
FontFamilySettingsPage Class	64
FontFamilySettingsPage.FontFamilySettingsPage Constructor	65
FontFamilySettingsPage Fields	66
FontFamilySettingsPage Methods	66
FontSizeSettingsPage Class	67
FontSizeSettingsPage.FontSizeSettingsPage Constructor	67
FontSizeSettingsPage Fields	68
FontSizeSettingsPage Methods	68
GeneralSettingsPage Class	69
GeneralSettingsPage.GeneralSettingsPage Constructor	70
GeneralSettingsPage Fields	73
GeneralSettingsPage Methods	73
LanguageSettingsPage Class	75
LanguageSettingsPage.LanguageSettingsPage Constructor	75
LanguageSettingsPage Methods	76
LocalizedStrings Class	76
LocalizedStrings Properties	76
MainPage Class	77
MainPage.MainPage Constructor	77
MainPage Fields	78
MainPage Methods	78

OrderSettingsPage Class	79
OrderSettingsPage.OrderSettingsPage Constructor	80
OrderSettingsPage Fields	80
OrderSettingsPage Methods	80
OrderStyleSettingsPage Class	82
OrderStyleSettingsPage.OrderStyleSettingsPage Constructor	83
OrderStyleSettingsPage Fields	83
OrderStyleSettingsPage Methods	83
PopupAddChange Class	85
PopupAddChange.PopupAddChange Constructor	85
SearchPage Class	86
SearchPage.SearchPage Constructor	86
SearchPage Methods	86
StatisticsPage Class	87
StatisticsPage.StatisticsPage Constructor	87
StatisticsPage Methods	88
Files	88
AboutPage.xaml.cs	89
AddCategoryPage.xaml.cs	92
Album.cs	94
AlbumArtist.cs	96
AlbumPage.xaml.cs	96
App.xaml.cs	107
AppResources.Designer.cs	114
AppSettings.cs	134
Artist.cs	135
ArtistPage.xaml.cs	136
ArtistSettingsPage.xaml.cs	144
AssemblyInfo.cs	147
AwesomeMusic.csproj	148
AwesomeMusic.sln	148
AwesomeMusicDataContext.cs	148
BackgroundColorSettingsPage.xaml.cs	149
Category.cs	152
CategoryArtist.cs	153
CategoryPage.xaml.cs	154
CategorySettingsPage.xaml.cs	162
FontFamilySettingsPage.xaml.cs	165
FontSizeSettingsPage.xaml.cs	168
GeneralSettingsPage.xaml.cs	170

LanguageSettingsPage.xaml.cs	183
LocalizedStrings.cs	187
MainPage.xaml.cs	187
OrderSettingsPage.xaml.cs	193
OrderStyleSettingsPage.xaml.cs	198
PopupAddChange.xaml.cs	202
SearchPage.xaml.cs	203
StatisticsPage.xaml.cs	207

Index

a

1 Symbol Reference

























1.1 AwesomeMusic Namespace




This is namespace AwesomeMusic.

Namespaces

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

Classes

	Name	Description
	AboutPage (see page 24)	This is class AwesomeMusic>AboutPage.
	AddCategoryPage (see page 25)	This is class AwesomeMusic.AddCategoryPage.
	Album (see page 27)	This is class AwesomeMusic.Album.
	AlbumArtist (see page 30)	This is class AwesomeMusic.AlbumArtist.
	AlbumPage (see page 31)	This is class AwesomeMusic.AlbumPage.
	App (see page 36)	This is class AwesomeMusic.App.
	AppSettings (see page 38)	This is class AwesomeMusic.AppSettings.
	Artist (see page 40)	This is class AwesomeMusic.Artist.
	ArtistPage (see page 42)	This is class AwesomeMusic.ArtistPage.
	ArtistSettingsPage (see page 47)	This is class AwesomeMusic.ArtistSettingsPage.
	AwesomeMusicDataContext (see page 51)	This is class AwesomeMusic.AwesomeMusicDataContext.
	BackgroundColorSettingsPage (see page 52)	This is class AwesomeMusic.BackgroundColorSettingsPage.
	Category (see page 55)	This is class AwesomeMusic.Category.
	CategoryArtist (see page 57)	This is class AwesomeMusic.CategoryArtist.
	CategoryPage (see page 58)	This is class AwesomeMusic.CategoryPage.
	CategorySettingsPage (see page 62)	This is class AwesomeMusic.CategorySettingsPage.
	FontFamilySettingsPage (see page 64)	This is class AwesomeMusic.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 67)	This is class AwesomeMusic.FontSizeSettingsPage.
	GeneralSettingsPage (see page 69)	This is class AwesomeMusic.GeneralSettingsPage.
	LanguageSettingsPage (see page 75)	This is class AwesomeMusic.LanguageSettingsPage.
	LocalizedStrings (see page 76)	Provides access to string resources.
	MainPage (see page 77)	This is class AwesomeMusic.MainPage.
	OrderSettingsPage (see page 79)	This is class AwesomeMusic.OrderSettingsPage.
	OrderStyleSettingsPage (see page 82)	This is class AwesomeMusic.OrderStyleSettingsPage.

	PopupAddChange (see page 85)	This is class AwesomeMusic.PopupAddChange.
	SearchPage (see page 86)	This is class AwesomeMusic.SearchPage.
	StatisticsPage (see page 87)	This is class AwesomeMusic.StatisticsPage.

1.1.1 AwesomeMusic.Resources Namespace

This is namespace AwesomeMusic.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



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 114](#))



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 7)	This is AppResources, a member of class AppResources.

AppResources Properties





	Name	Description
	About (see page 7)	Looks up a localized string similar to About.
	AboutTheApp (see page 7)	Looks up a localized string similar to About (see page 7) The App (see page 36).

	AboutTheAppText (see page 7)	Looks up a localized string similar to Hi. I like listening album a lot. And after Awesome Library app, i decide to create an app which has similar properties like PitchFork. You can add categories, add artists and add albums on it. You can send information of your albums via SMS, E-Mail and Social Media share (like Facebook etc.). I hope that you will like this app. If you rate app and write your suggestions to marketplace and coderserdar@outlook.com I will be so appreciated to you. With my best regards. CMS Software..
	AboutTheAwesomeMusic (see page 7)	Looks up a localized string similar to About (see page 7) The Awesome Music.
	AddAlbum (see page 7)	Looks up a localized string similar to Add Album (see page 27).
	AddArtist (see page 8)	Looks up a localized string similar to Add Artist (see page 40).
	AddCategory (see page 8)	Looks up a localized string similar to Add Category (see page 55).
	AlbumComment (see page 8)	Looks up a localized string similar to Album (see page 27) Comment.
	AlbumCount (see page 8)	Looks up a localized string similar to Album (see page 27) Count.
	AlbumDeleteSuccess (see page 8)	Looks up a localized string similar to Album (see page 27) Has Been Removed Successfully.
	AlbumList (see page 8)	Looks up a localized string similar to Album (see page 27) List.
	AlbumName (see page 8)	Looks up a localized string similar to Album (see page 27) Name (see page 18).
	AlbumNameMustBe (see page 8)	Looks up a localized string similar to You Have To Enter Album (see page 27) Name (see page 18) At Least.
	AlbumOrderBy (see page 8)	Looks up a localized string similar to Album (see page 27) Order By.
	AlbumOrderStyle (see page 9)	Looks up a localized string similar to Album (see page 27) Order Style.
	AlbumOrderStyleChangeSuccess (see page 9)	Looks up a localized string similar to Album (see page 27) Order Style Has Been Changed Successfully.
	AlbumOrderTypeChangeSuccess (see page 9)	Looks up a localized string similar to Album (see page 27) Order Type Has Been Changed Successfully.
	AlbumRating (see page 9)	Looks up a localized string similar to Album (see page 27) Rating.
	AlbumSaveSuccess (see page 9)	Looks up a localized string similar to Album (see page 27) Has Been Saved Successfully.
	Arabic (see page 9)	Looks up a localized string similar to Arabic.
	ArtistAddSuccess (see page 9)	Looks up a localized string similar to Artist (see page 40) Has Been Added Successfully.
	ArtistAlreadySameCategory (see page 9)	Looks up a localized string similar to Artist (see page 40) Has This Category (see page 55) Already.
	ArtistCategoryAddSuccess (see page 9)	Looks up a localized string similar to Category (see page 55) Has Been Added To Artist (see page 40) Successfully.
	ArtistDeleteSuccess (see page 10)	Looks up a localized string similar to Artist (see page 40) Has Been Removed Successfully.
	ArtistExists (see page 10)	Looks up a localized string similar to This Artist (see page 40) Has Already Exists.
	ArtistList (see page 10)	Looks up a localized string similar to Artist (see page 40) List.
	ArtistName (see page 10)	Looks up a localized string similar to Artist (see page 40) Name (see page 18).
	ArtistNameChangeSuccess (see page 10)	Looks up a localized string similar to Artist (see page 40) Name (see page 18) Has Been Changed Successfully.
	ArtistOrderBy (see page 10)	Looks up a localized string similar to Artist (see page 40) Order By.
	ArtistOrderStyle (see page 10)	Looks up a localized string similar to Artist (see page 40) Order Style.
	ArtistOrderStyleChangeSuccess (see page 10)	Looks up a localized string similar to Artist (see page 40) Order Style Has Been Changed Successfully.

	ArtistOrderTypeChangeSuccess (see page 10)	Looks up a localized string similar to Artist (see page 40) Order Type Has Been Changed Successfully.
	ArtistSettings (see page 11)	Looks up a localized string similar to Artist (see page 40) Settings (see page 21).
	Ascending (see page 11)	Looks up a localized string similar to Ascending.
	Background (see page 11)	Looks up a localized string similar to Background.
	BackgroundColor (see page 11)	Looks up a localized string similar to Background (see page 11) Color.
	BackgroundColorChangeSuccess (see page 11)	Looks up a localized string similar to Background (see page 11) Color Has Been Changed Successfully.
	BackgroundImage (see page 11)	Looks up a localized string similar to Background (see page 11) Image.
	BackgroundImageChangeSuccess (see page 11)	Looks up a localized string similar to Background (see page 11) Image Has Been Changed Successfully.
	BackgroundImageRemoveSuccess (see page 11)	Looks up a localized string similar to Background (see page 11) Image Has Been Removed Successfully.
	BestAlbum (see page 11)	Looks up a localized string similar to Best Album (see page 27).
	BestSong (see page 12)	Looks up a localized string similar to Best Song.
	Black (see page 12)	Looks up a localized string similar to Black.
	Blue (see page 12)	Looks up a localized string similar to Blue.
	Brown (see page 12)	Looks up a localized string similar to Brown.
	Cancel (see page 12)	Looks up a localized string similar to Cancel.
	Categories (see page 12)	Looks up a localized string similar to Categories.
	CategoryAddSuccess (see page 12)	Looks up a localized string similar to Category (see page 55) Has Been Added Successfully.
	CategoryDeleteSuccess (see page 12)	Looks up a localized string similar to Category (see page 55) Has Been Removed Successfully.
	CategoryExists (see page 12)	Looks up a localized string similar to This Category (see page 55) Has Already Exists.
	CategoryName (see page 13)	Looks up a localized string similar to Category (see page 55) Name (see page 18).
	CategoryNameChangeSuccess (see page 13)	Looks up a localized string similar to Category (see page 55) Name (see page 18) Has Been Changed Successfully.
	CategoryOrderBy (see page 13)	Looks up a localized string similar to Category (see page 55) Order By.
	CategoryOrderStyle (see page 13)	Looks up a localized string similar to Category (see page 55) Order Style.
	CategoryOrderStyleChangeSuccess (see page 13)	Looks up a localized string similar to Category (see page 55) Order Style Has Been Changed Successfully.
	CategoryOrderTypeChangeSuccess (see page 13)	Looks up a localized string similar to Category (see page 55) Order Type Has Been Changed Successfully.
	CategorySettings (see page 13)	Looks up a localized string similar to Category (see page 55) Settings (see page 21).
	Chinese (see page 13)	Looks up a localized string similar to Chinese.
	ContactWithUs (see page 13)	Looks up a localized string similar to Contact With Us.
	CreationDate (see page 14)	Looks up a localized string similar to Creation Date (see page 14).
	Culture (see page 14)	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
	Date (see page 14)	Looks up a localized string similar to Date.
	DeleteAlbum (see page 14)	Looks up a localized string similar to Delete Album (see page 27).
	DeleteAlbumQuestion (see page 14)	Looks up a localized string similar to You Will Delete This Album (see page 27). Are You Sure?.

	DeleteArtist (see page 14)	Looks up a localized string similar to Delete Artist (see page 40).
	DeleteArtistQuestion (see page 14)	Looks up a localized string similar to You Will Delete The Artist (see page 40) With All Contents. Are You Sure?.
	DeleteCategory (see page 14)	Looks up a localized string similar to Delete Category (see page 55).
	DeleteCategoryQuestion (see page 15)	Looks up a localized string similar to You Will Delete Category (see page 55) With All Contents. Do You Agree?.
	Descending (see page 15)	Looks up a localized string similar to Descending.
	English (see page 15)	Looks up a localized string similar to English.
	EnterArtistName (see page 15)	Looks up a localized string similar to Enter Artist (see page 40) Name (see page 18).
	EnterCategoryName (see page 15)	Looks up a localized string similar to Enter Category (see page 55) Name (see page 18).
	ExitApp (see page 15)	Looks up a localized string similar to Exit App (see page 36).
	ExitAppQuestion (see page 15)	Looks up a localized string similar to Do You Want To Exit App (see page 36)?.
	FinishDate (see page 15)	Looks up a localized string similar to Finish Date (see page 14).
	Font (see page 15)	Looks up a localized string similar to Font.
	FontFamily (see page 16)	Looks up a localized string similar to Font (see page 15) Family.
	FontFamilyChangeSuccess (see page 16)	Looks up a localized string similar to Font (see page 15) Family Has Been Changed Successfully.
	FontSize (see page 16)	Looks up a localized string similar to Font (see page 15) Size.
	FontSizeChangeSuccess (see page 16)	Looks up a localized string similar to Font (see page 15) Size Has Been Changed Successfully.
	French (see page 16)	Looks up a localized string similar to French.
	GeneralSettings (see page 16)	Looks up a localized string similar to General Settings (see page 21).
	German (see page 16)	Looks up a localized string similar to German.
	Gray (see page 16)	Looks up a localized string similar to Gray.
	Green (see page 16)	Looks up a localized string similar to Green.
	Italian (see page 17)	Looks up a localized string similar to Italian.
	Japanese (see page 17)	Looks up a localized string similar to Japanese.
	LabelName (see page 17)	Looks up a localized string similar to Label Name (see page 18).
	Language (see page 17)	Looks up a localized string similar to Language.
	LanguageWarning (see page 17)	Looks up a localized string similar to You May Restart Application To Change Effect.
	ModificationDate (see page 17)	Looks up a localized string similar to Last Modification Date (see page 14).
	MostListenArtist (see page 17)	Looks up a localized string similar to Artist (see page 40) You Most Listen.
	MostListenCategory (see page 17)	Looks up a localized string similar to Category (see page 55) You Most Listen.
	MostListenLabel (see page 17)	Looks up a localized string similar to Label You Most Listen.
	Name (see page 18)	Looks up a localized string similar to Name.
	OK (see page 18)	Looks up a localized string similar to Ok.
	OneDrive (see page 18)	Looks up a localized string similar to OneDrive.
	OneDriveSyncCompleted (see page 18)	Looks up a localized string similar to OneDrive (see page 18) Sync (see page 22) Completed.
	Orange (see page 18)	Looks up a localized string similar to Orange.
	OtherSettings (see page 18)	Looks up a localized string similar to Other Settings (see page 21).
	Persian (see page 18)	Looks up a localized string similar to Persian.
	Portuguese (see page 18)	Looks up a localized string similar to Portuguese.

	Purple (see page 18)	Looks up a localized string similar to Purple.
	Rate (see page 19)	Looks up a localized string similar to Rate.
	Red (see page 19)	Looks up a localized string similar to Red.
	ReleaseYear (see page 19)	Looks up a localized string similar to Release Year.
	RemoveBackgroundImage (see page 19)	Looks up a localized string similar to Remove Background (see page 11) Image.
	ResetSettings (see page 19)	Looks up a localized string similar to Reset Settings (see page 21).
	ResourceFlowDirection (see page 19)	Looks up a localized string similar to LeftToRight.
	ResourceLanguage (see page 19)	Looks up a localized string similar to en-US.
	ResourceManager (see page 19)	Returns the cached ResourceManager instance used by this class.
	Russian (see page 19)	Looks up a localized string similar to Russian.
	Sanskrit (see page 20)	Looks up a localized string similar to Sanskrit.
	Save (see page 20)	Looks up a localized string similar to Save.
	Search (see page 20)	Looks up a localized string similar to Search.
	SearchCompleted (see page 20)	Looks up a localized string similar to Search (see page 20) Completed.
	SearchResults (see page 20)	Looks up a localized string similar to Search (see page 20) Results.
	SearchTrimFault (see page 20)	Looks up a localized string similar to You Have To Fill Search (see page 20) Criteria.
	Select (see page 20)	Looks up a localized string similar to Select.
	SelectBackgroundColor (see page 20)	Looks up a localized string similar to Select (see page 20) Background (see page 11) Color.
	Selected (see page 20)	Looks up a localized string similar to Selected.
	SelectFontFamily (see page 21)	Looks up a localized string similar to Select (see page 20) Font (see page 15) Family.
	SelectFontSize (see page 21)	Looks up a localized string similar to Select (see page 20) Font (see page 15) Size.
	SelectLanguage (see page 21)	Looks up a localized string similar to Select (see page 20) Language (see page 17).
	SendWithApp (see page 21)	Looks up a localized string similar to Send With Awesome Music App (see page 36).
	SendWithMail (see page 21)	Looks up a localized string similar to Send With Mail.
	SendWithSMS (see page 21)	Looks up a localized string similar to Send With SMS.
	Settings (see page 21)	Looks up a localized string similar to Settings.
	ShareAlbum (see page 21)	Looks up a localized string similar to Share Album (see page 27).
	SongCount (see page 21)	Looks up a localized string similar to Song Count.
	Spanish (see page 22)	Looks up a localized string similar to Spanish.
	StartDate (see page 22)	Looks up a localized string similar to Start Date (see page 14).
	Statistics (see page 22)	Looks up a localized string similar to Statistics.
	SuccessfulResetSettings (see page 22)	Looks up a localized string similar to Background (see page 11) Settings (see page 21) Has Been Reset Successfully.
	Sync (see page 22)	Looks up a localized string similar to Sync.
	Synchronizing (see page 22)	Looks up a localized string similar to Synchronizing.
	SyncOnOneFile (see page 22)	Looks up a localized string similar to Sync (see page 22) All In One File.
	SystemFault (see page 22)	Looks up a localized string similar to System Has A Fault. Please Try Again Later.
	Thai (see page 22)	Looks up a localized string similar to Thai.

	TotalAlbumCount (see page 23)	Looks up a localized string similar to Album (see page 27) Count You Listen.
	Turkish (see page 23)	Looks up a localized string similar to Turkish.
	WorstAlbum (see page 23)	Looks up a localized string similar to Worst Album (see page 27).
	Yellow (see page 23)	Looks up a localized string similar to Yellow.

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 About.

C#

```
public static string About;
```

1.1.1.1.1.2.2 AppResources.AboutTheApp Property

Looks up a localized string similar to About ([see page 7](#)) The App ([see page 36](#)).

C#

```
public static string AboutTheApp;
```

1.1.1.1.1.2.3 AppResources.AboutTheAppText Property

Looks up a localized string similar to Hi. I like listening album a lot. And after Awesome Library app, i decide to create an app which has similar properties like PitchFork. You can add categories, add artists and add albums on it. You can send information of your albums via SMS, E-Mail and Social Media share (like Facebook etc.). I hope that you will like this app. If you rate app and write your suggestions to marketplace and coderserdar@outlook.com I will be so appreciated to you. With my best regards. ÇMS Software..

C#

```
public static string AboutTheAppText;
```

1.1.1.1.1.2.4 AppResources.AboutTheAwesomeMusic Property

Looks up a localized string similar to About ([see page 7](#)) The Awesome Music.

C#

```
public static string AboutTheAwesomeMusic;
```

1.1.1.1.1.2.5 AppResources.AddAlbum Property

Looks up a localized string similar to Add Album ([see page 27](#)).

C#

```
public static string AddAlbum;
```

1.1.1.1.1.2.6 AppResources.AddArtist Property

Looks up a localized string similar to Add Artist (see page 40).

C#

```
public static string AddArtist;
```

1.1.1.1.1.2.7 AppResources.AddCategory Property

Looks up a localized string similar to Add Category (see page 55).

C#

```
public static string AddCategory;
```

1.1.1.1.1.2.8 AppResources.AlbumComment Property

Looks up a localized string similar to Album (see page 27) Comment.

C#

```
public static string AlbumComment;
```

1.1.1.1.1.2.9 AppResources.AlbumCount Property

Looks up a localized string similar to Album (see page 27) Count.

C#

```
public static string AlbumCount;
```

1.1.1.1.1.2.10 AppResources.AlbumDeleteSuccess Property

Looks up a localized string similar to Album (see page 27) Has Been Removed Successfully.

C#

```
public static string AlbumDeleteSuccess;
```

1.1.1.1.1.2.11 AppResources.AlbumList Property

Looks up a localized string similar to Album (see page 27) List.

C#

```
public static string AlbumList;
```

1.1.1.1.1.2.12 AppResources.AlbumName Property

Looks up a localized string similar to Album (see page 27) Name (see page 18).

C#

```
public static string AlbumName;
```

1.1.1.1.1.2.13 AppResources.AlbumNameMustBe Property

Looks up a localized string similar to You Have To Enter Album (see page 27) Name (see page 18) At Least.

C#

```
public static string AlbumNameMustBe;
```

1.1.1.1.1.2.14 AppResources.AlbumOrderBy Property

Looks up a localized string similar to Album (see page 27) Order By.

C#

```
public static string AlbumOrderBy;
```

1.1.1.1.1.2.15 AppResources.AlbumOrderStyle Property

Looks up a localized string similar to Album (see page 27) Order Style.

C#

```
public static string AlbumOrderStyle;
```

1.1.1.1.1.2.16 AppResources.AlbumOrderStyleChangeSuccess Property

Looks up a localized string similar to Album (see page 27) Order Style Has Been Changed Successfully.

C#

```
public static string AlbumOrderStyleChangeSuccess;
```

1.1.1.1.1.2.17 AppResources.AlbumOrderTypeChangeSuccess Property

Looks up a localized string similar to Album (see page 27) Order Type Has Been Changed Successfully.

C#

```
public static string AlbumOrderTypeChangeSuccess;
```

1.1.1.1.1.2.18 AppResources.AlbumRating Property

Looks up a localized string similar to Album (see page 27) Rating.

C#

```
public static string AlbumRating;
```

1.1.1.1.1.2.19 AppResources.AlbumSaveSuccess Property

Looks up a localized string similar to Album (see page 27) Has Been Saved Successfully.

C#

```
public static string AlbumSaveSuccess;
```

1.1.1.1.1.2.20 AppResources.Arabic Property

Looks up a localized string similar to Arabic.

C#

```
public static string Arabic;
```

1.1.1.1.1.2.21 AppResources.ArtistAddSuccess Property

Looks up a localized string similar to Artist (see page 40) Has Been Added Successfully.

C#

```
public static string ArtistAddSuccess;
```

1.1.1.1.1.2.22 AppResources.ArtistAlreadySameCategory Property

Looks up a localized string similar to Artist (see page 40) Has This Category (see page 55) Already.

C#

```
public static string ArtistAlreadySameCategory;
```

1.1.1.1.1.2.23 AppResources.ArtistCategoryAddSuccess Property

Looks up a localized string similar to Category (see page 55) Has Been Added To Artist (see page 40) Successfully.

C#

```
public static string ArtistCategoryAddSuccess;
```

1.1.1.1.1.2.24 AppResources.ArtistDeleteSuccess Property

Looks up a localized string similar to Artist (see page 40) Has Been Removed Successfully.

C#

```
public static string ArtistDeleteSuccess;
```

1.1.1.1.1.2.25 AppResources.ArtistExists Property

Looks up a localized string similar to This Artist (see page 40) Has Already Exists.

C#

```
public static string ArtistExists;
```

1.1.1.1.1.2.26 AppResources.ArtistList Property

Looks up a localized string similar to Artist (see page 40) List.

C#

```
public static string ArtistList;
```

1.1.1.1.1.2.27 AppResources.ArtistName Property

Looks up a localized string similar to Artist (see page 40) Name (see page 18).

C#

```
public static string ArtistName;
```

1.1.1.1.1.2.28 AppResources.ArtistNameChangeSuccess Property

Looks up a localized string similar to Artist (see page 40) Name (see page 18) Has Been Changed Successfully.

C#

```
public static string ArtistNameChangeSuccess;
```

1.1.1.1.1.2.29 AppResources.ArtistOrderBy Property

Looks up a localized string similar to Artist (see page 40) Order By.

C#

```
public static string ArtistOrderBy;
```

1.1.1.1.1.2.30 AppResources.ArtistOrderStyle Property

Looks up a localized string similar to Artist (see page 40) Order Style.

C#

```
public static string ArtistOrderStyle;
```

1.1.1.1.1.2.31 AppResources.ArtistOrderStyleChangeSuccess Property

Looks up a localized string similar to Artist (see page 40) Order Style Has Been Changed Successfully.

C#

```
public static string ArtistOrderStyleChangeSuccess;
```

1.1.1.1.1.2.32 AppResources.ArtistOrderTypeChangeSuccess Property

Looks up a localized string similar to Artist (see page 40) Order Type Has Been Changed Successfully.

C#

```
public static string ArtistOrderTypeChangeSuccess;
```

1.1.1.1.1.2.33 AppResources.ArtistSettings Property

Looks up a localized string similar to Artist ([see page 40](#)) Settings ([see page 21](#)).

C#

```
public static string ArtistSettings;
```

1.1.1.1.1.2.34 AppResources.Ascending Property

Looks up a localized string similar to Ascending.

C#

```
public static string Ascending;
```

1.1.1.1.1.2.35 AppResources.Background Property

Looks up a localized string similar to Background.

C#

```
public static string Background;
```

1.1.1.1.1.2.36 AppResources.BackgroundColor Property

Looks up a localized string similar to Background ([see page 11](#)) Color.

C#

```
public static string BackgroundColor;
```

1.1.1.1.1.2.37 AppResources.BackgroundColorChangeSuccess Property

Looks up a localized string similar to Background ([see page 11](#)) Color Has Been Changed Successfully.

C#

```
public static string BackgroundColorChangeSuccess;
```

1.1.1.1.1.2.38 AppResources.BackgroundImage Property

Looks up a localized string similar to Background ([see page 11](#)) Image.

C#

```
public static string BackgroundImage;
```

1.1.1.1.1.2.39 AppResources.BackgroundImageChangeSuccess Property

Looks up a localized string similar to Background ([see page 11](#)) Image Has Been Changed Successfully.

C#

```
public static string BackgroundImageChangeSuccess;
```

1.1.1.1.1.2.40 AppResources.BackgroundImageRemoveSuccess Property

Looks up a localized string similar to Background ([see page 11](#)) Image Has Been Removed Successfully.

C#

```
public static string BackgroundImageRemoveSuccess;
```

1.1.1.1.1.2.41 AppResources.BestAlbum Property

Looks up a localized string similar to Best Album ([see page 27](#)).

C#

```
public static string BestAlbum;
```

1.1.1.1.1.2.42 AppResources.BestSong Property

Looks up a localized string similar to Best Song.

C#

```
public static string BestSong;
```

1.1.1.1.1.2.43 AppResources.Black Property

Looks up a localized string similar to Black.

C#

```
public static string Black;
```

1.1.1.1.1.2.44 AppResources.Blue Property

Looks up a localized string similar to Blue.

C#

```
public static string Blue;
```

1.1.1.1.1.2.45 AppResources.Brown Property

Looks up a localized string similar to Brown.

C#

```
public static string Brown;
```

1.1.1.1.1.2.46 AppResources.Cancel Property

Looks up a localized string similar to Cancel.

C#

```
public static string Cancel;
```

1.1.1.1.1.2.47 AppResources.Categories Property

Looks up a localized string similar to Categories.

C#

```
public static string Categories;
```

1.1.1.1.1.2.48 AppResources.CategoryAddSuccess Property

Looks up a localized string similar to Category (see page 55) Has Been Added Successfully.

C#

```
public static string CategoryAddSuccess;
```

1.1.1.1.1.2.49 AppResources.CategoryDeleteSuccess Property

Looks up a localized string similar to Category (see page 55) Has Been Removed Successfully.

C#

```
public static string CategoryDeleteSuccess;
```

1.1.1.1.1.2.50 AppResources.CategoryExists Property

Looks up a localized string similar to This Category (see page 55) Has Already Exists.

C#

```
public static string CategoryExists;
```

1.1.1.1.1.2.51 AppResources.CategoryName Property

Looks up a localized string similar to Category (see page 55) Name (see page 18).

C#

```
public static string CategoryName;
```

1.1.1.1.1.2.52 AppResources.CategoryNameChangeSuccess Property

Looks up a localized string similar to Category (see page 55) Name (see page 18) Has Been Changed Successfully.

C#

```
public static string CategoryNameChangeSuccess;
```

1.1.1.1.1.2.53 AppResources.CategoryOrderBy Property

Looks up a localized string similar to Category (see page 55) Order By.

C#

```
public static string CategoryOrderBy;
```

1.1.1.1.1.2.54 AppResources.CategoryOrderStyle Property

Looks up a localized string similar to Category (see page 55) Order Style.

C#

```
public static string CategoryOrderStyle;
```

1.1.1.1.1.2.55 AppResources.CategoryOrderStyleChangeSuccess Property

Looks up a localized string similar to Category (see page 55) Order Style Has Been Changed Successfully.

C#

```
public static string CategoryOrderStyleChangeSuccess;
```

1.1.1.1.1.2.56 AppResources.CategoryOrderTypeChangeSuccess Property

Looks up a localized string similar to Category (see page 55) Order Type Has Been Changed Successfully.

C#

```
public static string CategoryOrderTypeChangeSuccess;
```

1.1.1.1.1.2.57 AppResources.CategorySettings Property

Looks up a localized string similar to Category (see page 55) Settings (see page 21).

C#

```
public static string CategorySettings;
```

1.1.1.1.1.2.58 AppResources.Chinese Property

Looks up a localized string similar to Chinese.

C#

```
public static string Chinese;
```

1.1.1.1.1.2.59 AppResources.ContactWithUs Property

Looks up a localized string similar to Contact With Us.

C#

```
public static string ContactWithUs;
```

1.1.1.1.1.2.60 AppResources.CreationDate Property

Looks up a localized string similar to Creation Date (see page 14).

C#

```
public static string CreationDate;
```

1.1.1.1.1.2.61 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;
```

1.1.1.1.1.2.62 AppResources.Date Property

Looks up a localized string similar to Date.

C#

```
public static string Date;
```

1.1.1.1.1.2.63 AppResources.DeleteAlbum Property

Looks up a localized string similar to Delete Album (see page 27).

C#

```
public static string DeleteAlbum;
```

1.1.1.1.1.2.64 AppResources.DeleteAlbumQuestion Property

Looks up a localized string similar to You Will Delete This Album (see page 27). Are You Sure?.

C#

```
public static string DeleteAlbumQuestion;
```

1.1.1.1.1.2.65 AppResources.DeleteArtist Property

Looks up a localized string similar to Delete Artist (see page 40).

C#

```
public static string DeleteArtist;
```

1.1.1.1.1.2.66 AppResources.DeleteArtistQuestion Property

Looks up a localized string similar to You Will Delete The Artist (see page 40) With All Contents. Are You Sure?.

C#

```
public static string DeleteArtistQuestion;
```

1.1.1.1.1.2.67 AppResources.DeleteCategory Property

Looks up a localized string similar to Delete Category (see page 55).

C#

```
public static string DeleteCategory;
```

1.1.1.1.1.2.68 AppResources.DeleteCategoryQuestion Property

Looks up a localized string similar to You Will Delete Category (see page 55) With All Contents. Do You Agree?.

C#

```
public static string DeleteCategoryQuestion;
```

1.1.1.1.1.2.69 AppResources.Descending Property

Looks up a localized string similar to Descending.

C#

```
public static string Descending;
```

1.1.1.1.1.2.70 AppResources.English Property

Looks up a localized string similar to English.

C#

```
public static string English;
```

1.1.1.1.1.2.71 AppResources.EnterArtistName Property

Looks up a localized string similar to Enter Artist (see page 40) Name (see page 18).

C#

```
public static string EnterArtistName;
```

1.1.1.1.1.2.72 AppResources.EnterCategoryName Property

Looks up a localized string similar to Enter Category (see page 55) Name (see page 18).

C#

```
public static string EnterCategoryName;
```

1.1.1.1.1.2.73 AppResources.ExitApp Property

Looks up a localized string similar to Exit App (see page 36).

C#

```
public static string ExitApp;
```

1.1.1.1.1.2.74 AppResources.ExitAppQuestion Property

Looks up a localized string similar to Do You Want To Exit App (see page 36)?.

C#

```
public static string ExitAppQuestion;
```

1.1.1.1.1.2.75 AppResources.FinishDate Property

Looks up a localized string similar to Finish Date (see page 14).

C#

```
public static string FinishDate;
```

1.1.1.1.1.2.76 AppResources.Font Property

Looks up a localized string similar to Font.

C#

```
public static string Font;
```

1.1.1.1.1.2.77 AppResources.FontFamily Property

Looks up a localized string similar to Font (see page 15) Family.

C#

```
public static string FontFamily;
```

1.1.1.1.1.2.78 AppResources.FontFamilyChangeSuccess Property

Looks up a localized string similar to Font (see page 15) Family Has Been Changed Successfully.

C#

```
public static string FontFamilyChangeSuccess;
```

1.1.1.1.1.2.79 AppResources.FontSize Property

Looks up a localized string similar to Font (see page 15) Size.

C#

```
public static string FontSize;
```

1.1.1.1.1.2.80 AppResources.FontSizeChangeSuccess Property

Looks up a localized string similar to Font (see page 15) Size Has Been Changed Successfully.

C#

```
public static string FontSizeChangeSuccess;
```

1.1.1.1.1.2.81 AppResources.French Property

Looks up a localized string similar to French.

C#

```
public static string French;
```

1.1.1.1.1.2.82 AppResources.GeneralSettings Property

Looks up a localized string similar to General Settings (see page 21).

C#

```
public static string GeneralSettings;
```

1.1.1.1.1.2.83 AppResources.German Property

Looks up a localized string similar to German.

C#

```
public static string German;
```

1.1.1.1.1.2.84 AppResources.Gray Property

Looks up a localized string similar to Gray.

C#

```
public static string Gray;
```

1.1.1.1.1.2.85 AppResources.Green Property

Looks up a localized string similar to Green.

C#

```
public static string Green;
```

1.1.1.1.1.2.86 AppResources.Italian Property

Looks up a localized string similar to Italian.

C#

```
public static string Italian;
```

1.1.1.1.1.2.87 AppResources.Japanese Property

Looks up a localized string similar to Japanese.

C#

```
public static string Japanese;
```

1.1.1.1.1.2.88 AppResources.LabelName Property

Looks up a localized string similar to Label Name (see page 18).

C#

```
public static string LabelName;
```

1.1.1.1.1.2.89 AppResources.Language Property

Looks up a localized string similar to Language.

C#

```
public static string Language;
```

1.1.1.1.1.2.90 AppResources.LanguageWarning Property

Looks up a localized string similar to You May Restart Application To Change Effect.

C#

```
public static string LanguageWarning;
```

1.1.1.1.1.2.91 AppResources.ModificationDate Property

Looks up a localized string similar to Last Modification Date (see page 14).

C#

```
public static string ModificationDate;
```

1.1.1.1.1.2.92 AppResources.MostListenArtist Property

Looks up a localized string similar to Artist (see page 40) You Most Listen.

C#

```
public static string MostListenArtist;
```

1.1.1.1.1.2.93 AppResources.MostListenCategory Property

Looks up a localized string similar to Category (see page 55) You Most Listen.

C#

```
public static string MostListenCategory;
```

1.1.1.1.1.2.94 AppResources.MostListenLabel Property

Looks up a localized string similar to Label You Most Listen.

C#

```
public static string MostListenLabel;
```

1.1.1.1.1.2.95 AppResources.Name Property

Looks up a localized string similar to Name.

C#

```
public static string Name;
```

1.1.1.1.1.2.96 AppResources.OK Property

Looks up a localized string similar to Ok.

C#

```
public static string OK;
```

1.1.1.1.1.2.97 AppResources.OneDrive Property

Looks up a localized string similar to OneDrive.

C#

```
public static string OneDrive;
```

1.1.1.1.1.2.98 AppResources.OneDriveSyncCompleted Property

Looks up a localized string similar to OneDrive (see page 18) Sync (see page 22) Completed.

C#

```
public static string OneDriveSyncCompleted;
```

1.1.1.1.1.2.99 AppResources.Orange Property

Looks up a localized string similar to Orange.

C#

```
public static string Orange;
```

1.1.1.1.1.2.100 AppResources.OtherSettings Property

Looks up a localized string similar to Other Settings (see page 21).

C#

```
public static string OtherSettings;
```

1.1.1.1.1.2.101 AppResources.Persian Property

Looks up a localized string similar to Persian.

C#

```
public static string Persian;
```

1.1.1.1.1.2.102 AppResources.Portuguese Property

Looks up a localized string similar to Portuguese.

C#

```
public static string Portuguese;
```

1.1.1.1.1.2.103 AppResources.Purple Property

Looks up a localized string similar to Purple.

C#

```
public static string Purple;
```

1.1.1.1.2.104 AppResources.Rate Property

Looks up a localized string similar to Rate.

C#

```
public static string Rate;
```

1.1.1.1.2.105 AppResources.Red Property

Looks up a localized string similar to Red.

C#

```
public static string Red;
```

1.1.1.1.2.106 AppResources.ReleaseYear Property

Looks up a localized string similar to Release Year.

C#

```
public static string ReleaseYear;
```

1.1.1.1.2.107 AppResources.RemoveBackgroundImage Property

Looks up a localized string similar to Remove Background (see page 11) Image.

C#

```
public static string RemoveBackgroundImage;
```

1.1.1.1.2.108 AppResources.ResetSettings Property

Looks up a localized string similar to Reset Settings (see page 21).

C#

```
public static string ResetSettings;
```

1.1.1.1.2.109 AppResources.ResourceFlowDirection Property

Looks up a localized string similar to LeftToRight.

C#

```
public static string ResourceFlowDirection;
```

1.1.1.1.2.110 AppResources.ResourceLanguage Property

Looks up a localized string similar to en-US.

C#

```
public static string ResourceLanguage;
```

1.1.1.1.2.111 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;
```

1.1.1.1.2.112 AppResources.Russian Property

Looks up a localized string similar to Russian.

C#

```
public static string Russian;
```

1.1.1.1.1.2.113 AppResources.Sanskrit Property

Looks up a localized string similar to Sanskrit.

C#

```
public static string Sanskrit;
```

1.1.1.1.1.2.114 AppResources.Save Property

Looks up a localized string similar to Save.

C#

```
public static string Save;
```

1.1.1.1.1.2.115 AppResources.Search Property

Looks up a localized string similar to Search.

C#

```
public static string Search;
```

1.1.1.1.1.2.116 AppResources.SearchCompleted Property

Looks up a localized string similar to Search (see page 20) Completed.

C#

```
public static string SearchCompleted;
```

1.1.1.1.1.2.117 AppResources.SearchResults Property

Looks up a localized string similar to Search (see page 20) Results.

C#

```
public static string SearchResults;
```

1.1.1.1.1.2.118 AppResources.SearchTrimFault Property

Looks up a localized string similar to You Have To Fill Search (see page 20) Criteria.

C#

```
public static string SearchTrimFault;
```

1.1.1.1.1.2.119 AppResources.Select Property

Looks up a localized string similar to Select.

C#

```
public static string Select;
```

1.1.1.1.1.2.120 AppResources.SelectBackgroundColor Property

Looks up a localized string similar to Select (see page 20) Background (see page 11) Color.

C#

```
public static string SelectBackgroundColor;
```

1.1.1.1.1.2.121 AppResources.Selected Property

Looks up a localized string similar to Selected.

C#

```
public static string Selected;
```

1.1.1.1.1.2.122 AppResources.SelectFontFamily Property

Looks up a localized string similar to Select (see page 20) Font (see page 15) Family.

C#

```
public static string SelectFontFamily;
```

1.1.1.1.1.2.123 AppResources.SelectFontSize Property

Looks up a localized string similar to Select (see page 20) Font (see page 15) Size.

C#

```
public static string SelectFontSize;
```

1.1.1.1.1.2.124 AppResources.SelectLanguage Property

Looks up a localized string similar to Select (see page 20) Language (see page 17).

C#

```
public static string SelectLanguage;
```

1.1.1.1.1.2.125 AppResources.SendWithApp Property

Looks up a localized string similar to Send With Awesome Music App (see page 36).

C#

```
public static string SendWithApp;
```

1.1.1.1.1.2.126 AppResources.SendWithMail Property

Looks up a localized string similar to Send With Mail.

C#

```
public static string SendWithMail;
```

1.1.1.1.1.2.127 AppResources.SendWithSMS Property

Looks up a localized string similar to Send With SMS.

C#

```
public static string SendWithSMS;
```

1.1.1.1.1.2.128 AppResources.Settings Property

Looks up a localized string similar to Settings.

C#

```
public static string Settings;
```

1.1.1.1.1.2.129 AppResources.ShareAlbum Property

Looks up a localized string similar to Share Album (see page 27).

C#

```
public static string ShareAlbum;
```

1.1.1.1.1.2.130 AppResources.SongCount Property

Looks up a localized string similar to Song Count.

C#

```
public static string SongCount;
```

1.1.1.1.1.2.131 AppResources.Spanish Property

Looks up a localized string similar to Spanish.

C#

```
public static string Spanish;
```

1.1.1.1.1.2.132 AppResources.StartDate Property

Looks up a localized string similar to Start Date (see page 14).

C#

```
public static string StartDate;
```

1.1.1.1.1.2.133 AppResources.Statistics Property

Looks up a localized string similar to Statistics.

C#

```
public static string Statistics;
```

1.1.1.1.1.2.134 AppResources.SuccessfulResetSettings Property

Looks up a localized string similar to Background (see page 11) Settings (see page 21) Has Been Reset Successfully.

C#

```
public static string SuccessfulResetSettings;
```

1.1.1.1.1.2.135 AppResources.Sync Property

Looks up a localized string similar to Sync.

C#

```
public static string Sync;
```

1.1.1.1.1.2.136 AppResources.Synchronizing Property

Looks up a localized string similar to Synchronizing.

C#

```
public static string Synchronizing;
```

1.1.1.1.1.2.137 AppResources.SyncOnOneFile Property

Looks up a localized string similar to Sync (see page 22) All In One File.

C#

```
public static string SyncOnOneFile;
```

1.1.1.1.1.2.138 AppResources.SystemFault Property

Looks up a localized string similar to System Has A Fault. Please Try Again Later.

C#

```
public static string SystemFault;
```

1.1.1.1.1.2.139 AppResources.Thai Property

Looks up a localized string similar to Thai.

C#

```
public static string Thai;
```

1.1.1.1.1.2.140 AppResources.TotalAlbumCount Property

Looks up a localized string similar to Album (see page 27) Count You Listen.

C#

```
public static string TotalAlbumCount;
```

1.1.1.1.1.2.141 AppResources.Turkish Property

Looks up a localized string similar to Turkish.

C#

```
public static string Turkish;
```

1.1.1.1.1.2.142 AppResources.WorstAlbum Property

Looks up a localized string similar to Worst Album (see page 27).

C#

```
public static string WorstAlbum;
```

1.1.1.1.1.2.143 AppResources.Yellow Property

Looks up a localized string similar to Yellow.
















C#













```
public static string Yellow;
```

1.1.2 Classes

The following table lists classes in this documentation.

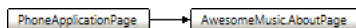
Classes

	Name	Description
	AboutPage (see page 24)	This is class AwesomeMusic>AboutPage.
	AddCategoryPage (see page 25)	This is class AwesomeMusic.AddCategoryPage.
	Album (see page 27)	This is class AwesomeMusic.Album.
	AlbumArtist (see page 30)	This is class AwesomeMusic.AlbumArtist.
	AlbumPage (see page 31)	This is class AwesomeMusic.AlbumPage.
	App (see page 36)	This is class AwesomeMusic.App.
	AppSettings (see page 38)	This is class AwesomeMusic.AppSettings.
	Artist (see page 40)	This is class AwesomeMusic.Artist.
	ArtistPage (see page 42)	This is class AwesomeMusic.ArtistPage.
	ArtistSettingsPage (see page 47)	This is class AwesomeMusic.ArtistSettingsPage.
	AwesomeMusicDataContext (see page 51)	This is class AwesomeMusic.AwesomeMusicDataContext.
	BackgroundColorSettingsPage (see page 52)	This is class AwesomeMusic.BackgroundColorSettingsPage.
	Category (see page 55)	This is class AwesomeMusic.Category.
	CategoryArtist (see page 57)	This is class AwesomeMusic.CategoryArtist.
	CategoryPage (see page 58)	This is class AwesomeMusic.CategoryPage.

	CategorySettingsPage (see page 62)	This is class AwesomeMusic.CategorySettingsPage.
	FontFamilySettingsPage (see page 64)	This is class AwesomeMusic.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 67)	This is class AwesomeMusic.FontSizeSettingsPage.
	GeneralSettingsPage (see page 69)	This is class AwesomeMusic.GeneralSettingsPage.
	LanguageSettingsPage (see page 75)	This is class AwesomeMusic.LanguageSettingsPage.
	LocalizedStrings (see page 76)	Provides access to string resources.
	MainPage (see page 77)	This is class AwesomeMusic.MainPage.
	OrderSettingsPage (see page 79)	This is class AwesomeMusic.OrderSettingsPage.
	OrderStyleSettingsPage (see page 82)	This is class AwesomeMusic.OrderStyleSettingsPage.
	PopupAddChange (see page 85)	This is class AwesomeMusic.PopupAddChange.
	SearchPage (see page 86)	This is class AwesomeMusic.SearchPage.
	StatisticsPage (see page 87)	This is class AwesomeMusic.StatisticsPage.

1.1.2.1 AboutPage Class

Class Hierarchy



C#

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

File

AboutPage.xaml.cs ([see page 89](#))

Description

This is class AwesomeMusic.AboutPage.

Methods

	Name	Description
	AboutPage (see page 24)	This is AboutPage, 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: }
  
```

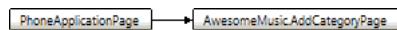
```

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: }

```

1.1.2.2 AddCategoryPage Class

Class Hierarchy



C#

```
public class AddCategoryPage : PhoneApplicationPage;
```

File

AddCategoryPage.xaml.cs (see page 92)

Description

This is class AwesomeMusic.AddCategoryPage.

Methods

	Name	Description
☞	AddCategoryPage (see page 25)	This is AddCategoryPage, a member of class AddCategoryPage.

AddCategoryPage Fields

	Name	Description
💡	artistId (see page 26)	This is artistId, a member of class AddCategoryPage.

AddCategoryPage Methods

	Name	Description
☞	OnFragmentNavigation (see page 26)	This is OnFragmentNavigation, a member of class AddCategoryPage.
☞	OnNavigatedFrom (see page 26)	This is OnNavigatedFrom, a member of class AddCategoryPage.
☞	OnNavigatedTo (see page 27)	This is OnNavigatedTo, a member of class AddCategoryPage.

1.1.2.2.1 AddCategoryPage.AddCategoryPage Constructor

C#

```
public AddCategoryPage();
```

Description

This is AddCategoryPage, a member of class AddCategoryPage.

Body Source

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

1.1.2.2.2 AddCategoryPage Fields

1.1.2.2.2.1 AddCategoryPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class AddCategoryPage.

1.1.2.2.3 AddCategoryPage Methods

1.1.2.2.3.1 AddCategoryPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class AddCategoryPage.

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:     artistId = int.Parse(e.Fragment);  
7:     using (var context = new  
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))  
8:     {  
9:         var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single()  
as Artist;  
10:         lstCategories.Items.Clear();  
11:         lblArtistName.Text = artist.ArtistName;  
12:         lblCategories.Text = AppResources.Categories;  
13:         var categories = context.Categories;  
14:         lstCategories.ItemsSource = categories;  
15:         lstCategories.DisplayMemberPath = "CategoryName";  
16:     }  
17: }
```

1.1.2.2.3.2 AddCategoryPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class AddCategoryPage.

Body Source

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

1.1.2.2.3.3 AddCategoryPage.OnNavigatedTo Method**C#**

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

Description

This is OnNavigatedTo, a member of class AddCategoryPage.

Body Source

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

1.1.2.3 Album Class**Class Hierarchy**

AwesomeMusic.Album

C#

```
[Table]
public class Album;
```















File

Album.cs ([see page 94](#))

Description

This is class AwesomeMusic.Album.

Album Properties

	Name	Description
	AlbumBestSong (see page 28)	This is AlbumBestSong, a member of class Album.
	AlbumCategoryId (see page 28)	This is AlbumCategoryId, a member of class Album.
	AlbumComment (see page 28)	This is AlbumComment, a member of class Album.
	AlbumGuid (see page 28)	This is AlbumGuid, a member of class Album.
	AlbumId (see page 28)	This is AlbumId, a member of class Album.
	AlbumInformation (see page 28)	This is AlbumInformation, a member of class Album.
	AlbumLabelName (see page 29)	This is AlbumLabelName, a member of class Album.
	AlbumName (see page 29)	This is AlbumName, a member of class Album.
	AlbumNameRating (see page 29)	This is AlbumNameRating, a member of class Album.
	AlbumRating (see page 29)	This is AlbumRating, a member of class Album.
	AlbumReleaseYear (see page 29)	This is AlbumReleaseYear, a member of class Album.
	AlbumSongCount (see page 29)	This is AlbumSongCount, a member of class Album.
	CreationDate (see page 30)	This is CreationDate, a member of class Album.
	ModificationDate (see page 30)	This is ModificationDate, a member of class Album.

1.1.2.3.1 Album Properties

1.1.2.3.1.1 Album.AlbumBestSong Property

C#

```
[Column]
public string AlbumBestSong;
```

Description

This is AlbumBestSong, a member of class Album.

1.1.2.3.1.2 Album.AlbumCategoryId Property

C#

```
[Column]
public int AlbumCategoryId;
```

Description

This is AlbumCategoryId, a member of class Album.

1.1.2.3.1.3 Album.AlbumComment Property

C#

```
[Column]
public string AlbumComment;
```

Description

This is AlbumComment, a member of class Album.

1.1.2.3.1.4 Album.AlbumGuid Property

C#

```
[Column]
public string AlbumGuid;
```

Description

This is AlbumGuid, a member of class Album.

1.1.2.3.1.5 Album.AlbumId Property

C#

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

Description

This is AlbumId, a member of class Album.

1.1.2.3.1.6 Album.AlbumInformation Property

C#

```
[Column]
public string AlbumInformation;
```

Description

This is AlbumInformation, a member of class Album.

1.1.2.3.1.7 Album.AlbumLabelName Property**C#**

```
[Column]  
public string AlbumLabelName;
```

Description

This is AlbumLabelName, a member of class Album.

1.1.2.3.1.8 Album.AlbumName Property**C#**

```
[Column]  
public string AlbumName;
```

Description

This is AlbumName, a member of class Album.

1.1.2.3.1.9 Album.AlbumNameRating Property**C#**

```
[Column]  
public string AlbumNameRating;
```

Description

This is AlbumNameRating, a member of class Album.

1.1.2.3.1.10 Album.AlbumRating Property**C#**

```
[Column]  
public int AlbumRating;
```

Description

This is AlbumRating, a member of class Album.

1.1.2.3.1.11 Album.AlbumReleaseYear Property**C#**

```
[Column]  
public int AlbumReleaseYear;
```

Description

This is AlbumReleaseYear, a member of class Album.

1.1.2.3.1.12 Album.AlbumSongCount Property**C#**

```
[Column]  
public int AlbumSongCount;
```

Description

This is AlbumSongCount, a member of class Album.

1.1.2.3.1.13 Album.CreationDate Property

C#

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

Description

This is CreationDate, a member of class Album.

1.1.2.3.1.14 Album.ModificationDate Property

C#

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

Description

This is ModificationDate, a member of class Album.

1.1.2.4 AlbumArtist Class

Class Hierarchy



C#

```
[Table]
public class AlbumArtist;
```

File

AlbumArtist.cs (📄 see page 96)

Description

This is class AwesomeMusic.AlbumArtist.

AlbumArtist Properties

	Name	Description
📄	AlbumArtistId (📄 see page 30)	This is AlbumArtistId, a member of class AlbumArtist.
📄	AlbumId (📄 see page 31)	This is AlbumId, a member of class AlbumArtist.
📄	ArtistId (📄 see page 31)	This is ArtistId, a member of class AlbumArtist.

1.1.2.4.1 AlbumArtist Properties

1.1.2.4.1.1 AlbumArtist.AlbumArtistId Property

C#

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

Description

This is AlbumArtistId, a member of class AlbumArtist.

1.1.2.4.1.2 AlbumArtist.AlbumId Property**C#**

```
[Column]
public int AlbumId;
```

Description

This is AlbumId, a member of class AlbumArtist.

1.1.2.4.1.3 AlbumArtist.ArtistId Property**C#**

```
[Column]
public int ArtistId;
```

Description

This is ArtistId, a member of class AlbumArtist.

1.1.2.5 AlbumPage Class**Class Hierarchy****C#**

```
public class AlbumPage : PhoneApplicationPage;
```

File

AlbumPage.xaml.cs (see page 96)

Description

This is class AwesomeMusic.AlbumPage.

Methods

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

AlbumPage Fields

	Name	Description
	albumId (see page 33)	This is albumId, a member of class AlbumPage.
	artistId (see page 33)	This is artistId, a member of class AlbumPage.
	artistName (see page 33)	This is artistName, a member of class AlbumPage.
	categoryId (see page 33)	This is categoryId, a member of class AlbumPage.
	categoryName (see page 33)	This is categoryName, a member of class AlbumPage.
	flag (see page 33)	This is flag, a member of class AlbumPage.
	isFilled (see page 34)	This is isFilled, a member of class AlbumPage.
	pageName (see page 34)	This is pageName, a member of class AlbumPage.
	ratingValue (see page 34)	This is ratingValue, a member of class AlbumPage.

AlbumPage Methods

	Name	Description
🔗	OnFragmentNavigation (🔗 see page 34)	This is OnFragmentNavigation, a member of class AlbumPage.
🔗	OnNavigatedFrom (🔗 see page 35)	This is OnNavigatedFrom, a member of class AlbumPage.
🔗	OnNavigatedTo (🔗 see page 35)	This is OnNavigatedTo, a member of class AlbumPage.

1.1.2.5.1 AlbumPage.AlbumPage Constructor**C#**

```
public AlbumPage();
```

Description

This is AlbumPage, a member of class AlbumPage.

Body Source

```

1: public AlbumPage()
2: {
3:     InitializeComponent();
4:
5:     SetBackgroundColor();
6:
7:     //pvArtist.Title = artistName;
8:     piAlbumName.Header = AppResources.AlbumName;
9:     piComment.Header = AppResources.AlbumComment;
10:    piLabelName.Header = AppResources.LabelName;
11:    piRating.Header = AppResources.AlbumRating;
12:    piReleaseYear.Header = AppResources.ReleaseYear;
13:    piBestSong.Header = AppResources.BestSong;
14:    piSongCount.Header = AppResources.SongCount;
15:
16:
17:    ApplicationBar = new ApplicationBar();
18:
19:    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
20:    button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
21:    button1.Text = AppResources.Save;
22:    ApplicationBar.Buttons.Add(button1);
23:    button1.Click += new EventHandler(SaveButton_Click);
24:
25:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
26:    button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
27:    button2.Text = AppResources.SendWithMail;
28:    ApplicationBar.Buttons.Add(button2);
29:    button2.Click += new EventHandler(SendMailButton_Click);
30:
31:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
32:    button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
33:    button3.Text = AppResources.SendWithSMS;
34:    ApplicationBar.Buttons.Add(button3);
35:    button3.Click += new EventHandler(SendSMSButton_Click);
36:
37:    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
38:    button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
39:    button4.Text = AppResources.ShareAlbum;
40:    ApplicationBar.Buttons.Add(button4);
41:    button4.Click += new EventHandler(ShareAlbumButton_Click);
42:
43:    isFilled = false;
44:
45:    ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
46:    menuItem1.Text = AppResources.DeleteAlbum;
47:    ApplicationBar.MenuItems.Add(menuItem1);

```

```
48:     menuItem1.Click += new EventHandler(DeleteAlbumMenuItem_Click);
49:
50: }
```

1.1.2.5.2 AlbumPage Fields

1.1.2.5.2.1 AlbumPage.albumId Field

C#

```
public int albumId;
```

Description

This is albumId, a member of class AlbumPage.

1.1.2.5.2.2 AlbumPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class AlbumPage.

1.1.2.5.2.3 AlbumPage.artistName Field

C#

```
public string artistName;
```

Description

This is artistName, a member of class AlbumPage.

1.1.2.5.2.4 AlbumPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class AlbumPage.

1.1.2.5.2.5 AlbumPage.categoryName Field

C#

```
public string categoryName;
```

Description

This is categoryName, a member of class AlbumPage.

1.1.2.5.2.6 AlbumPage.flag Field

C#

```
public bool flag;
```

Description

This is flag, a member of class AlbumPage.

1.1.2.5.2.7 AlbumPage.isFilled Field

C#

```
public bool isFilled;
```

Description

This is isFilled, a member of class AlbumPage.

1.1.2.5.2.8 AlbumPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class AlbumPage.

1.1.2.5.2.9 AlbumPage.ratingValue Field

C#

```
public double ratingValue = 0;
```

Description

This is ratingValue, a member of class AlbumPage.

1.1.2.5.3 AlbumPage Methods

1.1.2.5.3.1 AlbumPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class AlbumPage.

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:     albumId = int.Parse(e.Fragment);
7:     if (pageName.Contains("/ArtistPage.xaml"))
8:     {
9:         isFilled = true;
10:    }
11:    else
12:    {
13:        //using (var context2 = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
14:        //{
15:            //    var appSettings = context2.AppSettings; ;
16:            //    var album2 = context2.Albums.Where(j => j.AlbumId.Equals(albumId)) as
Album;
17:            //    var albumArtist = context2.AlbumArtists.Where(j =>
j.AlbumId.Equals(albumId)).ToList() as List<AlbumArtist>;
18:            //    var bArtist = albumArtist.First();
19:            //    var artist = context2.Artists.Where(j =>
j.ArtistId.Equals(bArtist.ArtistId)) as Artist;
```

```

20:         //      foreach (var item in appSettings)
21:         //      {
22:         //          item.CurrentArtistNumber = artist.ArtistId;
23:         //          item.CurrentCategoryNumber = album2.AlbumCategoryId;
24:         //      }
25:         //      context2.SubmitChanges();
26:         //      pvArtist.Title = artist.ArtistName;
27:         //    }
28:     }
29:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
30:     {
31:         var album = context.Albums.Where(j => j.AlbumId.Equals(e.Fragment)).Single() as
Album;
32:
33:         txtAlbumName.Text = album.AlbumName == "" ? "" : album.AlbumName;
34:         txtSongCount.Text = album.AlbumSongCount.ToString() == "" ? "" :
album.AlbumSongCount.ToString();
35:         txtReleaseYear.Text = album.AlbumReleaseYear.ToString() == "" ? "" :
album.AlbumReleaseYear.ToString();
36:         txtLabelName.Text = album.AlbumLabelName == "" ? "" : album.AlbumLabelName;
37:         txtBestSong.Text = album.AlbumBestSong == "" ? "" : album.AlbumBestSong;
38:         //dtStart.Value = album.ReadStartDate == null ? DateTime.Now :
album.ReadStartDate;
39:         //dtFinish.Value = album.ReadFinishDate == null ? DateTime.Now :
album.ReadFinishDate;
40:         rtRating.Value = album.AlbumRating == null ? 0 : album.AlbumRating;
41:         txtAlbumComment.Text = album.AlbumComment == "" ? "" : album.AlbumComment;
42:     }
43:
44:     SetBackgroundColor();
45:     pvArtist.SelectedIndex = 0;
46:     //pvArtist.Name = artistName;
47:     txtAlbumName.Focus();
48: }

```

1.1.2.5.3.2 AlbumPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class AlbumPage.

Body Source

```

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

```

1.1.2.5.3.3 AlbumPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class AlbumPage.

Body Source

```

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

```



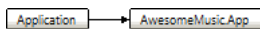
```

4:      using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
5:      {
6:          var appSettings = context.AppSettings.First();
7:          categoryId = appSettings.CurrentCategoryNumber;
8:          artistId = appSettings.CurrentArtistNumber;
9:
10:         // sayfanin font ayarlari için yapılan bir degisiklik
11:         FontFamily temp = new FontFamily(appSettings.FontFamily);
12:         double fontsize = double.Parse(appSettings.FontSize);
13:         txtAlbumComment.FontFamily = temp;
14:         txtAlbumComment.FontSize = fontsize;
15:         txtReleaseYear.FontFamily = temp;
16:         txtReleaseYear.FontSize = fontsize;
17:         txtAlbumName.FontFamily = temp;
18:         txtAlbumName.FontSize = fontsize;
19:         txtAlbumComment.FontFamily = temp;
20:         txtAlbumComment.FontSize = fontsize;
21:         txtLabelName.FontFamily = temp;
22:         txtLabelName.FontSize = fontsize;
23:         txtBestSong.FontFamily = temp;
24:         txtBestSong.FontSize = fontsize;
25:         txtSongCount.FontFamily = temp;
26:         txtSongCount.FontSize = fontsize;
27:         // oylamada kolaylik olmasi için otomatik olarak 5 veriliyor
28:         // sonradan istenirse 0 da verilebilir.
29:         rtRating.Value = 5;
30:
31:         var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single()
as Artist;
32:         artistName = artist.ArtistName;
33:
34:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
35:         categoryName = category.CategoryName;
36:     }
37:
38:     var lastPage = NavigationService.BackStack.FirstOrDefault();
39:     pageName = lastPage.Source.ToString();
40:     pvArtist.SelectedIndex = 0;
41:     txtAlbumName.Focus();
42:     // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
43:     pvArtist.Title = artistName;
44:     SetBackgroundColor();
45: }

```

1.1.2.6 App Class

Class Hierarchy



C#

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

File

App.xaml.cs (see page 107)

Description

This is class AwesomeMusic.App.



Methods

	Name	Description
	App (see page 37)	Constructor for the Application object.

App Fields

	Name	Description
	categoryNumber (see page 38)	This is categoryNumber, a member of class App.

App Properties

	Name	Description
	IsTrial (see page 38)	This is IsTrial, a member of class App.
	RootFrame (see page 38)	This is RootFrame, a member of class App.

1.1.2.6.1 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:     // ayarlardan temasi açık renk bile olsa
10:    // kapali gibi çalışmasını sağlayacak bir nuget paketi yüklendi
11:    // bu sorunu gideriyor
12:    ThemeManager.ToDarkTheme();
13:
14:    // Phone-specific initialization
15:    InitializePhoneApplication();
16:
17:    // Language display initialization
18:    InitializeLanguage();
19:
20:    // Show graphics profiling information while debugging.
21:    if (Debugger.IsAttached)
22:    {
23:        // Display the current frame rate counters.
24:        Application.Current.Host.Settings.EnableFrameRateCounter = true;
25:
26:        // Show the areas of the app that are being redrawn in each frame.
27:        //Application.Current.Host.Settings.EnableRedrawRegions = true;
28:
29:        // Enable non-production analysis visualization mode,
30:        // which shows areas of a page that are handed off to GPU with a colored
31:        // overlay.
32:        //Application.Current.Host.Settings.EnableCacheVisualization = true;
33:
34:        // Prevent the screen from turning off while under the debugger by disabling
35:        // the application's idle detection.
36:        // Caution:- Use this under debug mode only. Application that disables user
37:        // idle detection will continue to run
38:        // and consume battery power when the user is not using the phone.
39:        PhoneApplicationService.Current.UserIdleDetectionMode =
40:        IdleDetectionMode.Disabled;
41:    }
42: }
```

1.1.2.6.2 App Fields

1.1.2.6.2.1 App.categoryNumber Field

C#

```
public int categoryNumber;
```

Description

This is categoryNumber, a member of class App.

1.1.2.6.3 App Properties

1.1.2.6.3.1 App.IsTrial Property

C#

```
public bool IsTrial;
```

Description

This is IsTrial, a member of class App.

1.1.2.6.3.2 App.RootFrame Property

C#

```
public static PhoneApplicationFrame RootFrame;
```

Description

This is RootFrame, a member of class App.

1.1.2.7 AppSettings Class

Class Hierarchy

AwesomeMusic.AppSettings

C#

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






File






AppSettings.cs ([see page 134](#))

Description

This is class AwesomeMusic.AppSettings.

AppSettings Properties

	Name	Description
	AppBackgroundColor (see page 39)	This is AppBackgroundColor, a member of class AppSettings.
	AppBackgroundImage (see page 39)	This is AppBackgroundImage, a member of class AppSettings.
	AppLangName (see page 39)	This is AppLangName, a member of class AppSettings.
	AppSettingsId (see page 39)	This is AppSettingsId, a member of class AppSettings.
	CategoryOrderBy (see page 39)	This is CategoryOrderBy, a member of class AppSettings.

	CategoryOrderStyle (🔗 see page 40)	This is CategoryOrderStyle, a member of class AppSettings.
	CurrentArtistNumber (🔗 see page 40)	This is CurrentArtistNumber, a member of class AppSettings.
	CurrentCategoryNumber (🔗 see page 40)	This is CurrentCategoryNumber, a member of class AppSettings.
	FontFamily (🔗 see page 40)	This is FontFamily, a member of class AppSettings.
	FontSize (🔗 see page 40)	This is FontSize, a member of class AppSettings.

1.1.2.7.1 AppSettings Properties

1.1.2.7.1.1 AppSettings.AppBackgroundColor Property

C#

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

Description

This is AppBackgroundColor, a member of class AppSettings.

1.1.2.7.1.2 AppSettings.AppBackgroundImage Property

C#

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

Description

This is AppBackgroundImage, a member of class AppSettings.

1.1.2.7.1.3 AppSettings.AppLangName Property

C#

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

Description

This is AppLangName, a member of class AppSettings.

1.1.2.7.1.4 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.

1.1.2.7.1.5 AppSettings.CategoryOrderBy Property

C#

```
[Column]
public string CategoryOrderBy;
```

Description

This is CategoryOrderBy, a member of class AppSettings.

1.1.2.7.1.6 AppSettings.CategoryOrderStyle Property**C#**

```
[Column]
public string CategoryOrderStyle;
```

Description

This is CategoryOrderStyle, a member of class AppSettings.

1.1.2.7.1.7 AppSettings.CurrentArtistNumber Property**C#**

```
[Column]
public int CurrentArtistNumber;
```

Description

This is CurrentArtistNumber, a member of class AppSettings.

1.1.2.7.1.8 AppSettings.CurrentCategoryNumber Property**C#**

```
[Column]
public int CurrentCategoryNumber;
```

Description

This is CurrentCategoryNumber, a member of class AppSettings.

1.1.2.7.1.9 AppSettings.FontFamily Property**C#**

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

Description

This is FontFamily, a member of class AppSettings.

1.1.2.7.1.10 AppSettings.FontSize Property**C#**

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

Description

This is FontSize, a member of class AppSettings.

1.1.2.8 Artist Class**Class Hierarchy**

AwesomeMusicArtist

C#

```
[Table]
public class Artist;
```









File

Artist.cs ([see page 135](#))

Description

This is class AwesomeMusic.Artist.

Artist Properties

	Name	Description
	AlbumOrderBy (see page 41)	This is AlbumOrderBy, a member of class Artist.
	AlbumOrderStyle (see page 41)	This is AlbumOrderStyle, a member of class Artist.
	ArtistAlbumCount (see page 41)	This is ArtistAlbumCount, a member of class Artist.
	ArtistId (see page 42)	This is ArtistId, a member of class Artist.
	ArtistName (see page 42)	This is ArtistName, a member of class Artist.
	ArtistNameCount (see page 42)	This is ArtistNameCount, a member of class Artist.
	CreationDate (see page 42)	This is CreationDate, a member of class Artist.
	ModificationDate (see page 42)	This is ModificationDate, a member of class Artist.

1.1.2.8.1 Artist Properties

1.1.2.8.1.1 Artist.AlbumOrderBy Property

C#

```
[Column]
public string AlbumOrderBy;
```

Description

This is AlbumOrderBy, a member of class Artist.

1.1.2.8.1.2 Artist.AlbumOrderStyle Property

C#

```
[Column]
public string AlbumOrderStyle;
```

Description

This is AlbumOrderStyle, a member of class Artist.

1.1.2.8.1.3 Artist.ArtistAlbumCount Property

C#

```
[Column]
public int ArtistAlbumCount;
```

Description

This is ArtistAlbumCount, a member of class Artist.

1.1.2.8.1.4 Artist.ArtistId Property

C#

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

Description

This is ArtistId, a member of class Artist.

1.1.2.8.1.5 Artist.ArtistName Property

C#

```
[Column]  
public string ArtistName;
```

Description

This is ArtistName, a member of class Artist.

1.1.2.8.1.6 Artist.ArtistNameCount Property

C#

```
[Column]  
public string ArtistNameCount;
```

Description

This is ArtistNameCount, a member of class Artist.

1.1.2.8.1.7 Artist.CreationDate Property

C#

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

Description

This is CreationDate, a member of class Artist.

1.1.2.8.1.8 Artist.ModificationDate Property

C#

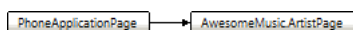
```
[Column]  
public DateTime ModificationDate;
```

Description

This is ModificationDate, a member of class Artist.

1.1.2.9 ArtistPage Class

Class Hierarchy



C#

```
public class ArtistPage : PhoneApplicationPage;
```

File

ArtistPage.xaml.cs (🔗 see page 136)

Description

This is class AwesomeMusic.ArtistPage.

Methods

	Name	Description
🔗	ArtistPage (🔗 see page 43)	This is ArtistPage, a member of class ArtistPage.

ArtistPage Fields

	Name	Description
🔗	albumId (🔗 see page 44)	This is albumId, a member of class ArtistPage.
🔗	artistId (🔗 see page 44)	This is artistId, a member of class ArtistPage.
🔗	categoryId (🔗 see page 44)	This is categoryId, a member of class ArtistPage.
🔗	oldArtistName (🔗 see page 44)	This is oldArtistName, a member of class ArtistPage.
🔗	popup (🔗 see page 44)	This is popup, a member of class ArtistPage.

ArtistPage Methods

	Name	Description
🔗	OnFragmentNavigation (🔗 see page 45)	This is OnFragmentNavigation, a member of class ArtistPage.
🔗	OnNavigatedFrom (🔗 see page 47)	This is OnNavigatedFrom, a member of class ArtistPage.
🔗	OnNavigatedTo (🔗 see page 47)	This is OnNavigatedTo, a member of class ArtistPage.

1.1.2.9.1 ArtistPage.ArtistPage Constructor

C#

```
public ArtistPage();
```

Description

This is ArtistPage, a member of class ArtistPage.

Body Source

```
1: public ArtistPage()
2: {
3:     InitializeComponent();
4:
5:     ApplicationBar = new ApplicationBar();
6:
7:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();
8:     button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
9:     button1.Text = AppResources.AddAlbum;
10:    ApplicationBar.Buttons.Add(button1);
11:    button1.Click += new EventHandler(AddAlbumButton_Click);
12:
13:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
14:    button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
15:    button2.Text = AppResources.DeleteArtist;
16:    ApplicationBar.Buttons.Add(button2);
17:    button2.Click += new EventHandler(DeleteArtistButton_Click);
18:
19:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
20:    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
21:    button3.Text = AppResources.ArtistSettings;
22:    ApplicationBar.Buttons.Add(button3);
23:    button3.Click += new EventHandler(ArtistSettingsButton_Click);
```



```
24:
25:     ApplicationBarIconButton button4 = new ApplicationBarIconButton();
26:     button4.IconUri = new Uri("/Assets/AddCategory.png", UriKind.Relative);
27:     button4.Text = AppResources.AddCategory;
28:     ApplicationBar.Buttons.Add(button4);
29:     button4.Click += new EventHandler(AddCategoryButton_Click);
30:
31:     popup = new Popup();
32:
33: }
```

1.1.2.9.2 ArtistPage Fields

1.1.2.9.2.1 ArtistPage.albumId Field

C#

```
public int albumId;
```

Description

This is albumId, a member of class ArtistPage.

1.1.2.9.2.2 ArtistPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class ArtistPage.

1.1.2.9.2.3 ArtistPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class ArtistPage.

1.1.2.9.2.4 ArtistPage.oldArtistName Field

C#

```
public string oldArtistName;
```

Description

This is oldArtistName, a member of class ArtistPage.

1.1.2.9.2.5 ArtistPage.popup Field

C#

```
public Popup popup;
```

Description

This is popup, a member of class ArtistPage.

1.1.2.9.3 ArtistPage Methods

1.1.2.9.3.1 ArtistPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class ArtistPage.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     List<Album> albums = new List<Album>();
4:     List<Album> albumsOrdered = new List<Album>();
5:
6:     // displays "Fragment: Detail"
7:     //MessageBox.Show("Folder Id: " + e.Fragment);
8:     base.OnFragmentNavigation(e);
9:
10:    lstAlbums.Items.Clear();
11:
12:    using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
13:    {
14:
15:        var appSettings = context.AppSettings.First();
16:        categoryId = appSettings.CurrentCategoryNumber;
17:
18:        var artist = context.Artists.Where(j =>
j.ArtistId.Equals(e.Fragment)).Single() as Artist;
19:        artistId = artist.ArtistId;
20:
21:        var appSettings2 = context.AppSettings;
22:        foreach (var item in appSettings2)
23:        {
24:            item.CurrentArtistNumber = artistId;
25:        }
26:        context.SubmitChanges();
27:
28:        var artistAlbums = context.AlbumArtists.Where(j =>
j.ArtistId.Equals(e.Fragment)).ToList() as List<AlbumArtist>;
29:        if (artistAlbums.Count != 0)
30:        {
31:            foreach (var item in artistAlbums)
32:            {
33:                try
34:                {
35:                    var album = context.Albums.Where(j =>
j.AlbumCategoryId.Equals(categoryId) && j.AlbumId.Equals(item.AlbumId)).Single() as Album;
36:                    albums.Add(album);
37:                }
38:                catch (Exception)
39:                {
40:                }
41:            }
42:
43:        }
44:
45:
46:        string orderStyle = artist.AlbumOrderStyle;
47:
48:        switch (artist.AlbumOrderBy)
49:        {
50:            case "NAME":
51:                if (orderStyle == "A")
52:                {
53:                    albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
```

```

54:         }
55:         else
56:         {
57:             albumsOrdered = albums.OrderByDescending(j =>
j.AlbumName).ToList();
58:         }
59:         break;
60:         case "CDATE":
61:             if (orderStyle == "A")
62:             {
63:                 albumsOrdered = albums.OrderBy(j => j.CreationDate).ToList();
64:             }
65:             else
66:             {
67:                 albumsOrdered = albums.OrderByDescending(j =>
j.CreationDate).ToList();
68:             }
69:             break;
70:         case "MDATE":
71:             if (orderStyle == "A")
72:             {
73:                 albumsOrdered = albums.OrderBy(j => j.ModificationDate).ToList();
74:             }
75:             else
76:             {
77:                 albumsOrdered = albums.OrderByDescending(j =>
j.ModificationDate).ToList();
78:             }
79:             break;
80:         case "RATING":
81:             if (orderStyle == "A")
82:             {
83:                 albumsOrdered = albums.OrderBy(j => j.AlbumRating).ToList();
84:             }
85:             else
86:             {
87:                 albumsOrdered = albums.OrderByDescending(j =>
j.AlbumRating).ToList();
88:             }
89:             break;
90:             //case "SDATE":
91:             //    if (orderStyle == "A")
92:             //    {
93:             //        albumsOrdered = albums.OrderBy(j => j.ReadStartDate).ToList();
94:             //    }
95:             //    else
96:             //    {
97:             //        albumsOrdered = albums.OrderByDescending(j =>
j.ReadStartDate).ToList();
98:             //    }
99:             //    break;
100:            //case "FDATE":
101:            //    if (orderStyle == "A")
102:            //    {
103:            //        albumsOrdered = albums.OrderBy(j => j.ReadFinishDate).ToList();
104:            //    }
105:            //    else
106:            //    {
107:            //        albumsOrdered = albums.OrderByDescending(j =>
j.ReadFinishDate).ToList();
108:            //    }
109:            //    break;
110:            default:
111:                if (orderStyle == "A")
112:                {
113:                    albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
114:                }
115:                else
116:                {

```

```

117:             albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
118:         }
119:         break;
120:     }
121:
122:     lblArtistName.Text = artist.ArtistName;
123:     lblAlbumList.Text = AppResources.AlbumList + " (" + artist.ArtistName + ")";
124:     lstAlbums.ItemsSource = albumsOrdered;
125:     lstAlbums.DisplayMemberPath = "AlbumNameRating";
126:     SetBackgroundColor();
127:     //lstNoteList.DisplayMemberPath = "NameCreation";
128: }
129: }

```

1.1.2.9.3.2 ArtistPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class ArtistPage.

Body Source

```

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

```

1.1.2.9.3.3 ArtistPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class ArtistPage.

Body Source

```

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

```

1.1.2.10 ArtistSettingsPage Class

Class Hierarchy



C#

```
public class ArtistSettingsPage : PhoneApplicationPage;
```


File

ArtistSettingsPage.xaml.cs (see page 144)



Description

This is class AwesomeMusic.ArtistSettingsPage.




Methods

	Name	Description
	ArtistSettingsPage (see page 48)	This is ArtistSettingsPage, a member of class ArtistSettingsPage.

ArtistSettingsPage Fields

	Name	Description
	artistId (see page 49)	This is artistId, a member of class ArtistSettingsPage.
	categoryId (see page 49)	This is categoryId, a member of class ArtistSettingsPage.

ArtistSettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 49)	This is OnFragmentNavigation, a member of class ArtistSettingsPage.
	OnNavigatedFrom (see page 50)	This is OnNavigatedFrom, a member of class ArtistSettingsPage.
	OnNavigatedTo (see page 50)	This is OnNavigatedTo, a member of class ArtistSettingsPage.

1.1.2.10.1 ArtistSettingsPage.ArtistSettingsPage Constructor**C#**

```
public ArtistSettingsPage();
```

Description

This is ArtistSettingsPage, a member of class ArtistSettingsPage.

Body Source

```
1: public ArtistSettingsPage()
2: {
3:     InitializeComponent();
4:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
5:     {
6:         var appSettings = context.AppSettings.First();
7:         lblFontFamily.Text = AppResources.FontFamily + " (" + AppResources.Selected +
": " + appSettings.FontFamily + ")";
8:         lblFontSize.Text = AppResources.FontSize + " (" + AppResources.Selected + ": "
+ appSettings.FontSize + ")";
9:     }
10:
11:     pvArtistSettings.Title = AppResources.ArtistSettings;
12:     piFont.Header = AppResources.Font;
13:     piOtherSettings.Header = AppResources.OtherSettings;
14:
15:     btnFontFamily.Content = AppResources.Select;
16:     btnFontSize.Content = AppResources.Select;
17:     btnAlbumOrder.Content = AppResources.Select;
18:     btnAlbumOrderStyle.Content = AppResources.Select;
19: }
```

1.1.2.10.2 ArtistSettingsPage Fields

1.1.2.10.2.1 ArtistSettingsPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class ArtistSettingsPage.

1.1.2.10.2.2 ArtistSettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class ArtistSettingsPage.

1.1.2.10.3 ArtistSettingsPage Methods

1.1.2.10.3.1 ArtistSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class ArtistSettingsPage.

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:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
7:     {
8:         var artist = context.Artists.Where(j => j.ArtistId.Equals(e.Fragment)).Single()
as Artist;
9:         artistId = artist.ArtistId;
10:        var appSettings = context.AppSettings.First();
11:        categoryId = appSettings.CurrentCategoryNumber;
12:        string orderStyle = artist.AlbumOrderStyle;
13:
14:        if (artist.AlbumOrderBy == "NAME")
15:        {
16:            lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
17:        }
18:        if (artist.AlbumOrderBy == "CDATE")
19:        {
20:            lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
21:        }
22:        if (artist.AlbumOrderBy == "MDATE")
23:        {
24:            lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
25:        }
26:        if (artist.AlbumOrderBy == "RATING")
27:        {
28:            lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
```

```

AppResources.Selected + ": " + AppResources.AlbumRating + "));
29:     }
30:     //if (artist.AlbumOrderBy == "SDATE")
31:     //{
32:     //    lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.StartDate + "));
33:     //}
34:     //if (artist.AlbumOrderBy == "FDATE")
35:     //{
36:     //    lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.FinishDate + "));
37:     //}
38:     if (artist.AlbumOrderStyle == "A")
39:     {
40:         lblAlbumOrderStyle.Text = AppResources.AlbumOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + "));
41:     }
42:     if (artist.AlbumOrderStyle == "D")
43:     {
44:         lblAlbumOrderStyle.Text = AppResources.AlbumOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + "));
45:     }
46:     //lstNoteList.DisplayMemberPath = "NameCreation";
47:     SetBackgroundColor();
48: }
49: }

```

1.1.2.10.3.2 ArtistSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class ArtistSettingsPage.

Body Source

```

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

```

1.1.2.10.3.3 ArtistSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class ArtistSettingsPage.

Body Source

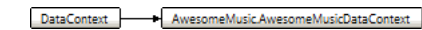
```

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

```

1.1.2.11 AwesomeMusicDataContext Class

Class Hierarchy



C#

```
public class AwesomeMusicDataContext : DataContext;
```

File

AwesomeMusicDataContext.cs (see page 148)

Description

This is class AwesomeMusicDataContext.

Methods

	Name	Description
	AwesomeMusicDataContext (see page 51)	This is AwesomeMusicDataContext, a member of class AwesomeMusicDataContext.

AwesomeMusicDataContext Fields

	Name	Description
	AlbumArtists (see page 51)	This is AlbumArtists, a member of class AwesomeMusicDataContext.
	Albums (see page 52)	This is Albums, a member of class AwesomeMusicDataContext.
	AppSettings (see page 52)	This is AppSettings, a member of class AwesomeMusicDataContext.
	Artists (see page 52)	This is Artists, a member of class AwesomeMusicDataContext.
	Categories (see page 52)	This is Categories, a member of class AwesomeMusicDataContext.
	CategoryArtists (see page 52)	This is CategoryArtists, a member of class AwesomeMusicDataContext.
	ConnectionString (see page 52)	This is ConnectionString, a member of class AwesomeMusicDataContext.

1.1.2.11.1 AwesomeMusicDataContext.AwesomeMusicDataContext Constructor

C#

```
public AwesomeMusicDataContext(string connectionString);
```

Description

This is AwesomeMusicDataContext, a member of class AwesomeMusicDataContext.

Body Source

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

1.1.2.11.2 AwesomeMusicDataContext Fields

1.1.2.11.2.1 AwesomeMusicDataContext.AlbumArtists Field

C#

```
public Table<AlbumArtist> AlbumArtists;
```

Description

This is AlbumArtists, a member of class AwesomeMusicDataContext.

1.1.2.11.2.2 AwesomeMusicDataContext.Albums Field

C#

```
public Table<Album> Albums;
```

Description

This is Albums, a member of class AwesomeMusicDataContext.

1.1.2.11.2.3 AwesomeMusicDataContext.AppSettings Field

C#

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

Description

This is AppSettings, a member of class AwesomeMusicDataContext.

1.1.2.11.2.4 AwesomeMusicDataContext.Artists Field

C#

```
public Table<Artist> Artists;
```

Description

This is Artists, a member of class AwesomeMusicDataContext.

1.1.2.11.2.5 AwesomeMusicDataContext.Categories Field

C#

```
public Table<Category> Categories;
```

Description

This is Categories, a member of class AwesomeMusicDataContext.

1.1.2.11.2.6 AwesomeMusicDataContext.CategoryArtists Field

C#

```
public Table<CategoryArtist> CategoryArtists;
```

Description

This is CategoryArtists, a member of class AwesomeMusicDataContext.

1.1.2.11.2.7 AwesomeMusicDataContext.ConnectionString Field

C#

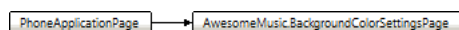
```
public const string ConnectionString = @"Data Source=isostore:/MyMusicLibrary.sdf";
```

Description

This is ConnectionString, a member of class AwesomeMusicDataContext.

1.1.2.12 BackgroundColorSettingsPage Class

Class Hierarchy



C#

```
public class BackgroundColorSettingsPage : PhoneApplicationPage;
```


File

BackgroundColorSettingsPage.xaml.cs ([🔗](#) see page 149)


Description

This is class AwesomeMusic.BackgroundColorSettingsPage.




Methods

	Name	Description
	BackgroundColorSettingsPage (🔗 see page 53)	This is BackgroundColorSettingsPage, a member of class BackgroundColorSettingsPage.

BackgroundColorSettingsPage Fields

	Name	Description
	artistId (🔗 see page 54)	This is artistId, a member of class BackgroundColorSettingsPage.

BackgroundColorSettingsPage Methods

	Name	Description
	OnFragmentNavigation (🔗 see page 54)	This is OnFragmentNavigation, a member of class BackgroundColorSettingsPage.
	OnNavigatedFrom (🔗 see page 54)	This is OnNavigatedFrom, a member of class BackgroundColorSettingsPage.
	OnNavigatedTo (🔗 see page 55)	This is OnNavigatedTo, a member of class BackgroundColorSettingsPage.

1.1.2.12.1 BackgroundColorSettingsPage.BackgroundColorSettingsPage Constructor

C#

```
public BackgroundColorSettingsPage();
```

Description

This is BackgroundColorSettingsPage, a member of class BackgroundColorSettingsPage.

Body Source

```
1: public BackgroundColorSettingsPage()
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:    lblGeneralSettings.Text = AppResources.GeneralSettings;
19:
20:    SetBackgroundColor();
```

```
21: }
```

1.1.2.12.2 BackgroundColorSettingsPage Fields

1.1.2.12.2.1 BackgroundColorSettingsPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class BackgroundColorSettingsPage.

1.1.2.12.3 BackgroundColorSettingsPage Methods

1.1.2.12.3.1 BackgroundColorSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class BackgroundColorSettingsPage.

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:     artistId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
8:     {
9:         var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single()
as Artist;
10:         lblGeneralSettings.Text = AppResources.GeneralSettings;
11:         lblBackgroundColor.Text = AppResources.SelectFontSize;
12:     }
13: }
```

1.1.2.12.3.2 BackgroundColorSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class BackgroundColorSettingsPage.

Body Source

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

1.1.2.12.3.3 BackgroundColorSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class BackgroundColorSettingsPage.

Body Source

```
1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     //SetBackgroundColor();
5:     //while (NavigationService.CanGoBack)
6:     //NavigationService.RemoveBackEntry();
7:
8: }
```

1.1.2.13 Category Class

Class Hierarchy

AwesomeMusic.Category

C#

```
[Table]
public class Category;
```









File

Category.cs ([see page 152](#))

Description

This is class AwesomeMusic.Category.

Category Properties

	Name	Description
	ArtistOrderBy (see page 55)	This is ArtistOrderBy, a member of class Category.
	ArtistOrderStyle (see page 56)	This is ArtistOrderStyle, a member of class Category.
	CategoryAlbumCount (see page 56)	This is CategoryAlbumCount, a member of class Category.
	CategoryId (see page 56)	This is CategoryId, a member of class Category.
	CategoryName (see page 56)	This is CategoryName, a member of class Category.
	CategoryNameCount (see page 56)	This is CategoryNameCount, a member of class Category.
	CreationDate (see page 56)	This is CreationDate, a member of class Category.
	ModificationDate (see page 57)	This is ModificationDate, a member of class Category.

1.1.2.13.1 Category Properties

1.1.2.13.1.1 Category.ArtistOrderBy Property

C#

```
[Column]
public string ArtistOrderBy;
```

Description

This is ArtistOrderBy, a member of class Category.

1.1.2.13.1.2 Category.ArtistOrderStyle Property**C#**

```
[Column]
public string ArtistOrderStyle;
```

Description

This is ArtistOrderStyle, a member of class Category.

1.1.2.13.1.3 Category.CategoryAlbumCount Property**C#**

```
[Column]
public int CategoryAlbumCount;
```

Description

This is CategoryAlbumCount, a member of class Category.

1.1.2.13.1.4 Category.CategoryId Property**C#**

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

Description

This is CategoryId, a member of class Category.

1.1.2.13.1.5 Category.CategoryName Property**C#**

```
[Column]
public string CategoryName;
```

Description

This is CategoryName, a member of class Category.

1.1.2.13.1.6 Category.CategoryNameCount Property**C#**

```
[Column]
public string CategoryNameCount;
```

Description

This is CategoryNameCount, a member of class Category.

1.1.2.13.1.7 Category.CreationDate Property**C#**

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

Description

This is CreationDate, a member of class Category.

1.1.2.13.1.8 Category.ModificationDate Property

C#

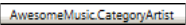
```
[Column]
public DateTime ModificationDate;
```

Description

This is ModificationDate, a member of class Category.

1.1.2.14 CategoryArtist Class

Class Hierarchy



C#

```
[Table]
public class CategoryArtist;
```

File

CategoryArtist.cs (📄 see page 153)

Description

This is class AwesomeMusic.CategoryArtist.

CategoryArtist Properties

	Name	Description
📄	ArtistId (📄 see page 57)	This is ArtistId, a member of class CategoryArtist.
📄	CategoryArtistId (📄 see page 57)	This is CategoryArtistId, a member of class CategoryArtist.
📄	CategoryId (📄 see page 58)	This is CategoryId, a member of class CategoryArtist.

1.1.2.14.1 CategoryArtist Properties

1.1.2.14.1.1 CategoryArtist.ArtistId Property

C#

```
[Column]
public int ArtistId;
```

Description

This is ArtistId, a member of class CategoryArtist.

1.1.2.14.1.2 CategoryArtist.CategoryArtistId Property

C#

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

Description

This is CategoryArtistId, a member of class CategoryArtist.

1.1.2.14.1.3 CategoryArtist.CategoryId Property

C#

```
[Column]
public int CategoryId;
```

Description

This is CategoryId, a member of class CategoryArtist.

1.1.2.15 CategoryPage Class

Class Hierarchy



C#

```
public class CategoryPage : PhoneApplicationPage;
```

File

CategoryPage.xaml.cs (🔗 see page 154)

Description

This is class AwesomeMusic.CategoryPage.

Methods

	Name	Description
🔗	CategoryPage (🔗 see page 58)	This is CategoryPage, a member of class CategoryPage.

CategoryPage Fields

	Name	Description
🔗	categoryId (🔗 see page 59)	This is categoryId, a member of class CategoryPage.
🔗	oldCategoryName (🔗 see page 59)	This is oldCategoryName, a member of class CategoryPage.
🔗	popup (🔗 see page 59)	This is popup, a member of class CategoryPage.

CategoryPage Methods

	Name	Description
🔗🔗	OnFragmentNavigation (🔗 see page 60)	This is OnFragmentNavigation, a member of class CategoryPage.
🔗🔗	OnNavigatedFrom (🔗 see page 61)	This is OnNavigatedFrom, a member of class CategoryPage.
🔗🔗	OnNavigatedTo (🔗 see page 61)	This is OnNavigatedTo, a member of class CategoryPage.

1.1.2.15.1 CategoryPage.CategoryPage Constructor

C#

```
public CategoryPage ( ) ;
```

Description

This is CategoryPage, a member of class CategoryPage.

Body Source

```
1: public CategoryPage()  
2: {  
3:     InitializeComponent();  
4:  
5:     ApplicationBar = new ApplicationBar();  
6:  
7:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();  
8:     button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);  
9:     button1.Text = AppResources.AddArtist;  
10:    ApplicationBar.Buttons.Add(button1);  
11:    button1.Click += new EventHandler(AddArtistButton_Click);  
12:  
13:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();  
14:    button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);  
15:    button2.Text = AppResources.DeleteCategory;  
16:    ApplicationBar.Buttons.Add(button2);  
17:    button2.Click += new EventHandler(DeleteCategoryButton_Click);  
18:  
19:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();  
20:    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);  
21:    button3.Text = AppResources.CategorySettings;  
22:    ApplicationBar.Buttons.Add(button3);  
23:    button3.Click += new EventHandler(CategorySettingsButton_Click);  
24:  
25:    SetBackgroundColor();  
26:    popup = new Popup();  
27: }
```

1.1.2.15.2 CategoryPage Fields**1.1.2.15.2.1 CategoryPage.categoryId Field****C#**

```
public int categoryId;
```

Description

This is categoryId, a member of class CategoryPage.

1.1.2.15.2.2 CategoryPage.oldCategoryName Field**C#**

```
public string oldCategoryName;
```

Description

This is oldCategoryName, a member of class CategoryPage.

1.1.2.15.2.3 CategoryPage.popup Field**C#**

```
public Popup popup;
```

Description

This is popup, a member of class CategoryPage.

1.1.2.15.3 CategoryPage Methods

1.1.2.15.3.1 CategoryPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class CategoryPage.

Body Source

```
1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     List<Artist> artists = new List<Artist>();
4:     List<Artist> artistsOrdered = new List<Artist>();
5:
6:     // displays "Fragment: Detail"
7:     //MessageBox.Show("Folder Id: " + e.Fragment);
8:     base.OnFragmentNavigation(e);
9:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
10:     {
11:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
12:         string orderStyle = category.ArtistOrderStyle;
13:         var categoryArtist = context.CategoryArtists.Where(j =>
j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryArtist>;
14:
15:         foreach (var item in categoryArtist)
16:         {
17:             try
18:             {
19:                 artists.Add(context.Artists.Where(j =>
j.ArtistId.Equals(item.ArtistId)).Single());
20:             }
21:             catch (Exception)
22:             {
23:             }
24:
25:         }
26:
27:         switch (category.ArtistOrderBy)
28:         {
29:             case "NAME":
30:                 if (orderStyle == "A")
31:                 {
32:                     artistsOrdered = artists.OrderBy(j => j.ArtistName).ToList();
33:                 }
34:                 else
35:                 {
36:                     artistsOrdered = artists.OrderByDescending(j =>
j.ArtistName).ToList();
37:                 }
38:                 break;
39:             case "ALBUMCOUNT":
40:                 if (orderStyle == "A")
41:                 {
42:                     artistsOrdered = artists.OrderBy(j => j.ArtistAlbumCount).ToList();
43:                 }
44:                 else
45:                 {
46:                     artistsOrdered = artists.OrderByDescending(j =>
j.ArtistAlbumCount).ToList();
47:                 }
48:                 break;
49:             case "CDATE":
50:                 if (orderStyle == "A")
51:                 {
```

```

52:             artistsOrdered = artists.OrderBy(j => j.CreationDate).ToList();
53:         }
54:         else
55:         {
56:             artistsOrdered = artists.OrderByDescending(j =>
j.CreationDate).ToList();
57:         }
58:         break;
59:         case "MDATE":
60:             if (orderStyle == "A")
61:             {
62:                 artistsOrdered = artists.OrderBy(j => j.ModificationDate).ToList();
63:             }
64:             else
65:             {
66:                 artistsOrdered = artists.OrderByDescending(j =>
j.ModificationDate).ToList();
67:             }
68:             break;
69:         default:
70:             if (orderStyle == "A")
71:             {
72:                 artistsOrdered = artists.OrderBy(j => j.ArtistName).ToList();
73:             }
74:             else
75:             {
76:                 artistsOrdered = artists.OrderByDescending(j =>
j.ArtistName).ToList();
77:             }
78:             break;
79:         }
80:
81:         lstArtists.Items.Clear();
82:         categoryId = category.CategoryId;
83:         lblCategoryName.Text = category.CategoryName;
84:         lblArtistList.Text = AppResources.ArtistList + " (" + category.CategoryName +
")";
85:         lstArtists.ItemsSource = artistsOrdered;
86:         lstArtists.DisplayMemberPath = "ArtistNameCount";
87:         SetBackgroundColor();
88:         //lstNoteList.DisplayMemberPath = "NameCreation";
89:     }
90: }

```

1.1.2.15.3.2 CategoryPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class CategoryPage.

Body Source

```

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

```

1.1.2.15.3.3 CategoryPage.OnNavigatedTo Method

C#

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

Description

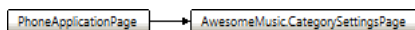
This is OnNavigatedTo, a member of class CategoryPage.

Body Source

```

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

1.1.2.16 CategorySettingsPage Class

Class Hierarchy**C#**

```
public class CategorySettingsPage : PhoneApplicationPage;
```

File

CategorySettingsPage.xaml.cs (see page 162)

Description

This is class AwesomeMusic.CategorySettingsPage.

Methods

	Name	Description
	CategorySettingsPage (see page 62)	This is CategorySettingsPage, a member of class CategorySettingsPage.

CategorySettingsPage Fields

	Name	Description
	categoryId (see page 63)	This is categoryId, a member of class CategorySettingsPage.

CategorySettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 63)	This is OnFragmentNavigation, a member of class CategorySettingsPage.
	OnNavigatedFrom (see page 64)	This is OnNavigatedFrom, a member of class CategorySettingsPage.
	OnNavigatedTo (see page 64)	This is OnNavigatedTo, a member of class CategorySettingsPage.

1.1.2.16.1 CategorySettingsPage.CategorySettingsPage Constructor

C#

```
public CategorySettingsPage();
```

Description

This is CategorySettingsPage, a member of class CategorySettingsPage.

Body Source

```

1: public CategorySettingsPage()
2: {
3:     InitializeComponent();
```

```

4:
5:     pvCategorySettings.Title = AppResources.CategorySettings;
6:
7:     piOtherSettings.Header = AppResources.OtherSettings;
8:     btnArtistOrder.Content = AppResources.Select;
9:     btnArtistOrderStyle.Content = AppResources.Select;
10:    SetBackgroundColor();
11:
12: }

```

1.1.2.16.2 CategorySettingsPage Fields

1.1.2.16.2.1 CategorySettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class CategorySettingsPage.

1.1.2.16.3 CategorySettingsPage Methods

1.1.2.16.3.1 CategorySettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class CategorySettingsPage.

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:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
7:     {
8:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
9:         string orderStyle = category.ArtistOrderStyle;
10:        categoryId = category.CategoryId;
11:
12:        if (category.ArtistOrderBy == "NAME")
13:        {
14:            lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
15:        }
16:        if (category.ArtistOrderBy == "ALBUMCOUNT")
17:        {
18:            lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.AlbumCount + ")";
19:        }
20:        if (category.ArtistOrderBy == "CDATE")
21:        {
22:            lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
23:        }
24:        if (category.ArtistOrderBy == "MDATE")
25:        {

```

```

26:         lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
27:     }
28:     if (category.ArtistOrderStyle == "A")
29:     {
30:         lblArtistOrderStyle.Text = AppResources.ArtistOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
31:     }
32:     if (category.ArtistOrderStyle == "D")
33:     {
34:         lblArtistOrderStyle.Text = AppResources.ArtistOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
35:     }
36:     //lstNoteList.DisplayMemberPath = "NameCreation";
37:     SetBackgroundColor();
38:     }
39: }

```

1.1.2.16.3.2 CategorySettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class CategorySettingsPage.

Body Source

```

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

```

1.1.2.16.3.3 CategorySettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class CategorySettingsPage.

Body Source

```

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

```

1.1.2.17 FontFamilySettingsPage Class

Class Hierarchy



C#

```
public class FontFamilySettingsPage : PhoneApplicationPage;
```

File

FontFamilySettingsPage.xaml.cs (🔗 see page 165)

Description

This is class AwesomeMusic.FontFamilySettingsPage.

Methods

	Name	Description
🔗	FontFamilySettingsPage (🔗 see page 65)	This is FontFamilySettingsPage, a member of class FontFamilySettingsPage.

FontFamilySettingsPage Fields

	Name	Description
🔗	artistId (🔗 see page 66)	This is artistId, a member of class FontFamilySettingsPage.

FontFamilySettingsPage Methods

	Name	Description
🔗	OnFragmentNavigation (🔗 see page 66)	This is OnFragmentNavigation, a member of class FontFamilySettingsPage.
🔗	OnNavigatedFrom (🔗 see page 66)	This is OnNavigatedFrom, a member of class FontFamilySettingsPage.
🔗	OnNavigatedTo (🔗 see page 67)	This is OnNavigatedTo, a member of class FontFamilySettingsPage.

1.1.2.17.1 FontFamilySettingsPage.FontFamilySettingsPage Constructor**C#**

```
public FontFamilySettingsPage();
```

Description

This is FontFamilySettingsPage, a member of class FontFamilySettingsPage.

Body Source

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

```
29:     lstFontFamily.Items.Add("Times New Roman");
30:     lstFontFamily.Items.Add("Verdana");
31:     lstFontFamily.SelectedIndex = -1;
32: }
```

1.1.2.17.2 FontFamilySettingsPage Fields

1.1.2.17.2.1 FontFamilySettingsPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class FontFamilySettingsPage.

1.1.2.17.3 FontFamilySettingsPage Methods

1.1.2.17.3.1 FontFamilySettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class FontFamilySettingsPage.

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:     artistId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
8:     {
9:         var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single()
as Artist;
10:         lblArtistName.Text = artist.ArtistName;
11:         lblFontFamily.Text = AppResources.SelectFontFamily;
12:     }
13:     SetBackgroundColor();
14: }
```

1.1.2.17.3.2 FontFamilySettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class FontFamilySettingsPage.

Body Source

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

1.1.2.17.3.3 FontFamilySettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class FontFamilySettingsPage.

Body Source

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

1.1.2.18 FontSizeSettingsPage Class

Class Hierarchy



C#

```
public class FontSizeSettingsPage : PhoneApplicationPage;
```

File

FontSizeSettingsPage.xaml.cs (🔗 see page 168)

Description

This is class AwesomeMusic.FontSizeSettingsPage.

Methods

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

FontSizeSettingsPage Fields

	Name	Description
🔗	artistId (🔗 see page 68)	This is artistId, a member of class FontSizeSettingsPage.

FontSizeSettingsPage Methods

	Name	Description
🔗🔗	OnFragmentNavigation (🔗 see page 68)	This is OnFragmentNavigation, a member of class FontSizeSettingsPage.
🔗🔗	OnNavigatedFrom (🔗 see page 69)	This is OnNavigatedFrom, a member of class FontSizeSettingsPage.
🔗🔗	OnNavigatedTo (🔗 see page 69)	This is OnNavigatedTo, a member of class FontSizeSettingsPage.

1.1.2.18.1 FontSizeSettingsPage.FontSizeSettingsPage Constructor

C#

```
public FontSizeSettingsPage();
```

Description

This is FontSizeSettingsPage, a member of class FontSizeSettingsPage.

Body Source

```

1: public FontSizeSettingsPage()
2: {
3:     InitializeComponent();
4:
5:     lstFontSize.Items.Clear();
6:     lstFontSize.Items.Add("14");
7:     lstFontSize.Items.Add("18");
8:     lstFontSize.Items.Add("22");
9:     lstFontSize.Items.Add("26");
10:    lstFontSize.Items.Add("28");
11:    lstFontSize.Items.Add("30");
12:    lstFontSize.Items.Add("32");
13:    lstFontSize.Items.Add("34");
14:    lstFontSize.Items.Add("36");
15:    lstFontSize.Items.Add("38");
16:    lstFontSize.Items.Add("40");
17:    lstFontSize.Items.Add("42");
18:    lstFontSize.Items.Add("44");
19:    lstFontSize.Items.Add("64");
20:    lstFontSize.Items.Add("72");
21:    lstFontSize.SelectedIndex = -1;
22: }

```

1.1.2.18.2 FontSizeSettingsPage Fields**1.1.2.18.2.1 FontSizeSettingsPage.artistId Field****C#**

```
public int artistId;
```

Description

This is artistId, a member of class FontSizeSettingsPage.

1.1.2.18.3 FontSizeSettingsPage Methods**1.1.2.18.3.1 FontSizeSettingsPage.OnFragmentNavigation Method****C#**

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

Description

This is OnFragmentNavigation, a member of class FontSizeSettingsPage.

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:     artistId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
8:     {
9:         var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single()
as Artist;
10:         lblArtistName.Text = artist.ArtistName;
11:         lblFontSize.Text = AppResources.SelectFontSize;
12:     }
13:     SetBackgroundColor();

```

```
14: }
```

1.1.2.18.3.2 FontSizeSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class FontSizeSettingsPage.

Body Source

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

1.1.2.18.3.3 FontSizeSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class FontSizeSettingsPage.

Body Source

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

1.1.2.19 GeneralSettingsPage Class

Class Hierarchy



C#

```
public class GeneralSettingsPage : PhoneApplicationPage;
```

File

GeneralSettingsPage.xaml.cs (🔗 see page 170)

Description

This is class AwesomeMusic.GeneralSettingsPage.

Methods

	Name	Description
🔗	GeneralSettingsPage (🔗 see page 70)	This is GeneralSettingsPage, a member of class GeneralSettingsPage.

GeneralSettingsPage Fields

	Name	Description
🔗	signIn (🔗 see page 73)	This is signIn, a member of class GeneralSettingsPage.

GeneralSettingsPage Methods

	Name	Description
🔗	CreateDirectoryAsync (🔗 see page 73)	This is CreateDirectoryAsync, a member of class GeneralSettingsPage.
🔗	DesignFileName (🔗 see page 74)	This is DesignFileName, a member of class GeneralSettingsPage.
🔗🔗	OnNavigatedFrom (🔗 see page 74)	This is OnNavigatedFrom, a member of class GeneralSettingsPage.
🔗🔗	OnNavigatedTo (🔗 see page 74)	This is OnNavigatedTo, a member of class GeneralSettingsPage.

1.1.2.19.1 GeneralSettingsPage.GeneralSettingsPage Constructor**C#**

```
public GeneralSettingsPage();
```

Description

This is GeneralSettingsPage, a member of class GeneralSettingsPage.

Body Source

```

1: public GeneralSettingsPage()
2: {
3:     InitializeComponent();
4:     InitializePage();
5:
6:     pvGeneralSettings.Title = AppResources.GeneralSettings;
7:
8:     piLanguage.Header = AppResources.Language;
9:     piSync.Header = AppResources.Sync;
10:    piOtherSettings.Header = AppResources.OtherSettings;
11:    piBackground.Header = AppResources.Background;
12:
13:    //lblOneDrive.Text = AppResources.OneDrive;
14:
15:    btnCategoryOrder.Content = AppResources.Select;
16:    btnCategoryOrderStyle.Content = AppResources.Select;
17:    btnLanguage.Content = AppResources.Select;
18:    btnBackgroundColor.Content = AppResources.Select;
19:    //btnOneDrive.Content = AppResources.Login;
20:    //btnOneDrive.SignInText = AppResources.SignIn;
21:    //btnOneDrive.SignOutText = AppResources.SignOut;
22:    btnOneDriveSync.Content = AppResources.Sync;
23:    lblOneDrive.Text = AppResources.OneDrive;
24:    txtSynchronizing.Text = AppResources.Synchronizing;
25:
26:    pbSync.Visibility = Visibility.Collapsed;
27:    txtSynchronizing.Visibility = Visibility.Collapsed;
28:    txtSynchronizing.BorderBrush = this.LayoutRoot.Background;
29:
30:    btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
31:    lblBackgroundImage.Text = AppResources.BackgroundImage;
32:    btnBackgroundImage.Content = AppResources.Select;
33:    btnResetSettings.Content = AppResources.ResetSettings;
34:
35:    btnOneDriveSync.IsEnabled = false;
36:    cbSync.Content = AppResources.SyncOnOneFile;
37:    cbSync.IsEnabled = false;
38:    btnOneDrive.Content = "Sign In";
39:
40:    SetBackgroundColor();
41:
42:    using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
43:    {
44:        var appSettings = context.AppSettings.First() as AppSettings;
```

```
45:         if (appSettings.AppLangName == "EN")
46:         {
47:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.English + ")";
48:         }
49:         if (appSettings.AppLangName == "TR")
50:         {
51:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Turkish + ")";
52:         }
53:         if (appSettings.AppLangName == "DE")
54:         {
55:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.German + ")";
56:         }
57:         if (appSettings.AppLangName == "ES")
58:         {
59:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Spanish + ")";
60:         }
61:
62:         if (appSettings.AppLangName == "PT")
63:         {
64:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Portuguese + ")";
65:         }
66:         if (appSettings.AppLangName == "AR")
67:         {
68:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Arabic + ")";
69:         }
70:         if (appSettings.AppLangName == "FA")
71:         {
72:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Persian + ")";
73:         }
74:         if (appSettings.AppLangName == "IT")
75:         {
76:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Italian + ")";
77:         }
78:         if (appSettings.AppLangName == "FR")
79:         {
80:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.French + ")";
81:         }
82:         if (appSettings.AppLangName == "RU")
83:         {
84:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Russian + ")";
85:         }
86:         if (appSettings.AppLangName == "ZH")
87:         {
88:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Chinese + ")";
89:         }
90:         if (appSettings.AppLangName == "JA")
91:         {
92:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Japanese + ")";
93:         }
94:         if (appSettings.AppLangName == "SA")
95:         {
96:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
": " + AppResources.Sanskrit + ")";
97:         }
98:         if (appSettings.AppLangName == "TH")
99:         {
100:             lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected +
```

```

": " + AppResources.Thai + "));
101:         }
102:
103:         if (appSettings.CategoryOrderBy == "NAME")
104:         {
105:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
106:         }
107:         if (appSettings.CategoryOrderBy == "CDATE")
108:         {
109:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
110:         }
111:         if (appSettings.CategoryOrderBy == "MDATE")
112:         {
113:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
114:         }
115:         if (appSettings.CategoryOrderBy == "ALBUMCOUNT")
116:         {
117:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.AlbumCount + ")";
118:         }
119:         if (appSettings.CategoryOrderStyle == "A")
120:         {
121:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
122:         }
123:         if (appSettings.CategoryOrderStyle == "D")
124:         {
125:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
126:         }
127:         if (appSettings.AppBackgroundColor == "BLA")
128:         {
129:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + ")";
130:         }
131:         if (appSettings.AppBackgroundColor == "BLU")
132:         {
133:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + ")";
134:         }
135:         if (appSettings.AppBackgroundColor == "BRO")
136:         {
137:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + ")";
138:         }
139:         if (appSettings.AppBackgroundColor == "RED")
140:         {
141:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + ")";
142:         }
143:         if (appSettings.AppBackgroundColor == "GRE")
144:         {
145:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + ")";
146:         }
147:         if (appSettings.AppBackgroundColor == "YEL")
148:         {
149:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + ")";
150:         }
151:         if (appSettings.AppBackgroundColor == "GRA")
152:         {
153:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + ")";
154:         }
155:         if (appSettings.AppBackgroundColor == "ORA")

```

```

156:         {
157:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + ")";
158:         }
159:         if (appSettings.AppBackgroundColor == "PUR")
160:         {
161:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + ")";
162:         }
163:     }
164: }

```

1.1.2.19.2 GeneralSettingsPage Fields

1.1.2.19.2.1 GeneralSettingsPage.signIn Field

C#

```
public int signIn;
```

Description

This is signIn, a member of class GeneralSettingsPage.

1.1.2.19.3 GeneralSettingsPage Methods

1.1.2.19.3.1 GeneralSettingsPage.CreateDirectoryAsync Method

C#

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

Description

This is CreateDirectoryAsync, a member of class GeneralSettingsPage.

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: }
```

1.1.2.19.3.2 GeneralSettingsPage.DesignFileName Method

C#

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

Description

This is DesignFileName, a member of class GeneralSettingsPage.

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: }
```

1.1.2.19.3.3 GeneralSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class GeneralSettingsPage.

Body Source

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

1.1.2.19.3.4 GeneralSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class GeneralSettingsPage.

Body Source

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

```
7:
8: }
```

1.1.2.20 LanguageSettingsPage Class

Class Hierarchy



C#

```
public class LanguageSettingsPage : PhoneApplicationPage;
```

File

LanguageSettingsPage.xaml.cs (see page 183)

Description

This is class AwesomeMusic.LanguageSettingsPage.

Methods

	Name	Description
	LanguageSettingsPage (see page 75)	This is LanguageSettingsPage, a member of class LanguageSettingsPage.

LanguageSettingsPage Methods

	Name	Description
	OnNavigatedFrom (see page 76)	This is OnNavigatedFrom, a member of class LanguageSettingsPage.
	OnNavigatedTo (see page 76)	This is OnNavigatedTo, a member of class LanguageSettingsPage.

1.1.2.20.1 LanguageSettingsPage.LanguageSettingsPage Constructor

C#

```
public LanguageSettingsPage();
```

Description

This is LanguageSettingsPage, a member of class LanguageSettingsPage.

Body Source

```

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



```
23:
24:     SetBackgroundColor();
25: }
```

1.1.2.20.2 LanguageSettingsPage Methods

1.1.2.20.2.1 LanguageSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class LanguageSettingsPage.

Body Source

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

1.1.2.20.2.2 LanguageSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class LanguageSettingsPage.

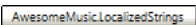
Body Source

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

1.1.2.21 LocalizedStrings Class

Provides access to string resources.

Class Hierarchy



C#

```
public class LocalizedStrings;
```

File

LocalizedStrings.cs (🔗 see page 187)

LocalizedStrings Properties

	Name	Description
	LocalizedResources (🔗 see page 77)	This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.21.1 LocalizedStrings Properties

1.1.2.21.1 LocalizedStrings.LocalizedResources Property

C#

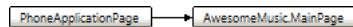
```
public AppResources LocalizedResources;
```

Description

This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.22 MainPage Class

Class Hierarchy



C#

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

File

MainPage.xaml.cs (see page 187)

Description

This is class AwesomeMusic.MainPage.

Methods

	Name	Description
	MainPage (see page 77)	Constructor

MainPage Fields

	Name	Description
	popup (see page 78)	This is popup, a member of class MainPage.

MainPage Methods

	Name	Description
	OnNavigatedFrom (see page 78)	This is OnNavigatedFrom, a member of class MainPage.
	OnNavigatedTo (see page 79)	This is OnNavigatedTo, a member of class MainPage.

1.1.2.22.1 MainPage.MainPage Constructor

C#

```
public MainPage();
```

Description

Constructor

Body Source

```
1: public MainPage()  
2: {  
3:     InitializeComponent();  
4:  
5:  
6:     ApplicationBar = new ApplicationBar();  
7:  
8:     ApplicationBarIconButton button1 = new ApplicationBarIconButton();  
9:     button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);  
10:    button1.Text = AppResources.AddCategory;
```

```

11:     ApplicationBar.Buttons.Add(button1);
12:     button1.Click += new EventHandler(AddCategoryButton_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/Statistics.png", UriKind.Relative);
28:     button4.Text = AppResources.Statistics;
29:     ApplicationBar.Buttons.Add(button4);
30:     button4.Click += new EventHandler(StatisticsButton_Click);
31:
32:     ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
33:     menuItem1.Text = AppResources.About;
34:     ApplicationBar.MenuItems.Add(menuItem1);
35:     menuItem1.Click += new EventHandler(AboutMenuItem_Click);
36:
37:     lblCategories.Text = AppResources.Categories;
38:     // Sample code to localize the ApplicationBar
39:     //BuildLocalizedApplicationBar();
40:
41:     SetBackgroundColor();
42:
43:     popup = new Popup();
44: }

```

1.1.2.22.2 MainPage Fields

1.1.2.22.2.1 MainPage.popup Field

C#

```
public Popup popup;
```

Description

This is popup, a member of class MainPage.

1.1.2.22.3 MainPage Methods

1.1.2.22.3.1 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: }
```

1.1.2.22.3.2 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: }
```

1.1.2.23 OrderSettingsPage Class

Class Hierarchy



C#

```
public class OrderSettingsPage : PhoneApplicationPage;
```

File

OrderSettingsPage.xaml.cs (🔗 see page 193)

Description

This is class AwesomeMusic.OrderSettingsPage.

Methods

	Name	Description
🔗	OrderSettingsPage (🔗 see page 80)	This is OrderSettingsPage, a member of class OrderSettingsPage.

OrderSettingsPage Fields

	Name	Description
🔗	artistId (🔗 see page 80)	This is artistId, a member of class OrderSettingsPage.
🔗	categoryId (🔗 see page 80)	This is categoryId, a member of class OrderSettingsPage.
🔗	pageName (🔗 see page 80)	This is pageName, a member of class OrderSettingsPage.

OrderSettingsPage Methods

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

1.1.2.23.1 OrderSettingsPage.OrderSettingsPage Constructor

C#

```
public OrderSettingsPage();
```

Description

This is OrderSettingsPage, a member of class OrderSettingsPage.

Body Source

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

1.1.2.23.2 OrderSettingsPage Fields

1.1.2.23.2.1 OrderSettingsPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class OrderSettingsPage.

1.1.2.23.2.2 OrderSettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class OrderSettingsPage.

1.1.2.23.2.3 OrderSettingsPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class OrderSettingsPage.

1.1.2.23.3 OrderSettingsPage Methods

1.1.2.23.3.1 OrderSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class OrderSettingsPage.

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:     lstOrderBy.Items.Clear();
7:     if (pageName.Contains("/CategorySettingsPage.xaml"))
8:     {
9:         categoryId = int.Parse(e.Fragment);
10:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
11:        {
12:            var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
13:            lblSettings.Text = AppResources.CategorySettings + " (" +
category.CategoryName + ")";
14:            lblOrderBy.Text = AppResources.ArtistOrderBy;
15:            lstOrderBy.Items.Add(AppResources.Name);
16:            lstOrderBy.Items.Add(AppResources.AlbumCount);
17:            lstOrderBy.Items.Add(AppResources.CreationDate);
18:            lstOrderBy.Items.Add(AppResources.ModificationDate);
19:        }
20:    }
21:    else
22:    {
23:        artistId = int.Parse(e.Fragment);
24:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
25:        {
26:            var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
27:            lblSettings.Text = AppResources.ArtistSettings + " (" + artist.ArtistName +
")";
28:            lblOrderBy.Text = AppResources.AlbumOrderBy;
29:            lstOrderBy.Items.Add(AppResources.Name);
30:            lstOrderBy.Items.Add(AppResources.CreationDate);
31:            lstOrderBy.Items.Add(AppResources.ModificationDate);
32:            lstOrderBy.Items.Add(AppResources.AlbumRating);
33:        }
34:    }
35:    lstOrderBy.SelectedIndex = -1;
36:    SetBackgroundColor();
37: }

```

1.1.2.23.3.2 OrderSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class OrderSettingsPage.

Body Source

```

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

```

1.1.2.23.3.3 OrderSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class OrderSettingsPage.

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:     lstOrderBy.Items.Clear();
8:     if (pageName.Contains("/GeneralSettingsPage.xaml"))
9:     {
10:         lblSettings.Text = AppResources.GeneralSettings;
11:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
12:         {
13:             var appSettings =
14:                 context.AppSettings.First();
15:             lblOrderBy.Text = AppResources.CategoryOrderBy;
16:
17:             lstOrderBy.Items.Add(AppResources.Name);
18:             lstOrderBy.Items.Add(AppResources.AlbumCount);
19:             lstOrderBy.Items.Add(AppResources.CreationDate);
20:             lstOrderBy.Items.Add(AppResources.ModificationDate);
21:
22:         }
23:     }
24: }

```

1.1.2.24 OrderStyleSettingsPage Class

Class Hierarchy**C#**

```
public class OrderStyleSettingsPage : PhoneApplicationPage;
```


File

OrderStyleSettingsPage.xaml.cs (see page 198)




Description

This is class AwesomeMusic.OrderStyleSettingsPage.



Methods


	Name	Description
	OrderStyleSettingsPage (see page 83)	This is OrderStyleSettingsPage, a member of class OrderStyleSettingsPage.

OrderStyleSettingsPage Fields

	Name	Description
	artistId (see page 83)	This is artistId, a member of class OrderStyleSettingsPage.
	categoryId (see page 83)	This is categoryId, a member of class OrderStyleSettingsPage.
	pageName (see page 83)	This is pageName, a member of class OrderStyleSettingsPage.

OrderStyleSettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 84)	This is OnFragmentNavigation, a member of class OrderStyleSettingsPage.
	OnNavigatedFrom (see page 84)	This is OnNavigatedFrom, a member of class OrderStyleSettingsPage.

	OnNavigatedTo (see page 84)	This is OnNavigatedTo, a member of class OrderStyleSettingsPage.
---	-----------------------------	--

1.1.2.24.1 OrderStyleSettingsPage.OrderStyleSettingsPage Constructor

C#

```
public OrderStyleSettingsPage();
```

Description

This is OrderStyleSettingsPage, a member of class OrderStyleSettingsPage.

Body Source

```
1: public OrderStyleSettingsPage()  
2: {  
3:     InitializeComponent();  
4:  
5:     lstOrderStyle.Items.Clear();  
6:  
7:     lstOrderStyle.Items.Add(AppResources.Ascending);  
8:     lstOrderStyle.Items.Add(AppResources.Descending);  
9:  
10:    lstOrderStyle.SelectedIndex = -1;  
11:  
12:    SetBackgroundColor();  
13: }
```

1.1.2.24.2 OrderStyleSettingsPage Fields

1.1.2.24.2.1 OrderStyleSettingsPage.artistId Field

C#

```
public int artistId;
```

Description

This is artistId, a member of class OrderStyleSettingsPage.

1.1.2.24.2.2 OrderStyleSettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class OrderStyleSettingsPage.

1.1.2.24.2.3 OrderStyleSettingsPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class OrderStyleSettingsPage.

1.1.2.24.3 OrderStyleSettingsPage Methods

1.1.2.24.3.1 OrderStyleSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class OrderStyleSettingsPage.

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:     if (pageName.Contains("/CategorySettingsPage.xaml"))
7:     {
8:         categoryId = int.Parse(e.Fragment);
9:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
10:        {
11:            var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
12:            lblSettings.Text = AppResources.CategorySettings + " (" +
category.CategoryName + ")";
13:            lblOrderStyle.Text = AppResources.ArtistOrderStyle;
14:        }
15:    }
16:    else
17:    {
18:        artistId = int.Parse(e.Fragment);
19:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
20:        {
21:            var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
22:            lblSettings.Text = AppResources.ArtistSettings + " (" + artist.ArtistName +
")";
23:            lblOrderStyle.Text = AppResources.AlbumOrderStyle;
24:        }
25:    }
26:    SetBackgroundColor();
27: }
```

1.1.2.24.3.2 OrderStyleSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class OrderStyleSettingsPage.

Body Source

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

1.1.2.24.3.3 OrderStyleSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class OrderStyleSettingsPage.

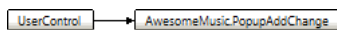
Body Source

```

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:     if (pageName.Contains("/GeneralSettingsPage.xaml"))
8:     {
9:         lblSettings.Text = AppResources.GeneralSettings;
10:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
11:        {
12:            var appSettings =
13:                context.AppSettings.First();
14:            lblOrderStyle.Text = AppResources.CategoryOrderStyle;
15:        }
16:    }
17: }

```

1.1.2.25 PopupAddChange Class

Class Hierarchy**C#**

```
public class PopupAddChange : UserControl;
```

File

PopupAddChange.xaml.cs (see page 202)

Description

This is class AwesomeMusic.PopupAddChange.

Methods

	Name	Description
	PopupAddChange (see page 85)	This is PopupAddChange, a member of class PopupAddChange.

1.1.2.25.1 PopupAddChange.PopupAddChange Constructor

C#

```
public PopupAddChange();
```

Description

This is PopupAddChange, a member of class PopupAddChange.

Body Source

```

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

```

1.1.2.26 SearchPage Class

Class Hierarchy



C#

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

File

SearchPage.xaml.cs (🔗 see page 203)

Description

This is class AwesomeMusic.SearchPage.

Methods

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

SearchPage Methods

	Name	Description
🔗	OnNavigatedFrom (🔗 see page 86)	This is OnNavigatedFrom, a member of class SearchPage.
🔗	OnNavigatedTo (🔗 see page 87)	This is OnNavigatedTo, a member of class SearchPage.

1.1.2.26.1 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: }
```

1.1.2.26.2 SearchPage Methods

1.1.2.26.2.1 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: }

```

1.1.2.26.2 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: }

```

1.1.2.27 StatisticsPage Class**Class Hierarchy****C#**

```
public class StatisticsPage : PhoneApplicationPage;
```

File

StatisticsPage.xaml.cs (see page 207)

Description

This is class AwesomeMusic.StatisticsPage.

Methods

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

StatisticsPage Methods

	Name	Description
	OnNavigatedFrom (see page 88)	This is OnNavigatedFrom, a member of class StatisticsPage.
	OnNavigatedTo (see page 88)	This is OnNavigatedTo, a member of class StatisticsPage.

1.1.2.27.1 StatisticsPage.StatisticsPage Constructor**C#**

```
public StatisticsPage();
```

Description

This is StatisticsPage, a member of class StatisticsPage.

Body Source

```

1: public StatisticsPage()
2: {
3:     InitializeComponent();

```

```
4:     lblStatistics.Text = AppResources.Statistics;
5:     SetBackgroundColor();
6:     SetStatistic();
7: }
```

1.1.2.27.2 StatisticsPage Methods

1.1.2.27.2.1 StatisticsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class StatisticsPage.

Body Source

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

1.1.2.27.2.2 StatisticsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class StatisticsPage.

Body Source

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

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AboutPage.xaml.cs (see page 89)	This is file AboutPage.xaml.cs.
AddCategoryPage.xaml.cs (see page 92)	This is file AddCategoryPage.xaml.cs.
Album.cs (see page 94)	This is file Album.cs.
AlbumArtist.cs (see page 96)	This is file AlbumArtist.cs.
AlbumPage.xaml.cs (see page 96)	This is file AlbumPage.xaml.cs.
App.xaml.cs (see page 107)	This is file App.xaml.cs.
AppResources.Designer.cs (see page 114)	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 134)	This is file AppSettings.cs.
Artist.cs (see page 135)	This is file Artist.cs.
ArtistPage.xaml.cs (see page 136)	This is file ArtistPage.xaml.cs.
ArtistSettingsPage.xaml.cs (see page 144)	This is file ArtistSettingsPage.xaml.cs.
AssemblyInfo.cs (see page 147)	This is file AssemblyInfo.cs.
AwesomeMusic.csproj (see page 148)	This is file AwesomeMusic.csproj.
AwesomeMusic.sln (see page 148)	This is file AwesomeMusic.sln.
AwesomeMusicDataContext.cs (see page 148)	This is file AwesomeMusicDataContext.cs.
BackgroundColorSettingsPage.xaml.cs (see page 149)	This is file BackgroundColorSettingsPage.xaml.cs.
Category.cs (see page 152)	This is file Category.cs.
CategoryArtist.cs (see page 153)	This is file CategoryArtist.cs.
CategoryPage.xaml.cs (see page 154)	This is file CategoryPage.xaml.cs.
CategorySettingsPage.xaml.cs (see page 162)	This is file CategorySettingsPage.xaml.cs.
FontFamilySettingsPage.xaml.cs (see page 165)	This is file FontFamilySettingsPage.xaml.cs.
FontSizeSettingsPage.xaml.cs (see page 168)	This is file FontSizeSettingsPage.xaml.cs.
GeneralSettingsPage.xaml.cs (see page 170)	This is file GeneralSettingsPage.xaml.cs.
LanguageSettingsPage.xaml.cs (see page 183)	This is file LanguageSettingsPage.xaml.cs.
LocalizedStrings.cs (see page 187)	This is file LocalizedStrings.cs.
MainPage.xaml.cs (see page 187)	This is file MainPage.xaml.cs.
OrderSettingsPage.xaml.cs (see page 193)	This is file OrderSettingsPage.xaml.cs.
OrderStyleSettingsPage.xaml.cs (see page 198)	This is file OrderStyleSettingsPage.xaml.cs.
PopupAddChange.xaml.cs (see page 202)	This is file PopupAddChange.xaml.cs.
SearchPage.xaml.cs (see page 203)	This is file SearchPage.xaml.cs.
StatisticsPage.xaml.cs (see page 207)	This is file StatisticsPage.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.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 AwesomeMusic.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:

```

```

19: namespace AwesomeMusic
20: {
21:     public partial class AboutPage : PhoneApplicationPage
22:     {
23:         public AboutPage()
24:         {
25:             InitializeComponent();
26:             SetBackgroundColor();
27:
28:             ApplicationBar = new ApplicationBar();
29:
30:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
31:             button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
32:             button2.Text = AppResources.ContactWithUs;
33:             ApplicationBar.Buttons.Add(button2);
34:             button2.Click += new EventHandler(SendMailButton_Click);
35:
36:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
37:             button3.IconUri = new Uri("/Assets/Rate.png", UriKind.Relative);
38:             button3.Text = AppResources.Rate;
39:             ApplicationBar.Buttons.Add(button3);
40:             button3.Click += new EventHandler(RateButton_Click);
41:
42:             lblAboutTheApp.Text = AppResources.AboutTheApp;
43:             //txtAbout2.Text = AppResources.AboutTheAppText;
44:             //var paragraph = new Paragraph();
45:             //paragraph.Inlines.Add(AppResources.AboutTheAppText);
46:             //txtAbout.Blocks.Add(paragraph);
47:             txtAbout.Text = AppResources.AboutTheAppText;
48:             //txtAbout.IsEnabled = false;
49:             txtAbout.IsReadOnly = true;
50:             //this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
51:         }
52:
53:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
54:         {
55:             if (this.NavigationService.CanGoBack)
56:             {
57:                 this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
58:             }
59:         }
60:
61:         private void SendMailButton_Click(object sender, EventArgs e)
62:         {
63:             // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
64:             //EmailAddressChooserTask emailAddressChooserTask;
65:             //emailAddressChooserTask = new EmailAddressChooserTask();
66:             //emailAddressChooserTask.Completed += new
EventHandler<EmailResult>(emailAddressChooserTask_Completed);
67:             //emailAddressChooserTask.Show();
68:             StringBuilder sb = new StringBuilder();
69:             EmailComposeTask emailComposeTask = new EmailComposeTask();
70:
71:
72:             sb.AppendLine();
73:             sb.AppendLine();
74:             sb.AppendLine(AppResources.SendWithApp);
75:
76:             emailComposeTask.Subject = AppResources.AboutTheAwesomeMusic;
77:             emailComposeTask.Body = sb.ToString();
78:             emailComposeTask.To = "coderserdar@outlook.com";
79:             emailComposeTask.Cc = "";
80:             emailComposeTask.Bcc = "";
81:
82:             emailComposeTask.Show();
83:             //MessageBox.Show(AppResources.SuccessfulSendWithMail);
84:         }

```

```

85:
86:     private void RateButton_Click(object sender, EventArgs e)
87:     {
88:         MarketplaceReviewTask marketplaceReviewTask = new MarketplaceReviewTask();
89:         marketplaceReviewTask.Show();
90:     }
91:
92:     private void SetBackgroundColor()
93:     {
94:         AppSettings appSettings = new AppSettings();
95:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
96:         {
97:             appSettings = context.AppSettings.First() as AppSettings;
98:         }
99:
100:         if (appSettings.AppBackgroundImage != null)
101:         {
102:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
103:             BitmapImage image = new BitmapImage();
104:             image.SetSource(stream);
105:             ImageBrush ib = new ImageBrush();
106:             ib.ImageSource = image;
107:             this.LayoutRoot.Background = ib;
108:         }
109:         else
110:         {
111:             switch (appSettings.AppBackgroundColor)
112:             {
113:                 case "BLA":
114:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
115:                     break;
116:                 case "BLU":
117:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
118:                     break;
119:                 case "BRO":
120:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
121:                     break;
122:                 case "RED":
123:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
124:                     break;
125:                 case "GRE":
126:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
127:                     break;
128:                 case "GRA":
129:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
130:                     break;
131:                 case "YEL":
132:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
133:                     break;
134:                 case "ORA":
135:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
136:                     break;
137:                 case "PUR":
138:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
139:                     break;
140:                 default:
141:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
142:                     break;
143:             }
144:         }
145:     }
146: }
147: }

```


Namespaces

Name	Description
AwesomeMusic (🔗 see page 1)	This is namespace AwesomeMusic.

1.2.2 AddCategoryPage.xaml.cs

This is file AddCategoryPage.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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
18: namespace AwesomeMusic
19: {
20:     public partial class AddCategoryPage : PhoneApplicationPage
21:     {
22:
23:         public int artistId;
24:
25:         public AddCategoryPage()
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:             artistId = int.Parse(e.Fragment);
47:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
48:             {
49:                 var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
50:                 lstCategories.Items.Clear();
51:                 lblArtistName.Text = artist.ArtistName;

```

```

52:         lblCategories.Text = AppResources.Categories;
53:         var categories = context.Categories;
54:         lstCategories.ItemsSource = categories;
55:         lstCategories.DisplayMemberPath = "CategoryName";
56:     }
57: }
58:
59:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
60:     {
61:         if (this.NavigationService.CanGoBack)
62:         {
63:             this.NavigationService.Navigate(new Uri("/ArtistPage.xaml#" +
artistId, UriKind.Relative));
64:         }
65:     }
66:
67:     private void SetBackgroundColor()
68:     {
69:         AppSettings appSettings = new AppSettings();
70:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
71:         {
72:             appSettings = context.AppSettings.First() as AppSettings;
73:         }
74:
75:         if (appSettings.AppBackgroundImage != null)
76:         {
77:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
78:             BitmapImage image = new BitmapImage();
79:             image.SetSource(stream);
80:             ImageBrush ib = new ImageBrush();
81:             ib.ImageSource = image;
82:             this.LayoutRoot.Background = ib;
83:         }
84:         else
85:         {
86:             switch (appSettings.AppBackgroundColor)
87:             {
88:                 case "BLA":
89:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
90:                     break;
91:                 case "BLU":
92:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
93:                     break;
94:                 case "BRO":
95:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
96:                     break;
97:                 case "RED":
98:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
99:                     break;
100:                 case "GRE":
101:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
102:                     break;
103:                 case "GRA":
104:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
105:                     break;
106:                 case "YEL":
107:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
108:                     break;
109:                 case "ORA":
110:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
111:                     break;
112:                 case "PUR":
113:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
114:                     break;

```

```

115:                 default:
116:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
117:                     break;
118:             }
119:         }
120:     }
121:
122:     private void lstCategories_SelectionChanged(object sender,
SelectionChangedEventArgs e)
123:     {
124:         CategoryArtist categoryArtist2 = null;
125:         Category category = lstCategories.SelectedItem as Category;
126:         CategoryArtist categoryArtist = new CategoryArtist();
127:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
128:         {
129:             categoryArtist.ArtistId = artistId;
130:             categoryArtist.CategoryId = category.CategoryId;
131:             try
132:             {
133:                 categoryArtist2 = context.CategoryArtists.Where(j =>
j.CategoryId.Equals(categoryArtist.CategoryId) &&
j.ArtistId.Equals(categoryArtist.ArtistId)).Single() as CategoryArtist;
134:             }
135:             catch (Exception)
136:             {
137:                 context.CategoryArtists.InsertOnSubmit(categoryArtist);
138:                 context.SubmitChanges();
139:                 var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Select(j => j);
140:                 foreach (var item in artist)
141:                 {
142:                     item.ModificationDate = DateTime.Now;
143:                 }
144:                 context.SubmitChanges();
145:                 MessageBox.Show(AppResources.ArtistCategoryAddSuccess);
146:             }
147:             if (categoryArtist2 != null)
148:             {
149:                 MessageBox.Show(AppResources.ArtistAlreadySameCategory);
150:             }
151:             else
152:             {
153:             }
154:         }
155:     }
156:     this.NavigationService.Navigate(new Uri("/ArtistPage.xaml#" + artistId,
UriKind.Relative));
157: }
158: }
159: }

```

Namespaces

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

1.2.3 Album.cs

This is file Album.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 AwesomeMusic
10: {
11:     [Table]
12:     public class Album
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int AlbumId { get; set; }
19:
20:         [Column]
21:         public string AlbumGuid { get; set; }
22:
23:         [Column]
24:         public int AlbumCategoryId { get; set; }
25:
26:         [Column]
27:         public string AlbumName { get; set; }
28:
29:         [Column]
30:         public int AlbumReleaseYear { get; set; }
31:
32:         [Column]
33:         public int AlbumSongCount { get; set; }
34:
35:         [Column]
36:         public string AlbumLabelName { get; set; }
37:
38:         [Column]
39:         public string AlbumBestSong { get; set; }
40:
41:         [Column]
42:         public int AlbumRating { get; set; }
43:
44:         [Column]
45:         public string AlbumComment { get; set; }
46:
47:         [Column]
48:         public DateTime CreationDate { get; set; }
49:
50:         [Column]
51:         public DateTime ModificationDate { get; set; }
52:
53:         [Column]
54:         public string AlbumInformation { get; set; }
55:
56:         [Column]
57:         public string AlbumNameRating { get; set; }
58:     }
59: }
```

Namespaces

Name	Description
AwesomeMusic (🔗 see page 1)	This is namespace AwesomeMusic.

1.2.4 AlbumArtist.cs

This is file AlbumArtist.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 AwesomeMusic
10: {
11:     [Table]
12:     public class AlbumArtist
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int AlbumArtistId { get; set; }
19:
20:         [Column]
21:         public int AlbumId { get; set; }
22:
23:         [Column]
24:         public int ArtistId { get; set; }
25:     }
26: }
```

Namespaces

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

1.2.5 AlbumPage.xaml.cs

This is file AlbumPage.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 AwesomeMusic.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
```

```

18:
19: namespace AwesomeMusic
20: {
21:     public partial class AlbumPage : PhoneApplicationPage
22:     {
23:         public int artistId;
24:         public string artistName;
25:         public string categoryName;
26:         public int categoryId;
27:         public int albumId;
28:         public string pageName;
29:         double InputHeight = 0.0;
30:         public bool flag;
31:         public bool isFilled;
32:         public double ratingValue = 0;
33:
34:         public AlbumPage()
35:         {
36:             InitializeComponent();
37:
38:             SetBackgroundColor();
39:
40:             //pvArtist.Title = artistName;
41:             piAlbumName.Header = AppResources.AlbumName;
42:             piComment.Header = AppResources.AlbumComment;
43:             piLabelName.Header = AppResources.LabelName;
44:             piRating.Header = AppResources.AlbumRating;
45:             piReleaseYear.Header = AppResources.ReleaseYear;
46:             piBestSong.Header = AppResources.BestSong;
47:             piSongCount.Header = AppResources.SongCount;
48:
49:
50:             ApplicationBar = new ApplicationBar();
51:
52:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
53:             button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
54:             button1.Text = AppResources.Save;
55:             ApplicationBar.Buttons.Add(button1);
56:             button1.Click += new EventHandler(SaveButton_Click);
57:
58:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
59:             button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
60:             button2.Text = AppResources.SendWithMail;
61:             ApplicationBar.Buttons.Add(button2);
62:             button2.Click += new EventHandler(SendMailButton_Click);
63:
64:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
65:             button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
66:             button3.Text = AppResources.SendWithSMS;
67:             ApplicationBar.Buttons.Add(button3);
68:             button3.Click += new EventHandler(SendSMSButton_Click);
69:
70:             ApplicationBarIconButton button4 = new ApplicationBarIconButton();
71:             button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
72:             button4.Text = AppResources.ShareAlbum;
73:             ApplicationBar.Buttons.Add(button4);
74:             button4.Click += new EventHandler(ShareAlbumButton_Click);
75:
76:             isFilled = false;
77:
78:             ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
79:             menuItem1.Text = AppResources.DeleteAlbum;
80:             ApplicationBar.MenuItems.Add(menuItem1);
81:             menuItem1.Click += new EventHandler>DeleteAlbumMenuItem_Click);
82:
83:         }
84:
85:         private void SendSMSButton_Click(object sender, EventArgs e)
86:         {

```

```

87:         SmsComposeTask smsComposeTask = new SmsComposeTask();
88:
89:         smsComposeTask.To = "";
90:         smsComposeTask.Body = CreateSendMaterial();
91:
92:         smsComposeTask.Show();
93:         //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
94:     }
95:
96:     private void ShareAlbumButton_Click(object sender, EventArgs e)
97:     {
98:         ShareStatusTask shareStatusTask = new ShareStatusTask();
99:
100:        shareStatusTask.Status = CreateSendMaterial();
101:
102:        shareStatusTask.Show();
103:        //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
104:    }
105:
106:    private void SendMailButton_Click(object sender, EventArgs e)
107:    {
108:        // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
109:        //EmailAddressChooserTask emailAddressChooserTask;
110:        //emailAddressChooserTask = new EmailAddressChooserTask();
111:        //emailAddressChooserTask.Completed += new
112:        //EventHandler<EmailResult>(emailAddressChooserTask_Completed);
113:        //emailAddressChooserTask.Show();
114:
115:        EmailComposeTask emailComposeTask = new EmailComposeTask();
116:
117:        emailComposeTask.Subject = txtAlbumName.Text;
118:        emailComposeTask.Body = CreateSendMaterial();
119:        emailComposeTask.To = "";
120:        emailComposeTask.Cc = "";
121:        emailComposeTask.Bcc = "";
122:
123:        emailComposeTask.Show();
124:        //MessageBox.Show(AppResources.SuccessfulSendWithMail);
125:    }
126:
127:    private string CreateSendMaterial()
128:    {
129:        StringBuilder sb = new StringBuilder();
130:        sb.AppendLine(AppResources.AlbumName + ": " + txtAlbumName.Text);
131:        sb.AppendLine(AppResources.CategoryName + ": " + categoryName);
132:        sb.AppendLine(AppResources.ArtistName + ": " + artistName);
133:        sb.AppendLine(AppResources.ReleaseYear + ": " + txtReleaseYear.Text);
134:        sb.AppendLine(AppResources.SongCount + ": " + txtSongCount.Text);
135:        sb.AppendLine(AppResources.LabelName + ": " + txtLabelName.Text);
136:        sb.AppendLine(AppResources.BestSong + ": " + txtBestSong.Text);
137:        sb.AppendLine(AppResources.AlbumComment + ": " + txtAlbumComment.Text);
138:        sb.AppendLine(AppResources.AlbumRating + ": " + rtRating.Value.ToString()
139:        + "/10");
140:        sb.AppendLine();
141:        sb.AppendLine();
142:        sb.AppendLine(AppResources.SendWithApp);
143:        return sb.ToString();
144:    }
145:
146:    private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
147:    {
148:        //SetBackgroundColor();
149:
150:        // yazarin adi sayfanin en üstünde görünsün diye yapıyor bu
151:        //pvArtist.Title = artistName;
152:
153:        //pvArtist.Title = artistName;
154:        //piAlbumName.Header = AppResources.AlbumName;
155:        //piComment.Header = AppResources.AlbumComment;

```

```

154:         //piLabelName.Header = AppResources.LabelName;
155:         //piRating.Header = AppResources.AlbumRating;
156:         //piStartFinishDate.Header = AppResources.Date;
157:         //lblStartDate.Text = AppResources.StartDate;
158:         //lblFinishDate.Text = AppResources.FinishDate;
159:         //piReleaseYear.Header = AppResources.ReleaseYear;
160:     }
161:
162:     private void SetBackgroundColor()
163:     {
164:         AppSettings appSettings = new AppSettings();
165:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
166:         {
167:             appSettings = context.AppSettings.First() as AppSettings;
168:         }
169:
170:         if (appSettings.AppBackgroundImage != null)
171:         {
172:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
173:             BitmapImage image = new BitmapImage();
174:             image.SetSource(stream);
175:             ImageBrush ib = new ImageBrush();
176:             ib.ImageSource = image;
177:             this.LayoutRoot.Background = ib;
178:         }
179:         else
180:         {
181:             switch (appSettings.AppBackgroundColor)
182:             {
183:                 case "BLA":
184:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
185:                     break;
186:                 case "BLU":
187:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
188:                     break;
189:                 case "BRO":
190:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
191:                     break;
192:                 case "RED":
193:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
194:                     break;
195:                 case "GRE":
196:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
197:                     break;
198:                 case "GRA":
199:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
200:                     break;
201:                 case "YEL":
202:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
203:                     break;
204:                 case "ORA":
205:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
206:                     break;
207:                 case "PUR":
208:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
209:                     break;
210:                 default:
211:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
212:                     break;
213:             }
214:         }
215:     }
216:
217:     protected override void OnNavigatedTo(NavigationEventArgs e)
218:     {

```



```

219:         base.OnNavigatedTo(e);
220:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
221:         {
222:             var appSettings = context.AppSettings.First();
223:             categoryId = appSettings.CurrentCategoryNumber;
224:             artistId = appSettings.CurrentArtistNumber;
225:
226:             // sayfanin font ayarlari için yapılan bir degisiklik
227:             FontFamily temp = new FontFamily(appSettings.FontFamily);
228:             double fontsize = double.Parse(appSettings.FontSize);
229:             txtAlbumComment.FontFamily = temp;
230:             txtAlbumComment.FontSize = fontsize;
231:             txtReleaseYear.FontFamily = temp;
232:             txtReleaseYear.FontSize = fontsize;
233:             txtAlbumName.FontFamily = temp;
234:             txtAlbumName.FontSize = fontsize;
235:             txtAlbumComment.FontFamily = temp;
236:             txtAlbumComment.FontSize = fontsize;
237:             txtLabelName.FontFamily = temp;
238:             txtLabelName.FontSize = fontsize;
239:             txtBestSong.FontFamily = temp;
240:             txtBestSong.FontSize = fontsize;
241:             txtSongCount.FontFamily = temp;
242:             txtSongCount.FontSize = fontsize;
243:             // oylamada kolaylik olmasi için otomatik olarak 5 veriliyor
244:             // sonradan istenirse 0 da verilebilir.
245:             rtRating.Value = 5;
246:
247:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
248:             artistName = artist.ArtistName;
249:
250:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
251:             categoryName = category.CategoryName;
252:         }
253:
254:         var lastPage = NavigationService.BackStack.FirstOrDefault();
255:         pageName = lastPage.Source.ToString();
256:         pvArtist.SelectedIndex = 0;
257:         txtAlbumName.Focus();
258:         // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
259:         pvArtist.Title = artistName;
260:         SetBackgroundColor();
261:     }
262:
263:     protected override void OnNavigatedFrom(NavigationEventArgs e)
264:     {
265:         base.OnNavigatedFrom(e);
266:         //while (NavigationService.CanGoBack)
267:         //NavigationService.RemoveBackEntry();
268:     }
269:
270:
271:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
272:     {
273:         // displays "Fragment: Detail"
274:         //MessageBox.Show("Folder Id: " + e.Fragment);
275:         base.OnFragmentNavigation(e);
276:         albumId = int.Parse(e.Fragment);
277:         if (pageName.Contains("/ArtistPage.xaml"))
278:         {
279:             isFilled = true;
280:         }
281:         else
282:         {
283:             //using (var context2 = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))

```

```

284:         //{
285:         //     var appSettings = context2.AppSettings; ;
286:         //     var album2 = context2.Albums.Where(j =>
j.AlbumId.Equals(albumId)) as Album;
287:         //     var albumArtist = context2.AlbumArtists.Where(j =>
j.AlbumId.Equals(albumId)).ToList() as List<AlbumArtist>;
288:         //     var bArtist = albumArtist.First();
289:         //     var artist = context2.Artists.Where(j =>
j.ArtistId.Equals(bArtist.ArtistId)) as Artist;
290:         //     foreach (var item in appSettings)
291:         //     {
292:         //         item.CurrentArtistNumber = artist.ArtistId;
293:         //         item.CurrentCategoryNumber = album2.AlbumCategoryId;
294:         //     }
295:         //     context2.SubmitChanges();
296:         //     pvArtist.Title = artist.ArtistName;
297:         //}
298:     }
299:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
300:     {
301:         var album = context.Albums.Where(j =>
j.AlbumId.Equals(e.Fragment)).Single() as Album;
302:
303:         txtAlbumName.Text = album.AlbumName == "" ? "" : album.AlbumName;
304:         txtSongCount.Text = album.AlbumSongCount.ToString() == "" ? "" :
album.AlbumSongCount.ToString();
305:         txtReleaseYear.Text = album.AlbumReleaseYear.ToString() == "" ? "" :
album.AlbumReleaseYear.ToString();
306:         txtLabelName.Text = album.AlbumLabelName == "" ? "" :
album.AlbumLabelName;
307:         txtBestSong.Text = album.AlbumBestSong == "" ? "" :
album.AlbumBestSong;
308:         //dtStart.Value = album.ReadStartDate == null ? DateTime.Now :
album.ReadStartDate;
309:         //dtFinish.Value = album.ReadFinishDate == null ? DateTime.Now :
album.ReadFinishDate;
310:         rtRating.Value = album.AlbumRating == null ? 0 : album.AlbumRating;
311:         txtAlbumComment.Text = album.AlbumComment == "" ? "" :
album.AlbumComment;
312:     }
313:
314:     SetBackgroundColor();
315:     pvArtist.SelectedIndex = 0;
316:     //pvArtist.Name = artistName;
317:     txtAlbumName.Focus();
318: }
319:
320: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
321: {
322:     //SaveButton_Click(this, new EventArgs());
323:     if (pageName.Contains("/SearchPage.xaml"))
324:     {
325:         //this.NavigationService.Navigate(new Uri("/SearchPage.xaml",
UriKind.Relative));
326:     }
327:     else
328:     {
329:         this.NavigationService.Navigate(new Uri("/ArtistPage.xaml#" +
artistId, UriKind.Relative));
330:     }
331: }
332:
333: private void SaveButton_Click(object sender, EventArgs e)
334: {
335:     txtAlbumComment_LostFocus(this, new RoutedEventArgs());
336:     this.pnlKeyboardPlaceHolder.Visibility = Visibility.Collapsed;
337:     if (txtAlbumName.Text.Trim().Length < 1)

```

```

338:         {
339:             MessageBox.Show(AppResources.AlbumNameMustBe);
340:         }
341:         else
342:         {
343:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
344:             {
345:                 if (isFilled || pageName.Contains("/SearchPage.xaml"))
346:                 {
347:                     var album = context.Albums.Where(j =>
j.AlbumId.Equals(albumId)).Select(j => j);
348:                     foreach (var item in album)
349:                     {
350:                         item.AlbumCategoryId = categoryId;
351:                         item.AlbumName = txtAlbumName.Text == "" ? "" :
txtAlbumName.Text;
352:                         item.AlbumReleaseYear = txtReleaseYear.Text == "" ? 0 :
int.Parse(txtReleaseYear.Text);
353:                         item.AlbumLabelName = txtLabelName.Text == "" ? "" :
txtLabelName.Text;
354:                         item.AlbumSongCount = txtSongCount.Text == "" ? 0 :
int.Parse(txtSongCount.Text);
355:                         item.AlbumBestSong = txtBestSong.Text == "" ? "" :
txtBestSong.Text;
356:                         //item.ReadStartDate =
DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtStart.Value.ToString());
357:                         //item.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString());
358:                         //item.AlbumRating = rtRating.Value.ToString() == "" ? 0 :
int.Parse(rtRating.Value.ToString());
359:                         item.AlbumRating = int.Parse(ratingValue.ToString()) == 0
? 0 : int.Parse(ratingValue.ToString());
360:                         item.AlbumComment = txtAlbumComment.Text == "" ? "" :
txtAlbumComment.Text;
361:                         item.ModificationDate = DateTime.Now;
362:                         item.AlbumInformation = categoryName + " " + artistName +
" " + txtAlbumName.Text + " " + txtLabelName.Text + " " + txtAlbumComment.Text;
363:                         item.AlbumNameRating = item.AlbumName + " (" +
item.AlbumRating + "/10)";
364:                     }
365:                     context.SubmitChanges();
366:                 }
367:                 else
368:                 {
369:                     Album album = new Album();
370:                     album.AlbumCategoryId = categoryId;
371:                     album.AlbumGuid = Guid.NewGuid().ToString();
372:                     album.AlbumName = txtAlbumName.Text == "" ? "" :
txtAlbumName.Text;
373:                     album.AlbumReleaseYear = txtReleaseYear.Text == "" ? 0 :
int.Parse(txtReleaseYear.Text);
374:                     album.AlbumLabelName = txtLabelName.Text == "" ? "" :
txtLabelName.Text;
375:                     album.AlbumSongCount = txtSongCount.Text == "" ? 0 :
int.Parse(txtSongCount.Text);
376:                     album.AlbumBestSong = txtBestSong.Text == "" ? "" :
txtBestSong.Text;
377:                     //album.ReadStartDate =
DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtStart.Value.ToString());
378:                     //album.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString());
379:                     //album.AlbumRating = rtRating.Value.ToString() == "" ? 0 :
int.Parse(rtRating.Value.ToString());
380:                     //album.AlbumRating = rtRating.Value.ToString() == "" ? 0 :
int.Parse(rtRating.Value.ToString());

```

```

381:                album.AlbumRating = int.Parse(ratingValue.ToString()) == 0 ? 0
: int.Parse(ratingValue.ToString());
382:                album.AlbumComment = txtAlbumComment.Text == "" ? "" :
txtAlbumComment.Text;
383:                album.ModificationDate = DateTime.Now;
384:                album.AlbumInformation = categoryName + " " + artistName + " "
+ album.AlbumName + " " + album.AlbumReleaseYear.ToString() + " " + album.AlbumLabelName +
" " + album.AlbumComment;
385:                album.CreationDate = DateTime.Now;
386:                album.AlbumNameRating = album.AlbumName + " (" +
album.AlbumRating + "/10)";
387:                context.Albums.InsertOnSubmit(album);
388:                context.SubmitChanges();
389:
390:                Album album2 = context.Albums.Where(j =>
j.AlbumGuid.Equals(album.AlbumGuid)).Single() as Album;
391:
392:                AlbumArtist albumArtist = new AlbumArtist();
393:                albumArtist.ArtistId = artistId;
394:                albumArtist.AlbumId = album2.AlbumId;
395:                context.AlbumArtists.InsertOnSubmit(albumArtist);
396:                context.SubmitChanges();
397:
398:                var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
399:                foreach (var item in category)
400:                {
401:                    item.CategoryAlbumCount = item.CategoryAlbumCount + 1;
402:                    item.CategoryNameCount = item.CategoryName + " (" +
item.CategoryAlbumCount + ")";
403:                    item.ModificationDate = DateTime.Now;
404:                }
405:                context.SubmitChanges();
406:
407:                var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Select(j => j);
408:                foreach (var item in artist)
409:                {
410:                    item.ArtistAlbumCount = item.ArtistAlbumCount + 1;
411:                    item.ArtistNameCount = item.ArtistName + " (" +
item.ArtistAlbumCount + ")";
412:                    item.ModificationDate = DateTime.Now;
413:                }
414:                context.SubmitChanges();
415:            }
416:        }
417:        MessageBox.Show(AppResources.AlbumSaveSuccess);
418:    }
419:    isFilled = false;
420:}
421:
422:private void txtAlbumComment_TextChanged(object sender, TextChangedEventArgs e)
423:    {
424:        Dispatcher.BeginInvoke(() =>
425:        {
426:            double CurrentInputHeight = txtAlbumComment.ActualHeight;
427:
428:            if (CurrentInputHeight > InputHeight)
429:            {
430:
svAlbumComment.ScrollToVerticalOffset(svAlbumComment.VerticalOffset + CurrentInputHeight -
InputHeight);
431:            }
432:
433:            InputHeight = CurrentInputHeight;
434:        });
435:    }
436:
437:    private void txtAlbumComment_GotFocus(object sender, RoutedEventArgs e)

```

```

438:         {
439:             App.RootFrame.RenderTransform = new CompositeTransform();
440:             flag = true;
441:         }
442:
443:         private void txtAlbumComment_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
444:         {
445:             txtAlbumComment.Focus();
446:             //txtAlbumComment.Select(txtAlbumComment.Text.Length, 1);
447:             svAlbumComment.ScrollToVerticalOffset(e.GetPosition(txtAlbumComment).Y -
80);
448:         }
449:
450:         private void txtAlbumComment_LostFocus(object sender, RoutedEventArgs e)
451:         {
452:             if (!flag) return;
453:             txtAlbumComment.Focus();
454:             flag = false;
455:             this.pnlKeyboardPlaceholder.Visibility = Visibility.Collapsed;
456:         }
457:
458:         private void txtAlbumComment_KeyDown(object sender, KeyEventArgs e)
459:         {
460:             if (e.Key == Key.Enter)
461:             {
462:                 svAlbumComment.ScrollToVerticalOffset(txtAlbumComment.ActualHeight);
463:             }
464:         }
465:
466:         private void svAlbumComment_GotFocus(object sender, RoutedEventArgs e)
467:         {
468:             this.svAlbumComment.ScrollToVerticalOffset(this.txtAlbumComment.ActualHeight);
469:             this.svAlbumComment.UpdateLayout();
470:         }
471:
472:         private void txtAlbumName_KeyDown(object sender, KeyEventArgs e)
473:         {
474:             if (e.Key == Key.Enter)
475:             {
476:                 pvArtist.SelectedIndex = 1;
477:                 txtReleaseYear.Focus();
478:             }
479:         }
480:
481:         private void txtReleaseYear_KeyDown(object sender, KeyEventArgs e)
482:         {
483:             if (e.Key == Key.Enter)
484:             {
485:                 pvArtist.SelectedIndex = 2;
486:                 txtSongCount.Focus();
487:             }
488:         }
489:
490:         private void txtLabelName_KeyDown(object sender, KeyEventArgs e)
491:         {
492:             if (e.Key == Key.Enter)
493:             {
494:                 pvArtist.SelectedIndex = 4;
495:                 txtBestSong.Focus();
496:             }
497:         }
498:
499:
500:         private void rtRating_KeyDown(object sender, KeyEventArgs e)
501:         {
502:             //if (e.Key == Key.Enter)
503:             //{

```

```

504:         //     pvArtist.SelectedIndex = 5;
505:         //     txtAlbumComment.Focus();
506:         //}
507:     }
508:
509:     private void rtRating_ValueChanged(object sender, EventArgs e)
510:     {
511:         //pvArtist.SelectedIndex = 5;
512:         ratingValue = rtRating.Value;
513:         //txtAlbumComment.Focus();
514:     }
515:
516:     private void dtFinish_ValueChanged(object sender,
DateTimeValueChangedEventArgs e)
517:     {
518:         //if (isFilled)
519:         //{
520:             //     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
521:             //     {
522:                 //         var album = context.Albums.Where(j =>
j.AlbumId.Equals(albumId)).Select(j => j);
523:                 //         foreach (var item in album)
524:                 //         {
525:                     //             item.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString());
526:                     //             item.ModificationDate = DateTime.Now;
527:                 //         }
528:                 //         context.SubmitChanges();
529:             //     }
530:         //}
531:         //pvArtist.SelectedIndex = 4;
532:         //rtRating.Focus();
533:     }
534:
535:     private void dtStart_ValueChanged(object sender, DateTimeValueChangedEventArgs
e)
536:     {
537:         //if (isFilled)
538:         //{
539:             //     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
540:             //     {
541:                 //         var album = context.Albums.Where(j =>
j.AlbumId.Equals(albumId)).Select(j => j);
542:                 //         foreach (var item in album)
543:                 //         {
544:                     //             item.ReadStartDate =
DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtStart.Value.ToString());
545:                     //             item.ModificationDate = DateTime.Now;
546:                 //         }
547:                 //         context.SubmitChanges();
548:             //     }
549:         //}
550:         //pvArtist.SelectedIndex = 3;
551:         //dtFinish.Focus();
552:     }
553:
554:     private void rtRating_Tap(object sender, System.Windows.Input.GestureEventArgs
e)
555:     {
556:         //pvArtist.SelectedIndex = 5;
557:         //ratingValue = rtRating.Value;
558:         //txtAlbumComment.Focus();
559:     }
560:
561:     private void DeleteAlbumMenuItem_Click(object sender, EventArgs e)

```

```

562:         {
563:             if (MessageBox.Show(AppResources.DeleteAlbumQuestion,
564:                 AppResources.DeleteAlbum, MessageBoxButton.OKCancel)
565:                 != MessageBoxResult.OK)
566:             {
567:             }
568:             else
569:             {
570:                 using (var context = new
571: AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
572:                 {
573:                     var album = context.Albums.Where(j =>
574: j.AlbumId.Equals(albumId)).Single() as Album;
575:                     var albumArtists = context.AlbumArtists.Where(j =>
576: j.AlbumId.Equals(albumId)).ToList() as List<AlbumArtist>;
577:                     context.AlbumArtists.DeleteAllOnSubmit(albumArtists);
578:                     context.Albums.DeleteOnSubmit(album);
579:                     var artists = context.Artists.Where(j =>
580: j.ArtistId.Equals(artistId)).Select(j => j);
581:                     foreach (var item in artists)
582:                     {
583:                         item.ModificationDate = DateTime.Now;
584:                         item.ArtistAlbumCount = item.ArtistAlbumCount - 1;
585:                         item.ArtistNameCount = item.ArtistNameCount + " (" +
586: item.ArtistAlbumCount + ")";
587:                     }
588:                     context.SubmitChanges();
589:                     var categories = context.Categories.Where(j =>
590: j.CategoryId.Equals(categoryId)).Select(j => j);
591:                     foreach (var item in categories)
592:                     {
593:                         item.ModificationDate = DateTime.Now;
594:                         item.CategoryAlbumCount = item.CategoryAlbumCount - 1;
595:                         item.CategoryNameCount = item.CategoryNameCount + " (" +
596: item.CategoryAlbumCount + ")";
597:                     }
598:                     context.SubmitChanges();
599:                     MessageBox.Show(AppResources.AlbumDeleteSuccess);
600:                     NavigationService.Navigate(new Uri("/MainPage.xaml",
601: UriKind.Relative));
602:                 }
603:                 //MessageBox.Show(AppResources.NoteSaved);
604:             }
605:         }
606:         private void btnIncrease_Click(object sender, RoutedEventArgs e)
607:         {
608:             if (rtRating.Value != 10.0)
609:             {
610:                 rtRating.Value = rtRating.Value + 1.0;
611:             }
612:         }
613:         private void btnDecrease_Click(object sender, RoutedEventArgs e)
614:         {
615:             if (rtRating.Value != 0.0)
616:             {
617:                 rtRating.Value = rtRating.Value - 1.0;
618:             }
619:         }
620:         private void txtSongCount_KeyDown(object sender, KeyEventArgs e)
621:         {
622:             if (e.Key == Key.Enter)
623:             {
624:                 pvArtist.SelectedIndex = 3;

```

```
623:         txtLabelName.Focus();
624:     }
625: }
626:
627: private void txtBestSong_KeyDown(object sender, KeyEventArgs e)
628: {
629:     if (e.Key == Key.Enter)
630:     {
631:         pvArtist.SelectedIndex = 5;
632:         rtRating.Focus();
633:     }
634: }
635: }
636: }
```

Namespaces

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

1.2.6 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 AwesomeMusic.Resources;
13: using System.Collections.Generic;
14: using Microsoft.Phone.Marketplace;
15:
16:
17: namespace AwesomeMusic
18: {
19:     public partial class App : Application
20:     {
21:         /// <summary>
22:         /// Provides easy access to the root frame of the Phone Application.
23:         /// </summary>
24:         /// <returns>The root frame of the Phone Application.</returns>
25:         ///
26:
27:
28:         // lisans bilgisi için gerekli olan bir sey
29:         private static bool _isTrial = true;
30:
31:         private static LicenseInformation _licenseInfo = new LicenseInformation();
32:         public bool IsTrial
33:         {
34:             get
35:             {
36:                 return _isTrial;
37:             }
38:         }
39:     }
40: }
```



```

40:
41:     public int categoryNumber;
42:
43:     public static PhoneApplicationFrame RootFrame { get; private set; }
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:         // ayarlardan temasi açık renk bile olsa
57:         // kapali gibi çalışmasını sağlayacak bir nuget paketi yüklendi
58:         // bu sorunu gideriyor
59:         ThemeManager.ToDarkTheme();
60:
61:         // Phone-specific initialization
62:         InitializePhoneApplication();
63:
64:         // Language display initialization
65:         InitializeLanguage();
66:
67:         // Show graphics profiling information while debugging.
68:         if (Debugger.IsAttached)
69:         {
70:             // Display the current frame rate counters.
71:             Application.Current.Host.Settings.EnableFrameRateCounter = true;
72:
73:             // Show the areas of the app that are being redrawn in each frame.
74:             //Application.Current.Host.Settings.EnableRedrawRegions = true;
75:
76:             // Enable non-production analysis visualization mode,
77:             // which shows areas of a page that are handed off to GPU with a
colored overlay.
78:             //Application.Current.Host.Settings.EnableCacheVisualization = true;
79:
80:             // Prevent the screen from turning off while under the debugger by
disabling
81:             // the application's idle detection.
82:             // Caution:- Use this under debug mode only. Application that disables
user idle detection will continue to run
83:             // and consume battery power when the user is not using the phone.
84:             PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
85:         }
86:
87:     }
88:
89:     // Code to execute when the application is launching (eg, from Start)
90:     // This code will not execute when the application is reactivated
91:     private void Application_Launching(object sender, LaunchingEventArgs e)
92:     {
93:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
94:         {
95:             if (!context.DatabaseExists())
96:             {
97:                 context.CreateDatabase();
98:                 DilAyariOlustur(context);
99:             }
100:             else
101:             {
102:                 using (var context2 = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))

```

```

103:         {
104:
105:             AppSettings lang =
106:                 context2.AppSettings.First() as AppSettings;
107:             string culture = "";
108:             switch (lang.AppLangName)
109:             {
110:                 case "TR":
111:                     culture = "tr";
112:                     break;
113:                 case "EN":
114:                     culture = "en";
115:                     break;
116:                 case "DE":
117:                     culture = "de";
118:                     break;
119:                 case "ES":
120:                     culture = "es";
121:                     break;
122:                 case "FR":
123:                     culture = "fr";
124:                     break;
125:                 case "IT":
126:                     culture = "it";
127:                     break;
128:                 case "AR":
129:                     culture = "ar";
130:                     break;
131:                 case "FA":
132:                     culture = "fa-IR";
133:                     break;
134:                 case "ZH":
135:                     culture = "zh";
136:                     break;
137:                 case "PT":
138:                     culture = "pt";
139:                     break;
140:                 case "RU":
141:                     culture = "ru";
142:                     break;
143:                 case "JA":
144:                     culture = "ja";
145:                     break;
146:                 case "SA":
147:                     culture = "sa";
148:                     break;
149:                 case "TH":
150:                     culture = "th";
151:                     break;
152:                 default:
153:                     culture = "tr-TR";
154:                     break;
155:             }
156:             CultureInfo newCulture = new CultureInfo(culture);
157:             Thread.CurrentThread.CurrentCulture = newCulture;
158:             Thread.CurrentThread.CurrentUICulture = newCulture;
159:         }
160:     }
161:
162:     // kullanıcının programla ilgili bilgilendirici notları kendi dilinde
163:     // görebilmesi için burada ekliyoruz.
164: }
165:
166: // Code to execute when the application is activated (brought to foreground)
167: // This code will not execute when the application is first launched
168: private void Application_Activated(object sender, ActivatedEventArgs e)
169: {
170: }

```

```

171:
172:     // Code to execute when the application is deactivated (sent to background)
173:     // This code will not execute when the application is closing
174:     private void Application_Deactivated(object sender, DeactivatedEventArgs e)
175:     {
176:     }
177:
178:     // Code to execute when the application is closing (eg, user hit Back)
179:     // This code will not execute when the application is deactivated
180:     private void Application_Closing(object sender, ClosingEventArgs e)
181:     {
182:     }
183:
184:     // Code to execute if a navigation fails
185:     private void RootFrame_NavigationFailed(object sender,
NavigationFailedEventArgs e)
186:     {
187:         if (Debugger.IsAttached)
188:         {
189:             // A navigation has failed; break into the debugger
190:             Debugger.Break();
191:         }
192:     }
193:
194:     // Code to execute on Unhandled Exceptions
195:     private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
196:     {
197:         if (Debugger.IsAttached)
198:         {
199:             // An unhandled exception has occurred; break into the debugger
200:             Debugger.Break();
201:         }
202:     }
203:
204:     #region Phone application initialization
205:
206:     // Avoid double-initialization
207:     private bool phoneApplicationInitialized = false;
208:
209:     // Do not add any additional code to this method
210:     private void InitializePhoneApplication()
211:     {
212:         if (phoneApplicationInitialized)
213:             return;
214:
215:         // Create the frame but don't set it as RootVisual yet; this allows the
splash
216:         // screen to remain active until the application is ready to render.
217:         RootFrame = new PhoneApplicationFrame();
218:         RootFrame.Navigated += CompleteInitializePhoneApplication;
219:
220:         // Handle navigation failures
221:         RootFrame.NavigationFailed += RootFrame_NavigationFailed;
222:
223:         // Handle reset requests for clearing the backstack
224:         RootFrame.Navigated += CheckForResetNavigation;
225:
226:         // Ensure we don't initialize again
227:         phoneApplicationInitialized = true;
228:     }
229:
230:     // Do not add any additional code to this method
231:     private void CompleteInitializePhoneApplication(object sender,
NavigationEventArgs e)
232:     {
233:         // Set the root visual to allow the application to render
234:         if (RootVisual != RootFrame)
235:             RootVisual = RootFrame;

```

```

236:
237:         // Remove this handler since it is no longer needed
238:         RootFrame.Navigated -= CompleteInitializePhoneApplication;
239:     }
240:
241:     private void CheckForResetNavigation(object sender, NavigationEventArgs e)
242:     {
243:         // If the app has received a 'reset' navigation, then we need to check
244:         // on the next navigation to see if the page stack should be reset
245:         if (e.NavigationMode == NavigationMode.Reset)
246:             RootFrame.Navigated += ClearBackStackAfterReset;
247:     }
248:
249:     private void ClearBackStackAfterReset(object sender, NavigationEventArgs e)
250:     {
251:         // Unregister the event so it doesn't get called again
252:         RootFrame.Navigated -= ClearBackStackAfterReset;
253:
254:         // Only clear the stack for 'new' (forward) and 'refresh' navigations
255:         if (e.NavigationMode != NavigationMode.New && e.NavigationMode !=
NavigationMode.Refresh)
256:             return;
257:
258:         // For UI consistency, clear the entire page stack
259:         while (RootFrame.RemoveBackEntry() != null)
260:         {
261:             ; // do nothing
262:         }
263:     }
264:
265:     #endregion
266:
267:     // Initialize the app's font and flow direction as defined in its localized
resource strings.
268:     //
269:     // To ensure that the font of your application is aligned with its supported
languages and that the
270:     // FlowDirection for each of those languages follows its traditional
direction, ResourceLanguage
271:     // and ResourceFlowDirection should be initialized in each resx file to match
these values with that
272:     // file's culture. For example:
273:     //
274:     // AppResources.es-ES.resx
275:     //     ResourceLanguage's value should be "es-ES"
276:     //     ResourceFlowDirection's value should be "LeftToRight"
277:     //
278:     // AppResources.ar-SA.resx
279:     //     ResourceLanguage's value should be "ar-SA"
280:     //     ResourceFlowDirection's value should be "RightToLeft"
281:     //
282:     // For more info on localizing Windows Phone apps see
http://go.microsoft.com/fwlink/?LinkId=262072.
283:     //
284:     private void InitializeLanguage()
285:     {
286:         try
287:         {
288:             // Set the font to match the display language defined by the
289:             // ResourceLanguage resource string for each supported language.
290:             //
291:             // Fall back to the font of the neutral language if the Display
292:             // language of the phone is not supported.
293:             //
294:             // If a compiler error is hit then ResourceLanguage is missing from
295:             // the resource file.
296:             RootFrame.Language =
XmlLanguage.GetLanguage(AppResources.ResourceLanguage);
297:

```

```

298:         // Set the FlowDirection of all elements under the root frame based
299:         // on the ResourceFlowDirection resource string for each
300:         // supported language.
301:         //
302:         // If a compiler error is hit then ResourceFlowDirection is missing
from
303:         // the resource file.
304:         FlowDirection flow = (FlowDirection)Enum.Parse(typeof(FlowDirection),
AppResources.ResourceFlowDirection);
305:         RootFrame.FlowDirection = flow;
306:     }
307:     catch
308:     {
309:         // If an exception is caught here it is most likely due to either
310:         // ResourceLanguage not being correctly set to a supported language
311:         // code or ResourceFlowDirection is set to a value other than
LeftToRight
312:         // or RightToLeft.
313:
314:         if (Debugger.IsAttached)
315:         {
316:             Debugger.Break();
317:         }
318:
319:         throw;
320:     }
321: }
322:
323: private void CheckLicense()
324: {
325:     // When debugging, we want to simulate a trial mode experience. The
following conditional allows us to set the _isTrial
326:     // property to simulate trial mode being on or off.
327:     #if DEBUG
328:         string message = "This sample demonstrates the implementation of a trial
mode in an application." +
329:             "Press 'OK' to simulate trial mode. Press 'Cancel' to
run the application in normal mode.";
330:         if (MessageBox.Show(message, "Debug Trial",
331:             MessageBoxButton.OKCancel) == MessageBoxResult.OK)
332:         {
333:             _isTrial = true;
334:         }
335:         else
336:         {
337:             _isTrial = false;
338:         }
339:     #else
340:         _isTrial = _licenseInfo.IsTrial();
341:     #endif
342: }
343:
344: private void Application_Startup(object sender, StartupEventArgs e)
345: {
346:     //using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
347:     //{
348:         // if (!context.DatabaseExists())
349:         //{
350:             // context.CreateDatabase();
351:             // DilAyariOlustur(context);
352:         // }
353:         // else
354:         //{
355:             // using (var context2 = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
356:             //{
357:
358:                 // AppSettings lang =

```

```

359:         // context2.AppSettings.First() as AppSettings;
360:         // string culture = "";
361:         // switch (lang.AppLangName)
362:         // {
363:         //     case "TR":
364:         //         culture = "tr";
365:         //         break;
366:         //     case "EN":
367:         //         culture = "en";
368:         //         break;
369:         //     case "DE":
370:         //         culture = "de";
371:         //         break;
372:         //     case "ES":
373:         //         culture = "es";
374:         //         break;
375:         //     case "FR":
376:         //         culture = "fr";
377:         //         break;
378:         //     case "IT":
379:         //         culture = "it";
380:         //         break;
381:         //     case "AR":
382:         //         culture = "ar";
383:         //         break;
384:         //     case "FA":
385:         //         culture = "fa-IR";
386:         //         break;
387:         //     case "ZH":
388:         //         culture = "zh";
389:         //         break;
390:         //     case "PT":
391:         //         culture = "pt";
392:         //         break;
393:         //     case "RU":
394:         //         culture = "ru";
395:         //         break;
396:         //     case "JA":
397:         //         culture = "ja";
398:         //         break;
399:         //     default:
400:         //         culture = "tr-TR";
401:         //         break;
402:         // }
403:         // CultureInfo newCulture = new CultureInfo(culture);
404:         // Thread.CurrentThread.CurrentCulture = newCulture;
405:         // Thread.CurrentThread.CurrentUICulture = newCulture;
406:         // }
407:         // }
408:
409:         // // kullanıcının programla ilgili bilgilendirici notları kendi
410:         // // dilinde görebilmesi için burada ekliyoruz.
411:         // }
412:
413:         private static void DilAyariOlustur(AwesomeMusicDataContext context)
414:         {
415:             var appSettings = new AppSettings()
416:             {
417:                 //AppLangId = 42,
418:                 AppLangName = "EN",
419:                 AppBackgroundColor = "BLA",
420:                 CategoryOrderBy = "NAME",
421:                 CategoryOrderStyle = "A",
422:                 CurrentCategoryNumber = 0,
423:                 CurrentArtistNumber = 0,
424:                 FontFamily = "Verdana",
425:                 FontSize = "30",
426:                 AppBackgroundImage = null

```

```
427:         };
428:
429:         context.AppSettings.InsertOnSubmit(appSettings);
430:         context.SubmitChanges();
431:
432:         CultureInfo newCulture = new CultureInfo("en");
433:         Thread.CurrentThread.CurrentCulture = newCulture;
434:         Thread.CurrentThread.CurrentUICulture = newCulture;
435:     }
436: }
437: }
```

Namespaces

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

1.2.7 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 AwesomeMusic.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:
23: [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
24: "4.0.0.0")]
25:     [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
26:     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
27:     public class AppResources {
28:
29:         private static global::System.Resources.ResourceManager resourceMan;
30:
31:         private static global::System.Globalization.CultureInfo resourceCulture;
32:
33: [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
34: "CA1811:AvoidUncalledPrivateCode")]
35:         internal AppResources() {
36:
37:             /// <summary>
```

```

36:         /// Returns the cached ResourceManager instance used by this class.
37:         /// </summary>
38:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.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("AwesomeMusic.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.EditorBrowsableState.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 About.
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 About The App.
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 Hi. I like listening album a
lot. And after Awesome Library app, i decide to create an app which has similar properties
like PitchFork. You can add categories, add artists and add albums on it. You can send
information of your albums via SMS, E-Mail and Social Media share (like Facebook etc.). I
hope that you will like this app. If you rate app and write your suggestions to marketplace
and coderserdar@outlook.com I will be so appreciated to you. With my best regards. ÇMS
Software..
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 About The Awesome Music.
92:         /// </summary>

```



```
93:         public static string AboutTheAwesomeMusic {
94:             get {
95:                 return ResourceManager.GetString("AboutTheAwesomeMusic",
resourceCulture);
96:             }
97:         }
98:
99:         /// <summary>
100:        ///     Looks up a localized string similar to Add Album.
101:        /// </summary>
102:        public static string AddAlbum {
103:            get {
104:                return ResourceManager.GetString("AddAlbum", resourceCulture);
105:            }
106:        }
107:
108:        /// <summary>
109:        ///     Looks up a localized string similar to Add Artist.
110:        /// </summary>
111:        public static string AddArtist {
112:            get {
113:                return ResourceManager.GetString("AddArtist", resourceCulture);
114:            }
115:        }
116:
117:        /// <summary>
118:        ///     Looks up a localized string similar to Add Category.
119:        /// </summary>
120:        public static string AddCategory {
121:            get {
122:                return ResourceManager.GetString("AddCategory", resourceCulture);
123:            }
124:        }
125:
126:        /// <summary>
127:        ///     Looks up a localized string similar to Album Comment.
128:        /// </summary>
129:        public static string AlbumComment {
130:            get {
131:                return ResourceManager.GetString("AlbumComment", resourceCulture);
132:            }
133:        }
134:
135:        /// <summary>
136:        ///     Looks up a localized string similar to Album Count.
137:        /// </summary>
138:        public static string AlbumCount {
139:            get {
140:                return ResourceManager.GetString("AlbumCount", resourceCulture);
141:            }
142:        }
143:
144:        /// <summary>
145:        ///     Looks up a localized string similar to Album Has Been Removed
Successfully.
146:        /// </summary>
147:        public static string AlbumDeleteSuccess {
148:            get {
149:                return ResourceManager.GetString("AlbumDeleteSuccess",
resourceCulture);
150:            }
151:        }
152:
153:        /// <summary>
154:        ///     Looks up a localized string similar to Album List.
155:        /// </summary>
156:        public static string AlbumList {
157:            get {
158:                return ResourceManager.GetString("AlbumList", resourceCulture);
```

```

159:         }
160:     }
161:
162:     /// <summary>
163:     /// Looks up a localized string similar to Album Name.
164:     /// </summary>
165:     public static string AlbumName {
166:         get {
167:             return ResourceManager.GetString("AlbumName", resourceCulture);
168:         }
169:     }
170:
171:     /// <summary>
172:     /// Looks up a localized string similar to You Have To Enter Album Name At
Least.
173:     /// </summary>
174:     public static string AlbumNameMustBe {
175:         get {
176:             return ResourceManager.GetString("AlbumNameMustBe", resourceCulture);
177:         }
178:     }
179:
180:     /// <summary>
181:     /// Looks up a localized string similar to Album Order By.
182:     /// </summary>
183:     public static string AlbumOrderBy {
184:         get {
185:             return ResourceManager.GetString("AlbumOrderBy", resourceCulture);
186:         }
187:     }
188:
189:     /// <summary>
190:     /// Looks up a localized string similar to Album Order Style.
191:     /// </summary>
192:     public static string AlbumOrderStyle {
193:         get {
194:             return ResourceManager.GetString("AlbumOrderStyle", resourceCulture);
195:         }
196:     }
197:
198:     /// <summary>
199:     /// Looks up a localized string similar to Album Order Style Has Been
Changed Successfully.
200:     /// </summary>
201:     public static string AlbumOrderStyleChangeSuccess {
202:         get {
203:             return ResourceManager.GetString("AlbumOrderStyleChangeSuccess",
resourceCulture);
204:         }
205:     }
206:
207:     /// <summary>
208:     /// Looks up a localized string similar to Album Order Type Has Been
Changed Successfully.
209:     /// </summary>
210:     public static string AlbumOrderTypeChangeSuccess {
211:         get {
212:             return ResourceManager.GetString("AlbumOrderTypeChangeSuccess",
resourceCulture);
213:         }
214:     }
215:
216:     /// <summary>
217:     /// Looks up a localized string similar to Album Rating.
218:     /// </summary>
219:     public static string AlbumRating {
220:         get {
221:             return ResourceManager.GetString("AlbumRating", resourceCulture);
222:         }

```

```

223:     }
224:
225:     /// <summary>
226:     /// Looks up a localized string similar to Album Has Been Saved
Successfully.
227:     /// </summary>
228:     public static string AlbumSaveSuccess {
229:         get {
230:             return ResourceManager.GetString("AlbumSaveSuccess", resourceCulture);
231:         }
232:     }
233:
234:     /// <summary>
235:     /// Looks up a localized string similar to Arabic.
236:     /// </summary>
237:     public static string Arabic {
238:         get {
239:             return ResourceManager.GetString("Arabic", resourceCulture);
240:         }
241:     }
242:
243:     /// <summary>
244:     /// Looks up a localized string similar to Artist Has Been Added
Successfully.
245:     /// </summary>
246:     public static string ArtistAddSuccess {
247:         get {
248:             return ResourceManager.GetString("ArtistAddSuccess", resourceCulture);
249:         }
250:     }
251:
252:     /// <summary>
253:     /// Looks up a localized string similar to Artist Has This Category Already.
254:     /// </summary>
255:     public static string ArtistAlreadySameCategory {
256:         get {
257:             return ResourceManager.GetString("ArtistAlreadySameCategory",
resourceCulture);
258:         }
259:     }
260:
261:     /// <summary>
262:     /// Looks up a localized string similar to Category Has Been Added To
Artist Successfully.
263:     /// </summary>
264:     public static string ArtistCategoryAddSuccess {
265:         get {
266:             return ResourceManager.GetString("ArtistCategoryAddSuccess",
resourceCulture);
267:         }
268:     }
269:
270:     /// <summary>
271:     /// Looks up a localized string similar to Artist Has Been Removed
Successfully.
272:     /// </summary>
273:     public static string ArtistDeleteSuccess {
274:         get {
275:             return ResourceManager.GetString("ArtistDeleteSuccess",
resourceCulture);
276:         }
277:     }
278:
279:     /// <summary>
280:     /// Looks up a localized string similar to This Artist Has Already Exists.
281:     /// </summary>
282:     public static string ArtistExists {
283:         get {
284:             return ResourceManager.GetString("ArtistExists", resourceCulture);

```

```

285:         }
286:     }
287:
288:     /// <summary>
289:     /// Looks up a localized string similar to Artist List.
290:     /// </summary>
291:     public static string ArtistList {
292:         get {
293:             return ResourceManager.GetString("ArtistList", resourceCulture);
294:         }
295:     }
296:
297:     /// <summary>
298:     /// Looks up a localized string similar to Artist Name.
299:     /// </summary>
300:     public static string ArtistName {
301:         get {
302:             return ResourceManager.GetString("ArtistName", resourceCulture);
303:         }
304:     }
305:
306:     /// <summary>
307:     /// Looks up a localized string similar to Artist Name Has Been Changed
308:     /// </summary>
309:     public static string ArtistNameChangeSuccess {
310:         get {
311:             return ResourceManager.GetString("ArtistNameChangeSuccess",
312: resourceCulture);
313:         }
314:     }
315:
316:     /// <summary>
317:     /// Looks up a localized string similar to Artist Order By.
318:     /// </summary>
319:     public static string ArtistOrderBy {
320:         get {
321:             return ResourceManager.GetString("ArtistOrderBy", resourceCulture);
322:         }
323:     }
324:
325:     /// <summary>
326:     /// Looks up a localized string similar to Artist Order Style.
327:     /// </summary>
328:     public static string ArtistOrderStyle {
329:         get {
330:             return ResourceManager.GetString("ArtistOrderStyle", resourceCulture);
331:         }
332:     }
333:
334:     /// <summary>
335:     /// Looks up a localized string similar to Artist Order Style Has Been
336:     /// </summary>
337:     public static string ArtistOrderStyleChangeSuccess {
338:         get {
339:             return ResourceManager.GetString("ArtistOrderStyleChangeSuccess",
340: resourceCulture);
341:         }
342:     }
343:
344:     /// <summary>
345:     /// Looks up a localized string similar to Artist Order Type Has Been
346:     /// </summary>
347:     public static string ArtistOrderTypeChangeSuccess {
348:         get {
349:             return ResourceManager.GetString("ArtistOrderTypeChangeSuccess",
350: resourceCulture);

```

```
348:         }
349:     }
350:
351:     /// <summary>
352:     /// Looks up a localized string similar to Artist Settings.
353:     /// </summary>
354:     public static string ArtistSettings {
355:         get {
356:             return ResourceManager.GetString("ArtistSettings", resourceCulture);
357:         }
358:     }
359:
360:     /// <summary>
361:     /// Looks up a localized string similar to Ascending.
362:     /// </summary>
363:     public static string Ascending {
364:         get {
365:             return ResourceManager.GetString("Ascending", resourceCulture);
366:         }
367:     }
368:
369:     /// <summary>
370:     /// Looks up a localized string similar to Background.
371:     /// </summary>
372:     public static string Background {
373:         get {
374:             return ResourceManager.GetString("Background", resourceCulture);
375:         }
376:     }
377:
378:     /// <summary>
379:     /// Looks up a localized string similar to Background Color.
380:     /// </summary>
381:     public static string BackgroundColor {
382:         get {
383:             return ResourceManager.GetString("BackgroundColor", resourceCulture);
384:         }
385:     }
386:
387:     /// <summary>
388:     /// Looks up a localized string similar to Background Color Has Been
389:     /// </summary>
390:     public static string BackgroundColorChangeSuccess {
391:         get {
392:             return ResourceManager.GetString("BackgroundColorChangeSuccess",
393:             resourceCulture);
394:         }
395:     }
396:
397:     /// <summary>
398:     /// Looks up a localized string similar to Background Image.
399:     /// </summary>
400:     public static string BackgroundImage {
401:         get {
402:             return ResourceManager.GetString("BackgroundImage", resourceCulture);
403:         }
404:     }
405:
406:     /// <summary>
407:     /// Looks up a localized string similar to Background Image Has Been
408:     /// </summary>
409:     public static string BackgroundImageChangeSuccess {
410:         get {
411:             return ResourceManager.GetString("BackgroundImageChangeSuccess",
412:             resourceCulture);
413:         }
414:     }
```

```
413:
414:     /// <summary>
415:     /// Looks up a localized string similar to Background Image Has Been
Removed Successfully.
416:     /// </summary>
417:     public static string BackgroundImageRemoveSuccess {
418:         get {
419:             return ResourceManager.GetString("BackgroundImageRemoveSuccess",
resourceCulture);
420:         }
421:     }
422:
423:     /// <summary>
424:     /// Looks up a localized string similar to Best Album.
425:     /// </summary>
426:     public static string BestAlbum {
427:         get {
428:             return ResourceManager.GetString("BestAlbum", resourceCulture);
429:         }
430:     }
431:
432:     /// <summary>
433:     /// Looks up a localized string similar to Best Song.
434:     /// </summary>
435:     public static string BestSong {
436:         get {
437:             return ResourceManager.GetString("BestSong", resourceCulture);
438:         }
439:     }
440:
441:     /// <summary>
442:     /// Looks up a localized string similar to Black.
443:     /// </summary>
444:     public static string Black {
445:         get {
446:             return ResourceManager.GetString("Black", resourceCulture);
447:         }
448:     }
449:
450:     /// <summary>
451:     /// Looks up a localized string similar to Blue.
452:     /// </summary>
453:     public static string Blue {
454:         get {
455:             return ResourceManager.GetString("Blue", resourceCulture);
456:         }
457:     }
458:
459:     /// <summary>
460:     /// Looks up a localized string similar to Brown.
461:     /// </summary>
462:     public static string Brown {
463:         get {
464:             return ResourceManager.GetString("Brown", resourceCulture);
465:         }
466:     }
467:
468:     /// <summary>
469:     /// Looks up a localized string similar to Cancel.
470:     /// </summary>
471:     public static string Cancel {
472:         get {
473:             return ResourceManager.GetString("Cancel", resourceCulture);
474:         }
475:     }
476:
477:     /// <summary>
478:     /// Looks up a localized string similar to Categories.
479:     /// </summary>
```

```

480:         public static string Categories {
481:             get {
482:                 return ResourceManager.GetString("Categories", resourceCulture);
483:             }
484:         }
485:
486:         /// <summary>
487:         /// Looks up a localized string similar to Category Has Been Added
Successfully.
488:         /// </summary>
489:         public static string CategoryAddSuccess {
490:             get {
491:                 return ResourceManager.GetString("CategoryAddSuccess",
resourceCulture);
492:             }
493:         }
494:
495:         /// <summary>
496:         /// Looks up a localized string similar to Category Has Been Removed
Successfully.
497:         /// </summary>
498:         public static string CategoryDeleteSuccess {
499:             get {
500:                 return ResourceManager.GetString("CategoryDeleteSuccess",
resourceCulture);
501:             }
502:         }
503:
504:         /// <summary>
505:         /// Looks up a localized string similar to This Category Has Already Exists.
506:         /// </summary>
507:         public static string CategoryExists {
508:             get {
509:                 return ResourceManager.GetString("CategoryExists", resourceCulture);
510:             }
511:         }
512:
513:         /// <summary>
514:         /// Looks up a localized string similar to Category Name.
515:         /// </summary>
516:         public static string CategoryName {
517:             get {
518:                 return ResourceManager.GetString("CategoryName", resourceCulture);
519:             }
520:         }
521:
522:         /// <summary>
523:         /// Looks up a localized string similar to Category Name Has Been Changed
Successfully.
524:         /// </summary>
525:         public static string CategoryNameChangeSuccess {
526:             get {
527:                 return ResourceManager.GetString("CategoryNameChangeSuccess",
resourceCulture);
528:             }
529:         }
530:
531:         /// <summary>
532:         /// Looks up a localized string similar to Category Order By.
533:         /// </summary>
534:         public static string CategoryOrderBy {
535:             get {
536:                 return ResourceManager.GetString("CategoryOrderBy", resourceCulture);
537:             }
538:         }
539:
540:         /// <summary>
541:         /// Looks up a localized string similar to Category Order Style.
542:         /// </summary>

```

```

543:         public static string CategoryOrderStyle {
544:             get {
545:                 return ResourceManager.GetString("CategoryOrderStyle",
resourceCulture);
546:             }
547:         }
548:
549:         /// <summary>
550:         /// Looks up a localized string similar to Category Order Style Has Been
Changed Successfully.
551:         /// </summary>
552:         public static string CategoryOrderStyleChangeSuccess {
553:             get {
554:                 return ResourceManager.GetString("CategoryOrderStyleChangeSuccess",
resourceCulture);
555:             }
556:         }
557:
558:         /// <summary>
559:         /// Looks up a localized string similar to Category Order Type Has Been
Changed Successfully.
560:         /// </summary>
561:         public static string CategoryOrderTypeChangeSuccess {
562:             get {
563:                 return ResourceManager.GetString("CategoryOrderTypeChangeSuccess",
resourceCulture);
564:             }
565:         }
566:
567:         /// <summary>
568:         /// Looks up a localized string similar to Category Settings.
569:         /// </summary>
570:         public static string CategorySettings {
571:             get {
572:                 return ResourceManager.GetString("CategorySettings", resourceCulture);
573:             }
574:         }
575:
576:         /// <summary>
577:         /// Looks up a localized string similar to Chinese.
578:         /// </summary>
579:         public static string Chinese {
580:             get {
581:                 return ResourceManager.GetString("Chinese", resourceCulture);
582:             }
583:         }
584:
585:         /// <summary>
586:         /// Looks up a localized string similar to Contact With Us.
587:         /// </summary>
588:         public static string ContactWithUs {
589:             get {
590:                 return ResourceManager.GetString("ContactWithUs", resourceCulture);
591:             }
592:         }
593:
594:         /// <summary>
595:         /// Looks up a localized string similar to Creation Date.
596:         /// </summary>
597:         public static string CreationDate {
598:             get {
599:                 return ResourceManager.GetString("CreationDate", resourceCulture);
600:             }
601:         }
602:
603:         /// <summary>
604:         /// Looks up a localized string similar to Date.
605:         /// </summary>
606:         public static string Date {

```



```

607:         get {
608:             return ResourceManager.GetString("Date", resourceCulture);
609:         }
610:     }
611:
612:     /// <summary>
613:     /// Looks up a localized string similar to Delete Album.
614:     /// </summary>
615:     public static string DeleteAlbum {
616:         get {
617:             return ResourceManager.GetString("DeleteAlbum", resourceCulture);
618:         }
619:     }
620:
621:     /// <summary>
622:     /// Looks up a localized string similar to You Will Delete This Album. Are
You Sure?.
623:     /// </summary>
624:     public static string DeleteAlbumQuestion {
625:         get {
626:             return ResourceManager.GetString("DeleteAlbumQuestion",
resourceCulture);
627:         }
628:     }
629:
630:     /// <summary>
631:     /// Looks up a localized string similar to Delete Artist.
632:     /// </summary>
633:     public static string DeleteArtist {
634:         get {
635:             return ResourceManager.GetString("DeleteArtist", resourceCulture);
636:         }
637:     }
638:
639:     /// <summary>
640:     /// Looks up a localized string similar to You Will Delete The Artist With
All Contents. Are You Sure?.
641:     /// </summary>
642:     public static string DeleteArtistQuestion {
643:         get {
644:             return ResourceManager.GetString("DeleteArtistQuestion",
resourceCulture);
645:         }
646:     }
647:
648:     /// <summary>
649:     /// Looks up a localized string similar to Delete Category.
650:     /// </summary>
651:     public static string DeleteCategory {
652:         get {
653:             return ResourceManager.GetString("DeleteCategory", resourceCulture);
654:         }
655:     }
656:
657:     /// <summary>
658:     /// Looks up a localized string similar to You Will Delete Category With
All Contents. Do You Agree?.
659:     /// </summary>
660:     public static string DeleteCategoryQuestion {
661:         get {
662:             return ResourceManager.GetString("DeleteCategoryQuestion",
resourceCulture);
663:         }
664:     }
665:
666:     /// <summary>
667:     /// Looks up a localized string similar to Descending.
668:     /// </summary>
669:     public static string Descending {

```

```
670:         get {
671:             return ResourceManager.GetString("Descending", resourceCulture);
672:         }
673:     }
674:
675:     /// <summary>
676:     /// Looks up a localized string similar to English.
677:     /// </summary>
678:     public static string English {
679:         get {
680:             return ResourceManager.GetString("English", resourceCulture);
681:         }
682:     }
683:
684:     /// <summary>
685:     /// Looks up a localized string similar to Enter Artist Name.
686:     /// </summary>
687:     public static string EnterArtistName {
688:         get {
689:             return ResourceManager.GetString("EnterArtistName", resourceCulture);
690:         }
691:     }
692:
693:     /// <summary>
694:     /// Looks up a localized string similar to Enter Category Name.
695:     /// </summary>
696:     public static string EnterCategoryName {
697:         get {
698:             return ResourceManager.GetString("EnterCategoryName",
resourceCulture);
699:         }
700:     }
701:
702:     /// <summary>
703:     /// Looks up a localized string similar to Exit App.
704:     /// </summary>
705:     public static string ExitApp {
706:         get {
707:             return ResourceManager.GetString("ExitApp", resourceCulture);
708:         }
709:     }
710:
711:     /// <summary>
712:     /// Looks up a localized string similar to Do You Want To Exit App?.
713:     /// </summary>
714:     public static string ExitAppQuestion {
715:         get {
716:             return ResourceManager.GetString("ExitAppQuestion", resourceCulture);
717:         }
718:     }
719:
720:     /// <summary>
721:     /// Looks up a localized string similar to Finish Date.
722:     /// </summary>
723:     public static string FinishDate {
724:         get {
725:             return ResourceManager.GetString("FinishDate", resourceCulture);
726:         }
727:     }
728:
729:     /// <summary>
730:     /// Looks up a localized string similar to Font.
731:     /// </summary>
732:     public static string Font {
733:         get {
734:             return ResourceManager.GetString("Font", resourceCulture);
735:         }
736:     }
737:
```

```
738:         /// <summary>
739:         /// Looks up a localized string similar to Font Family.
740:         /// </summary>
741:         public static string FontFamily {
742:             get {
743:                 return ResourceManager.GetString("FontFamily", resourceCulture);
744:             }
745:         }
746:
747:         /// <summary>
748:         /// Looks up a localized string similar to Font Family Has Been Changed
Successfully.
749:         /// </summary>
750:         public static string FontFamilyChangeSuccess {
751:             get {
752:                 return ResourceManager.GetString("FontFamilyChangeSuccess",
resourceCulture);
753:             }
754:         }
755:
756:         /// <summary>
757:         /// Looks up a localized string similar to Font Size.
758:         /// </summary>
759:         public static string FontSize {
760:             get {
761:                 return ResourceManager.GetString("FontSize", resourceCulture);
762:             }
763:         }
764:
765:         /// <summary>
766:         /// Looks up a localized string similar to Font Size Has Been Changed
Successfully.
767:         /// </summary>
768:         public static string FontSizeChangeSuccess {
769:             get {
770:                 return ResourceManager.GetString("FontSizeChangeSuccess",
resourceCulture);
771:             }
772:         }
773:
774:         /// <summary>
775:         /// Looks up a localized string similar to French.
776:         /// </summary>
777:         public static string French {
778:             get {
779:                 return ResourceManager.GetString("French", resourceCulture);
780:             }
781:         }
782:
783:         /// <summary>
784:         /// Looks up a localized string similar to General Settings.
785:         /// </summary>
786:         public static string GeneralSettings {
787:             get {
788:                 return ResourceManager.GetString("GeneralSettings", resourceCulture);
789:             }
790:         }
791:
792:         /// <summary>
793:         /// Looks up a localized string similar to German.
794:         /// </summary>
795:         public static string German {
796:             get {
797:                 return ResourceManager.GetString("German", resourceCulture);
798:             }
799:         }
800:
801:         /// <summary>
802:         /// Looks up a localized string similar to Gray.
```

```
803:         /// </summary>
804:         public static string Gray {
805:             get {
806:                 return ResourceManager.GetString("Gray", resourceCulture);
807:             }
808:         }
809:
810:         /// <summary>
811:         /// Looks up a localized string similar to Green.
812:         /// </summary>
813:         public static string Green {
814:             get {
815:                 return ResourceManager.GetString("Green", resourceCulture);
816:             }
817:         }
818:
819:         /// <summary>
820:         /// Looks up a localized string similar to Italian.
821:         /// </summary>
822:         public static string Italian {
823:             get {
824:                 return ResourceManager.GetString("Italian", resourceCulture);
825:             }
826:         }
827:
828:         /// <summary>
829:         /// Looks up a localized string similar to Japanese.
830:         /// </summary>
831:         public static string Japanese {
832:             get {
833:                 return ResourceManager.GetString("Japanese", resourceCulture);
834:             }
835:         }
836:
837:         /// <summary>
838:         /// Looks up a localized string similar to Label Name.
839:         /// </summary>
840:         public static string LabelName {
841:             get {
842:                 return ResourceManager.GetString("LabelName", resourceCulture);
843:             }
844:         }
845:
846:         /// <summary>
847:         /// Looks up a localized string similar to Language.
848:         /// </summary>
849:         public static string Language {
850:             get {
851:                 return ResourceManager.GetString("Language", resourceCulture);
852:             }
853:         }
854:
855:         /// <summary>
856:         /// Looks up a localized string similar to You May Restart Application To
Change Effect.
857:         /// </summary>
858:         public static string LanguageWarning {
859:             get {
860:                 return ResourceManager.GetString("LanguageWarning", resourceCulture);
861:             }
862:         }
863:
864:         /// <summary>
865:         /// Looks up a localized string similar to Last Modification Date.
866:         /// </summary>
867:         public static string ModificationDate {
868:             get {
869:                 return ResourceManager.GetString("ModificationDate", resourceCulture);
870:             }
871:         }
```

```
871:     }
872:
873:     /// <summary>
874:     /// Looks up a localized string similar to Artist You Most Listen.
875:     /// </summary>
876:     public static string MostListenArtist {
877:         get {
878:             return ResourceManager.GetString("MostListenArtist", resourceCulture);
879:         }
880:     }
881:
882:     /// <summary>
883:     /// Looks up a localized string similar to Category You Most Listen.
884:     /// </summary>
885:     public static string MostListenCategory {
886:         get {
887:             return ResourceManager.GetString("MostListenCategory",
resourceCulture);
888:         }
889:     }
890:
891:     /// <summary>
892:     /// Looks up a localized string similar to Label You Most Listen.
893:     /// </summary>
894:     public static string MostListenLabel {
895:         get {
896:             return ResourceManager.GetString("MostListenLabel", resourceCulture);
897:         }
898:     }
899:
900:     /// <summary>
901:     /// Looks up a localized string similar to Name.
902:     /// </summary>
903:     public static string Name {
904:         get {
905:             return ResourceManager.GetString("Name", resourceCulture);
906:         }
907:     }
908:
909:     /// <summary>
910:     /// Looks up a localized string similar to Ok.
911:     /// </summary>
912:     public static string OK {
913:         get {
914:             return ResourceManager.GetString("OK", resourceCulture);
915:         }
916:     }
917:
918:     /// <summary>
919:     /// Looks up a localized string similar to OneDrive.
920:     /// </summary>
921:     public static string OneDrive {
922:         get {
923:             return ResourceManager.GetString("OneDrive", resourceCulture);
924:         }
925:     }
926:
927:     /// <summary>
928:     /// Looks up a localized string similar to OneDrive Sync Completed.
929:     /// </summary>
930:     public static string OneDriveSyncCompleted {
931:         get {
932:             return ResourceManager.GetString("OneDriveSyncCompleted",
resourceCulture);
933:         }
934:     }
935:
936:     /// <summary>
937:     /// Looks up a localized string similar to Orange.
```

```
938:         /// </summary>
939:         public static string Orange {
940:             get {
941:                 return ResourceManager.GetString("Orange", resourceCulture);
942:             }
943:         }
944:
945:         /// <summary>
946:         /// Looks up a localized string similar to Other Settings.
947:         /// </summary>
948:         public static string OtherSettings {
949:             get {
950:                 return ResourceManager.GetString("OtherSettings", resourceCulture);
951:             }
952:         }
953:
954:         /// <summary>
955:         /// Looks up a localized string similar to Persian.
956:         /// </summary>
957:         public static string Persian {
958:             get {
959:                 return ResourceManager.GetString("Persian", resourceCulture);
960:             }
961:         }
962:
963:         /// <summary>
964:         /// Looks up a localized string similar to Portuguese.
965:         /// </summary>
966:         public static string Portuguese {
967:             get {
968:                 return ResourceManager.GetString("Portuguese", resourceCulture);
969:             }
970:         }
971:
972:         /// <summary>
973:         /// Looks up a localized string similar to Purple.
974:         /// </summary>
975:         public static string Purple {
976:             get {
977:                 return ResourceManager.GetString("Purple", resourceCulture);
978:             }
979:         }
980:
981:         /// <summary>
982:         /// Looks up a localized string similar to Rate.
983:         /// </summary>
984:         public static string Rate {
985:             get {
986:                 return ResourceManager.GetString("Rate", resourceCulture);
987:             }
988:         }
989:
990:         /// <summary>
991:         /// Looks up a localized string similar to Red.
992:         /// </summary>
993:         public static string Red {
994:             get {
995:                 return ResourceManager.GetString("Red", resourceCulture);
996:             }
997:         }
998:
999:         /// <summary>
1000:        /// Looks up a localized string similar to Release Year.
1001:        /// </summary>
1002:        public static string ReleaseYear {
1003:            get {
1004:                return ResourceManager.GetString("ReleaseYear", resourceCulture);
1005:            }
1006:        }
```

```
1007:
1008:     /// <summary>
1009:     /// Looks up a localized string similar to Remove Background Image.
1010:     /// </summary>
1011:     public static string RemoveBackgroundImage {
1012:         get {
1013:             return ResourceManager.GetString("RemoveBackgroundImage",
resourceCulture);
1014:         }
1015:     }
1016:
1017:     /// <summary>
1018:     /// Looks up a localized string similar to Reset Settings.
1019:     /// </summary>
1020:     public static string ResetSettings {
1021:         get {
1022:             return ResourceManager.GetString("ResetSettings", resourceCulture);
1023:         }
1024:     }
1025:
1026:     /// <summary>
1027:     /// Looks up a localized string similar to LeftToRight.
1028:     /// </summary>
1029:     public static string ResourceFlowDirection {
1030:         get {
1031:             return ResourceManager.GetString("ResourceFlowDirection",
resourceCulture);
1032:         }
1033:     }
1034:
1035:     /// <summary>
1036:     /// Looks up a localized string similar to en-US.
1037:     /// </summary>
1038:     public static string ResourceLanguage {
1039:         get {
1040:             return ResourceManager.GetString("ResourceLanguage", resourceCulture);
1041:         }
1042:     }
1043:
1044:     /// <summary>
1045:     /// Looks up a localized string similar to Russian.
1046:     /// </summary>
1047:     public static string Russian {
1048:         get {
1049:             return ResourceManager.GetString("Russian", resourceCulture);
1050:         }
1051:     }
1052:
1053:     /// <summary>
1054:     /// Looks up a localized string similar to Sanskrit.
1055:     /// </summary>
1056:     public static string Sanskrit {
1057:         get {
1058:             return ResourceManager.GetString("Sanskrit", resourceCulture);
1059:         }
1060:     }
1061:
1062:     /// <summary>
1063:     /// Looks up a localized string similar to Save.
1064:     /// </summary>
1065:     public static string Save {
1066:         get {
1067:             return ResourceManager.GetString("Save", resourceCulture);
1068:         }
1069:     }
1070:
1071:     /// <summary>
1072:     /// Looks up a localized string similar to Search.
1073:     /// </summary>
```

```
1074:         public static string Search {
1075:             get {
1076:                 return ResourceManager.GetString("Search", resourceCulture);
1077:             }
1078:         }
1079:
1080:         /// <summary>
1081:         /// Looks up a localized string similar to Search Completed.
1082:         /// </summary>
1083:         public static string SearchCompleted {
1084:             get {
1085:                 return ResourceManager.GetString("SearchCompleted", resourceCulture);
1086:             }
1087:         }
1088:
1089:         /// <summary>
1090:         /// Looks up a localized string similar to Search Results.
1091:         /// </summary>
1092:         public static string SearchResults {
1093:             get {
1094:                 return ResourceManager.GetString("SearchResults", resourceCulture);
1095:             }
1096:         }
1097:
1098:         /// <summary>
1099:         /// Looks up a localized string similar to You Have To Fill Search Criteria.
1100:         /// </summary>
1101:         public static string SearchTrimFault {
1102:             get {
1103:                 return ResourceManager.GetString("SearchTrimFault", resourceCulture);
1104:             }
1105:         }
1106:
1107:         /// <summary>
1108:         /// Looks up a localized string similar to Select.
1109:         /// </summary>
1110:         public static string Select {
1111:             get {
1112:                 return ResourceManager.GetString("Select", resourceCulture);
1113:             }
1114:         }
1115:
1116:         /// <summary>
1117:         /// Looks up a localized string similar to Select Background Color.
1118:         /// </summary>
1119:         public static string SelectBackgroundColor {
1120:             get {
1121:                 return ResourceManager.GetString("SelectBackgroundColor",
resourceCulture);
1122:             }
1123:         }
1124:
1125:         /// <summary>
1126:         /// Looks up a localized string similar to Selected.
1127:         /// </summary>
1128:         public static string Selected {
1129:             get {
1130:                 return ResourceManager.GetString("Selected", resourceCulture);
1131:             }
1132:         }
1133:
1134:         /// <summary>
1135:         /// Looks up a localized string similar to Select Font Family.
1136:         /// </summary>
1137:         public static string SelectFontFamily {
1138:             get {
1139:                 return ResourceManager.GetString("SelectFontFamily", resourceCulture);
1140:             }
1141:         }
```



```
1142:
1143:     /// <summary>
1144:     /// Looks up a localized string similar to Select Font Size.
1145:     /// </summary>
1146:     public static string SelectFontSize {
1147:         get {
1148:             return ResourceManager.GetString("SelectFontSize", resourceCulture);
1149:         }
1150:     }
1151:
1152:     /// <summary>
1153:     /// Looks up a localized string similar to Select Language.
1154:     /// </summary>
1155:     public static string SelectLanguage {
1156:         get {
1157:             return ResourceManager.GetString("SelectLanguage", resourceCulture);
1158:         }
1159:     }
1160:
1161:     /// <summary>
1162:     /// Looks up a localized string similar to Send With Awesome Music App.
1163:     /// </summary>
1164:     public static string SendWithApp {
1165:         get {
1166:             return ResourceManager.GetString("SendWithApp", resourceCulture);
1167:         }
1168:     }
1169:
1170:     /// <summary>
1171:     /// Looks up a localized string similar to Send With Mail.
1172:     /// </summary>
1173:     public static string SendWithMail {
1174:         get {
1175:             return ResourceManager.GetString("SendWithMail", resourceCulture);
1176:         }
1177:     }
1178:
1179:     /// <summary>
1180:     /// Looks up a localized string similar to Send With SMS.
1181:     /// </summary>
1182:     public static string SendWithSMS {
1183:         get {
1184:             return ResourceManager.GetString("SendWithSMS", resourceCulture);
1185:         }
1186:     }
1187:
1188:     /// <summary>
1189:     /// Looks up a localized string similar to Settings.
1190:     /// </summary>
1191:     public static string Settings {
1192:         get {
1193:             return ResourceManager.GetString("Settings", resourceCulture);
1194:         }
1195:     }
1196:
1197:     /// <summary>
1198:     /// Looks up a localized string similar to Share Album.
1199:     /// </summary>
1200:     public static string ShareAlbum {
1201:         get {
1202:             return ResourceManager.GetString("ShareAlbum", resourceCulture);
1203:         }
1204:     }
1205:
1206:     /// <summary>
1207:     /// Looks up a localized string similar to Song Count.
1208:     /// </summary>
1209:     public static string SongCount {
1210:         get {
```

```

1211:         return ResourceManager.GetString("SongCount", resourceCulture);
1212:     }
1213: }
1214:
1215: /// <summary>
1216: /// Looks up a localized string similar to Spanish.
1217: /// </summary>
1218: public static string Spanish {
1219:     get {
1220:         return ResourceManager.GetString("Spanish", resourceCulture);
1221:     }
1222: }
1223:
1224: /// <summary>
1225: /// Looks up a localized string similar to Start Date.
1226: /// </summary>
1227: public static string StartDate {
1228:     get {
1229:         return ResourceManager.GetString("StartDate", resourceCulture);
1230:     }
1231: }
1232:
1233: /// <summary>
1234: /// Looks up a localized string similar to Statistics.
1235: /// </summary>
1236: public static string Statistics {
1237:     get {
1238:         return ResourceManager.GetString("Statistics", resourceCulture);
1239:     }
1240: }
1241:
1242: /// <summary>
1243: /// Looks up a localized string similar to Background Settings Has Been
Reset Successfully.
1244: /// </summary>
1245: public static string SuccessfulResetSettings {
1246:     get {
1247:         return ResourceManager.GetString("SuccessfulResetSettings",
resourceCulture);
1248:     }
1249: }
1250:
1251: /// <summary>
1252: /// Looks up a localized string similar to Sync.
1253: /// </summary>
1254: public static string Sync {
1255:     get {
1256:         return ResourceManager.GetString("Sync", resourceCulture);
1257:     }
1258: }
1259:
1260: /// <summary>
1261: /// Looks up a localized string similar to Synchronizing.
1262: /// </summary>
1263: public static string Synchronizing {
1264:     get {
1265:         return ResourceManager.GetString("Synchronizing", resourceCulture);
1266:     }
1267: }
1268:
1269: /// <summary>
1270: /// Looks up a localized string similar to Sync All In One File.
1271: /// </summary>
1272: public static string SyncOnOneFile {
1273:     get {
1274:         return ResourceManager.GetString("SyncOnOneFile", resourceCulture);
1275:     }
1276: }
1277:

```

```
1278:         /// <summary>
1279:         /// Looks up a localized string similar to System Has A Fault. Please Try
Again Later.
1280:         /// </summary>
1281:         public static string SystemFault {
1282:             get {
1283:                 return ResourceManager.GetString("SystemFault", resourceCulture);
1284:             }
1285:         }
1286:
1287:         /// <summary>
1288:         /// Looks up a localized string similar to Thai.
1289:         /// </summary>
1290:         public static string Thai {
1291:             get {
1292:                 return ResourceManager.GetString("Thai", resourceCulture);
1293:             }
1294:         }
1295:
1296:         /// <summary>
1297:         /// Looks up a localized string similar to Album Count You Listen.
1298:         /// </summary>
1299:         public static string TotalAlbumCount {
1300:             get {
1301:                 return ResourceManager.GetString("TotalAlbumCount", resourceCulture);
1302:             }
1303:         }
1304:
1305:         /// <summary>
1306:         /// Looks up a localized string similar to Turkish.
1307:         /// </summary>
1308:         public static string Turkish {
1309:             get {
1310:                 return ResourceManager.GetString("Turkish", resourceCulture);
1311:             }
1312:         }
1313:
1314:         /// <summary>
1315:         /// Looks up a localized string similar to Worst Album.
1316:         /// </summary>
1317:         public static string WorstAlbum {
1318:             get {
1319:                 return ResourceManager.GetString("WorstAlbum", resourceCulture);
1320:             }
1321:         }
1322:
1323:         /// <summary>
1324:         /// Looks up a localized string similar to Yellow.
1325:         /// </summary>
1326:         public static string Yellow {
1327:             get {
1328:                 return ResourceManager.GetString("Yellow", resourceCulture);
1329:             }
1330:         }
1331:     }
1332: }
```

Namespaces

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

1.2.8 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 AwesomeMusic
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 int CurrentCategoryNumber { get; set; }
22:
23:         [Column]
24:         public int CurrentArtistNumber { get; set; }
25:
26:         [Column]
27:         public string AppLangName { get; set; }
28:
29:         [Column]
30:         public string AppBackgroundColor { get; set; }
31:
32:         [Column]
33:         public string CategoryOrderBy { get; set; }
34:
35:         [Column]
36:         public string CategoryOrderStyle { get; set; }
37:
38:         [Column]
39:         public string FontFamily { get; set; }
40:         [Column]
41:         public string FontSize { get; set; }
42:
43:         [Column(DbType = "Image", UpdateCheck = UpdateCheck.Never)]
44:         public byte[] AppBackgroundImage { get; set; }
45:     }
46: }
```

Namespaces

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

1.2.9 Artist.cs

This is file Artist.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 AwesomeMusic
10: {
11:     [Table]
12:     public class Artist
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int ArtistId { get; set; }
19:
20:         [Column]
21:         public string ArtistName { get; set; }
22:
23:         [Column]
24:         public int ArtistAlbumCount { get; set; }
25:
26:         [Column]
27:         public string AlbumOrderBy { get; set; }
28:
29:         [Column]
30:         public string AlbumOrderStyle { get; set; }
31:
32:         [Column]
33:         public string ArtistNameCount { get; set; }
34:
35:         [Column]
36:         public DateTime CreationDate { get; set; }
37:
38:         [Column]
39:         public DateTime ModificationDate { get; set; }
40:     }
41: }
```

Namespaces

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

1.2.10 ArtistPage.xaml.cs

This is file ArtistPage.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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
```

```

18: namespace AwesomeMusic
19: {
20:     public partial class ArtistPage : PhoneApplicationPage
21:     {
22:
23:         public int artistId;
24:         public int categoryId;
25:         public int albumId;
26:         public Popup popup;
27:         public string oldArtistName;
28:
29:         public ArtistPage()
30:         {
31:             InitializeComponent();
32:
33:             ApplicationBar = new ApplicationBar();
34:
35:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
36:             button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
37:             button1.Text = AppResources.AddAlbum;
38:             ApplicationBar.Buttons.Add(button1);
39:             button1.Click += new EventHandler(AddAlbumButton_Click);
40:
41:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
42:             button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
43:             button2.Text = AppResources.DeleteArtist;
44:             ApplicationBar.Buttons.Add(button2);
45:             button2.Click += new EventHandler(DeleteArtistButton_Click);
46:
47:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
48:             button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
49:             button3.Text = AppResources.ArtistSettings;
50:             ApplicationBar.Buttons.Add(button3);
51:             button3.Click += new EventHandler(ArtistSettingsButton_Click);
52:
53:             ApplicationBarIconButton button4 = new ApplicationBarIconButton();
54:             button4.IconUri = new Uri("/Assets/AddCategory.png", UriKind.Relative);
55:             button4.Text = AppResources.AddCategory;
56:             ApplicationBar.Buttons.Add(button4);
57:             button4.Click += new EventHandler(AddCategoryButton_Click);
58:
59:             popup = new Popup();
60:         }
61:
62:         protected override void OnNavigatedTo(NavigationEventArgs e)
63:         {
64:             base.OnNavigatedTo(e);
65:             //while (NavigationService.CanGoBack)
66:             //NavigationService.RemoveBackEntry();
67:
68:         }
69:
70:         protected override void OnNavigatedFrom(NavigationEventArgs e)
71:         {
72:             base.OnNavigatedFrom(e);
73:             //while (NavigationService.CanGoBack)
74:             //NavigationService.RemoveBackEntry();
75:
76:         }
77:
78:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
79:         {
80:             List<Album> albums = new List<Album>();
81:             List<Album> albumsOrdered = new List<Album>();
82:
83:             // displays "Fragment: Detail"
84:             //MessageBox.Show("Folder Id: " + e.Fragment);
85:             base.OnFragmentNavigation(e);
86:

```

```

87:
88:         lstAlbums.Items.Clear();
89:
90:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
91:         {
92:
93:             var appSettings = context.AppSettings.First();
94:             categoryId = appSettings.CurrentCategoryNumber;
95:
96:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(e.Fragment)).Single() as Artist;
97:             artistId = artist.ArtistId;
98:
99:             var appSettings2 = context.AppSettings;
100:             foreach (var item in appSettings2)
101:             {
102:                 item.CurrentArtistNumber = artistId;
103:             }
104:             context.SubmitChanges();
105:
106:             var artistAlbums = context.AlbumArtists.Where(j =>
j.ArtistId.Equals(e.Fragment)).ToList() as List<AlbumArtist>;
107:             if (artistAlbums.Count != 0)
108:             {
109:                 foreach (var item in artistAlbums)
110:                 {
111:                     try
112:                     {
113:                         var album = context.Albums.Where(j =>
j.AlbumCategoryId.Equals(categoryId) && j.AlbumId.Equals(item.AlbumId)).Single() as Album;
114:                         albums.Add(album);
115:                     }
116:                     catch (Exception)
117:                     {
118:                     }
119:                 }
120:
121:             }
122:
123:
124:             string orderStyle = artist.AlbumOrderStyle;
125:
126:             switch (artist.AlbumOrderBy)
127:             {
128:                 case "NAME":
129:                     if (orderStyle == "A")
130:                     {
131:                         albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
132:                     }
133:                     else
134:                     {
135:                         albumsOrdered = albums.OrderByDescending(j =>
j.AlbumName).ToList();
136:                     }
137:                     break;
138:                 case "CDATE":
139:                     if (orderStyle == "A")
140:                     {
141:                         albumsOrdered = albums.OrderBy(j =>
j.CreationDate).ToList();
142:                     }
143:                     else
144:                     {
145:                         albumsOrdered = albums.OrderByDescending(j =>
j.CreationDate).ToList();
146:                     }
147:                     break;
148:                 case "MDATE":

```

```

149:             if (orderStyle == "A")
150:             {
151:                 albumsOrdered = albums.OrderBy(j =>
j.ModificationDate).ToList();
152:             }
153:             else
154:             {
155:                 albumsOrdered = albums.OrderByDescending(j =>
j.ModificationDate).ToList();
156:             }
157:             break;
158:         case "RATING":
159:             if (orderStyle == "A")
160:             {
161:                 albumsOrdered = albums.OrderBy(j =>
j.AlbumRating).ToList();
162:             }
163:             else
164:             {
165:                 albumsOrdered = albums.OrderByDescending(j =>
j.AlbumRating).ToList();
166:             }
167:             break;
168:         //case "SDATE":
169:         //     if (orderStyle == "A")
170:         //     {
171:         //         albumsOrdered = albums.OrderBy(j =>
j.ReadStartDate).ToList();
172:         //     }
173:         //     else
174:         //     {
175:         //         albumsOrdered = albums.OrderByDescending(j =>
j.ReadStartDate).ToList();
176:         //     }
177:         //     break;
178:         //case "FDATE":
179:         //     if (orderStyle == "A")
180:         //     {
181:         //         albumsOrdered = albums.OrderBy(j =>
j.ReadFinishDate).ToList();
182:         //     }
183:         //     else
184:         //     {
185:         //         albumsOrdered = albums.OrderByDescending(j =>
j.ReadFinishDate).ToList();
186:         //     }
187:         //     break;
188:         default:
189:             if (orderStyle == "A")
190:             {
191:                 albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
192:             }
193:             else
194:             {
195:                 albumsOrdered = albums.OrderBy(j => j.AlbumName).ToList();
196:             }
197:             break;
198:     }
199:
200:     lblArtistName.Text = artist.ArtistName;
201:     lblAlbumList.Text = AppResources.AlbumList + " (" + artist.ArtistName
+ ")";
202:     lstAlbums.ItemsSource = albumsOrdered;
203:     lstAlbums.DisplayMemberPath = "AlbumNameRating";
204:     SetBackgroundColor();
205:     //lstNoteList.DisplayMemberPath = "NameCreation";
206:     }
207: }
208:

```



```

209:         private void SetBackgroundColor()
210:         {
211:             AppSettings appSettings = new AppSettings();
212:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
213:             {
214:                 appSettings = context.AppSettings.First() as AppSettings;
215:             }
216:
217:             if (appSettings.AppBackgroundImage != null)
218:             {
219:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
220:                 BitmapImage image = new BitmapImage();
221:                 image.SetSource(stream);
222:                 ImageBrush ib = new ImageBrush();
223:                 ib.ImageSource = image;
224:                 this.LayoutRoot.Background = ib;
225:             }
226:             else
227:             {
228:                 switch (appSettings.AppBackgroundColor)
229:                 {
230:                     case "BLA":
231:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
232:                         break;
233:                     case "BLU":
234:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
235:                         break;
236:                     case "BRO":
237:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
238:                         break;
239:                     case "RED":
240:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
241:                         break;
242:                     case "GRE":
243:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
244:                         break;
245:                     case "GRA":
246:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
247:                         break;
248:                     case "YEL":
249:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
250:                         break;
251:                     case "ORA":
252:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
253:                         break;
254:                     case "PUR":
255:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
256:                         break;
257:                     default:
258:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
259:                         break;
260:                 }
261:             }
262:         }
263:
264:         private void lstAlbums_SelectionChanged(object sender,
SelectionChangedEventArgs e)
265:         {
266:             var album = (Album)lstAlbums.SelectedItem;
267:             int albumId = album.AlbumId;
268:             NavigationService.Navigate(new Uri("/AlbumPage.xaml#" + albumId,
UriKind.Relative));
269:         }
270:
271:         private void AddAlbumButton_Click(object sender, EventArgs e)

```

```

272:         {
273:             NavigationService.Navigate(new Uri("/AlbumPage.xaml", UriKind.Relative));
274:         }
275:
276:         private void DeleteArtistButton_Click(object sender, EventArgs e)
277:         {
278:             if (MessageBox.Show(AppResources.DeleteArtistQuestion,
279:                 AppResources.DeleteArtist, MessageBoxButton.OKCancel)
280:                 != MessageBoxResult.OK)
281:             {
282:             }
283:             else
284:             {
285:                 using (var context = new
286: AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
287:                 {
288:                     var albumArtists = context.AlbumArtists.Where(j =>
289: j.ArtistId.Equals(artistId)).ToList() as List<AlbumArtist>;
290:                     foreach (var item in albumArtists)
291:                     {
292:                         var album = context.Albums.Where(j =>
293: j.AlbumId.Equals(item.AlbumId)).Single() as Album;
294:                         var albumArtists2 = context.AlbumArtists.Where(j =>
295: j.AlbumId.Equals(albumId)).ToList() as List<AlbumArtist>;
296:                         context.AlbumArtists.DeleteAllOnSubmit(albumArtists2);
297:                         context.Albums.DeleteOnSubmit(album);
298:                     }
299:                     var categoryArtists = context.CategoryArtists.Where(j =>
300: j.ArtistId.Equals(artistId)).ToList() as List<CategoryArtist>;
301:                     context.CategoryArtists.DeleteAllOnSubmit(categoryArtists);
302:                     var artists = context.Artists.Where(j =>
303: j.ArtistId.Equals(artistId)).Single() as Artist;
304:                     context.Artists.DeleteOnSubmit(artists);
305:                     context.SubmitChanges();
306:                     var category = context.Categories.Where(j =>
307: j.CategoryId.Equals(categoryId)).Select(j => j);
308:                     foreach (var item in category)
309:                     {
310:                         item.CategoryAlbumCount = context.Albums.Where(j =>
311: j.AlbumCategoryId.Equals(item.CategoryId)).ToList().Count;
312:                         item.CategoryNameCount = item.CategoryName + " (" +
313: item.CategoryAlbumCount + ")";
314:                         item.ModificationDate = DateTime.Now;
315:                         context.SubmitChanges();
316:                     }
317:                 }
318:                 MessageBox.Show(AppResources.ArtistDeleteSuccess);
319:                 NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId,
320: UriKind.Relative));
321:             }
322:             //MessageBox.Show(AppResources.NoteSaved);
323:         }
324:
325:         private void lblArtistName_Tap(object sender,
326: System.Windows.Input.GestureEventArgs e)
327:         {
328:             oldArtistName = lblArtistName.Text;
329:             popup = new Popup();
330:             popup.Height = 300;
331:             popup.Width = 400;
332:             popup.VerticalOffset = 20;
333:             PopupAddChange control = new PopupAddChange();
334:             control.txtLabel.Text = AppResources.EnterArtistName;
335:             control.btnCancel.Content = AppResources.Cancel;

```

```

330:         control.btnOK.Content = AppResources.OK;
331:         popup.Child = control;
332:         popup.IsOpen = true;
333:         control.txtName.Text = lblArtistName.Text;
334:         control.txtName.Focus();
335:         control.txtName.Select(0, control.txtName.Text.Length);
336:
337:         control.btnOK.Click += (s, args) =>
338:         {
339:             bool isCreated;
340:             string artistName;
341:             popup.IsOpen = false;
342:             int length = control.txtName.Text.Length;
343:             string space = control.txtName.Text.Substring(length - Math.Min(1,
length));
344:             if (space == " ")
345:             {
346:                 artistName = control.txtName.Text.Remove(length - 1, 1);
347:             }
348:             else
349:             {
350:                 artistName = control.txtName.Text;
351:             }
352:
353:             if (artistName != lblArtistName.Text)
354:             {
355:                 // ayni isimde bir klasörün daha önceden olusturulup
olusturulmadigini
356:                 // kontrol eden bir kod bölümü
357:                 using (var contextFolder = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
358:                 {
359:                     isCreated =
360:                         contextFolder.Artists.Any(j =>
j.ArtistName.Equals(artistName));
361:                 }
362:                 if (isCreated == true)
363:                 {
364:                     MessageBox.Show(AppResources.ArtistExists);
365:                 }
366:                 // eger bu isimde bir klasör olusturulmamissa
367:                 // olusturulmasi için gerekli kodlar asagidadir
368:                 else
369:                 {
370:                     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
371:                     {
372:
373:                         // buraya kitapla ilgili bilginin güncellenecegi kod da
eklenecek
374:
375:                         var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Select(j => j);
376:                         foreach (var item in artist)
377:                         {
378:                             item.ArtistName = artistName;
379:                             item.ModificationDate = DateTime.Now;
380:                             item.ArtistNameCount = artistName + " (" +
item.ArtistAlbumCount.ToString() + ")";
381:                         }
382:                         context.SubmitChanges();
383:
384:                         var albumArtists = context.AlbumArtists.Where(j =>
j.ArtistId.Equals(artistId)).Select(j => j);
385:                         foreach (var item in albumArtists)
386:                         {
387:                             var album = context.Albums.Where(j =>
j.AlbumId.Equals(item.AlbumId)).Select(j => j);
388:                             foreach (var item2 in album)

```

```

389:                {
390:                    item2.AlbumInformation =
item2.AlbumInformation.Replace(oldArtistName, artistName);
391:                    item2.ModificationDate = DateTime.Now;
392:                    context.SubmitChanges();
393:                }
394:            }
395:
396:            //lstFolders.ItemsSource = context.NoteFolders;
397:            //lstArtists.ItemsSource = context.Categories;
398:            MessageBox.Show(AppResources.ArtistNameChangeSuccess);
399:            popup.IsOpen = false;
400:            CategoryArtist categoryArtist =
context.CategoryArtists.Where(j => j.ArtistId.Equals(artistId) &&
j.CategoryId.Equals(categoryId)).Single() as CategoryArtist;
401:            Artist artist2 = context.Artists.Where(j =>
j.ArtistName.Equals(artistName) && j.ArtistId.Equals(categoryArtist.ArtistId)).Single() as
Artist;
402:            NavigationService.Navigate(new Uri("/ArtistPage.xaml#" +
categoryArtist.ArtistId, UriKind.Relative));
403:        }
404:    }
405:}
406:};
407:control.btnCancel.Click += (s, args) =>
408:{
409:    popup.IsOpen = false;
410:};
411:}
412:
413:private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
414:{
415:    if (popup.IsOpen)
416:    {
417:        popup.IsOpen = false;
418:    }
419:    if (this.NavigationService.CanGoBack)
420:    {
421:        this.NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
categoryId, UriKind.Relative));
422:    }
423:}
424:
425:private void ArtistSettingsButton_Click(object sender, EventArgs e)
426:{
427:    NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" + artistId,
UriKind.Relative));
428:}
429:
430:private void AddCategoryButton_Click(object sender, EventArgs e)
431:{
432:    NavigationService.Navigate(new Uri("/AddCategoryPage.xaml#" + artistId,
UriKind.Relative));
433:}
434:}
435:}
436:}

```

Namespaces

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

1.2.11 ArtistSettingsPage.xaml.cs

This is file ArtistSettingsPage.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 AwesomeMusic.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21: using Microsoft.Phone.Marketplace;
22:
23: namespace AwesomeMusic
24: {
25:     public partial class ArtistSettingsPage : PhoneApplicationPage
26:     {
27:         public int artistId;
28:         public int categoryId;
29:
30:         public ArtistSettingsPage()
31:         {
32:             InitializeComponent();
33:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
34:             {
35:                 var appSettings = context.AppSettings.First();
36:                 lblFontFamily.Text = AppResources.FontFamily + " (" +
AppResources.Selected + ": " + appSettings.FontFamily + ")";
37:                 lblFontSize.Text = AppResources.FontSize + " (" +
AppResources.Selected + ": " + appSettings.FontSize + ")";
38:             }
39:
40:             pvArtistSettings.Title = AppResources.ArtistSettings;
41:             piFont.Header = AppResources.Font;
42:             piOtherSettings.Header = AppResources.OtherSettings;
43:
44:             btnFontFamily.Content = AppResources.Select;
45:             btnFontSize.Content = AppResources.Select;
46:             btnAlbumOrder.Content = AppResources.Select;
47:             btnAlbumOrderStyle.Content = AppResources.Select;
48:         }
49:
50:         protected override void OnNavigatedTo(NavigationEventArgs e)
51:         {
52:             base.OnNavigatedTo(e);
53:             //while (NavigationService.CanGoBack)
54:             //NavigationService.RemoveBackEntry();
55:
56:         }

```

```

57:
58:         protected override void OnNavigatedFrom(NavigationEventArgs e)
59:         {
60:             base.OnNavigatedFrom(e);
61:             //while (NavigationService.CanGoBack)
62:             //NavigationService.RemoveBackEntry();
63:
64:         }
65:
66:         protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
67:         {
68:             // displays "Fragment: Detail"
69:             //MessageBox.Show("Folder Id: " + e.Fragment);
70:             base.OnFragmentNavigation(e);
71:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
72:             {
73:                 var artist = context.Artists.Where(j =>
j.ArtistId.Equals(e.Fragment)).Single() as Artist;
74:                 artistId = artist.ArtistId;
75:                 var appSettings = context.AppSettings.First();
76:                 categoryId = appSettings.CurrentCategoryNumber;
77:                 string orderStyle = artist.AlbumOrderStyle;
78:
79:                 if (artist.AlbumOrderBy == "NAME")
80:                 {
81:                     lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
82:                 }
83:                 if (artist.AlbumOrderBy == "CDATE")
84:                 {
85:                     lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
86:                 }
87:                 if (artist.AlbumOrderBy == "MDATE")
88:                 {
89:                     lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
90:                 }
91:                 if (artist.AlbumOrderBy == "RATING")
92:                 {
93:                     lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.AlbumRating + ")";
94:                 }
95:                 //if (artist.AlbumOrderBy == "SDATE")
96:                 //{
97:                     //    lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.StartDate + ")";
98:                 //}
99:                 //if (artist.AlbumOrderBy == "FDATE")
100:                 //{
101:                     //    lblAlbumOrder.Text = AppResources.AlbumOrderBy + " (" +
AppResources.Selected + ": " + AppResources.FinishDate + ")";
102:                 //}
103:                 if (artist.AlbumOrderStyle == "A")
104:                 {
105:                     lblAlbumOrderStyle.Text = AppResources.AlbumOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
106:                 }
107:                 if (artist.AlbumOrderStyle == "D")
108:                 {
109:                     lblAlbumOrderStyle.Text = AppResources.AlbumOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
110:                 }
111:                 //lstNoteList.DisplayMemberPath = "NameCreation";
112:                 SetBackgroundColor();
113:             }
114:         }
115:

```

```

116:         private void SetBackgroundColor()
117:         {
118:             AppSettings appSettings = new AppSettings();
119:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
120:             {
121:                 appSettings = context.AppSettings.First() as AppSettings;
122:             }
123:
124:             if (appSettings.AppBackgroundImage != null)
125:             {
126:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
127:                 BitmapImage image = new BitmapImage();
128:                 image.SetSource(stream);
129:                 ImageBrush ib = new ImageBrush();
130:                 ib.ImageSource = image;
131:                 this.LayoutRoot.Background = ib;
132:             }
133:             else
134:             {
135:                 switch (appSettings.AppBackgroundColor)
136:                 {
137:                     case "BLA":
138:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
139:                         break;
140:                     case "BLU":
141:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
142:                         break;
143:                     case "BRO":
144:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
145:                         break;
146:                     case "RED":
147:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
148:                         break;
149:                     case "GRE":
150:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
151:                         break;
152:                     case "GRA":
153:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
154:                         break;
155:                     case "YEL":
156:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
157:                         break;
158:                     case "ORA":
159:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
160:                         break;
161:                     case "PUR":
162:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
163:                         break;
164:                     default:
165:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
166:                         break;
167:                 }
168:             }
169:         }
170:
171:         private void btnAlbumOrder_Click(object sender, RoutedEventArgs e)
172:         {
173:             this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" +
artistId, UriKind.Relative));
174:         }
175:
176:         private void btnAlbumOrderStyle_Click(object sender, RoutedEventArgs e)
177:         {
178:             this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml#" +
artistId, UriKind.Relative));

```



```

179:         }
180:
181:         private void btnFontSize_Click(object sender, RoutedEventArgs e)
182:         {
183:             this.NavigationService.Navigate(new Uri("/FontSizeSettingsPage.xaml#" +
artistId, UriKind.Relative));
184:         }
185:
186:         private void btnFontFamily_Click(object sender, RoutedEventArgs e)
187:         {
188:             this.NavigationService.Navigate(new Uri("/FontFamilySettingsPage.xaml#" +
artistId, UriKind.Relative));
189:         }
190:
191:         private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
192:         {
193:             //pvArtistSettings.Title = AppResources.ArtistSettings;
194:             //piFont.Header = AppResources.Font;
195:             //piOtherSettings.Header = AppResources.OtherSettings;
196:
197:             //btnFontFamily.Content = AppResources.Select;
198:             //btnFontSize.Content = AppResources.Select;
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("/ArtistPage.xaml#" +
artistId, UriKind.Relative));
206:             }
207:         }
208:     }
209: }

```

Namespaces

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

1.2.12 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("AwesomeMusic")]
10: [assembly: AssemblyDescription("Best Music App Ever")]
11: [assembly: AssemblyConfiguration("")]
12: [assembly: AssemblyCompany("CoderSerdar")]
13: [assembly: AssemblyProduct("CoderSerdar")]
14: [assembly: AssemblyCopyright("Copyright © 2014")]
15: [assembly: AssemblyTrademark("CoderSerdar")]
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("75fd9600-fbc0-4730-9a0f-5601a1a33231")]
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("1.0.0.1")]
36: [assembly: AssemblyFileVersion("1.0.0.1")]
37: [assembly: NeutralResourcesLanguageAttribute("en-US")]

```

1.2.13 AwesomeMusic.csproj

This is file AwesomeMusic.csproj.

1.2.14 AwesomeMusic.sln

This is file AwesomeMusic.sln.

1.2.15 AwesomeMusicDataContext.cs

This is file AwesomeMusicDataContext.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 AwesomeMusic
10: {
11:     public class AwesomeMusicDataContext : DataContext
12:     {
13:         public const string ConnectionString = @"Data
Source=isostore:/MyMusicLibrary.sdf";
14:         public AwesomeMusicDataContext(string connectionString)
15:             : base(connectionString) { }
16:         public Table<Category> Categories;
17:         public Table<Artist> Artists;
18:         public Table<Album> Albums;
19:         public Table<AppSettings> AppSettings;
20:         public Table<AlbumArtist> AlbumArtists;
21:         public Table<CategoryArtist> CategoryArtists;
22:     }

```

```
23: }
```

Namespaces

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

1.2.16 BackgroundColorSettingsPage.xaml.cs

This is file BackgroundColorSettingsPage.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 AwesomeMusic.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeMusic
16: {
17:     public partial class BackgroundColorSettingsPage : PhoneApplicationPage
18:     {
19:         public int artistId;
20:
21:         public BackgroundColorSettingsPage()
22:         {
23:             InitializeComponent();
24:
25:             lstBackgroundColor.Items.Clear();
26:             lstBackgroundColor.Items.Add(AppResources.Black);
27:             lstBackgroundColor.Items.Add(AppResources.Blue);
28:             lstBackgroundColor.Items.Add(AppResources.Brown);
29:             lstBackgroundColor.Items.Add(AppResources.Gray);
30:             lstBackgroundColor.Items.Add(AppResources.Green);
31:             lstBackgroundColor.Items.Add(AppResources.Orange);
32:             lstBackgroundColor.Items.Add(AppResources.Purple);
33:             lstBackgroundColor.Items.Add(AppResources.Red);
34:             lstBackgroundColor.Items.Add(AppResources.Yellow);
35:             lstBackgroundColor.SelectedIndex = -1;
36:
37:             lblBackgroundColor.Text = AppResources.SelectBackgroundColor;
38:             lblGeneralSettings.Text = AppResources.GeneralSettings;
39:
40:             SetBackgroundColor();
41:         }
42:
43:         private void SetBackgroundColor()
44:         {
45:             AppSettings appSettings = new AppSettings();
46:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
47:             {
48:                 appSettings = context.AppSettings.First() as AppSettings;
49:             }
50:
51:             if (appSettings.AppBackgroundImage != null)

```

```

52:         {
53:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
54:             BitmapImage image = new BitmapImage();
55:             image.SetSource(stream);
56:             ImageBrush ib = new ImageBrush();
57:             ib.ImageSource = image;
58:             this.LayoutRoot.Background = ib;
59:         }
60:     else
61:     {
62:         switch (appSettings.AppBackgroundColor)
63:         {
64:             case "BLA":
65:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
66:                 break;
67:             case "BLU":
68:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
69:                 break;
70:             case "BRO":
71:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
72:                 break;
73:             case "RED":
74:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
75:                 break;
76:             case "GRE":
77:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
78:                 break;
79:             case "GRA":
80:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
81:                 break;
82:             case "YEL":
83:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
84:                 break;
85:             case "ORA":
86:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
87:                 break;
88:             case "PUR":
89:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
90:                 break;
91:             default:
92:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
93:                 break;
94:         }
95:     }
96: }
97:
98: protected override void OnNavigatedTo(NavigationEventArgs e)
99: {
100:     base.OnNavigatedTo(e);
101:     //SetBackgroundColor();
102:     //while (NavigationService.CanGoBack)
103:     //NavigationService.RemoveBackEntry();
104: }
105:
106: protected override void OnNavigatedFrom(NavigationEventArgs e)
107: {
108:     base.OnNavigatedFrom(e);
109:     //while (NavigationService.CanGoBack)
110:     //NavigationService.RemoveBackEntry();
111: }
112:
113: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
114: {
115:     // displays "Fragment: Detail"
116: }
117:

```

```

118:         //MessageBox.Show("Folder Id: " + e.Fragment);
119:         base.OnFragmentNavigation(e);
120:         artistId = int.Parse(e.Fragment);
121:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
122:         {
123:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
124:             lblGeneralSettings.Text = AppResources.GeneralSettings;
125:             lblBackgroundColor.Text = AppResources.SelectFontSize;
126:         }
127:     }
128:
129:     private void lstBackgroundColor_SelectionChanged(object sender,
SelectionChangedEventArgs e)
130:     {
131:         int index = lstBackgroundColor.SelectedIndex;
132:         string backgroundColor = "";
133:         if (index == 0)
134:         {
135:             backgroundColor = "BLA";
136:         }
137:         else if (index == 1)
138:         {
139:             backgroundColor = "BLU";
140:         }
141:         else if (index == 2)
142:         {
143:             backgroundColor = "BRO";
144:         }
145:         else if (index == 3)
146:         {
147:             backgroundColor = "GRA";
148:         }
149:         else if (index == 4)
150:         {
151:             backgroundColor = "GRE";
152:         }
153:         else if (index == 5)
154:         {
155:             backgroundColor = "ORA";
156:         }
157:         else if (index == 6)
158:         {
159:             backgroundColor = "PUR";
160:         }
161:         else if (index == 7)
162:         {
163:             backgroundColor = "RED";
164:         }
165:         else if (index == 8)
166:         {
167:             backgroundColor = "YEL";
168:         }
169:         else
170:         {
171:             backgroundColor = "BLA";
172:         }
173:
174:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
175:         {
176:             var appSettings = context.AppSettings;
177:             foreach (var appSetting in appSettings)
178:             {
179:                 appSetting.AppBackgroundColor = backgroundColor;
180:             }
181:             context.SubmitChanges();
182:             //CustomMessageBox messageBox = new CustomMessageBox()

```

```

183:         //{
184:         //     Caption = AppResources.BackgroundColor,
185:         //     Message = AppResources.SuccessfulBackgroundColorChanged,
186:         //     Background = messageBackGround
187:         //};
188:         //messageBox.Show();
189:         MessageBox.Show(AppResources.BackgroundColorChangeSuccess);
190:     }
191:     SetBackgroundColor();
192:     NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
193:     }
194:
195:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
196:     {
197:         if (this.NavigationService.CanGoBack)
198:         {
199:             this.NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
200:         }
201:     }
202:
203:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
204:     {
205:         //SetBackgroundColor();
206:     }
207: }
208: }

```

Namespaces

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

1.2.17 Category.cs

This is file Category.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 AwesomeMusic
10: {
11:     [Table]
12:     public class Category
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int CategoryId { get; set; }
19:
20:         [Column]
21:         public string CategoryName { get; set; }
22:
23:         [Column]
24:         public int CategoryAlbumCount { get; set; }

```

```
25:
26:     [Column]
27:     public string ArtistOrderBy { get; set; }
28:
29:     [Column]
30:     public string ArtistOrderStyle { get; set; }
31:
32:     [Column]
33:     public string CategoryNameCount { get; set; }
34:
35:     [Column]
36:     public DateTime CreationDate { get; set; }
37:
38:     [Column]
39:     public DateTime ModificationDate { get; set; }
40: }
41: }
```

Namespaces

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

1.2.18 CategoryArtist.cs

This is file CategoryArtist.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 AwesomeMusic
10: {
11:     [Table]
12:     public class CategoryArtist
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int CategoryArtistId { get; set; }
19:
20:         [Column]
21:         public int CategoryId { get; set; }
22:
23:         [Column]
24:         public int ArtistId { get; set; }
25:     }
26: }
```

Namespaces

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

1.2.19 CategoryPage.xaml.cs

This is file CategoryPage.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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
18: namespace AwesomeMusic
19: {
20:     public partial class CategoryPage : PhoneApplicationPage
21:     {
22:         public Popup popup;
23:         public int categoryId;
24:         public string oldCategoryName;
25:         public CategoryPage()
26:         {
27:             InitializeComponent();
28:
29:             ApplicationBar = new ApplicationBar();
30:
31:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
32:             button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
33:             button1.Text = AppResources.AddArtist;
34:             ApplicationBar.Buttons.Add(button1);
35:             button1.Click += new EventHandler(AddArtistButton_Click);
36:
37:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
38:             button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
39:             button2.Text = AppResources.DeleteCategory;
40:             ApplicationBar.Buttons.Add(button2);
41:             button2.Click += new EventHandler(DeleteCategoryButton_Click);
42:
43:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
44:             button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
45:             button3.Text = AppResources.CategorySettings;
46:             ApplicationBar.Buttons.Add(button3);
47:             button3.Click += new EventHandler(CategorySettingsButton_Click);
48:
49:             SetBackgroundColor();
50:             popup = new Popup();
51:         }
52:
53:         private void SetBackgroundColor()
54:         {
55:             AppSettings appSettings = new AppSettings();
56:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
57:             {
58:                 appSettings = context.AppSettings.First() as AppSettings;

```

```

59:         }
60:
61:         if (appSettings.AppBackgroundImage != null)
62:         {
63:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
64:             BitmapImage image = new BitmapImage();
65:             image.SetSource(stream);
66:             ImageBrush ib = new ImageBrush();
67:             ib.ImageSource = image;
68:             this.LayoutRoot.Background = ib;
69:         }
70:         else
71:         {
72:             switch (appSettings.AppBackgroundColor)
73:             {
74:                 case "BLA":
75:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
76:                     break;
77:                 case "BLU":
78:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
79:                     break;
80:                 case "BRO":
81:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
82:                     break;
83:                 case "RED":
84:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
85:                     break;
86:                 case "GRE":
87:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
88:                     break;
89:                 case "GRA":
90:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
91:                     break;
92:                 case "YEL":
93:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
94:                     break;
95:                 case "ORA":
96:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
97:                     break;
98:                 case "PUR":
99:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
100:                     break;
101:                 default:
102:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
103:                     break;
104:             }
105:         }
106:     }
107:
108:     protected override void OnNavigatedTo(NavigationEventArgs e)
109:     {
110:         base.OnNavigatedTo(e);
111:         //while (NavigationService.CanGoBack)
112:         //NavigationService.RemoveBackEntry();
113:     }
114:
115:
116:     protected override void OnNavigatedFrom(NavigationEventArgs e)
117:     {
118:         base.OnNavigatedFrom(e);
119:         //while (NavigationService.CanGoBack)
120:         //NavigationService.RemoveBackEntry();
121:     }
122:
123:
124:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)

```



```

125:         {
126:             List<Artist> artists = new List<Artist>();
127:             List<Artist> artistsOrdered = new List<Artist>();
128:
129:             // displays "Fragment: Detail"
130:             //MessageBox.Show("Folder Id: " + e.Fragment);
131:             base.OnFragmentNavigation(e);
132:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
133:             {
134:                 var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
135:                 string orderStyle = category.ArtistOrderStyle;
136:                 var categoryArtist = context.CategoryArtists.Where(j =>
j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryArtist>;
137:
138:                 foreach (var item in categoryArtist)
139:                 {
140:                     try
141:                     {
142:                         artists.Add(context.Artists.Where(j =>
j.ArtistId.Equals(item.ArtistId)).Single());
143:                     }
144:                     catch (Exception)
145:                     {
146:                     }
147:                 }
148:
149:                 switch (category.ArtistOrderBy)
150:                 {
151:                     case "NAME":
152:                         if (orderStyle == "A")
153:                         {
154:                             artistsOrdered = artists.OrderBy(j =>
j.ArtistName).ToList();
155:                         }
156:                         else
157:                         {
158:                             artistsOrdered = artists.OrderByDescending(j =>
j.ArtistName).ToList();
159:                         }
160:                         break;
161:                     case "ALBUMCOUNT":
162:                         if (orderStyle == "A")
163:                         {
164:                             artistsOrdered = artists.OrderBy(j =>
j.ArtistAlbumCount).ToList();
165:                         }
166:                         else
167:                         {
168:                             artistsOrdered = artists.OrderByDescending(j =>
j.ArtistAlbumCount).ToList();
169:                         }
170:                         break;
171:                     case "CDATE":
172:                         if (orderStyle == "A")
173:                         {
174:                             artistsOrdered = artists.OrderBy(j =>
j.CreationDate).ToList();
175:                         }
176:                         else
177:                         {
178:                             artistsOrdered = artists.OrderByDescending(j =>
j.CreationDate).ToList();
179:                         }
180:                         break;
181:                     case "MDATE":
182:                         if (orderStyle == "A")
183:

```

```

184:                {
185:                    artistsOrdered = artists.OrderBy(j =>
186:                        j.ModificationDate).ToList();
187:                }
188:                else
189:                {
190:                    artistsOrdered = artists.OrderByDescending(j =>
191:                        j.ModificationDate).ToList();
192:                }
193:                break;
194:            default:
195:                if (orderStyle == "A")
196:                {
197:                    artistsOrdered = artists.OrderBy(j =>
198:                        j.ArtistName).ToList();
199:                }
200:                else
201:                {
202:                    artistsOrdered = artists.OrderByDescending(j =>
203:                        j.ArtistName).ToList();
204:                }
205:                break;
206:            }
207:        }
208:        lstArtists.Items.Clear();
209:        categoryId = category.CategoryId;
210:        lblCategoryName.Text = category.CategoryName;
211:        lblArtistList.Text = AppResources.ArtistList + " (" +
212:            category.CategoryName + ")";
213:        lstArtists.ItemsSource = artistsOrdered;
214:        lstArtists.DisplayMemberPath = "ArtistNameCount";
215:        SetBackgroundColor();
216:        //lstNoteList.DisplayMemberPath = "NameCreation";
217:    }
218:}
219:
220:private void AddArtistButton_Click(object sender, EventArgs e)
221:{
222:    popup = new Popup();
223:    popup.Height = 300;
224:    popup.Width = 400;
225:    popup.VerticalOffset = 20;
226:    PopupAddChange control = new PopupAddChange();
227:    control.txtLabel.Text = AppResources.EnterArtistName;
228:    control.btnCancel.Content = AppResources.Cancel;
229:    control.btnOK.Content = AppResources.OK;
230:    popup.Child = control;
231:    popup.IsOpen = true;
232:    control.txtName.Focus();
233:
234:    control.btnOK.Click += (s, args) =>
235:    {
236:        bool isCreated;
237:        string artistName;
238:        popup.IsOpen = false;
239:
240:        int length = control.txtName.Text.Length;
241:        string space = control.txtName.Text.Substring(length - Math.Min(1,
242:            length));
243:
244:        if (space == " ")
245:        {
246:            artistName = control.txtName.Text.Remove(length - 1, 1);
247:        }
248:        else
249:        {
250:            artistName = control.txtName.Text;
251:        }
252:
253:        // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini

```

```

247:         // kontrol eden bir kod bölümü
248:         using (var contextArtist = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
249:         {
250:             isCreated =
251:                 contextArtist.Artists.Any(j =>
j.ArtistName.Equals(artistName));
252:         }
253:         if (isCreated == true)
254:         {
255:             MessageBox.Show(AppResources.ArtistExists);
256:         }
257:         // eger bu isimde bir klasör oluşturulmamissa
258:         // oluşturulmasi için gerekli kodlar asagidadir
259:         else
260:         {
261:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
262:             {
263:                 Artist artist = new Artist();
264:                 artist.ArtistName = artistName;
265:                 artist.CreationDate = DateTime.Now;
266:                 artist.ModificationDate = DateTime.Now;
267:                 artist.ArtistAlbumCount = 0;
268:                 // burada yazarin kitaplarini
269:                 // bitirme tarihine göre azalan bir sekilde ayarlamak için
gerekli düzenleme yapiliyor
270:                 artist.AlbumOrderBy = "MDATE";
271:                 artist.AlbumOrderStyle = "D";
272:                 artist.ArtistNameCount = artist.ArtistName + " (" +
artist.ArtistAlbumCount + ")";
273:                 //note.NameDescriptionWithoutNewline =
note.NameDescription.Replace(Environment.NewLine, " ");
274:                 //note.IsPasswordProtected = false;
275:
276:                 context.Artists.InsertOnSubmit(artist);
277:                 context.SubmitChanges();
278:
279:                 Artist artist3 = context.Artists.Where(j =>
j.ArtistName.Equals(artistName)).Single() as Artist;
280:
281:                 CategoryArtist categoryArtist = new CategoryArtist();
282:                 categoryArtist.CategoryId = categoryId;
283:                 categoryArtist.ArtistId = artist3.ArtistId;
284:                 context.CategoryArtists.InsertOnSubmit(categoryArtist);
285:                 context.SubmitChanges();
286:
287:                 var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
288:                 foreach (var item in category)
289:                 {
290:                     item.ModificationDate = DateTime.Now;
291:                     //item.CategoryNameCount = item.CategoryName + " (" +
item.auth + ")";
292:                 }
293:                 context.SubmitChanges();
294:
295:                 var appSettings = context.AppSettings;
296:                 foreach (var appSetting in appSettings)
297:                 {
298:                     appSetting.CurrentCategoryNumber = categoryId;
299:                 }
300:                 context.SubmitChanges();
301:
302:                 List<Artist> artists = new List<Artist>();
303:                 var categoryArtists = context.CategoryArtists.Where(j =>
j.CategoryId.Equals(categoryId)).ToList() as List<CategoryArtist>;
304:                 foreach (var item in categoryArtists)
305:                 {

```

```

306:             artists.Add(context.Artists.Where(j =>
j.ArtistId.Equals(item.ArtistId)).Single());
307:         }
308:         lstArtists.ItemsSource = artists;
309:         MessageBox.Show(AppResources.ArtistAddSuccess);
310:         //Artist artist2 = context.Artists.Where(j =>
j.ArtistName.Equals(artistName)).Single() as Artist;
311:
312:         var appSettings2 = context.AppSettings;
313:         foreach (var item in appSettings2)
314:         {
315:             item.CurrentArtistNumber = artist3.ArtistId;
316:         }
317:         context.SubmitChanges();
318:         NavigationService.Navigate(new Uri("/ArtistPage.xaml#" +
artist3.ArtistId, UriKind.Relative));
319:     }
320: }
321: };
322: control.btnCancel.Click += (s, args) =>
323: {
324:     popup.IsOpen = false;
325: };
326:
327: //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
328: }
329:
330: private void CategorySettingsButton_Click(object sender, EventArgs e)
331: {
332:     NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" +
categoryId, UriKind.Relative));
333: }
334:
335: private void lblCategoryName_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
336: {
337:     oldCategoryName = lblCategoryName.Text;
338:     popup = new Popup();
339:     popup.Height = 300;
340:     popup.Width = 400;
341:     popup.VerticalOffset = 20;
342:     PopupAddChange control = new PopupAddChange();
343:     control.txtLabel.Text = AppResources.EnterCategoryName;
344:     control.btnCancel.Content = AppResources.Cancel;
345:     control.btnOK.Content = AppResources.OK;
346:     popup.Child = control;
347:     popup.IsOpen = true;
348:     control.txtName.Text = lblCategoryName.Text;
349:     control.txtName.Focus();
350:     control.txtName.Select(0, control.txtName.Text.Length);
351:
352:     control.btnOK.Click += (s, args) =>
353:     {
354:         bool isCreated;
355:         string categoryName;
356:         popup.IsOpen = false;
357:
358:         int length = control.txtName.Text.Length;
359:         string space = control.txtName.Text.Substring(length - Math.Min(1,
length));
360:         if (space == " ")
361:         {
362:             categoryName = control.txtName.Text.Remove(length - 1, 1);
363:         }
364:         else
365:         {
366:             categoryName = control.txtName.Text;
367:         }
368:     }

```

```

369:         if (categoryName != lblCategoryName.Text)
370:         {
371:             // ayni isimde bir klasörün daha önceden olusturulup
olusturulmadigini
372:             // kontrol eden bir kod bölümü
373:             using (var contextFolder = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
374:             {
375:                 isCreated =
376:                     contextFolder.Categories.Any(j =>
j.CategoryName.Equals(categoryName));
377:             }
378:             if (isCreated == true)
379:             {
380:                 MessageBox.Show(AppResources.CategoryExists);
381:             }
382:             // eger bu isimde bir klasör olusturulmamissa
383:             // olusturulmasi için gerekli kodlar asagidadir
384:             else
385:             {
386:                 using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
387:                 {
388:
389:                     // buraya kitapla ilgili bilginin güncellenecegi kod da
eklenecek
390:
391:                     var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
392:                     foreach (var item in category)
393:                     {
394:                         item.CategoryName = categoryName;
395:                         item.ModificationDate = DateTime.Now;
396:                         item.CategoryNameCount = categoryName + " (" +
item.CategoryAlbumCount.ToString() + ")";
397:                     }
398:                     context.SubmitChanges();
399:
400:                     var album = context.Albums.Where(j =>
j.AlbumCategoryId.Equals(categoryId)).Select(j => j);
401:                     foreach (var item in album)
402:                     {
403:                         item.AlbumInformation =
item.AlbumInformation.Replace(oldCategoryName, categoryName);
404:                         item.ModificationDate = DateTime.Now;
405:                     }
406:                     context.SubmitChanges();
407:                     //lstFolders.ItemsSource = context.NoteFolders;
408:                     //lstArtists.ItemsSource = context.Categories;
409:                     MessageBox.Show(AppResources.CategoryNameChangeSuccess);
410:                     popup.IsOpen = false;
411:                     Category category2 = context.Categories.Where(j =>
j.CategoryName.Equals(categoryName)).Single() as Category;
412:                     NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
category2.CategoryId, UriKind.Relative));
413:                 }
414:             }
415:         }
416:     };
417:     control.btnCancel.Click += (s, args) =>
418:     {
419:         popup.IsOpen = false;
420:     };
421: }
422:
423: private void DeleteCategoryButton_Click(object sender, EventArgs e)
424: {
425:     if (MessageBox.Show(AppResources.DeleteCategoryQuestion,
AppResources.DeleteCategory, MessageBoxButton.OKCancel)
426:

```

```

427:         != MessageBoxResult.OK)
428:     {
429:
430:     }
431:     else
432:     {
433:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
434:         {
435:             var albums = context.Albums.Where(j =>
j.AlbumCategoryId.Equals(categoryId)).ToList() as List<Album>;
436:             foreach (var item in albums)
437:             {
438:                 var albumArtists = context.AlbumArtists.Where(j =>
j.AlbumId.Equals(item.AlbumId)).ToList() as List<AlbumArtist>;
439:                 context.AlbumArtists.DeleteAllOnSubmit(albumArtists);
440:             }
441:             context.Albums.DeleteAllOnSubmit(albums);
442:
443:             var artistCategories = context.CategoryArtists.Where(j =>
j.CategoryId.Equals(categoryId)).ToList() as List<CategoryArtist>;
444:             foreach (var item in artistCategories)
445:             {
446:                 var artist = context.Artists.Where(j =>
j.ArtistId.Equals(item.ArtistId)).ToList() as List<Artist>;
447:                 context.Artists.DeleteAllOnSubmit(artist);
448:             }
449:             context.CategoryArtists.DeleteAllOnSubmit(artistCategories);
450:
451:             var categories = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
452:             context.Categories.DeleteOnSubmit(categories);
453:
454:             context.SubmitChanges();
455:         }
456:         MessageBox.Show(AppResources.CategoryDeleteSuccess);
457:         NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
458:     }
459:     //MessageBox.Show(AppResources.NoteSaved);
460: }
461:
462: private void lstArtists_SelectionChanged(object sender,
SelectionChangedEventArgs e)
463: {
464:     var artist = (Artist)lstArtists.SelectedItem;
465:     int artistId = artist.ArtistId;
466:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
467:     {
468:         var appSettings = context.AppSettings;
469:         foreach (var item in appSettings)
470:         {
471:             item.CurrentArtistNumber = artistId;
472:         }
473:         context.SubmitChanges();
474:     }
475:     NavigationService.Navigate(new Uri("/ArtistPage.xaml#" + artistId,
UriKind.Relative));
476: }
477:
478: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
479: {
480:     if (popup.IsOpen)
481:     {
482:         popup.IsOpen = false;
483:     }
484:     if (this.NavigationService.CanGoBack)

```

```
485:         {
486:             this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
487:         }
488:     }
489:
490:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
491:     {
492:         //SetBackgroundColor();
493:     }
494: }
495: }
```

Namespaces

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

1.2.20 CategorySettingsPage.xaml.cs

This is file CategorySettingsPage.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 AwesomeMusic.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21:
22:
23: namespace AwesomeMusic
24: {
25:     public partial class CategorySettingsPage : PhoneApplicationPage
26:     {
27:         public int categoryId;
28:         public CategorySettingsPage()
29:         {
30:             InitializeComponent();
31:
32:             pvCategorySettings.Title = AppResources.CategorySettings;
33:
34:             piOtherSettings.Header = AppResources.OtherSettings;
35:             btnArtistOrder.Content = AppResources.Select;
36:             btnArtistOrderStyle.Content = AppResources.Select;
37:             SetBackgroundColor();
38:
39:         }
40:         private void SetBackgroundColor()
41:         {
```

```

42:         AppSettings appSettings = new AppSettings();
43:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
44:         {
45:             appSettings = context.AppSettings.First() as AppSettings;
46:         }
47:
48:         if (appSettings.AppBackgroundImage != null)
49:         {
50:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
51:             BitmapImage image = new BitmapImage();
52:             image.SetSource(stream);
53:             ImageBrush ib = new ImageBrush();
54:             ib.ImageSource = image;
55:             this.LayoutRoot.Background = ib;
56:         }
57:         else
58:         {
59:             switch (appSettings.AppBackgroundColor)
60:             {
61:                 case "BLA":
62:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
63:                     break;
64:                 case "BLU":
65:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
66:                     break;
67:                 case "BRO":
68:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
69:                     break;
70:                 case "RED":
71:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
72:                     break;
73:                 case "GRE":
74:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
75:                     break;
76:                 case "GRA":
77:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
78:                     break;
79:                 case "YEL":
80:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
81:                     break;
82:                 case "ORA":
83:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
84:                     break;
85:                 case "PUR":
86:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
87:                     break;
88:                 default:
89:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
90:                     break;
91:             }
92:         }
93:     }
94:
95:     protected override void OnNavigatedTo(NavigationEventArgs e)
96:     {
97:         base.OnNavigatedTo(e);
98:         //while (NavigationService.CanGoBack)
99:         //NavigationService.RemoveBackEntry();
100:     }
101:
102:
103:     protected override void OnNavigatedFrom(NavigationEventArgs e)
104:     {
105:         base.OnNavigatedFrom(e);
106:         //while (NavigationService.CanGoBack)

```



```

107:         //NavigationService.RemoveBackEntry();
108:
109:     }
110:
111:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
112:     {
113:         // displays "Fragment: Detail"
114:         //MessageBox.Show("Folder Id: " + e.Fragment);
115:         base.OnFragmentNavigation(e);
116:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
117:         {
118:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
119:             string orderStyle = category.ArtistOrderStyle;
120:             categoryId = category.CategoryId;
121:
122:             if (category.ArtistOrderBy == "NAME")
123:             {
124:                 lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
125:             }
126:             if (category.ArtistOrderBy == "ALBUMCOUNT")
127:             {
128:                 lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.AlbumCount + ")";
129:             }
130:             if (category.ArtistOrderBy == "CDATE")
131:             {
132:                 lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
133:             }
134:             if (category.ArtistOrderBy == "MDATE")
135:             {
136:                 lblArtistOrder.Text = AppResources.ArtistOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
137:             }
138:             if (category.ArtistOrderStyle == "A")
139:             {
140:                 lblArtistOrderStyle.Text = AppResources.ArtistOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
141:             }
142:             if (category.ArtistOrderStyle == "D")
143:             {
144:                 lblArtistOrderStyle.Text = AppResources.ArtistOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
145:             }
146:             //lstNoteList.DisplayMemberPath = "NameCreation";
147:             SetBackgroundColor();
148:         }
149:     }
150:
151:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
152:     {
153:         //pvCategorySettings.Title = AppResources.CategorySettings;
154:
155:         //piOtherSettings.Header = AppResources.OtherSettings;
156:         //btnArtistOrder.Content = AppResources.Select;
157:         //btnArtistOrderStyle.Content = AppResources.Select;
158:         //SetBackgroundColor();
159:     }
160:
161:     private void btnArtistOrder_Click(object sender, RoutedEventArgs e)
162:     {
163:         this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" +
categoryId, UriKind.Relative));
164:     }
165:
166:     private void btnArtistOrderStyle_Click(object sender, RoutedEventArgs e)

```

```
167:         {
168:             this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml#" +
categoryId, UriKind.Relative));
169:         }
170:
171:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
172:         {
173:             if (this.NavigationService.CanGoBack)
174:             {
175:                 this.NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
categoryId, UriKind.Relative));
176:             }
177:         }
178:     }
179: }
```

Namespaces

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

1.2.21 FontFamilySettingsPage.xaml.cs

This is file FontFamilySettingsPage.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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
18: namespace AwesomeMusic
19: {
20:     public partial class FontFamilySettingsPage : PhoneApplicationPage
21:     {
22:         public int artistId;
23:         public FontFamilySettingsPage()
24:         {
25:             InitializeComponent();
26:
27:             lstFontFamily.Items.Clear();
28:             lstFontFamily.Items.Add("Arial");
29:             lstFontFamily.Items.Add("Arial Black");
30:             lstFontFamily.Items.Add("Baskerville Old Face");
31:             lstFontFamily.Items.Add("Berlin Sans FB");
32:             lstFontFamily.Items.Add("Albumman Old Style");
33:             lstFontFamily.Items.Add("Calibri");
34:             lstFontFamily.Items.Add("Cambria");
35:             lstFontFamily.Items.Add("Candara");
36:             lstFontFamily.Items.Add("Comic Sans MS");
37:             lstFontFamily.Items.Add("Consolas");
```

```

38:         lstFontFamily.Items.Add("Constantia");
39:         lstFontFamily.Items.Add("Courier New");
40:         lstFontFamily.Items.Add("DokChampa");
41:         lstFontFamily.Items.Add("Ebrima");
42:         lstFontFamily.Items.Add("Georgia");
43:         lstFontFamily.Items.Add("Lucida Sans Unicode");
44:         lstFontFamily.Items.Add("Meiryo UI");
45:         lstFontFamily.Items.Add("Microsoft YaHei");
46:         lstFontFamily.Items.Add("Malgun Gothic");
47:         lstFontFamily.Items.Add("Segoe UI");
48:         lstFontFamily.Items.Add("Segoe WP");
49:         lstFontFamily.Items.Add("Tahoma");
50:         lstFontFamily.Items.Add("Trebuchet MS");
51:         lstFontFamily.Items.Add("Times New Roman");
52:         lstFontFamily.Items.Add("Verdana");
53:         lstFontFamily.SelectedIndex = -1;
54:     }
55:
56:     private void lstFontFamily_SelectionChanged(object sender,
SelectionChangedEventArgs e)
57:     {
58:         if (lstFontFamily.SelectedIndex != -1)
59:         {
60:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
61:             {
62:                 var appSettings = context.AppSettings;
63:                 foreach (var item in appSettings)
64:                 {
65:                     item.FontFamily = lstFontFamily.SelectedItem.ToString();
66:                 }
67:                 context.SubmitChanges();
68:                 MessageBox.Show(AppResources.FontFamilyChangeSuccess);
69:             }
70:         }
71:         NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" + artistId,
UriKind.Relative));
72:     }
73:
74:     protected override void OnNavigatedTo(NavigationEventArgs e)
75:     {
76:         base.OnNavigatedTo(e);
77:     }
78:
79:     protected override void OnNavigatedFrom(NavigationEventArgs e)
80:     {
81:         base.OnNavigatedFrom(e);
82:     }
83:
84:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
85:     {
86:         // displays "Fragment: Detail"
87:         //MessageBox.Show("Folder Id: " + e.Fragment);
88:         base.OnFragmentNavigation(e);
89:         artistId = int.Parse(e.Fragment);
90:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
91:         {
92:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
93:             lblArtistName.Text = artist.ArtistName;
94:             lblFontFamily.Text = AppResources.SelectFontFamily;
95:         }
96:         SetBackgroundColor();
97:     }
98:
99:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
100:    {

```

```

101:         if (this.NavigationService.CanGoBack)
102:         {
103:             this.NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" +
artistId, UriKind.Relative));
104:         }
105:     }
106:
107:     private void SetBackgroundColor()
108:     {
109:         AppSettings appSettings = new AppSettings();
110:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
111:         {
112:             appSettings = context.AppSettings.First() as AppSettings;
113:         }
114:
115:         if (appSettings.AppBackgroundImage != null)
116:         {
117:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
118:             BitmapImage image = new BitmapImage();
119:             image.SetSource(stream);
120:             ImageBrush ib = new ImageBrush();
121:             ib.ImageSource = image;
122:             this.LayoutRoot.Background = ib;
123:         }
124:         else
125:         {
126:             switch (appSettings.AppBackgroundColor)
127:             {
128:                 case "BLA":
129:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
130:                     break;
131:                 case "BLU":
132:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
133:                     break;
134:                 case "BRO":
135:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
136:                     break;
137:                 case "RED":
138:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
139:                     break;
140:                 case "GRE":
141:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
142:                     break;
143:                 case "GRA":
144:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
145:                     break;
146:                 case "YEL":
147:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
148:                     break;
149:                 case "ORA":
150:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
151:                     break;
152:                 case "PUR":
153:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
154:                     break;
155:                 default:
156:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
157:                     break;
158:             }
159:         }
160:     }
161: }
162: }

```

Namespaces

Name	Description
AwesomeMusic (🔗 see page 1)	This is namespace AwesomeMusic.

1.2.22 FontSizeSettingsPage.xaml.cs

This is file FontSizeSettingsPage.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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
18: namespace AwesomeMusic
19: {
20:     public partial class FontSizeSettingsPage : PhoneApplicationPage
21:     {
22:         public int artistId;
23:         public FontSizeSettingsPage()
24:         {
25:             InitializeComponent();
26:
27:             lstFontSize.Items.Clear();
28:             lstFontSize.Items.Add("14");
29:             lstFontSize.Items.Add("18");
30:             lstFontSize.Items.Add("22");
31:             lstFontSize.Items.Add("26");
32:             lstFontSize.Items.Add("28");
33:             lstFontSize.Items.Add("30");
34:             lstFontSize.Items.Add("32");
35:             lstFontSize.Items.Add("34");
36:             lstFontSize.Items.Add("36");
37:             lstFontSize.Items.Add("38");
38:             lstFontSize.Items.Add("40");
39:             lstFontSize.Items.Add("42");
40:             lstFontSize.Items.Add("44");
41:             lstFontSize.Items.Add("64");
42:             lstFontSize.Items.Add("72");
43:             lstFontSize.SelectedIndex = -1;
44:         }
45:
46:         private void lstFontSize_SelectionChanged(object sender,
47: SelectionChangedEventArgs e)
48:         {
49:             if (lstFontSize.SelectedIndex != -1)
50:             {
51:                 using (var context = new
52: AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
```

```

52:         var appSettings = context.AppSettings;
53:         foreach (var item in appSettings)
54:         {
55:             item.FontSize = lstFontSize.SelectedItem.ToString();
56:         }
57:         context.SubmitChanges();
58:         MessageBox.Show(AppResources.FontSizeChangeSuccess);
59:     }
60: }
61: NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" + artistId,
UriKind.Relative));
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: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
75: {
76:     // displays "Fragment: Detail"
77:     //MessageBox.Show("Folder Id: " + e.Fragment);
78:     base.OnFragmentNavigation(e);
79:     artistId = int.Parse(e.Fragment);
80:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
81:     {
82:         var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
83:         lblArtistName.Text = artist.ArtistName;
84:         lblFontSize.Text = AppResources.SelectFontSize;
85:     }
86:     SetBackgroundColor();
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("/ArtistSettingsPage.xaml#" +
artistId, UriKind.Relative));
94:     }
95: }
96:
97: private void SetBackgroundColor()
98: {
99:     AppSettings appSettings = new AppSettings();
100:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
101:     {
102:         appSettings = context.AppSettings.First() as AppSettings;
103:     }
104:
105:     if (appSettings.AppBackgroundImage != null)
106:     {
107:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
108:         BitmapImage image = new BitmapImage();
109:         image.SetSource(stream);
110:         ImageBrush ib = new ImageBrush();
111:         ib.ImageSource = image;
112:         this.LayoutRoot.Background = ib;
113:     }
114:     else

```

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

Namespaces

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

1.2.23 GeneralSettingsPage.xaml.cs

This is file GeneralSettingsPage.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 AwesomeMusic.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21:
22: namespace AwesomeMusic
23: {
24:     public partial class GeneralSettingsPage : PhoneApplicationPage
25:     {
26:
27:         private static readonly string[] scopes = new string[] { "wl.signin",
"wl.basic", "wl.offline_access", "wl.skydrive", "wl.skydrive_update" };
28:
29:         /// <summary>
30:         ///     Stores the LiveAuthClient instance.
31:         /// </summary>
32:         private LiveAuthClient authClient;
33:
34:         /// <summary>
35:         ///     Stores the LiveConnectClient instance.
36:         /// </summary>
37:         private LiveConnectClient liveClient;
38:
39:         public int signInIn;
40:
41:         public GeneralSettingsPage()
42:         {
43:             InitializeComponent();
44:             InitializePage();
45:
46:             pvGeneralSettings.Title = AppResources.GeneralSettings;
47:
48:             piLanguage.Header = AppResources.Language;
49:             piSync.Header = AppResources.Sync;
50:             piOtherSettings.Header = AppResources.OtherSettings;
51:             piBackground.Header = AppResources.Background;
52:
53:             //lblOneDrive.Text = AppResources.OneDrive;
54:
55:             btnCategoryOrder.Content = AppResources.Select;
56:             btnCategoryOrderStyle.Content = AppResources.Select;
57:             btnLanguage.Content = AppResources.Select;
58:             btnBackgroundColor.Content = AppResources.Select;
59:             //btnOneDrive.Content = AppResources.Login;
60:             //btnOneDrive.SignInText = AppResources.SignIn;
61:             //btnOneDrive.SignOutText = AppResources.SignOut;
62:             btnOneDriveSync.Content = AppResources.Sync;
63:             lblOneDrive.Text = AppResources.OneDrive;
64:             txtSynchronizing.Text = AppResources.Synchronizing;
65:
66:             pbSync.Visibility = Visibility.Collapsed;
67:             txtSynchronizing.Visibility = Visibility.Collapsed;
68:             txtSynchronizing.BorderBrush = this.LayoutRoot.Background;
69:
70:             btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
71:             lblBackgroundImage.Text = AppResources.BackgroundImage;
72:             btnBackgroundImage.Content = AppResources.Select;
73:             btnResetSettings.Content = AppResources.ResetSettings;
74:
75:             btnOneDriveSync.IsEnabled = false;
76:             cbSync.Content = AppResources.SyncOnOneFile;
77:             cbSync.IsEnabled = false;
78:             btnOneDrive.Content = "Sign In";
79:
80:             SetBackgroundColor();

```



```

81:
82:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
83:         {
84:             var appSettings = context.AppSettings.First() as AppSettings;
85:             if (appSettings.AppLangName == "EN")
86:             {
87:                 lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.English + ")";
88:             }
89:             if (appSettings.AppLangName == "TR")
90:             {
91:                 lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Turkish + ")";
92:             }
93:             if (appSettings.AppLangName == "DE")
94:             {
95:                 lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.German + ")";
96:             }
97:             if (appSettings.AppLangName == "ES")
98:             {
99:                 lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Spanish + ")";
100:            }
101:
102:            if (appSettings.AppLangName == "PT")
103:            {
104:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Portuguese + ")";
105:            }
106:            if (appSettings.AppLangName == "AR")
107:            {
108:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Arabic + ")";
109:            }
110:            if (appSettings.AppLangName == "FA")
111:            {
112:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Persian + ")";
113:            }
114:            if (appSettings.AppLangName == "IT")
115:            {
116:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Italian + ")";
117:            }
118:            if (appSettings.AppLangName == "FR")
119:            {
120:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.French + ")";
121:            }
122:            if (appSettings.AppLangName == "RU")
123:            {
124:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Russian + ")";
125:            }
126:            if (appSettings.AppLangName == "ZH")
127:            {
128:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Chinese + ")";
129:            }
130:            if (appSettings.AppLangName == "JA")
131:            {
132:                lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Japanese + ")";
133:            }
134:            if (appSettings.AppLangName == "SA")
135:            {
136:                lblLanguage.Text = AppResources.Language + " (" +

```

```

AppResources.Selected + ": " + AppResources.Sanskrit + "));
137:     }
138:     if (appSettings.AppLangName == "TH")
139:     {
140:         lblLanguage.Text = AppResources.Language + " (" +
AppResources.Selected + ": " + AppResources.Thai + "));
141:     }
142:
143:     if (appSettings.CategoryOrderBy == "NAME")
144:     {
145:         lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + "));
146:     }
147:     if (appSettings.CategoryOrderBy == "CDATE")
148:     {
149:         lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + "));
150:     }
151:     if (appSettings.CategoryOrderBy == "MDATE")
152:     {
153:         lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + "));
154:     }
155:     if (appSettings.CategoryOrderBy == "ALBUMCOUNT")
156:     {
157:         lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.AlbumCount + "));
158:     }
159:     if (appSettings.CategoryOrderStyle == "A")
160:     {
161:         lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + "
(" + AppResources.Selected + ": " + AppResources.Ascending + "));
162:     }
163:     if (appSettings.CategoryOrderStyle == "D")
164:     {
165:         lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + "
(" + AppResources.Selected + ": " + AppResources.Descending + "));
166:     }
167:     if (appSettings.AppBackgroundColor == "BLA")
168:     {
169:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + "));
170:     }
171:     if (appSettings.AppBackgroundColor == "BLU")
172:     {
173:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + "));
174:     }
175:     if (appSettings.AppBackgroundColor == "BRO")
176:     {
177:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + "));
178:     }
179:     if (appSettings.AppBackgroundColor == "RED")
180:     {
181:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + "));
182:     }
183:     if (appSettings.AppBackgroundColor == "GRE")
184:     {
185:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + "));
186:     }
187:     if (appSettings.AppBackgroundColor == "YEL")
188:     {
189:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + "));
190:     }
191:     if (appSettings.AppBackgroundColor == "GRA")

```

```

192:         {
193:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + ")";
194:         }
195:         if (appSettings.AppBackgroundColor == "ORA")
196:         {
197:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + ")";
198:         }
199:         if (appSettings.AppBackgroundColor == "PUR")
200:         {
201:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + ")";
202:         }
203:     }
204: }
205:
206: protected override void OnNavigatedTo(NavigationEventArgs e)
207: {
208:     base.OnNavigatedTo(e);
209:     SetBackgroundColor();
210:     //while (NavigationService.CanGoBack)
211:     //NavigationService.RemoveBackEntry();
212: }
213:
214:
215: protected override void OnNavigatedFrom(NavigationEventArgs e)
216: {
217:     base.OnNavigatedFrom(e);
218:     //while (NavigationService.CanGoBack)
219:     //NavigationService.RemoveBackEntry();
220: }
221:
222:
223: private async void btnOneDrive_Click(object sender, RoutedEventArgs e)
224: {
225:     try
226:     {
227:         if (this.btnOneDrive.Content.ToString() == "Sign In" ||
this.btnOneDrive.Content.ToString() == "Sign in")
228:         {
229:             LiveLoginResult loginResult = await
this.authClient.LoginAsync(scopes);
230:             if (loginResult.Status == LiveConnectSessionStatus.Connected)
231:             {
232:                 //this.btnOneDrive.Content = AppResources.SignOut;
233:                 this.btnOneDrive.Content = "Sign Out";
234:
235:                 this.liveClient = new LiveConnectClient(loginResult.Session);
236:                 this.GetMe();
237:                 btnOneDriveSync.IsEnabled = true;
238:                 cbSync.IsEnabled = true;
239:             }
240:         }
241:         else
242:         {
243:             this.authClient.Logout();
244:             //this.btnOneDrive.Content = AppResources.SignIn;
245:             this.btnOneDrive.Content = "Sign Out";
246:             btnOneDriveSync.IsEnabled = true;
247:             cbSync.IsEnabled = true;
248:             //this.tbResponse.Text = "";
249:         }
250:     }
251:     catch (LiveAuthException authExp)
252:     {
253:         //this.tbResponse.Text = authExp.ToString();
254:     }
255: }

```

```

256:
257:     private async void InitializePage()
258:     {
259:         try
260:         {
261:             // bu benim uygulamama ait bir client id
262:             this.authClient = new LiveAuthClient("0000000044125951");
263:             LiveLoginResult loginResult = await
this.authClient.InitializeAsync(scopes);
264:             btnOneDrive.Content = "Sign In";
265:             if (loginResult.Status == LiveConnectSessionStatus.Connected)
266:             {
267:                 //this.btnOneDrive.Content = AppResources.SignOut;
268:                 this.btnOneDrive.Content = "Sign Out";
269:
270:                 this.liveClient = new LiveConnectClient(loginResult.Session);
271:                 //this.GetMe();
272:             }
273:         }
274:         catch (LiveAuthException authExp)
275:         {
276:             //this.tbResponse.Text = authExp.ToString();
277:         }
278:     }
279:
280:     private async void GetMe()
281:     {
282:         try
283:         {
284:             LiveOperationResult operationResult = await
this.liveClient.GetAsync("me");
285:
286:             dynamic properties = operationResult.Result;
287:             //this.tbResponse.Text = properties.first_name + " " +
properties.last_name;
288:         }
289:         catch (LiveConnectException e)
290:         {
291:             //this.tbResponse.Text = e.ToString();
292:         }
293:     }
294:
295:     private void SetBackgroundColor()
296:     {
297:         AppSettings appSettings = new AppSettings();
298:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
299:         {
300:             appSettings = context.AppSettings.First() as AppSettings;
301:         }
302:
303:         if (appSettings.AppBackgroundImage != null)
304:         {
305:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
306:             BitmapImage image = new BitmapImage();
307:             image.SetSource(stream);
308:             ImageBrush ib = new ImageBrush();
309:             ib.ImageSource = image;
310:             this.LayoutRoot.Background = ib;
311:         }
312:         else
313:         {
314:             switch (appSettings.AppBackgroundColor)
315:             {
316:                 case "BLA":
317:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
318:                     break;
319:                 case "BLU":
320:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);

```

```

321:             break;
322:         case "BRO":
323:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
324:             break;
325:         case "RED":
326:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
327:             break;
328:         case "GRE":
329:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
330:             break;
331:         case "GRA":
332:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
333:             break;
334:         case "YEL":
335:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
336:             break;
337:         case "ORA":
338:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
339:             break;
340:         case "PUR":
341:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
342:             break;
343:         default:
344:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
345:             break;
346:     }
347: }
348: }
349:
350:     public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
351: string folderName, string parentFolder)
352:     {
353:         string folderId = null;
354:
355:         // Retrieves all the directories.
356:         var queryFolder = parentFolder + "/files?filter=folders,albums";
357:         var opResult = await client.GetAsync(queryFolder);
358:         dynamic result = opResult.Result;
359:
360:         foreach (dynamic folder in result.data)
361:         {
362:             // Checks if current folder has the passed name.
363:             if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
364:             {
365:                 folderId = folder.id;
366:                 break;
367:             }
368:         }
369:
370:         if (folderId == null)
371:         {
372:             // Directory hasn't been found, so creates it using the PostAsync
method.
373:             var folderData = new Dictionary<string, object>();
374:             folderData.Add("name", folderName);
375:             opResult = await client.PostAsync(parentFolder, folderData);
376:             result = opResult.Result;
377:
378:             // Retrieves the id of the created folder.
379:             folderId = result.id;
380:         }
381:
382:         return folderId;
383:     }
384:
385:     private async void btnOneDriveSync_Click(object sender, RoutedEventArgs e)

```

```

386:         {
387:
388:             IsolatedStorageFile myIsolatedStorage = null;
389:             StringBuilder sb = null;
390:
391:
392:             string folderName;
393:             try
394:             {
395:                 //var folderData = new Dictionary<string, object>();
396:                 folderName = "Awesome Music (" + DateTime.Now + ")";
397:                 //folderName = folderName.Replace(":", ".");
398:                 //folderName = folderName.Replace("/", ".");
399:                 folderName = DesignFileName(folderName);
400:
401:                 string skyDriveFolder = await CreateDirectoryAsync(liveClient,
402:                     folderName, "me/skydrive");
403:
404:                 if (cbSync.IsChecked == false)
405:                 {
406:                     btnOneDrive.IsEnabled = false;
407:                     pbSync.Visibility = Visibility.Visible;
408:                     txtSynchronizing.Visibility = Visibility.Visible;
409:
410:                     using (var context = new
411:                         AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
412:                     {
413:                         //var noteFolders = context.NoteFolders.ToList() as
414:                         List<NoteFolder>;
415:                         var albums = context.Albums.ToList() as List<Album>;
416:                         for (int i = 0; i < albums.Count; i++)
417:                         {
418:                             var albumArtist =
419:                                 context.AlbumArtists.Where(j =>
420:                                     j.AlbumId.Equals(albums[i].AlbumId)).ToList() as
421:                                     List<AlbumArtist>;
422:                             var category = context.Categories.Where(j =>
423:                                 j.CategoryId.Equals(albums[i].AlbumCategoryId)).Single() as
424:                                 Category;
425:                             List<Artist> artists = new List<Artist>();
426:                             for (int k = 0; k < albumArtist.Count; k++)
427:                             {
428:                                 artists.Add(context.Artists.Where(j =>
429:                                     j.ArtistId.Equals(albumArtist[k].ArtistId)).Single() as
430:                                     Artist);
431:                             }
432:                             string fileName = Guid.NewGuid() + ". " +
433:                                 albums[i].AlbumName + " (" + category.CategoryName +
434:                                 ").txt";
435:                             fileName = DesignFileName(fileName);
436:                             //fileName = fileName.Replace(":", ".");
437:                             //fileName = fileName.Replace("/", ".");
438:                             //StringBuilder sb = new StringBuilder();
439:                             //sb.AppendLine(AppResources.NoteName + ": " +
440:                                 notes[i].NoteName);
441:                             //sb.AppendLine(AppResources.FolderName + ": " +
442:                                 noteFolder.NoteFolderName);
443:                             //sb.AppendLine(AppResources.Password + ": " +
444:                                 noteFolder.IsPasswordProtected);
445:                             //sb.AppendLine(AppResources.CreationDate + ": " +
446:                                 notes[i].CreationDate);
447:                             //sb.AppendLine(AppResources.ModificationDate + ": " +
448:                                 notes[i].ModificationDate);

```

```

443:                                     //sb.AppendLine(AppResources.Note + ": " +
notes[i].NoteDescription);
444:
445:
446:                                     myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
447:                                     //if (myIsolatedStorage.FileExists(fileName))
448:                                     //{
449:                                     //myIsolatedStorage.DeleteFile(fileName);
450:                                     //} //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
451:                                     using (StreamWriter writeFile = new StreamWriter
452:                                     (new IsolatedStorageFileStream(fileName,
FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
453:                                     {
454:                                         writeFile.WriteLine(AppResources.AlbumName + ": " +
albums[i].AlbumName);
455:                                         writeFile.WriteLine(AppResources.CategoryName + ": " +
category.CategoryName);
456:                                         string artistNames = "";
457:                                         for (int l = 0; l < artists.Count; l++)
458:                                         {
459:                                             artistNames = artistNames + artists[l].ArtistName
+ ", ";
460:                                         }
461:                                         artistNames = artistNames.Substring(0,
artistNames.Length - 2);
462:                                         writeFile.WriteLine(AppResources.ArtistName + ": " +
artistNames);
463:                                         writeFile.WriteLine(AppResources.ReleaseYear + ": " +
albums[i].AlbumReleaseYear);
464:                                         writeFile.WriteLine(AppResources.SongCount + ": " +
albums[i].AlbumSongCount);
465:                                         writeFile.WriteLine(AppResources.LabelName + ": " +
albums[i].AlbumLabelName);
466:                                         writeFile.WriteLine(AppResources.BestSong + ": " +
albums[i].AlbumBestSong);
467:
468:                                     //writeFile.WriteLine(AppResources.StartDate + ": " +
albums[i].ReadStartDate.ToShortDateString());
469:                                     //writeFile.WriteLine(AppResources.FinishDate + ": " +
albums[i].ReadFinishDate.ToShortDateString());
470:                                     writeFile.WriteLine(AppResources.AlbumRating + ": " +
albums[i].AlbumRating + "/10");
471:                                     writeFile.WriteLine(AppResources.AlbumComment + ": " +
albums[i].AlbumComment);
472:                                     writeFile.Close();
473:                                     }
474:                                     IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.ReadWrite);
475:                                     var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
476:                                     pbSync.Value = (i + 1) * (100) / albums.Count;
477:                                     //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
478:                                     }
479:                                     }
480:                                     }
481:                                     else
482:                                     {
483:                                         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
484:                                         {
485:                                             //var noteFolders = context.NoteFolders.ToList() as
List<NoteFolder>;
486:                                             var albums = context.Albums.OrderBy(j =>
j.CreationDate).ToList() as List<Album>;
487:                                             var albumFirst = albums.First();

```



```

488:         var albumLast = albums.Last();
489:
490:         string fileName = Guid.NewGuid() + ". Awesome Music (" +
albumFirst.CreationDate.ToShortDateString() + " - " +
albumLast.CreationDate.ToShortDateString() + ").txt";
491:         fileName = DesignFileName(fileName);
492:
493:         myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
494:
495:         sb = new StringBuilder();
496:
497:         for (int i = 0; i < albums.Count; i++)
498:         {
499:             var albumArtist =
500:                 context.AlbumArtists.Where(j =>
j.AlbumId.Equals(albums[i].AlbumId)).ToList() as
501:                 List<AlbumArtist>;
502:
503:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(albums[i].AlbumCategoryId)).Single() as Category;
504:
505:             List<Artist> artists = new List<Artist>();
506:
507:             for (int k = 0; k < albumArtist.Count; k++)
508:             {
509:                 artists.Add(context.Artists.Where(j =>
j.ArtistId.Equals(albumArtist[k].ArtistId)).Single() as Artist);
510:             }
511:
512:             sb.AppendLine();
513:             sb.AppendLine(AppResources.AlbumName + ": " +
albums[i].AlbumName);
514:             sb.AppendLine(AppResources.CategoryName + ": " +
category.CategoryName);
515:             string artistNames = "";
516:             for (int l = 0; l < artists.Count; l++)
517:             {
518:                 artistNames = artistNames + artists[l].ArtistName + ",
";
519:             }
520:             artistNames = artistNames.Substring(0, artistNames.Length
- 2);
521:             sb.AppendLine(AppResources.ArtistName + ": " +
artistNames);
522:             sb.AppendLine(AppResources.ReleaseYear + ": " +
albums[i].AlbumReleaseYear);
523:             sb.AppendLine(AppResources.SongCount + ": " +
albums[i].AlbumSongCount);
524:             sb.AppendLine(AppResources.LabelName + ": " +
albums[i].AlbumLabelName);
525:             sb.AppendLine(AppResources.BestSong + ": " +
albums[i].AlbumBestSong);
526:             //sb.AppendLine(AppResources.StartDate + ": " +
albums[i].ReadStartDate.ToShortDateString());
527:             //sb.AppendLine(AppResources.FinishDate + ": " +
albums[i].ReadFinishDate.ToShortDateString());
528:             sb.AppendLine(AppResources.AlbumRating + ": " +
albums[i].AlbumRating + "/10");
529:             sb.AppendLine(AppResources.AlbumComment + ": " +
albums[i].AlbumComment);
530:             sb.AppendLine();
531:
532:             //if (myIsolatedStorage.FileExists(fileName))
533:             //{
534:             //    myIsolatedStorage.DeleteFile(fileName);
535:             //} //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
536:             //pbSync.Value = (i + 1) * (100) / albums.Count;

```



```

537:                                     //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
538:                                     }
539:                                     using (StreamWriter writeFile = new StreamWriter
540:                                         (new IsolatedStorageFileStream(fileName,
FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
541:                                     {
542:                                         writeFile.Write(sb.ToString());
543:                                         writeFile.Close();
544:                                     }
545:                                     IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.ReadWrite);
546:                                     var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
547:                                     }
548:                                 }
549:
550:                                 //this.infoTextBlock.Text = string.Join(" ", "Created folder:",
result.name, "ID:", result.id);
551:                                 MessageBox.Show(AppResources.OneDriveSyncCompleted);
552:
553:                                 pbSync.Visibility = Visibility.Collapsed;
554:                                 txtSynchronizing.Visibility = Visibility.Collapsed;
555:                                 pbSync.Value = 0;
556:                                 btnOneDrive.IsEnabled = true;
557:                             }
558:                             catch (Exception exception)
559:                             {
560:                                 //this.infoTextBlock.Text = "Error creating folder: " +
exception.Message;
561:                                 MessageBox.Show(AppResources.SystemFault);
562:                             }
563:                         }
564:
565:                         public string DesignFileName(string fileName)
566:                         {
567:                             fileName = fileName.Replace(":", ".");
568:                             fileName = fileName.Replace("?", ".");
569:                             fileName = fileName.Replace("\\", ".");
570:                             fileName = fileName.Replace("/", ".");
571:                             fileName = fileName.Replace("<", ".");
572:                             fileName = fileName.Replace(">", ".");
573:                             fileName = fileName.Replace("|", ".");
574:                             fileName = fileName.Replace("*", ".");
575:                             return fileName;
576:                         }
577:
578:                         private void btnBackgroundColor_Click(object sender, RoutedEventArgs e)
579:                         {
580:                             NavigationService.Navigate(new Uri("/BackgroundColorSettingsPage.xaml",
UriKind.Relative));
581:                         }
582:
583:                         private void btnBackgroundImage_Click(object sender, RoutedEventArgs e)
584:                         {
585:                             PhotoChooserTask objPhotoChooser = new PhotoChooserTask();
586:                             objPhotoChooser.Completed += new
EventHandler<PhotoResult>(PhotoChooseCall);
587:                             objPhotoChooser.Show();
588:                         }
589:
590:                         private void PhotoChooseCall(object sender, PhotoResult e)
591:                         {
592:                             switch (e.TaskResult)
593:                             {
594:                                 case TaskResult.OK:
595:                                     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))

```

```

596:         {
597:             var appSettings = context.AppSettings;
598:             foreach (var appSetting in appSettings)
599:             {
600:                 appSetting.AppBackgroundImage = new
byte[e.ChosenPhoto.Length];
601:                 e.ChosenPhoto.Position = 0;
602:                 e.ChosenPhoto.Read(appSetting.AppBackgroundImage, 0,
appSetting.AppBackgroundImage.Length);
603:                 //noteFolder.NoteFolderPassword = "";
604:             }
605:             context.SubmitChanges();
606:             MessageBox.Show(AppResources.BackgroundImageChangeSuccess);
607:         }
608:         break;
609:     case TaskResult.Cancel:
610:         //MessageBox.Show("Cancelled");
611:         break;
612:     case TaskResult.None:
613:         //MessageBox.Show("Nothing Entered");
614:         break;
615:     }
616:     SetBackgroundColor();
617: }
618:
619: private void btnRemoveBackgroundImage_Click(object sender, RoutedEventArgs e)
620: {
621:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
622:     {
623:         var appSettings = context.AppSettings;
624:         foreach (var appSetting in appSettings)
625:         {
626:             appSetting.AppBackgroundImage = null;
627:         }
628:         context.SubmitChanges();
629:         MessageBox.Show(AppResources.BackgroundImageRemoveSuccess);
630:     }
631: }
632:
633: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
634: {
635:
636:     //pvGeneralSettings.Title = AppResources.GeneralSettings;
637:
638:     //piLanguage.Header = AppResources.Language;
639:     //piSync.Header = AppResources.Sync;
640:     //piOtherSettings.Header = AppResources.OtherSettings;
641:     //piBackground.Header = AppResources.Background;
642:
643:     ////lblOneDrive.Text = AppResources.OneDrive;
644:
645:     //btnCategoryOrder.Content = AppResources.Select;
646:     //btnCategoryOrderStyle.Content = AppResources.Select;
647:     //btnLanguage.Content = AppResources.Select;
648:     //btnBackgroundColor.Content = AppResources.Select;
649:     ////btnOneDrive.Content = AppResources.Login;
650:     ////btnOneDrive.SignInText = AppResources.SignIn;
651:     ////btnOneDrive.SignOutText = AppResources.SignOut;
652:     //btnOneDriveSync.Content = AppResources.Sync;
653:     //lblOneDrive.Text = AppResources.OneDrive;
654:     //txtSynchronizing.Text = AppResources.Synchronizing;
655:
656:     //pbSync.Visibility = Visibility.Collapsed;
657:     //txtSynchronizing.Visibility = Visibility.Collapsed;
658:     //txtSynchronizing.BorderBrush = this.LayoutRoot.Background;
659:
660:     //btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
661:     //lblBackgroundImage.Text = AppResources.BackgroundImage;

```

```

662:         //btnBackgroundImage.Content = AppResources.Select;
663:         //btnResetSettings.Content = AppResources.ResetSettings;
664:
665:         //btnOneDriveSync.IsEnabled = false;
666:         //cbSync.Content = AppResources.SyncOnOneFile;
667:         //cbSync.IsEnabled = false;
668:         //btnOneDrive.Content = "Sign In";
669:
670:         //SetBackgroundColor();
671:     }
672:
673:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
674:     {
675:         if (this.NavigationService.CanGoBack)
676:         {
677:             this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
678:         }
679:     }
680:
681:     private void btnOneDrive_SessionChanged(object sender,
Microsoft.Live.Controls.LiveConnectSessionChangedEventArgs e)
682:     {
683:         if (e != null && e.Status == LiveConnectSessionStatus.Connected)
684:         {
685:             //the session status is connected so we need to set this session
status to client
686:             this.liveClient = new LiveConnectClient(e.Session);
687:         }
688:         else
689:         {
690:             this.liveClient = null;
691:         }
692:     }
693:
694:     private void btnLanguage_Click(object sender, RoutedEventArgs e)
695:     {
696:         this.NavigationService.Navigate(new Uri("/LanguageSettingsPage.xaml",
UriKind.Relative));
697:     }
698:
699:     private void btnCategoryOrder_Click(object sender, RoutedEventArgs e)
700:     {
701:         this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml",
UriKind.Relative));
702:     }
703:
704:     private void btnCategoryOrderStyle_Click(object sender, RoutedEventArgs e)
705:     {
706:         this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml",
UriKind.Relative));
707:     }
708:
709:     private void btnResetSettings_Click(object sender, RoutedEventArgs e)
710:     {
711:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
712:         {
713:             var appSettings = context.AppSettings;
714:             foreach (var appSetting in appSettings)
715:             {
716:                 appSetting.AppBackgroundImage = null;
717:                 appSetting.AppBackgroundColor = "BLA";
718:             }
719:             context.SubmitChanges();
720:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
721:             MessageBox.Show(AppResources.SuccessfulResetSettings);
722:         }

```

```

723:         }
724:     }
725: }

```

Namespaces

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

1.2.24 LanguageSettingsPage.xaml.cs

This is file LanguageSettingsPage.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 AwesomeMusic.Resources;
14: using Microsoft.Phone.Controls;
15: using Microsoft.Phone.Shell;
16:
17: namespace AwesomeMusic
18: {
19:     public partial class LanguageSettingsPage : PhoneApplicationPage
20:     {
21:         public LanguageSettingsPage()
22:         {
23:             InitializeComponent();
24:
25:             lstLanguage.Items.Clear();
26:             lstLanguage.Items.Add(AppResources.English);
27:             lstLanguage.Items.Add(AppResources.Turkish);
28:             lstLanguage.Items.Add(AppResources.German);
29:             //lstLanguage.Items.Add(AppResources.Spanish);
30:             lstLanguage.Items.Add(AppResources.Russian);
31:             lstLanguage.Items.Add(AppResources.Arabic);
32:             lstLanguage.Items.Add(AppResources.Persian);
33:             lstLanguage.Items.Add(AppResources.Chinese);
34:             lstLanguage.Items.Add(AppResources.Italian);
35:             lstLanguage.Items.Add(AppResources.French);
36:             lstLanguage.Items.Add(AppResources.Japanese);
37:             lstLanguage.Items.Add(AppResources.Sanskrit);
38:             lstLanguage.Items.Add(AppResources.Thai);
39:
40:             lstLanguage.SelectedIndex = -1;
41:             lblLanguage.Text = AppResources.SelectLanguage;
42:             lblGeneralSettings.Text = AppResources.GeneralSettings;
43:
44:             SetBackgroundColor();
45:         }
46:
47:         protected override void OnNavigatedTo(NavigationEventArgs e)
48:         {
49:             base.OnNavigatedTo(e);
50:             SetBackgroundColor();

```

```

51:         }
52:
53:         protected override void OnNavigatedFrom(NavigationEventArgs e)
54:         {
55:             base.OnNavigatedFrom(e);
56:         }
57:
58:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
59:         {
60:             if (this.NavigationService.CanGoBack)
61:             {
62:                 this.NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
63:             }
64:         }
65:
66:         private void SetBackgroundColor()
67:         {
68:             AppSettings appSettings = new AppSettings();
69:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
70:             {
71:                 appSettings = context.AppSettings.First() as AppSettings;
72:             }
73:
74:             if (appSettings.AppBackgroundImage != null)
75:             {
76:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
77:                 BitmapImage image = new BitmapImage();
78:                 image.SetSource(stream);
79:                 ImageBrush ib = new ImageBrush();
80:                 ib.ImageSource = image;
81:                 this.LayoutRoot.Background = ib;
82:             }
83:             else
84:             {
85:                 switch (appSettings.AppBackgroundColor)
86:                 {
87:                     case "BLA":
88:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
89:                         break;
90:                     case "BLU":
91:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
92:                         break;
93:                     case "BRO":
94:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
95:                         break;
96:                     case "RED":
97:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
98:                         break;
99:                     case "GRE":
100:                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
101:                        break;
102:                     case "GRA":
103:                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
104:                        break;
105:                     case "YEL":
106:                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
107:                        break;
108:                     case "ORA":
109:                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
110:                        break;
111:                     case "PUR":
112:                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
113:                        break;

```

```
114:             default:
115:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
116:                 break;
117:             }
118:         }
119:     }
120:
121:     private void lstLanguage_SelectionChanged(object sender,
SelectionChangedEventArgs e)
122:     {
123:         int index = lstLanguage.SelectedIndex;
124:         string culture = "";
125:         string lang = "";
126:         if (index == 0)
127:         {
128:             culture = "en";
129:             lang = "EN";
130:         }
131:         else if (index == 1)
132:         {
133:             culture = "tr";
134:             lang = "TR";
135:         }
136:         else if (index == 2)
137:         {
138:             culture = "de";
139:             lang = "DE";
140:         }
141:         else if (index == 3)
142:         {
143:             culture = "ru";
144:             lang = "RU";
145:         }
146:         else if (index == 4)
147:         {
148:             culture = "ar";
149:             lang = "AR";
150:         }
151:         else if (index == 5)
152:         {
153:             culture = "fa-IR";
154:             lang = "FA";
155:         }
156:         else if (index == 6)
157:         {
158:             culture = "zh";
159:             lang = "ZH";
160:         }
161:         else if (index == 7)
162:         {
163:             culture = "it";
164:             lang = "IT";
165:         }
166:         else if (index == 8)
167:         {
168:             culture = "fr";
169:             lang = "FR";
170:         }
171:         else if (index == 9)
172:         {
173:             culture = "ja";
174:             lang = "JA";
175:         }
176:         else if (index == 10)
177:         {
178:             culture = "sa";
179:             lang = "SA";
180:         }
181:         else if (index == 11)
```

```

182:         {
183:             culture = "th";
184:             lang = "TH";
185:         }
186:         //else if (index == 3)
187:         //{
188:             culture = "es";
189:             lang = "ES";
190:         //}
191:         //else if (index == 4)
192:         //{
193:             culture = "ru";
194:             lang = "RU";
195:         //}
196:         //else if (index == 5)
197:         //{
198:             culture = "zh";
199:             lang = "AR";
200:         //}
201:         //else if (index == 6)
202:         //{
203:             culture = "ar";
204:             lang = "AR";
205:         //}
206:         //else if (index == 7)
207:         //{
208:             culture = "fa-IR";
209:             lang = "FA";
210:         //}
211:         //else if (index == 8)
212:         //{
213:             culture = "it";
214:             lang = "IT";
215:         //}
216:         //else if (index == 9)
217:         //{
218:             culture = "fr";
219:             lang = "FR";
220:         //}
221:         //else if (index == 10)
222:         //{
223:             culture = "pt";
224:             lang = "PT";
225:         //}
226:         else
227:         {
228:             culture = "en";
229:             lang = "EN";
230:         }
231:
232:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
233:         {
234:             var appSettings = context.AppSettings;
235:             foreach (var appSetting in appSettings)
236:             {
237:                 appSetting.AppLangName = lang;
238:             }
239:             context.SubmitChanges();
240:         }
241:
242:         CultureInfo newCulture = new CultureInfo(culture);
243:         Thread.CurrentThread.CurrentCulture = newCulture;
244:         Thread.CurrentThread.CurrentUICulture = newCulture;
245:         MessageBox.Show(AppResources.LanguageWarning);
246:         NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
247:     }
248:

```

```
249:         private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
250:         {
251:             //SetBackgroundColor();
252:         }
253:     }
254: }
```

Namespaces

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

1.2.25 LocalizedStrings.cs

This is file LocalizedStrings.cs.

Body Source

```
1: ?using AwesomeMusic.Resources;
2:
3: namespace AwesomeMusic
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
AwesomeMusic (see page 1)	This is namespace AwesomeMusic.

1.2.26 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 System.Data.Common;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Controls.Primitives;
15: using Microsoft.Phone.Shell;
16: using AwesomeMusic.Resources;
17:
```



```

18: namespace AwesomeMusic
19: {
20:     public partial class MainPage : PhoneApplicationPage
21:     {
22:         public Popup popup;
23:         // Constructor
24:         public MainPage()
25:         {
26:             InitializeComponent();
27:
28:
29:             ApplicationBar = new ApplicationBar();
30:
31:             ApplicationBarIconButton button1 = new ApplicationBarIconButton();
32:             button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
33:             button1.Text = AppResources.AddCategory;
34:             ApplicationBar.Buttons.Add(button1);
35:             button1.Click += new EventHandler(AddCategoryButton_Click);
36:
37:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
38:             button2.IconUri = new Uri("/Assets/Search.png", UriKind.Relative);
39:             button2.Text = AppResources.Search;
40:             ApplicationBar.Buttons.Add(button2);
41:             button2.Click += new EventHandler(SearchButton_Click);
42:
43:             ApplicationBarIconButton button3 = new ApplicationBarIconButton();
44:             button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
45:             button3.Text = AppResources.Settings;
46:             ApplicationBar.Buttons.Add(button3);
47:             button3.Click += new EventHandler(SettingsButton_Click);
48:
49:             ApplicationBarIconButton button4 = new ApplicationBarIconButton();
50:             button4.IconUri = new Uri("/Assets/Statistics.png", UriKind.Relative);
51:             button4.Text = AppResources.Statistics;
52:             ApplicationBar.Buttons.Add(button4);
53:             button4.Click += new EventHandler(StatisticsButton_Click);
54:
55:             ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
56:             menuItem1.Text = AppResources.About;
57:             ApplicationBar.MenuItems.Add(menuItem1);
58:             menuItem1.Click += new EventHandler(AboutMenuItem_Click);
59:
60:             lblCategories.Text = AppResources.Categories;
61:             // Sample code to localize the ApplicationBar
62:             //BuildLocalizedApplicationBar();
63:
64:             SetBackgroundColor();
65:
66:             popup = new Popup();
67:         }
68:
69:         private void SetBackgroundColor()
70:         {
71:             AppSettings appSettings = new AppSettings();
72:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
73:             {
74:                 appSettings = context.AppSettings.First() as AppSettings;
75:             }
76:
77:             if (appSettings.AppBackgroundImage != null)
78:             {
79:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
80:                 BitmapImage image = new BitmapImage();
81:                 image.SetSource(stream);
82:                 ImageBrush ib = new ImageBrush();
83:                 ib.ImageSource = image;
84:                 this.LayoutRoot.Background = ib;
85:             }

```

```
86:         else
87:         {
88:             switch (appSettings.AppBackgroundColor)
89:             {
90:                 case "BLA":
91:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
92:                     break;
93:                 case "BLU":
94:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
95:                     break;
96:                 case "BRO":
97:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
98:                     break;
99:                 case "RED":
100:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
101:                     break;
102:                 case "GRE":
103:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
104:                     break;
105:                 case "GRA":
106:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
107:                     break;
108:                 case "YEL":
109:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
110:                     break;
111:                 case "ORA":
112:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
113:                     break;
114:                 case "PUR":
115:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
116:                     break;
117:                 default:
118:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
119:                     break;
120:             }
121:         }
122:     }
123:
124:     protected override void OnNavigatedTo(NavigationEventArgs e)
125:     {
126:         base.OnNavigatedTo(e);
127:         //while (NavigationService.CanGoBack)
128:         //NavigationService.RemoveBackEntry();
129:     }
130:
131:
132:     protected override void OnNavigatedFrom(NavigationEventArgs e)
133:     {
134:         base.OnNavigatedFrom(e);
135:         //while (NavigationService.CanGoBack)
136:         //NavigationService.RemoveBackEntry();
137:     }
138:
139:
140:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
141:     {
142:         if (popup.IsOpen)
143:         {
144:             popup.IsOpen = false;
145:         }
146:         if (MessageBox.Show(AppResources.ExitAppQuestion,
AppResources.ExitApp, MessageBoxButton.OKCancel)
147:             != MessageBoxResult.OK)
148:         {
149:             e.Cancel = true;
150:         }
151:     }
```

```

151:         }
152:         else
153:         {
154:             Application.Current.Terminate();
155:         }
156:     }
157:
158:     private void lstCategories_SelectionChanged(object sender,
SelectionChangedEventArgs e)
159:     {
160:         var category = (Category)lstCategories.SelectedItem;
161:         int categoryId = category.CategoryId;
162:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
163:         {
164:             var appSettings = context.AppSettings;
165:             foreach (var appSetting in appSettings)
166:             {
167:                 appSetting.CurrentCategoryNumber = categoryId;
168:             }
169:             context.SubmitChanges();
170:         }
171:         NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId,
UriKind.Relative));
172:     }
173:
174:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
175:     {
176:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
177:         {
178:             var appSettings = context.AppSettings.First() as AppSettings;
179:             string orderStyle = appSettings.CategoryOrderStyle;
180:             //lstCategories.ItemsSource = null;
181:             lstCategories.Items.Clear();
182:             if (context.Categories.Count() != 0)
183:             {
184:                 switch (appSettings.CategoryOrderBy)
185:                 {
186:                     case "NAME":
187:                         if (orderStyle == "A")
188:                         {
189:                             lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.CategoryName).ToList();
190:                         }
191:                         else
192:                         {
193:                             lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.CategoryName).ToList();
194:                         }
195:                         break;
196:                     case "CDATE":
197:                         if (orderStyle == "A")
198:                         {
199:                             lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.CreationDate).ToList();
200:                         }
201:                         else
202:                         {
203:                             lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.CreationDate).ToList();
204:                         }
205:                         break;
206:                     case "MDATE":
207:                         if (orderStyle == "A")
208:                         {
209:                             lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.ModificationDate).ToList();
210:                         }

```

```

211:                 else
212:                 {
213:                     lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.ModificationDate).ToList();
214:                 }
215:                 break;
216:                 case "ALBUMCOUNT":
217:                     if (orderStyle == "A")
218:                     {
219:                         lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.CategoryAlbumCount).ToList();
220:                     }
221:                     else
222:                     {
223:                         lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.CategoryAlbumCount).ToList();
224:                     }
225:                     break;
226:                 default:
227:                     if (orderStyle == "A")
228:                     {
229:                         lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.CategoryName).ToList();
230:                     }
231:                     else
232:                     {
233:                         lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.CategoryName).ToList();
234:                     }
235:                     break;
236:                 }
237:                 lstCategories.DisplayMemberPath = "CategoryNameCount";
238:                 SetBackgroundColor();
239:             }
240:         }
241:
242:         // Sample code for building a localized ApplicationBar
243:         //private void BuildLocalizedApplicationBar()
244:         //{
245:             // // Set the page's ApplicationBar to a new instance of ApplicationBar.
246:             // ApplicationBar = new ApplicationBar();
247:
248:             // // Create a new button and set the text value to the localized
string from AppResources.
249:             // ApplicationBarIconButton appBarButton = new
ApplicationBarIconButton(new Uri("/Assets/AppBar/appbar.add.rest.png", UriKind.Relative));
250:             // appBarButton.Text = AppResources.AppBarButtonText;
251:             // ApplicationBar.Buttons.Add(appBarButton);
252:
253:             // // Create a new menu item with the localized string from
AppResources.
254:             // ApplicationBarItem appBarMenuItem = new
ApplicationBarItem(AppResources.AppBarMenuItemText);
255:             // ApplicationBar.MenuItems.Add(appBarMenuItem);
256:             //}
257:         }
258:
259:         private void AddCategoryButton_Click(object sender, EventArgs e)
260:         {
261:             int categoryId;
262:
263:             popup = new Popup();
264:             popup.Height = 300;
265:             popup.Width = 400;
266:             popup.VerticalOffset = 20;
267:             PopupAddChange control = new PopupAddChange();
268:             control.txtLabel.Text = AppResources.EnterCategoryName;
269:             control.btnCancel.Content = AppResources.Cancel;
270:             control.btnOK.Content = AppResources.OK;

```

```

271:         popup.Child = control;
272:         popup.IsOpen = true;
273:         control.txtName.Focus();
274:
275:         control.btnOK.Click += (s, args) =>
276:         {
277:             bool folder = false;
278:             string categoryName;
279:             popup.IsOpen = false;
280:
281:             int length = control.txtName.Text.Length;
282:             string space = control.txtName.Text.Substring(length - Math.Min(1,
length));
283:             if (space == " ")
284:             {
285:                 categoryName = control.txtName.Text.Remove(length - 1, 1);
286:             }
287:             else
288:             {
289:                 categoryName = control.txtName.Text;
290:             }
291:
292:             // aynı isimde bir klasörün daha önceden oluşturulup oluşturulmadığını
293:             // kontrol eden bir kod bölümü
294:             using (var contextFolder = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
295:             {
296:                 folder =
297:                     contextFolder.Categories.Any(j =>
j.CategoryName.Equals(categoryName));
298:             }
299:             if (folder == true)
300:             {
301:                 MessageBox.Show(AppResources.CategoryExists);
302:             }
303:             // eğer bu isimde bir klasör oluşturulmamışsa
304:             // oluşturulması için gerekli kodlar aşağıdadır
305:             else
306:             {
307:                 using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
308:                 {
309:                     //noteFolderId = context.NoteFolders.Count() + 1;
310:                     Category category = new Category();
311:                     category.CategoryName = categoryName;
312:                     //noteFolder.NoteFolderId = noteFolderId;
313:                     category.CategoryAlbumCount = 0;
314:                     category.CreationDate = DateTime.Now;
315:                     category.ModificationDate = DateTime.Now;
316:                     category.ArtistOrderBy = "NAME";
317:                     category.ArtistOrderStyle = "A";
318:                     category.CategoryNameCount = category.CategoryName + " (" +
category.CategoryAlbumCount.ToString() + ")";
319:
320:                     context.Categories.InsertOnSubmit(category);
321:                     context.SubmitChanges();
322:
323:                     //lstCategories.ItemsSource = context.Categories;
324:                     MessageBox.Show(AppResources.CategoryAddSuccess);
325:                     popup.IsOpen = false;
326:                     Category category2 = context.Categories.Where(j =>
j.CategoryName.Equals(categoryName)).Single() as Category;
327:                     NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
category2.CategoryId, UriKind.Relative));
328:                 }
329:             }
330:         }
331:     };
332:     control.btnCancel.Click += (s, args) =>

```

```
333:         {
334:             popup.IsOpen = false;
335:         };
336:
337:         //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
338:     }
339:
340:     private void SettingsButton_Click(object sender, EventArgs e)
341:     {
342:         NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
343:     }
344:
345:     private void SearchButton_Click(object sender, EventArgs e)
346:     {
347:         NavigationService.Navigate(new Uri("/SearchPage.xaml", UriKind.Relative));
348:     }
349:
350:     private void AboutMenuItem_Click(object sender, EventArgs e)
351:     {
352:         NavigationService.Navigate(new Uri("/AboutPage.xaml", UriKind.Relative));
353:     }
354:     private void StatisticsButton_Click(object sender, EventArgs e)
355:     {
356:         NavigationService.Navigate(new Uri("/StatisticsPage.xaml",
UriKind.Relative));
357:     }
358: }
359: }
360: }
```

Namespaces

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

1.2.27 OrderSettingsPage.xaml.cs

This is file OrderSettingsPage.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 AwesomeMusic.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16: namespace AwesomeMusic
17: {
18:     public partial class OrderSettingsPage : PhoneApplicationPage
19:     {
20:         public string pageName;
21:         public int categoryId;
22:         public int artistId;
23:         public OrderSettingsPage()
```

```

24:         {
25:             InitializeComponent();
26:             SetBackgroundColor();
27:         }
28:
29:         private void lstOrderBy_SelectionChanged(object sender,
SelectionChangedEventArgs e)
30:         {
31:             int index = lstOrderBy.SelectedIndex;
32:             string orderType = "";
33:
34:             if (pageName.Contains("/GeneralSettingsPage.xaml"))
35:             {
36:                 if (index == 0)
37:                 {
38:                     orderType = "NAME";
39:                 }
40:                 else if (index == 1)
41:                 {
42:                     orderType = "ALBUMCOUNT";
43:                 }
44:                 else if (index == 2)
45:                 {
46:                     orderType = "CDATE";
47:                 }
48:                 else if (index == 3)
49:                 {
50:                     orderType = "MDATE";
51:                 }
52:                 else
53:                 {
54:                     orderType = "NAME";
55:                 }
56:
57:                 using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
58:                 {
59:                     var appSettings = context.AppSettings;
60:                     foreach (var appSetting in appSettings)
61:                     {
62:                         appSetting.CategoryOrderBy = orderType;
63:                     }
64:                     context.SubmitChanges();
65:                     MessageBox.Show(AppResources.CategoryOrderTypeChangeSuccess);
66:                 }
67:                 NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
68:             }
69:             else if (pageName.Contains("/CategorySettingsPage.xaml"))
70:             {
71:                 if (index == 0)
72:                 {
73:                     orderType = "NAME";
74:                 }
75:                 else if (index == 1)
76:                 {
77:                     orderType = "ALBUMCOUNT";
78:                 }
79:                 else if (index == 2)
80:                 {
81:                     orderType = "CDATE";
82:                 }
83:                 else if (index == 3)
84:                 {
85:                     orderType = "MDATE";
86:                 }
87:                 else
88:                 {
89:                     orderType = "NAME";

```

```

90:         }
91:
92:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
93:         {
94:             var categories = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).ToList() as List<Category>;
95:             foreach (var category in categories)
96:             {
97:                 category.ArtistOrderBy = orderType;
98:             }
99:             context.SubmitChanges();
100:            MessageBox.Show(AppResources.ArtistOrderTypeChangeSuccess);
101:        }
102:        NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" +
categoryId, UriKind.Relative));
103:    }
104:    else
105:    {
106:        if (index == 0)
107:        {
108:            orderType = "NAME";
109:        }
110:        else if (index == 1)
111:        {
112:            orderType = "CDATE";
113:        }
114:        else if (index == 2)
115:        {
116:            orderType = "MDATE";
117:        }
118:        else if (index == 3)
119:        {
120:            orderType = "RATING";
121:        }
122:        else
123:        {
124:            orderType = "NAME";
125:        }
126:
127:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
128:        {
129:            var artists = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).ToList() as List<Artist>;
130:            foreach (var artist in artists)
131:            {
132:                artist.AlbumOrderBy = orderType;
133:            }
134:            context.SubmitChanges();
135:            MessageBox.Show(AppResources.AlbumOrderTypeChangeSuccess);
136:        }
137:        NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" +
artistId, UriKind.Relative));
138:    }
139:    }
140:
141:    protected override void OnNavigatedTo(NavigationEventArgs e)
142:    {
143:        base.OnNavigatedTo(e);
144:        // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor
bu bölüm
145:        var lastPage = NavigationService.BackStack.FirstOrDefault();
146:        pageName = lastPage.Source.ToString();
147:        lstOrderBy.Items.Clear();
148:        if (pageName.Contains("/GeneralSettingsPage.xaml"))
149:        {
150:            lblSettings.Text = AppResources.GeneralSettings;
151:            using (var context = new

```



```

AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
152:         {
153:             var appSettings =
154:                 context.AppSettings.First();
155:             lblOrderBy.Text = AppResources.CategoryOrderBy;
156:
157:             lstOrderBy.Items.Add(AppResources.Name);
158:             lstOrderBy.Items.Add(AppResources.AlbumCount);
159:             lstOrderBy.Items.Add(AppResources.CreationDate);
160:             lstOrderBy.Items.Add(AppResources.ModificationDate);
161:
162:         }
163:     }
164: }
165:
166: protected override void OnNavigatedFrom(NavigationEventArgs e)
167: {
168:     base.OnNavigatedFrom(e);
169: }
170:
171: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
172: {
173:     // displays "Fragment: Detail"
174:     //MessageBox.Show("Folder Id: " + e.Fragment);
175:     base.OnFragmentNavigation(e);
176:     lstOrderBy.Items.Clear();
177:     if (pageName.Contains("/CategorySettingsPage.xaml"))
178:     {
179:         categoryId = int.Parse(e.Fragment);
180:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
181:         {
182:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
183:             lblSettings.Text = AppResources.CategorySettings + " (" +
category.CategoryName + ")";
184:             lblOrderBy.Text = AppResources.ArtistOrderBy;
185:             lstOrderBy.Items.Add(AppResources.Name);
186:             lstOrderBy.Items.Add(AppResources.AlbumCount);
187:             lstOrderBy.Items.Add(AppResources.CreationDate);
188:             lstOrderBy.Items.Add(AppResources.ModificationDate);
189:         }
190:     }
191:     else
192:     {
193:         artistId = int.Parse(e.Fragment);
194:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
195:         {
196:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
197:             lblSettings.Text = AppResources.ArtistSettings + " (" +
artist.ArtistName + ")";
198:             lblOrderBy.Text = AppResources.AlbumOrderBy;
199:             lstOrderBy.Items.Add(AppResources.Name);
200:             lstOrderBy.Items.Add(AppResources.CreationDate);
201:             lstOrderBy.Items.Add(AppResources.ModificationDate);
202:             lstOrderBy.Items.Add(AppResources.AlbumRating);
203:         }
204:     }
205:     lstOrderBy.SelectedIndex = -1;
206:     SetBackgroundColor();
207: }
208:
209: private void SetBackgroundColor()
210: {
211:     AppSettings appSettings = new AppSettings();
212:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))

```

```

213:         {
214:             appSettings = context.AppSettings.First() as AppSettings;
215:         }
216:
217:         if (appSettings.AppBackgroundImage != null)
218:         {
219:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
220:             BitmapImage image = new BitmapImage();
221:             image.SetSource(stream);
222:             ImageBrush ib = new ImageBrush();
223:             ib.ImageSource = image;
224:             this.LayoutRoot.Background = ib;
225:         }
226:         else
227:         {
228:             switch (appSettings.AppBackgroundColor)
229:             {
230:                 case "BLA":
231:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
232:                     break;
233:                 case "BLU":
234:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
235:                     break;
236:                 case "BRO":
237:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
238:                     break;
239:                 case "RED":
240:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
241:                     break;
242:                 case "GRE":
243:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
244:                     break;
245:                 case "GRA":
246:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
247:                     break;
248:                 case "YEL":
249:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
250:                     break;
251:                 case "ORA":
252:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
253:                     break;
254:                 case "PUR":
255:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
256:                     break;
257:                 default:
258:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
259:                     break;
260:             }
261:         }
262:     }
263:
264:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
265:     {
266:         if (this.NavigationService.CanGoBack)
267:         {
268:             if (pageName.Contains("/GeneralSettingsPage.xaml"))
269:             {
270:                 this.NavigationService.Navigate(new
Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
271:             }
272:             else if (pageName.Contains("/CategorySettingsPage.xaml"))
273:             {
274:                 this.NavigationService.Navigate(new
Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
275:             }

```

```
276:         else
277:         {
278:             this.NavigationService.Navigate(new
Uri("/ArtistSettingsPage.xaml#" + artistId, UriKind.Relative));
279:         }
280:     }
281: }
282:
283: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
284: {
285:     //SetBackgroundColor();
286: }
287: }
288: }
```

Namespaces

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

1.2.28 OrderStyleSettingsPage.xaml.cs

This is file OrderStyleSettingsPage.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 AwesomeMusic.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16:
17: namespace AwesomeMusic
18: {
19:     public partial class OrderStyleSettingsPage : PhoneApplicationPage
20:     {
21:         public string pageName;
22:         public int categoryId;
23:         public int artistId;
24:
25:         public OrderStyleSettingsPage()
26:         {
27:             InitializeComponent();
28:
29:             lstOrderStyle.Items.Clear();
30:
31:             lstOrderStyle.Items.Add(AppResources.Ascending);
32:             lstOrderStyle.Items.Add(AppResources.Descending);
33:
34:             lstOrderStyle.SelectedIndex = -1;
35:
36:             SetBackgroundColor();
37:         }
38:
39:         private void PhoneApplicationPage_BackKeyPress(object sender,
```

```

System.ComponentModel.CancelEventArgs e)
40:     {
41:         if (this.NavigationService.CanGoBack)
42:         {
43:             if (pageName.Contains("/GeneralSettingsPage.xaml"))
44:             {
45:                 this.NavigationService.Navigate(new
Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
46:             }
47:             else if (pageName.Contains("/CategorySettingsPage.xaml"))
48:             {
49:                 this.NavigationService.Navigate(new
Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
50:             }
51:             else
52:             {
53:                 this.NavigationService.Navigate(new
Uri("/ArtistSettingsPage.xaml#" + artistId, UriKind.Relative));
54:             }
55:         }
56:     }
57:
58:     private void lstOrderStyle_SelectionChanged(object sender,
SelectionChangedEventArgs e)
59:     {
60:         int index = lstOrderStyle.SelectedIndex;
61:         string orderStyle = "";
62:
63:         if (pageName.Contains("/GeneralSettingsPage.xaml"))
64:         {
65:             if (index == 0)
66:             {
67:                 orderStyle = "A";
68:             }
69:             else
70:             {
71:                 orderStyle = "D";
72:             }
73:
74:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
75:             {
76:                 var appSettings = context.AppSettings;
77:                 foreach (var appSetting in appSettings)
78:                 {
79:                     appSetting.CategoryOrderStyle = orderStyle;
80:                 }
81:                 context.SubmitChanges();
82:                 MessageBox.Show(AppResources.CategoryOrderStyleChangeSuccess);
83:             }
84:             NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
85:         }
86:         else if (pageName.Contains("/CategorySettingsPage.xaml"))
87:         {
88:             if (index == 0)
89:             {
90:                 orderStyle = "A";
91:             }
92:             else
93:             {
94:                 orderStyle = "D";
95:             }
96:
97:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
98:             {
99:                 var categories = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).ToList() as List<Category>;

```

```

100:                foreach (var category in categories)
101:                {
102:                    category.ArtistOrderStyle = orderStyle;
103:                }
104:                context.SubmitChanges();
105:                MessageBox.Show(AppResources.ArtistOrderStyleChangeSuccess);
106:            }
107:            NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" +
categoryId, UriKind.Relative));
108:        }
109:        else
110:        {
111:            if (index == 0)
112:            {
113:                orderStyle = "A";
114:            }
115:            else
116:            {
117:                orderStyle = "D";
118:            }
119:        }
120:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
121:        {
122:            var artists = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).ToList() as List<Artist>;
123:            foreach (var artist in artists)
124:            {
125:                artist.AlbumOrderStyle = orderStyle;
126:            }
127:            context.SubmitChanges();
128:            MessageBox.Show(AppResources.AlbumOrderStyleChangeSuccess);
129:        }
130:        NavigationService.Navigate(new Uri("/ArtistSettingsPage.xaml#" +
artistId, UriKind.Relative));
131:    }
132:}
133:
134:protected override void OnNavigatedTo(NavigationEventArgs e)
135:{
136:    base.OnNavigatedTo(e);
137:    // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor
bu bölüm
138:    var lastPage = NavigationService.BackStack.FirstOrDefault();
139:    pageName = lastPage.Source.ToString();
140:    if (pageName.Contains("/GeneralSettingsPage.xaml"))
141:    {
142:        lblSettings.Text = AppResources.GeneralSettings;
143:        using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
144:        {
145:            var appSettings =
146:                context.AppSettings.First();
147:            lblOrderStyle.Text = AppResources.CategoryOrderStyle;
148:        }
149:    }
150:}
151:
152:protected override void OnNavigatedFrom(NavigationEventArgs e)
153:{
154:    base.OnNavigatedFrom(e);
155:}
156:
157:protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
158:{
159:    // displays "Fragment: Detail"
160:    //MessageBox.Show("Folder Id: " + e.Fragment);
161:    base.OnFragmentNavigation(e);
162:    if (pageName.Contains("/CategorySettingsPage.xaml"))

```

```

163:         {
164:             categoryId = int.Parse(e.Fragment);
165:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
166:             {
167:                 var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
168:                 lblSettings.Text = AppResources.CategorySettings + " (" +
category.CategoryName + ")";
169:                 lblOrderStyle.Text = AppResources.ArtistOrderStyle;
170:             }
171:         }
172:     else
173:     {
174:         artistId = int.Parse(e.Fragment);
175:         using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
176:         {
177:             var artist = context.Artists.Where(j =>
j.ArtistId.Equals(artistId)).Single() as Artist;
178:             lblSettings.Text = AppResources.ArtistSettings + " (" +
artist.ArtistName + ")";
179:             lblOrderStyle.Text = AppResources.AlbumOrderStyle;
180:         }
181:     }
182:     SetBackgroundColor();
183: }
184:
185: private void SetBackgroundColor()
186: {
187:     AppSettings appSettings = new AppSettings();
188:     using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
189:     {
190:         appSettings = context.AppSettings.First() as AppSettings;
191:     }
192:
193:     if (appSettings.AppBackgroundImage != null)
194:     {
195:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
196:         BitmapImage image = new BitmapImage();
197:         image.SetSource(stream);
198:         ImageBrush ib = new ImageBrush();
199:         ib.ImageSource = image;
200:         this.LayoutRoot.Background = ib;
201:     }
202:     else
203:     {
204:         switch (appSettings.AppBackgroundColor)
205:         {
206:             case "BLA":
207:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
208:                 break;
209:             case "BLU":
210:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
211:                 break;
212:             case "BRO":
213:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
214:                 break;
215:             case "RED":
216:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
217:                 break;
218:             case "GRE":
219:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
220:                 break;
221:             case "GRA":
222:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
223:                 break;
224:             case "YEL":

```

```
225:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
226:             break;
227:             case "ORA":
228:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
229:             break;
230:             case "PUR":
231:             this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
232:             break;
233:             default:
234:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
235:             break;
236:         }
237:     }
238: }
239:
240: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
241: {
242:     //SetBackgroundColor();
243: }
244: }
245: }
```

Namespaces

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

1.2.29 PopupAddChange.xaml.cs

This is file PopupAddChange.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:
13: namespace AwesomeMusic
14: {
15:     public partial class PopupAddChange : UserControl
16:     {
17:         public PopupAddChange()
18:         {
19:             InitializeComponent();
20:             SetPopupBackgroundColor();
21:         }
22:         private void SetPopupBackgroundColor()
23:         {
24:             AppSettings appSettings = new AppSettings();
25:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.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
AwesomeMusic (↗ see page 1)	This is namespace AwesomeMusic.

1.2.30 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.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 AwesomeMusic.Resources;
16: using Microsoft.Phone.Controls;
```



```

17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeMusic
20: {
21:     public partial class SearchPage : PhoneApplicationPage
22:     {
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 SetBackgroundColor()
35:         {
36:             AppSettings appSettings = new AppSettings();
37:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
38:             {
39:                 appSettings = context.AppSettings.First() as AppSettings;
40:             }
41:
42:             if (appSettings.AppBackgroundImage != null)
43:             {
44:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
45:                 BitmapImage image = new BitmapImage();
46:                 image.SetSource(stream);
47:                 ImageBrush ib = new ImageBrush();
48:                 ib.ImageSource = image;
49:                 this.LayoutRoot.Background = ib;
50:             }
51:             else
52:             {
53:                 switch (appSettings.AppBackgroundColor)
54:                 {
55:                     case "BLA":
56:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
57:                         break;
58:                     case "BLU":
59:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
60:                         break;
61:                     case "BRO":
62:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
63:                         break;
64:                     case "RED":
65:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
66:                         break;
67:                     case "GRE":
68:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
69:                         break;
70:                     case "GRA":
71:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
72:                         break;
73:                     case "YEL":
74:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
75:                         break;
76:                     case "ORA":
77:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
78:                         break;
79:                     case "PUR":
80:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
81:                         break;

```

```

82:                 default:
83:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
84:                     break;
85:             }
86:         }
87:     }
88:
89:     private void btnSearch_Click(object sender, RoutedEventArgs e)
90:     {
91:         if (txtSearch.Text.TrimStart().Length < 1)
92:         {
93:             MessageBox.Show(AppResources.SearchTrimFault);
94:         }
95:         else
96:         {
97:             lstSearch.Items.Clear();
98:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
99:             {
100:                 var albumList =
101:                     context.Albums.Where(j =>
j.AlbumInformation.ToLower().Contains(txtSearch.Text.ToLower())).ToList() as List<Album>;
102:                 //var noteList = context.Notes.ToList() as List<Note>;
103:
104:                 if (albumList != null)
105:                 {
106:                     txtSearchResult.Text = AppResources.SearchResults + " (" +
albumList.Count() + ")";
107:                 }
108:
109:                 //lstSearch.ItemsSource = noteList;
110:                 for (int i = 0; i < albumList.Count; i++)
111:                 {
112:                     //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Index
Of(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
113:                     //{
114:                         lstSearch.Items.Add(albumList[i] as Album);
115:                     //}
116:                 }
117:                 //lstSearch.ItemTemplate.
118:                 //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate"
+ ")";
119:                 lstSearch.DisplayMemberPath = "AlbumNameRating";
120:                 MessageBox.Show(AppResources.SearchCompleted);
121:             }
122:         }
123:     }
124:
125:     private void lstSearch_SelectionChanged(object sender,
SelectionChangedEventArgs e)
126:     {
127:         try
128:         {
129:             if (lstSearch.SelectedIndex != -1)
130:             {
131:
132:                 Album selectedAlbum = lstSearch.SelectedItem as Album;
133:                 using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
134:                 {
135:                     var appSettings = context.AppSettings;
136:                     var category =
137:                         context.Categories.Where(j =>
j.CategoryId.Equals(selectedAlbum.AlbumCategoryId)).Single() as
138:                         Category;
139:
140:                     var artistAlbum = context.AlbumArtists.Where(j =>

```

```

j.AlbumId.Equals(selectedAlbum.AlbumId)).Single() as AlbumArtist;
141:
142:         foreach (var item in appSettings)
143:         {
144:             item.CurrentCategoryNumber = category.CategoryId;
145:             item.CurrentArtistNumber = artistAlbum.ArtistId;
146:         }
147:         context.SubmitChanges();
148:
149:         NavigationService.Navigate(new Uri("/AlbumPage.xaml#" +
selectedAlbum.AlbumId, UriKind.Relative));
150:     }
151:     lstSearch.SelectedIndex = -1;
152: }
153:
154:     }
155:     catch (Exception)
156:     {
157:         MessageBox.Show(AppResources.SystemFault);
158:     }
159: }
160:
161: protected override void OnNavigatedTo(NavigationEventArgs e)
162: {
163:     base.OnNavigatedTo(e);
164: }
165: protected override void OnNavigatedFrom(NavigationEventArgs e)
166: {
167:     base.OnNavigatedFrom(e);
168: }
169:
170: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
171: {
172:     if (this.NavigationService.CanGoBack)
173:     {
174:         this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
175:     }
176: }
177:
178: private void txtSearch_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
179: {
180:     if (e.Key == Key.Enter)
181:     {
182:         if (txtSearch.Text.TrimStart().Length < 1)
183:         {
184:             MessageBox.Show(AppResources.SearchTrimFault);
185:         }
186:         else
187:         {
188:             lstSearch.Items.Clear();
189:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
190:             {
191:                 var albumList =
192:                     context.Albums.Where(j =>
j.AlbumInformation.ToLower().Contains(txtSearch.Text.ToLower())).ToList() as List<Album>;
193:                 //var noteList = context.Notes.ToList() as List<Note>;
194:
195:                 if (albumList != null)
196:                 {
197:                     txtSearchResult.Text = AppResources.SearchResults + " (" +
albumList.Count() + ")";
198:                 }
199:
200:                 //lstSearch.ItemsSource = noteList;
201:                 for (int i = 0; i < albumList.Count; i++)

```

```
202:                {
203:                    //if
204:                    (noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).IndexOf(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
205:                    != -1)
206:                    {
207:                        lstSearch.Items.Add(albumList[i] as Album);
208:                    }
209:                    //lstSearch.ItemTemplate.
210:                    //lstSearch.DisplayMemberPath = "NoteName" + " (" +
211:                    "CreationDate" + ")";
212:                    lstSearch.DisplayMemberPath = "AlbumNameRating";
213:                    MessageBox.Show(AppResources.SearchCompleted);
214:                }
215:            }
216:        }
217:        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
218:        {
219:            txtSearch.Focus();
220:        }
221:    }
222: }
```

Namespaces

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

1.2.31 StatisticsPage.xaml.cs

This is file StatisticsPage.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 AwesomeMusic.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeMusic
20: {
21:     public partial class StatisticsPage : PhoneApplicationPage
22:     {
23:         public StatisticsPage()
24:         {
25:             InitializeComponent();
26:             lblStatistics.Text = AppResources.Statistics;
27:             SetBackgroundColor();
28:             SetStatistic();
```

```

29:         }
30:
31:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
32:         {
33:             if (this.NavigationService.CanGoBack)
34:             {
35:                 this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
36:             }
37:         }
38:
39:         private void SetBackgroundColor()
40:         {
41:             AppSettings appSettings = new AppSettings();
42:             using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
43:             {
44:                 appSettings = context.AppSettings.First() as AppSettings;
45:             }
46:
47:             if (appSettings.AppBackgroundImage != null)
48:             {
49:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
50:                 BitmapImage image = new BitmapImage();
51:                 image.SetSource(stream);
52:                 ImageBrush ib = new ImageBrush();
53:                 ib.ImageSource = image;
54:                 this.LayoutRoot.Background = ib;
55:             }
56:             else
57:             {
58:                 switch (appSettings.AppBackgroundColor)
59:                 {
60:                     case "BLA":
61:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
62:                         break;
63:                     case "BLU":
64:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
65:                         break;
66:                     case "BRO":
67:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
68:                         break;
69:                     case "RED":
70:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
71:                         break;
72:                     case "GRE":
73:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
74:                         break;
75:                     case "GRA":
76:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
77:                         break;
78:                     case "YEL":
79:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
80:                         break;
81:                     case "ORA":
82:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
83:                         break;
84:                     case "PUR":
85:                         this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
86:                         break;
87:                     default:
88:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
89:                         break;
90:                 }
91:             }

```

```

92:         }
93:
94:         protected override void OnNavigatedTo(NavigationEventArgs e)
95:         {
96:             base.OnNavigatedTo(e);
97:         }
98:
99:         protected override void OnNavigatedFrom(NavigationEventArgs e)
100:        {
101:            base.OnNavigatedFrom(e);
102:            //while (NavigationService.CanGoBack)
103:            //NavigationService.RemoveBackEntry();
104:
105:        }
106:
107:        private void SetStatistic()
108:        {
109:            StringBuilder sb = new StringBuilder();
110:            string artistName, categoryName, bestAlbum = "", worstAlbum = "",
labelName;
111:            int albumCount;
112:
113:
114:            using (var context = new
AwesomeMusicDataContext(AwesomeMusicDataContext.ConnectionString))
115:            {
116:                var categories = context.Categories.OrderByDescending(j =>
j.CategoryAlbumCount).ToList() as List<Category>;
117:                categoryName = categories.First().CategoryNameCount;
118:
119:                var artists = context.Artists.OrderByDescending(j =>
j.ArtistAlbumCount).ToList() as List<Artist>;
120:                artistName = artists.First().ArtistNameCount;
121:
122:                albumCount = context.Albums.Count();
123:
124:                var albums3 = context.Albums.GroupBy(j => j.AlbumLabelName).Select(j
=> new { Name = j.Key, Total = j.Count() }).OrderByDescending(k => k.Total);
125:                labelName = albums3.First().Name + " (" + albums3.First().Total + ")";
126:
127:                var albums = context.Albums.OrderByDescending(j =>
j.AlbumRating).ToList() as List<Album>;
128:                int bestAlbumRating = albums.First().AlbumRating;
129:                int worstAlbumRating = albums.Last().AlbumRating;
130:
131:
132:                // en iyi ve en kötü puana sahip birden fazla albüm varsa hepsini
listelemeye yarayan
133:                // kod parçası buradadır.
134:                var bestAlbums = context.Albums.Where(j =>
j.AlbumRating.Equals(bestAlbumRating)).ToList() as List<Album>;
135:                if (bestAlbums.Count < 2)
136:                {
137:                    bestAlbum = albums.First().AlbumNameRating;
138:
139:                }
140:                else
141:                {
142:                    for (int i = 0; i < bestAlbums.Count; i++)
143:                    {
144:                        bestAlbum = bestAlbum + bestAlbums[i].AlbumNameRating + ", ";
145:                    }
146:                    bestAlbum = bestAlbum.Substring(0, bestAlbum.Length - 2);
147:                }
148:
149:                var worstAlbums = context.Albums.Where(j =>
j.AlbumRating.Equals(worstAlbumRating)).ToList() as List<Album>;
150:                if (worstAlbums.Count < 2)
151:                {

```

```
152:         worstAlbum = albums.First().AlbumNameRating;
153:
154:     }
155:     else
156:     {
157:         for (int i = 0; i < worstAlbums.Count; i++)
158:         {
159:             worstAlbum = worstAlbum + worstAlbums[i].AlbumNameRating + ",
160: ";
161:         }
162:         worstAlbum = worstAlbum.Substring(0, worstAlbum.Length - 2);
163:     }
164:
165:     sb.AppendLine(AppResources.TotalAlbumCount + ": " + albumCount);
166:     sb.AppendLine(AppResources.MostListenCategory + ": " + categoryName);
167:     sb.AppendLine(AppResources.MostListenArtist + ": " + artistName);
168:     sb.AppendLine(AppResources.MostListenLabel + ": " + labelName);
169:     sb.AppendLine(AppResources.BestAlbum + ": " + bestAlbum);
170:     sb.AppendLine(AppResources.WorstAlbum + ": " + worstAlbum);
171:
172:     txtStatistics.Text = sb.ToString();
173:     txtStatistics.IsReadOnly = true;
174: }
175: }
176: }
```

Namespaces

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

Index

A

AboutPage class 24

about AboutPage class 24

AboutPage 24

AboutPage.xaml.cs 89

AddCategoryPage class 25

about AddCategoryPage class 25

AddCategoryPage 25

artistId 26

OnFragmentNavigation 26

OnNavigatedFrom 26

OnNavigatedTo 27

AddCategoryPage.xaml.cs 92

Album class 27

about Album class 27

AlbumBestSong 28

AlbumCategoryId 28

AlbumComment 28

AlbumGuid 28

AlbumId 28

AlbumInformation 28

AlbumLabelName 29

AlbumName 29

AlbumNameRating 29

AlbumRating 29

AlbumReleaseYear 29

AlbumSongCount 29

CreationDate 30

ModificationDate 30

Album.cs 94

AlbumArtist class 30

about AlbumArtist class 30

AlbumArtistId 30

AlbumId 31

ArtistId 31

AlbumArtist.cs 96

AlbumPage class 31

about AlbumPage class 31

albumId 33

AlbumPage 32

artistId 33

artistName 33

categoryId 33

categoryName 33

flag 33

isFilled 34

OnFragmentNavigation 34

OnNavigatedFrom 35

OnNavigatedTo 35

pageName 34

ratingValue 34

AlbumPage.xaml.cs 96

App class 36

about App class 36

App 37

categoryNumber 38

IsTrial 38

RootFrame 38

App.xaml.cs 107

AppResources class 2

About 7

about AppResources class 2

AboutTheApp 7

AboutTheAppText 7

AboutTheAwesomeMusic 7

AddAlbum 7

AddArtist 8

AddCategory 8

AlbumComment 8

AlbumCount 8

AlbumDeleteSuccess 8

AlbumList 8

AlbumName 8

AlbumNameMustBe 8

AlbumOrderBy 8

AlbumOrderStyle 9

AlbumOrderStyleChangeSuccess 9

AlbumOrderTypeChangeSuccess 9

AlbumRating 9

AlbumSaveSuccess 9	ContactWithUs 13
AppResources 7	CreationDate 14
Arabic 9	Culture 14
ArtistAddSuccess 9	Date 14
ArtistAlreadySameCategory 9	DeleteAlbum 14
ArtistCategoryAddSuccess 9	DeleteAlbumQuestion 14
ArtistDeleteSuccess 10	DeleteArtist 14
ArtistExists 10	DeleteArtistQuestion 14
ArtistList 10	DeleteCategory 14
ArtistName 10	DeleteCategoryQuestion 15
ArtistNameChangeSuccess 10	Descending 15
ArtistOrderBy 10	English 15
ArtistOrderStyle 10	EnterArtistName 15
ArtistOrderStyleChangeSuccess 10	EnterCategoryName 15
ArtistOrderTypeChangeSuccess 10	ExitApp 15
ArtistSettings 11	ExitAppQuestion 15
Ascending 11	FinishDate 15
Background 11	Font 15
BackgroundColor 11	FontFamily 16
BackgroundColorChangeSuccess 11	FontFamilyChangeSuccess 16
BackgroundImage 11	FontSize 16
BackgroundImageChangeSuccess 11	FontSizeChangeSuccess 16
BackgroundImageRemoveSuccess 11	French 16
BestAlbum 11	GeneralSettings 16
BestSong 12	German 16
Black 12	Gray 16
Blue 12	Green 16
Brown 12	Italian 17
Cancel 12	Japanese 17
Categories 12	LabelName 17
CategoryAddSuccess 12	Language 17
CategoryDeleteSuccess 12	LanguageWarning 17
CategoryExists 12	ModificationDate 17
CategoryName 13	MostListenArtist 17
CategoryNameChangeSuccess 13	MostListenCategory 17
CategoryOrderBy 13	MostListenLabel 17
CategoryOrderStyle 13	Name 18
CategoryOrderStyleChangeSuccess 13	OK 18
CategoryOrderTypeChangeSuccess 13	OneDrive 18
CategorySettings 13	OneDriveSyncCompleted 18
Chinese 13	Orange 18

OtherSettings 18	Turkish 23
Persian 18	WorstAlbum 23
Portuguese 18	Yellow 23
Purple 18	AppResources.Designer.cs 114
Rate 19	AppSettings class 38
Red 19	about AppSettings class 38
ReleaseYear 19	AppBackgroundColor 39
RemoveBackgroundImage 19	AppBackgroundImage 39
ResetSettings 19	AppLangName 39
ResourceFlowDirection 19	AppSettingsId 39
ResourceLanguage 19	CategoryOrderBy 39
ResourceManager 19	CategoryOrderStyle 40
Russian 19	CurrentArtistNumber 40
Sanskrit 20	CurrentCategoryNumber 40
Save 20	FontFamily 40
Search 20	FontSize 40
SearchCompleted 20	AppSettings.cs 134
SearchResults 20	Artist class 40
SearchTrimFault 20	about Artist class 40
Select 20	AlbumOrderBy 41
SelectBackgroundColor 20	AlbumOrderStyle 41
Selected 20	ArtistAlbumCount 41
SelectFontFamily 21	ArtistId 42
SelectFontSize 21	ArtistName 42
SelectLanguage 21	ArtistNameCount 42
SendWithApp 21	CreationDate 42
SendWithMail 21	ModificationDate 42
SendWithSMS 21	Artist.cs 135
Settings 21	ArtistPage class 42
ShareAlbum 21	about ArtistPage class 42
SongCount 21	albumId 44
Spanish 22	artistId 44
StartDate 22	ArtistPage 43
Statistics 22	categoryId 44
SuccessfulResetSettings 22	oldArtistName 44
Sync 22	OnFragmentNavigation 45
Synchronizing 22	OnNavigatedFrom 47
SyncOnOneFile 22	OnNavigatedTo 47
SystemFault 22	popup 44
Thai 22	ArtistPage.xaml.cs 136
TotalAlbumCount 23	ArtistSettingsPage class 47

about ArtistSettingsPage class 47	AwesomeMusic.AlbumPage.AlbumPage 32
artistId 49	AwesomeMusic.AlbumPage.artistId 33
ArtistSettingsPage 48	AwesomeMusic.AlbumPage.artistName 33
categoryId 49	AwesomeMusic.AlbumPage.categoryId 33
OnFragmentNavigation 49	AwesomeMusic.AlbumPage.categoryName 33
OnNavigatedFrom 50	AwesomeMusic.AlbumPage.flag 33
OnNavigatedTo 50	AwesomeMusic.AlbumPage.isFilled 34
ArtistSettingsPage.xaml.cs 144	AwesomeMusic.AlbumPage.OnFragmentNavigation 34
AssemblyInfo.cs 147	AwesomeMusic.AlbumPage.OnNavigatedFrom 35
AwesomeMusic 1	AwesomeMusic.AlbumPage.OnNavigatedTo 35
AwesomeMusic namespace 1	AwesomeMusic.AlbumPage.pageName 34
Classes 23	AwesomeMusic.AlbumPage.ratingValue 34
AwesomeMusic.AboutPage 24	AwesomeMusic.App 36
AwesomeMusic.AboutPage.AboutPage 24	AwesomeMusic.App.App 37
AwesomeMusic.AddCategoryPage 25	AwesomeMusic.App.categoryNumber 38
AwesomeMusic.AddCategoryPage.AddCategoryPage 25	AwesomeMusic.App.IsTrial 38
AwesomeMusic.AddCategoryPage.artistId 26	AwesomeMusic.App.RootFrame 38
AwesomeMusic.AddCategoryPage.OnFragmentNavigation 26	AwesomeMusic.AppSettings 38
AwesomeMusic.AddCategoryPage.OnNavigatedFrom 26	AwesomeMusic.AppSettings.AppBackgroundColor 39
AwesomeMusic.AddCategoryPage.OnNavigatedTo 27	AwesomeMusic.AppSettings.AppBackgroundImage 39
AwesomeMusic.Album 27	AwesomeMusic.AppSettings.AppLangName 39
AwesomeMusic.Album.AlbumBestSong 28	AwesomeMusic.AppSettings.AppSettingsId 39
AwesomeMusic.Album.AlbumCategoryId 28	AwesomeMusic.AppSettings.CategoryOrderBy 39
AwesomeMusic.Album.AlbumComment 28	AwesomeMusic.AppSettings.CategoryOrderStyle 40
AwesomeMusic.Album.AlbumGuid 28	AwesomeMusic.AppSettings.CurrentArtistNumber 40
AwesomeMusic.Album.AlbumId 28	AwesomeMusic.AppSettings.CurrentCategoryNumber 40
AwesomeMusic.Album.AlbumInformation 28	AwesomeMusic.AppSettings.FontFamily 40
AwesomeMusic.Album.AlbumLabelName 29	AwesomeMusic.AppSettings.FontSize 40
AwesomeMusic.Album.AlbumName 29	AwesomeMusic.Artist 40
AwesomeMusic.Album.AlbumNameRating 29	AwesomeMusic.Artist.AlbumOrderBy 41
AwesomeMusic.Album.AlbumRating 29	AwesomeMusic.Artist.AlbumOrderStyle 41
AwesomeMusic.Album.AlbumReleaseYear 29	AwesomeMusic.Artist.ArtistAlbumCount 41
AwesomeMusic.Album.AlbumSongCount 29	AwesomeMusic.Artist.ArtistId 42
AwesomeMusic.Album.CreationDate 30	AwesomeMusic.Artist.ArtistName 42
AwesomeMusic.Album.ModificationDate 30	AwesomeMusic.Artist.ArtistNameCount 42
AwesomeMusic.AlbumArtist 30	AwesomeMusic.Artist.CreationDate 42
AwesomeMusic.AlbumArtist.AlbumArtistId 30	AwesomeMusic.Artist.ModificationDate 42
AwesomeMusic.AlbumArtist.AlbumId 31	AwesomeMusic.ArtistPage 42
AwesomeMusic.AlbumArtist.ArtistId 31	AwesomeMusic.ArtistPage.albumId 44
AwesomeMusic.AlbumPage 31	AwesomeMusic.ArtistPage.artistId 44
AwesomeMusic.AlbumPage.albumId 33	AwesomeMusic.ArtistPage.ArtistPage 43

AwesomeMusic.ArtistPage.categoryId 44	AwesomeMusic.Category.CategoryId 56
AwesomeMusic.ArtistPage.oldArtistName 44	AwesomeMusic.Category.CategoryName 56
AwesomeMusic.ArtistPage.OnFragmentNavigation 45	AwesomeMusic.Category.CategoryNameCount 56
AwesomeMusic.ArtistPage.OnNavigatedFrom 47	AwesomeMusic.Category.CreationDate 56
AwesomeMusic.ArtistPage.OnNavigatedTo 47	AwesomeMusic.Category.ModificationDate 57
AwesomeMusic.ArtistPage.popup 44	AwesomeMusic.CategoryArtist 57
AwesomeMusic.ArtistSettingsPage 47	AwesomeMusic.CategoryArtist.ArtistId 57
AwesomeMusic.ArtistSettingsPage.artistId 49	AwesomeMusic.CategoryArtist.CategoryArtistId 57
AwesomeMusic.ArtistSettingsPage.ArtistSettingsPage 48	AwesomeMusic.CategoryArtist.CategoryId 58
AwesomeMusic.ArtistSettingsPage.categoryId 49	AwesomeMusic.CategoryPage 58
AwesomeMusic.ArtistSettingsPage.OnFragmentNavigation 49	AwesomeMusic.CategoryPage.categoryId 59
AwesomeMusic.ArtistSettingsPage.OnNavigatedFrom 50	AwesomeMusic.CategoryPage.CategoryPage 58
AwesomeMusic.ArtistSettingsPage.OnNavigatedTo 50	AwesomeMusic.CategoryPage.oldCategoryName 59
AwesomeMusic.AwesomeMusicDataContext 51	AwesomeMusic.CategoryPage.OnFragmentNavigation 60
AwesomeMusic.AwesomeMusicDataContext.AlbumArtists 51	AwesomeMusic.CategoryPage.OnNavigatedFrom 61
AwesomeMusic.AwesomeMusicDataContext.Albums 52	AwesomeMusic.CategoryPage.OnNavigatedTo 61
AwesomeMusic.AwesomeMusicDataContext.AppSettings 52	AwesomeMusic.CategoryPage.popup 59
AwesomeMusic.AwesomeMusicDataContext.Artists 52	AwesomeMusic.CategorySettingsPage 62
AwesomeMusic.AwesomeMusicDataContext.AwesomeMusic DataContext 51	AwesomeMusic.CategorySettingsPage.categoryId 63
AwesomeMusic.AwesomeMusicDataContext.Categories 52	AwesomeMusic.CategorySettingsPage.CategorySettingsPage 62
AwesomeMusic.AwesomeMusicDataContext.CategoryArtists 52	AwesomeMusic.CategorySettingsPage.OnFragmentNavigatio n 63
AwesomeMusic.AwesomeMusicDataContext.ConnectionStrin g 52	AwesomeMusic.CategorySettingsPage.OnNavigatedFrom 64
AwesomeMusic.BackgroundColorSettingsPage 52	AwesomeMusic.CategorySettingsPage.OnNavigatedTo 64
AwesomeMusic.BackgroundColorSettingsPage.artistId 54	AwesomeMusic.csproj 148
AwesomeMusic.BackgroundColorSettingsPage.BackgroundColor SettingsPage 53	AwesomeMusic.FontFamilySettingsPage 64
AwesomeMusic.BackgroundColorSettingsPage.OnFragmentN avigation 54	AwesomeMusic.FontFamilySettingsPage.artistId 66
AwesomeMusic.BackgroundColorSettingsPage.OnNavigatedF rom 54	AwesomeMusic.FontFamilySettingsPage.FontFamilySettingsP age 65
AwesomeMusic.BackgroundColorSettingsPage.OnNavigatedT o 55	AwesomeMusic.FontFamilySettingsPage.OnFragmentNavigati on 66
AwesomeMusic.Category 55	AwesomeMusic.FontFamilySettingsPage.OnNavigatedFrom 66
AwesomeMusic.Category.ArtistOrderBy 55	AwesomeMusic.FontFamilySettingsPage.OnNavigatedTo 67
AwesomeMusic.Category.ArtistOrderStyle 56	AwesomeMusic.FontSizeSettingsPage 67
AwesomeMusic.Category.CategoryAlbumCount 56	AwesomeMusic.FontSizeSettingsPage.artistId 68
	AwesomeMusic.FontSizeSettingsPage.FontSizeSettingsPage 67
	AwesomeMusic.FontSizeSettingsPage.OnFragmentNavigatio n

68	age 83
AwesomeMusic.FontSizeSettingsPage.OnNavigatedFrom 69	AwesomeMusic.OrderStyleSettingsPage.pageName 83
AwesomeMusic.FontSizeSettingsPage.OnNavigatedTo 69	AwesomeMusic.PopupAddChange 85
AwesomeMusic.GeneralSettingsPage 69	AwesomeMusic.PopupAddChange.PopupAddChange 85
AwesomeMusic.GeneralSettingsPage.CreateDirectoryAsync 73	AwesomeMusic.Resources 2
AwesomeMusic.GeneralSettingsPage.DesignFileName 74	AwesomeMusic.Resources namespace 2
AwesomeMusic.GeneralSettingsPage.GeneralSettingsPage 70	Classes 2
AwesomeMusic.GeneralSettingsPage.OnNavigatedFrom 74	AwesomeMusic.Resources.AppResources 2
AwesomeMusic.GeneralSettingsPage.OnNavigatedTo 74	AwesomeMusic.Resources.AppResources.About 7
AwesomeMusic.GeneralSettingsPage.signIn 73	AwesomeMusic.Resources.AppResources.AboutTheApp 7
AwesomeMusic.LanguageSettingsPage 75	AwesomeMusic.Resources.AppResources.AboutTheAppText 7
AwesomeMusic.LanguageSettingsPage.LanguageSettingsPage 75	AwesomeMusic.Resources.AppResources.AboutTheAwesomeMusic 7
AwesomeMusic.LanguageSettingsPage.OnNavigatedFrom 76	AwesomeMusic.Resources.AppResources.AddAlbum 7
AwesomeMusic.LanguageSettingsPage.OnNavigatedTo 76	AwesomeMusic.Resources.AppResources.AddArtist 8
AwesomeMusic.LocalizedStrings 76	AwesomeMusic.Resources.AppResources.AddCategory 8
AwesomeMusic.LocalizedStrings.LocalizedResources 77	AwesomeMusic.Resources.AppResources.AlbumComment 8
AwesomeMusic.MainPage 77	AwesomeMusic.Resources.AppResources.AlbumCount 8
AwesomeMusic.MainPage.MainPage 77	AwesomeMusic.Resources.AppResources.AlbumDeleteSuccess 8
AwesomeMusic.MainPage.OnNavigatedFrom 78	AwesomeMusic.Resources.AppResources.AlbumList 8
AwesomeMusic.MainPage.OnNavigatedTo 79	AwesomeMusic.Resources.AppResources.AlbumName 8
AwesomeMusic.MainPage.popup 78	AwesomeMusic.Resources.AppResources.AlbumNameMustBe 8
AwesomeMusic.OrderSettingsPage 79	AwesomeMusic.Resources.AppResources.AlbumOrderBy 8
AwesomeMusic.OrderSettingsPage.artistId 80	AwesomeMusic.Resources.AppResources.AlbumOrderStyle 9
AwesomeMusic.OrderSettingsPage.categoryId 40	AwesomeMusic.Resources.AppResources.AlbumOrderStyleChangeSuccess 9
AwesomeMusic.OrderSettingsPage.OnFragmentNavigation 80	AwesomeMusic.Resources.AppResources.AlbumOrderTypeChangeSuccess 9
AwesomeMusic.OrderSettingsPage.OnNavigatedFrom 81	AwesomeMusic.Resources.AppResources.AlbumRating 9
AwesomeMusic.OrderSettingsPage.OnNavigatedTo 81	AwesomeMusic.Resources.AppResources.AlbumSaveSuccess 9
AwesomeMusic.OrderSettingsPage.OrderSettingsPage 80	AwesomeMusic.Resources.AppResources.AppResources 7
AwesomeMusic.OrderSettingsPage.pageName 80	AwesomeMusic.Resources.AppResources.Arabic 9
AwesomeMusic.OrderStyleSettingsPage 82	AwesomeMusic.Resources.AppResources.ArtistAddSuccess 9
AwesomeMusic.OrderStyleSettingsPage.artistId 83	AwesomeMusic.Resources.AppResources.ArtistAlreadySame
AwesomeMusic.OrderStyleSettingsPage.categoryId 83	
AwesomeMusic.OrderStyleSettingsPage.OnFragmentNavigation 84	
AwesomeMusic.OrderStyleSettingsPage.OnNavigatedFrom 84	
AwesomeMusic.OrderStyleSettingsPage.OnNavigatedTo 84	
AwesomeMusic.OrderStyleSettingsPage.OrderStyleSettingsPage	

Category 9	AwesomeMusic.Resources.AppResources.CategoryDeleteSuccess 12
AwesomeMusic.Resources.AppResources.ArtistCategoryAddSuccess 9	AwesomeMusic.Resources.AppResources.CategoryExists 12
AwesomeMusic.Resources.AppResources.ArtistDeleteSuccess 10	AwesomeMusic.Resources.AppResources.CategoryName 13
AwesomeMusic.Resources.AppResources.ArtistExists 10	AwesomeMusic.Resources.AppResources.CategoryNameChangeSuccess 13
AwesomeMusic.Resources.AppResources.ArtistList 10	AwesomeMusic.Resources.AppResources.CategoryOrderBy 13
AwesomeMusic.Resources.AppResources.ArtistName 10	AwesomeMusic.Resources.AppResources.CategoryOrderStyle 13
AwesomeMusic.Resources.AppResources.ArtistNameChangeSuccess 10	AwesomeMusic.Resources.AppResources.CategoryOrderStyleChangeSuccess 13
AwesomeMusic.Resources.AppResources.ArtistOrderBy 10	AwesomeMusic.Resources.AppResources.CategoryOrderType 13
AwesomeMusic.Resources.AppResources.ArtistOrderStyle 10	AwesomeMusic.Resources.AppResources.CategoryOrderTypeChangeSuccess 13
AwesomeMusic.Resources.AppResources.ArtistOrderStyleChangeSuccess 10	AwesomeMusic.Resources.AppResources.CategorySettings 13
AwesomeMusic.Resources.AppResources.ArtistOrderTypeChangeSuccess 10	AwesomeMusic.Resources.AppResources.Chinese 13
AwesomeMusic.Resources.AppResources.ArtistSettings 11	AwesomeMusic.Resources.AppResources.ContactWithUs 13
AwesomeMusic.Resources.AppResources.Ascending 11	AwesomeMusic.Resources.AppResources.CreationDate 14
AwesomeMusic.Resources.AppResources.Background 11	AwesomeMusic.Resources.AppResources.Culture 14
AwesomeMusic.Resources.AppResources.BackgroundColor 11	AwesomeMusic.Resources.AppResources.Date 14
AwesomeMusic.Resources.AppResources.BackgroundColorChangeSuccess 11	AwesomeMusic.Resources.AppResources.DeleteAlbum 14
AwesomeMusic.Resources.AppResources.BackgroundImage 11	AwesomeMusic.Resources.AppResources.DeleteAlbumQuestion 14
AwesomeMusic.Resources.AppResources.BackgroundImageChangeSuccess 11	AwesomeMusic.Resources.AppResources.DeleteArtist 14
AwesomeMusic.Resources.AppResources.BackgroundImageRemoveSuccess 11	AwesomeMusic.Resources.AppResources.DeleteArtistQuestion 14
AwesomeMusic.Resources.AppResources.BestAlbum 11	AwesomeMusic.Resources.AppResources.DeleteCategory 14
AwesomeMusic.Resources.AppResources.BestSong 12	AwesomeMusic.Resources.AppResources.DeleteCategoryQuestion 15
AwesomeMusic.Resources.AppResources.Black 12	AwesomeMusic.Resources.AppResources.Descending 15
AwesomeMusic.Resources.AppResources.Blue 12	AwesomeMusic.Resources.AppResources.English 15
AwesomeMusic.Resources.AppResources.Brown 12	AwesomeMusic.Resources.AppResources.EnterArtistName 15
AwesomeMusic.Resources.AppResources.Cancel 12	AwesomeMusic.Resources.AppResources.EnterCategoryName 15
AwesomeMusic.Resources.AppResources.Categories 12	AwesomeMusic.Resources.AppResources.ExitApp 15
AwesomeMusic.Resources.AppResources.CategoryAddSuccess 12	AwesomeMusic.Resources.AppResources.ExitAppQuestion 15

AwesomeMusic.Resources.AppResources.FinishDate 15	dImage 19
AwesomeMusic.Resources.AppResources.Font 15	AwesomeMusic.Resources.AppResources.ResetSettings 19
AwesomeMusic.Resources.AppResources.FontFamily 16	AwesomeMusic.Resources.AppResources.ResourceFlowDirection 19
AwesomeMusic.Resources.AppResources.FontFamilyChangeSuccess 16	AwesomeMusic.Resources.AppResources.ResourceLanguage 19
AwesomeMusic.Resources.AppResources.FontSize 16	AwesomeMusic.Resources.AppResources.ResourceManager 19
AwesomeMusic.Resources.AppResources.FontSizeChangeSuccess 16	AwesomeMusic.Resources.AppResources.Russian 19
AwesomeMusic.Resources.AppResources.French 16	AwesomeMusic.Resources.AppResources.Sanskrit 20
AwesomeMusic.Resources.AppResources.GeneralSettings 16	AwesomeMusic.Resources.AppResources.Save 20
AwesomeMusic.Resources.AppResources.German 16	AwesomeMusic.Resources.AppResources.Search 20
AwesomeMusic.Resources.AppResources.Gray 16	AwesomeMusic.Resources.AppResources.SearchCompleted 20
AwesomeMusic.Resources.AppResources.Green 16	AwesomeMusic.Resources.AppResources.SearchResults 20
AwesomeMusic.Resources.AppResources.Italian 17	AwesomeMusic.Resources.AppResources.SearchTrimFault 20
AwesomeMusic.Resources.AppResources.Japanese 17	AwesomeMusic.Resources.AppResources.Select 20
AwesomeMusic.Resources.AppResources.LabelName 17	AwesomeMusic.Resources.AppResources.SelectBackgroundColor 20
AwesomeMusic.Resources.AppResources.Language 17	AwesomeMusic.Resources.AppResources.Selected 20
AwesomeMusic.Resources.AppResources.LanguageWarning 17	AwesomeMusic.Resources.AppResources.SelectFontFamily 21
AwesomeMusic.Resources.AppResources.ModificationDate 17	AwesomeMusic.Resources.AppResources.SelectFontSize 21
AwesomeMusic.Resources.AppResources.MostListenArtist 17	AwesomeMusic.Resources.AppResources.SelectLanguage 21
AwesomeMusic.Resources.AppResources.MostListenCategory 17	AwesomeMusic.Resources.AppResources.SendWithApp 21
AwesomeMusic.Resources.AppResources.MostListenLabel 17	AwesomeMusic.Resources.AppResources.SendWithMail 21
AwesomeMusic.Resources.AppResources.Name 18	AwesomeMusic.Resources.AppResources.SendWithSMS 21
AwesomeMusic.Resources.AppResources.OK 18	AwesomeMusic.Resources.AppResources.Settings 21
AwesomeMusic.Resources.AppResources.OneDrive 18	AwesomeMusic.Resources.AppResources.ShareAlbum 21
AwesomeMusic.Resources.AppResources.OneDriveSyncCompleted 18	AwesomeMusic.Resources.AppResources.SongCount 21
AwesomeMusic.Resources.AppResources.Orange 18	AwesomeMusic.Resources.AppResources.Spanish 22
AwesomeMusic.Resources.AppResources.OtherSettings 18	AwesomeMusic.Resources.AppResources.StartDate 22
AwesomeMusic.Resources.AppResources.Persian 18	AwesomeMusic.Resources.AppResources.Statistics 22
AwesomeMusic.Resources.AppResources.Portuguese 18	AwesomeMusic.Resources.AppResources.SuccessfulResetSettings 22
AwesomeMusic.Resources.AppResources.Purple 18	AwesomeMusic.Resources.AppResources.Sync 22
AwesomeMusic.Resources.AppResources.Rate 19	AwesomeMusic.Resources.AppResources.Synchronizing 22
AwesomeMusic.Resources.AppResources.Red 19	AwesomeMusic.Resources.AppResources.SyncOnOneFile 22
AwesomeMusic.Resources.AppResources.ReleaseYear 19	AwesomeMusic.Resources.AppResources.SystemFault 22
AwesomeMusic.Resources.AppResources.RemoveBackground	

AwesomeMusic.Resources.AppResources.Thai 22
 AwesomeMusic.Resources.AppResources.TotalAlbumCount 23
 AwesomeMusic.Resources.AppResources.Turkish 23
 AwesomeMusic.Resources.AppResources.WorstAlbum 23
 AwesomeMusic.Resources.AppResources.Yellow 23
 AwesomeMusic.SearchPage 86
 AwesomeMusic.SearchPage.OnNavigatedFrom 86
 AwesomeMusic.SearchPage.OnNavigatedTo 87
 AwesomeMusic.SearchPage.SearchPage 86
 AwesomeMusic.sln 148
 AwesomeMusic.StatisticsPage 87
 AwesomeMusic.StatisticsPage.OnNavigatedFrom 88
 AwesomeMusic.StatisticsPage.OnNavigatedTo 88
 AwesomeMusic.StatisticsPage.StatisticsPage 87
 AwesomeMusicDataContext class 51
 about AwesomeMusicDataContext class 51
 AlbumArtists 51
 Albums 52
 AppSettings 52
 Artists 52
 AwesomeMusicDataContext 51
 Categories 52
 CategoryArtists 52
 ConnectionString 52
 AwesomeMusicDataContext.cs 148

B

BackgroundColorSettingsPage class 52
 about BackgroundColorSettingsPage class 52
 artistId 54
 BackgroundColorSettingsPage 53
 OnFragmentNavigation 54
 OnNavigatedFrom 54
 OnNavigatedTo 55
 BackgroundColorSettingsPage.xaml.cs 149

C

Category class 55
 about Category class 55
 ArtistOrderBy 55

ArtistOrderStyle 56
 CategoryAlbumCount 56
 CategoryId 56
 CategoryName 56
 CategoryNameCount 56
 CreationDate 56
 ModificationDate 57
 Category.cs 152
 CategoryArtist class 57
 about CategoryArtist class 57
 ArtistId 57
 CategoryArtistId 57
 CategoryId 58
 CategoryArtist.cs 153
 CategoryPage class 58
 about CategoryPage class 58
 categoryId 59
 CategoryPage 58
 oldCategoryName 59
 OnFragmentNavigation 60
 OnNavigatedFrom 61
 OnNavigatedTo 61
 popup 59
 CategoryPage.xaml.cs 154
 CategorySettingsPage class 62
 about CategorySettingsPage class 62
 categoryId 63
 CategorySettingsPage 62
 OnFragmentNavigation 63
 OnNavigatedFrom 64
 OnNavigatedTo 64
 CategorySettingsPage.xaml.cs 162

F

Files 88
 FontFamilySettingsPage class 64
 about FontFamilySettingsPage class 64
 artistId 66
 FontFamilySettingsPage 65
 OnFragmentNavigation 66

OnNavigatedFrom 66
 OnNavigatedTo 67
 FontFamilySettingsPage.xaml.cs 165
 FontSizeSettingsPage class 67
 about FontSizeSettingsPage class 67
 artistId 68
 FontSizeSettingsPage 67
 OnFragmentNavigation 68
 OnNavigatedFrom 69
 OnNavigatedTo 69
 FontSizeSettingsPage.xaml.cs 168

G

GeneralSettingsPage class 69
 about GeneralSettingsPage class 69
 CreateDirectoryAsync 73
 DesignFileName 74
 GeneralSettingsPage 70
 OnNavigatedFrom 74
 OnNavigatedTo 74
 signIn 73
 GeneralSettingsPage.xaml.cs 170

L

LanguageSettingsPage class 75
 about LanguageSettingsPage class 75
 LanguageSettingsPage 75
 OnNavigatedFrom 76
 OnNavigatedTo 76
 LanguageSettingsPage.xaml.cs 183
 LocalizedStrings class 76
 about LocalizedStrings class 76
 LocalizedResources 77
 LocalizedStrings.cs 187

M

MainPage class 77
 about MainPage class 77
 MainPage 77
 OnNavigatedFrom 78

OnNavigatedTo 79
 popup 78
 MainPage.xaml.cs 187

O

OrderSettingsPage class 79
 about OrderSettingsPage class 79
 artistId 80
 categoryId 80
 OnFragmentNavigation 80
 OnNavigatedFrom 81
 OnNavigatedTo 81
 OrderSettingsPage 80
 pageName 80
 OrderSettingsPage.xaml.cs 193
 OrderStyleSettingsPage class 82
 about OrderStyleSettingsPage class 82
 artistId 83
 categoryId 83
 OnFragmentNavigation 84
 OnNavigatedFrom 84
 OnNavigatedTo 84
 OrderStyleSettingsPage 83
 pageName 83
 OrderStyleSettingsPage.xaml.cs 198

P

PopupAddChange class 85
 about PopupAddChange class 85
 PopupAddChange 85
 PopupAddChange.xaml.cs 202

S

SearchPage class 86
 about SearchPage class 86
 OnNavigatedFrom 86
 OnNavigatedTo 87
 SearchPage 86
 SearchPage.xaml.cs 203
 StatisticsPage class 87

about StatisticsPage class 87

OnNavigatedFrom 88

OnNavigatedTo 88

StatisticsPage 87

StatisticsPage.xaml.cs 207