

Awesome Library

Okudugunuz kitaplari GoodReads, 1000Kitap gibi puanlayip incelemelerinizi kaydedebileceginiz bir Windows Phone uygulaması

Table of Contents

Symbol Reference	1
AwesomeLibrary Namespace	1
AwesomeLibrary.Resources Namespace	2
Classes	2
AppResources Class	2
Classes	23
AboutPage Class	24
AboutPage>AboutPage Constructor	24
AddCategoryPage Class	25
AddCategoryPage.AddCategoryPage Constructor	26
AddCategoryPage Fields	26
AddCategoryPage Methods	26
App Class	27
App.App Constructor	28
App Fields	28
App Properties	29
AppSettings Class	29
AppSettings Properties	30
Author Class	31
Author Properties	32
AuthorPage Class	33
AuthorPage.AuthorPage Constructor	34
AuthorPage Fields	35
AuthorPage Methods	35
AuthorSettingsPage Class	38
AuthorSettingsPage.AuthorSettingsPage Constructor	39
AuthorSettingsPage Fields	39
AuthorSettingsPage Methods	40
AwesomeLibraryDataContext Class	41
AwesomeLibraryDataContext.AwesomeLibraryDataContext Constructor	42
AwesomeLibraryDataContext Fields	42
BackgroundColorSettingsPage Class	43
BackgroundColorSettingsPage.BackgroundColorSettingsPage Constructor	44
BackgroundColorSettingsPage Fields	44
BackgroundColorSettingsPage Methods	45
Book Class	46

Book Properties	46
BookAuthor Class	49
BookAuthor Properties	49
BookPage Class	50
BookPage.BookPage Constructor	51
BookPage Fields	52
BookPage Methods	53
Category Class	55
Category Properties	56
CategoryAuthor Class	57
CategoryAuthor Properties	57
CategoryPage Class	58
CategoryPage.CategoryPage Constructor	59
CategoryPage Fields	59
CategoryPage Methods	60
CategorySettingsPage Class	62
CategorySettingsPage.CategorySettingsPage Constructor	63
CategorySettingsPage Fields	63
CategorySettingsPage Methods	63
FontFamilySettingsPage Class	65
FontFamilySettingsPage.FontFamilySettingsPage Constructor	65
FontFamilySettingsPage Fields	66
FontFamilySettingsPage Methods	66
FontSizeSettingsPage Class	67
FontSizeSettingsPage.FontSizeSettingsPage Constructor	68
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
LocalizedString Class	76
LocalizedString Properties	77
MainPage Class	77
MainPage.MainPage Constructor	78
MainPage Fields	78
MainPage Methods	79

OrderSettingsPage Class	79
OrderSettingsPage.OrderSettingsPage Constructor	80
OrderSettingsPage Fields	80
OrderSettingsPage Methods	81
OrderStyleSettingsPage Class	82
OrderStyleSettingsPage.OrderStyleSettingsPage Constructor	83
OrderStyleSettingsPage Fields	83
OrderStyleSettingsPage Methods	84
PopupAddChange Class	85
PopupAddChange.PopupAddChange Constructor	86
SearchPage Class	86
SearchPage.SearchPage Constructor	86
SearchPage Methods	87
StatisticsPage Class	87
StatisticsPage.StatisticsPage Constructor	88
StatisticsPage Methods	88
Files	89
AboutPage.xaml.cs	90
AddCategoryPage.xaml.cs	92
App.xaml.cs	95
AppResources.Designer.cs	101
AppSettings.cs	121
AssemblyInfo.cs	122
Author.cs	123
AuthorPage.xaml.cs	124
AuthorSettingsPage.xaml.cs	131
AwesomeLibrary.csproj	135
AwesomeLibrary.sln	135
AwesomeLibraryDataContext.cs	135
BackgroundColorSettingsPage.xaml.cs	136
Book.cs	139
BookAuthor.cs	140
BookPage.xaml.cs	141
Category.cs	151
CategoryAuthor.cs	152
CategoryPage.xaml.cs	153
CategorySettingsPage.xaml.cs	161
FontFamilySettingsPage.xaml.cs	164
FontSizeSettingsPage.xaml.cs	167
GeneralSettingsPage.xaml.cs	169

LanguageSettingsPage.xaml.cs	182
LocalizedStrings.cs	186
MainPage.xaml.cs	186
OrderSettingsPage.xaml.cs	192
OrderStyleSettingsPage.xaml.cs	197
PopupAddChange.xaml.cs	201
SearchPage.xaml.cs	202
StatisticsPage.xaml.cs	207

Index

a

1 Symbol Reference

1.1 AwesomeLibrary Namespace





This is namespace AwesomeLibrary.

Namespaces

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

Classes


	Name	Description
	AboutPage (see page 24)	This is class AwesomeLibrary>AboutPage.
	AddCategoryPage (see page 25)	This is class AwesomeLibrary.AddCategoryPage.
	App (see page 27)	This is class AwesomeLibrary.App.
	AppSettings (see page 29)	This is class AwesomeLibrary.AppSettings.
	Author (see page 31)	This is class AwesomeLibrary.Author.
	AuthorPage (see page 33)	This is class AwesomeLibrary.AuthorPage.
	AuthorSettingsPage (see page 38)	This is class AwesomeLibrary.AuthorSettingsPage.
	AwesomeLibraryDataContext (see page 41)	This is class AwesomeLibrary.AwesomeLibraryDataContext.
	BackgroundColorSettingsPage (see page 43)	This is class AwesomeLibrary.BackgroundColorSettingsPage.
	Book (see page 46)	This is class AwesomeLibrary.Book.
	BookAuthor (see page 49)	This is class AwesomeLibrary.BookAuthor.
	BookPage (see page 50)	This is class AwesomeLibrary.BookPage.
	Category (see page 55)	This is class AwesomeLibrary.Category.
	CategoryAuthor (see page 57)	This is class AwesomeLibrary.CategoryAuthor.
	CategoryPage (see page 58)	This is class AwesomeLibrary.CategoryPage.
	CategorySettingsPage (see page 62)	This is class AwesomeLibrary.CategorySettingsPage.
	FontFamilySettingsPage (see page 65)	This is class AwesomeLibrary.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 67)	This is class AwesomeLibrary.FontSizeSettingsPage.
	GeneralSettingsPage (see page 69)	This is class AwesomeLibrary.GeneralSettingsPage.
	LanguageSettingsPage (see page 75)	This is class AwesomeLibrary.LanguageSettingsPage.
	LocalizedStrings (see page 76)	Provides access to string resources.
	MainPage (see page 77)	This is class AwesomeLibrary.MainPage.
	OrderSettingsPage (see page 79)	This is class AwesomeLibrary.OrderSettingsPage.

	OrderStyleSettingsPage (see page 82)	This is class AwesomeLibrary.OrderStyleSettingsPage.
	PopupAddChange (see page 85)	This is class AwesomeLibrary.PopupAddChange.
	SearchPage (see page 86)	This is class AwesomeLibrary.SearchPage.
	StatisticsPage (see page 87)	This is class AwesomeLibrary.StatisticsPage.

1.1.1 AwesomeLibrary.Resources Namespace

This is namespace AwesomeLibrary.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

AwesomeLibrary.Resources.AppResources

C#

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


File

AppResources.Designer.cs (see page 101)


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 27).
	AboutTheAppText (see page 7)	Looks up a localized string similar to Hi. I like reading book a lot. And after Awesome Note app, i decide to create an app which has similar properties like GoodReads. You can add categories, add authors and add books on it. You can send information of your books 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..
	AboutTheAwesomeLibrary (see page 7)	Looks up a localized string similar to About (see page 7) The Awesome Library.
	AddAuthor (see page 7)	Looks up a localized string similar to Add Author (see page 31).
	AddBook (see page 8)	Looks up a localized string similar to Add Book (see page 46).
	AddCategory (see page 8)	Looks up a localized string similar to Add Category (see page 55).
	Arabic (see page 8)	Looks up a localized string similar to Arabic.
	Ascending (see page 8)	Looks up a localized string similar to Ascending.
	AuthorAddSuccess (see page 8)	Looks up a localized string similar to Author (see page 31) Has Been Added Successfully.
	AuthorAlreadySameCategory (see page 8)	Looks up a localized string similar to Author (see page 31) Has This Category (see page 55) Already.
	AuthorCategoryAddSuccess (see page 8)	Looks up a localized string similar to Category (see page 55) Has Been Added To Author (see page 31) Successfully.
	AuthorDeleteSuccess (see page 8)	Looks up a localized string similar to Author (see page 31) Has Been Removed Successfully.
	AuthorExists (see page 8)	Looks up a localized string similar to This Author (see page 31) Has Already Exists.
	AuthorList (see page 9)	Looks up a localized string similar to Author (see page 31) List.
	AuthorName (see page 9)	Looks up a localized string similar to Author (see page 31) Name (see page 17).
	AuthorNameChangeSuccess (see page 9)	Looks up a localized string similar to Author (see page 31) Name (see page 17) Has Been Changed Successfully.
	AuthorOrderBy (see page 9)	Looks up a localized string similar to Author (see page 31) Order By.
	AuthorOrderStyle (see page 9)	Looks up a localized string similar to Author (see page 31) Order Style.
	AuthorOrderStyleChangeSuccess (see page 9)	Looks up a localized string similar to Author (see page 31) Order Style Has Been Changed Successfully.
	AuthorOrderTypeChangeSuccess (see page 9)	Looks up a localized string similar to Author (see page 31) Order Type Has Been Changed Successfully.
	AuthorSettings (see page 9)	Looks up a localized string similar to Author (see page 31) Settings (see page 21).
	Background (see page 9)	Looks up a localized string similar to Background.
	BackgroundColor (see page 10)	Looks up a localized string similar to Background (see page 9) Color.
	BackgroundColorChangeSuccess (see page 10)	Looks up a localized string similar to Background (see page 9) Color Has Been Changed Successfully.
	BackgroundImage (see page 10)	Looks up a localized string similar to Background (see page 9) Image.
	BackgroundImageChangeSuccess (see page 10)	Looks up a localized string similar to Background (see page 9) Image Has Been Changed Successfully.
	BackgroundImageRemoveSuccess (see page 10)	Looks up a localized string similar to Background (see page 9) Image Has Been Removed Successfully.
	BestBook (see page 10)	Looks up a localized string similar to Best Book (see page 46).
	Black (see page 10)	Looks up a localized string similar to Black.
	Blue (see page 10)	Looks up a localized string similar to Blue.

	BookComment (see page 10)	Looks up a localized string similar to Book (see page 46) Comment.
	BookCount (see page 11)	Looks up a localized string similar to Book (see page 46) Count.
	BookDeleteSuccess (see page 11)	Looks up a localized string similar to Book (see page 46) Has Been Removed Successfully.
	BookList (see page 11)	Looks up a localized string similar to Book (see page 46) List.
	BookName (see page 11)	Looks up a localized string similar to Book (see page 46) Name (see page 17).
	BookNameMustBe (see page 11)	Looks up a localized string similar to You Have To Enter Book (see page 46) Name (see page 17) At Least.
	BookOrderBy (see page 11)	Looks up a localized string similar to Book (see page 46) Order By.
	BookOrderStyle (see page 11)	Looks up a localized string similar to Book (see page 46) Order Style.
	BookOrderStyleChangeSuccess (see page 11)	Looks up a localized string similar to Book (see page 46) Order Style Has Been Changed Successfully.
	BookOrderTypeChangeSuccess (see page 11)	Looks up a localized string similar to Book (see page 46) Order Type Has Been Changed Successfully.
	BookPerDay (see page 12)	Looks up a localized string similar to Average Book (see page 46) Read Per Day.
	BookRating (see page 12)	Looks up a localized string similar to Book (see page 46) Rating.
	BookSaveSuccess (see page 12)	Looks up a localized string similar to Book (see page 46) Has Been Saved Successfully.
	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 17).
	CategoryNameChangeSuccess (see page 13)	Looks up a localized string similar to Category (see page 55) Name (see page 17) 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.
	DeleteAuthor (see page 14)	Looks up a localized string similar to Delete Author (see page 31).
	DeleteAuthorQuestion (see page 14)	Looks up a localized string similar to You Will Delete The Author (see page 31) With All Contents. Are You Sure?.

	DeleteBook (see page 14)	Looks up a localized string similar to Delete Book (see page 46).
	DeleteBookQuestion (see page 14)	Looks up a localized string similar to You Will Delete This Book (see page 46). 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.
	EnterAuthorName (see page 15)	Looks up a localized string similar to Enter Author (see page 31) Name (see page 17).
	EnterCategoryName (see page 15)	Looks up a localized string similar to Enter Category (see page 55) Name (see page 17).
	ExitApp (see page 15)	Looks up a localized string similar to Exit App (see page 27).
	ExitAppQuestion (see page 15)	Looks up a localized string similar to Do You Want To Exit App (see page 27)?.
	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.
	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 Affect.
	ModificationDate (see page 17)	Looks up a localized string similar to Last Modification Date (see page 14).
	MostReadAuthor (see page 17)	Looks up a localized string similar to Author (see page 31) You Most Read.
	MostReadCategory (see page 17)	Looks up a localized string similar to Category (see page 55) You Most Read.
	MostReadPublisher (see page 17)	Looks up a localized string similar to Publisher You Most Read.
	Name (see page 17)	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).
	PageNumber (see page 18)	Looks up a localized string similar to Page Number.
	PagePerDay (see page 18)	Looks up a localized string similar to Average Page Read Per Day.
	Persian (see page 18)	Looks up a localized string similar to Persian.

	Portuguese (see page 18)	Looks up a localized string similar to Portuguese.
	PublisherName (see page 19)	Looks up a localized string similar to Publisher Name (see page 17).
	Purple (see page 19)	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.
	RemoveBackgroundImage (see page 19)	Looks up a localized string similar to Remove Background (see page 9) 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 20)	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 21)	Looks up a localized string similar to Select (see page 20) Background (see page 9) Color.
	Selected (see page 21)	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 Library App (see page 27).
	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.
	ShareBook (see page 22)	Looks up a localized string similar to Share Book (see page 46).
	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 9) 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 23)	Looks up a localized string similar to Thai.

	TotalBookCount (see page 23)	Looks up a localized string similar to Book (see page 46) Count You Read.
	Turkish (see page 23)	Looks up a localized string similar to Turkish.
	WorstBook (see page 23)	Looks up a localized string similar to Worst Book (see page 46).
	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 27](#)).

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 reading book a lot. And after Awesome Note app, i decide to create an app which has similar properties like GoodReads. You can add categories, add authors and add books on it. You can send information of your books 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.AboutTheAwesomeLibrary Property

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

C#

```
public static string AboutTheAwesomeLibrary;
```

1.1.1.1.1.2.5 AppResources.AddAuthor Property

Looks up a localized string similar to Add Author ([see page 31](#)).

C#

```
public static string AddAuthor;
```

1.1.1.1.1.2.6 AppResources.AddBook Property

Looks up a localized string similar to Add Book (see page 46).

C#

```
public static string AddBook;
```

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.Arabic Property

Looks up a localized string similar to Arabic.

C#

```
public static string Arabic;
```

1.1.1.1.1.2.9 AppResources.Ascending Property

Looks up a localized string similar to Ascending.

C#

```
public static string Ascending;
```

1.1.1.1.1.2.10 AppResources.AuthorAddSuccess Property

Looks up a localized string similar to Author (see page 31) Has Been Added Successfully.

C#

```
public static string AuthorAddSuccess;
```

1.1.1.1.1.2.11 AppResources.AuthorAlreadySameCategory Property

Looks up a localized string similar to Author (see page 31) Has This Category (see page 55) Already.

C#

```
public static string AuthorAlreadySameCategory;
```

1.1.1.1.1.2.12 AppResources.AuthorCategoryAddSuccess Property

Looks up a localized string similar to Category (see page 55) Has Been Added To Author (see page 31) Successfully.

C#

```
public static string AuthorCategoryAddSuccess;
```

1.1.1.1.1.2.13 AppResources.AuthorDeleteSuccess Property

Looks up a localized string similar to Author (see page 31) Has Been Removed Successfully.

C#

```
public static string AuthorDeleteSuccess;
```

1.1.1.1.1.2.14 AppResources.AuthorExists Property

Looks up a localized string similar to This Author (see page 31) Has Already Exists.

C#

```
public static string AuthorExists;
```

1.1.1.1.1.2.15 AppResources.AuthorList Property

Looks up a localized string similar to Author (see page 31) List.

C#

```
public static string AuthorList;
```

1.1.1.1.1.2.16 AppResources.AuthorName Property

Looks up a localized string similar to Author (see page 31) Name (see page 17).

C#

```
public static string AuthorName;
```

1.1.1.1.1.2.17 AppResources.AuthorNameChangeSuccess Property

Looks up a localized string similar to Author (see page 31) Name (see page 17) Has Been Changed Successfully.

C#

```
public static string AuthorNameChangeSuccess;
```

1.1.1.1.1.2.18 AppResources.AuthorOrderBy Property

Looks up a localized string similar to Author (see page 31) Order By.

C#

```
public static string AuthorOrderBy;
```

1.1.1.1.1.2.19 AppResources.AuthorOrderStyle Property

Looks up a localized string similar to Author (see page 31) Order Style.

C#

```
public static string AuthorOrderStyle;
```

1.1.1.1.1.2.20 AppResources.AuthorOrderStyleChangeSuccess Property

Looks up a localized string similar to Author (see page 31) Order Style Has Been Changed Successfully.

C#

```
public static string AuthorOrderStyleChangeSuccess;
```

1.1.1.1.1.2.21 AppResources.AuthorOrderTypeChangeSuccess Property

Looks up a localized string similar to Author (see page 31) Order Type Has Been Changed Successfully.

C#

```
public static string AuthorOrderTypeChangeSuccess;
```

1.1.1.1.1.2.22 AppResources.AuthorSettings Property

Looks up a localized string similar to Author (see page 31) Settings (see page 21).

C#

```
public static string AuthorSettings;
```

1.1.1.1.1.2.23 AppResources.Background Property

Looks up a localized string similar to Background.

C#

```
public static string Background;
```

1.1.1.1.1.2.24 AppResources.BackgroundColor Property

Looks up a localized string similar to Background (see page 9) Color.

C#

```
public static string BackgroundColor;
```

1.1.1.1.1.2.25 AppResources.BackgroundColorChangeSuccess Property

Looks up a localized string similar to Background (see page 9) Color Has Been Changed Successfully.

C#

```
public static string BackgroundColorChangeSuccess;
```

1.1.1.1.1.2.26 AppResources.BackgroundImage Property

Looks up a localized string similar to Background (see page 9) Image.

C#

```
public static string BackgroundImage;
```

1.1.1.1.1.2.27 AppResources.BackgroundImageChangeSuccess Property

Looks up a localized string similar to Background (see page 9) Image Has Been Changed Successfully.

C#

```
public static string BackgroundImageChangeSuccess;
```

1.1.1.1.1.2.28 AppResources.BackgroundImageRemoveSuccess Property

Looks up a localized string similar to Background (see page 9) Image Has Been Removed Successfully.

C#

```
public static string BackgroundImageRemoveSuccess;
```

1.1.1.1.1.2.29 AppResources.BestBook Property

Looks up a localized string similar to Best Book (see page 46).

C#

```
public static string BestBook;
```

1.1.1.1.1.2.30 AppResources.Black Property

Looks up a localized string similar to Black.

C#

```
public static string Black;
```

1.1.1.1.1.2.31 AppResources.Blue Property

Looks up a localized string similar to Blue.

C#

```
public static string Blue;
```

1.1.1.1.1.2.32 AppResources.BookComment Property

Looks up a localized string similar to Book (see page 46) Comment.

C#

```
public static string BookComment;
```

1.1.1.1.1.2.33 AppResources.BookCount Property

Looks up a localized string similar to Book (see page 46) Count.

C#

```
public static string BookCount;
```

1.1.1.1.1.2.34 AppResources.BookDeleteSuccess Property

Looks up a localized string similar to Book (see page 46) Has Been Removed Successfully.

C#

```
public static string BookDeleteSuccess;
```

1.1.1.1.1.2.35 AppResources.BookList Property

Looks up a localized string similar to Book (see page 46) List.

C#

```
public static string BookList;
```

1.1.1.1.1.2.36 AppResources.BookName Property

Looks up a localized string similar to Book (see page 46) Name (see page 17).

C#

```
public static string BookName;
```

1.1.1.1.1.2.37 AppResources.BookNameMustBe Property

Looks up a localized string similar to You Have To Enter Book (see page 46) Name (see page 17) At Least.

C#

```
public static string BookNameMustBe;
```

1.1.1.1.1.2.38 AppResources.BookOrderBy Property

Looks up a localized string similar to Book (see page 46) Order By.

C#

```
public static string BookOrderBy;
```

1.1.1.1.1.2.39 AppResources.BookOrderStyle Property

Looks up a localized string similar to Book (see page 46) Order Style.

C#

```
public static string BookOrderStyle;
```

1.1.1.1.1.2.40 AppResources.BookOrderStyleChangeSuccess Property

Looks up a localized string similar to Book (see page 46) Order Style Has Been Changed Successfully.

C#

```
public static string BookOrderStyleChangeSuccess;
```

1.1.1.1.1.2.41 AppResources.BookOrderTypeChangeSuccess Property

Looks up a localized string similar to Book (see page 46) Order Type Has Been Changed Successfully.

C#

```
public static string BookOrderTypeChangeSuccess;
```

1.1.1.1.1.2.42 AppResources.BookPerDay Property

Looks up a localized string similar to Average Book (see page 46) Read Per Day.

C#

```
public static string BookPerDay;
```

1.1.1.1.1.2.43 AppResources.BookRating Property

Looks up a localized string similar to Book (see page 46) Rating.

C#

```
public static string BookRating;
```

1.1.1.1.1.2.44 AppResources.BookSaveSuccess Property

Looks up a localized string similar to Book (see page 46) Has Been Saved Successfully.

C#

```
public static string BookSaveSuccess;
```

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 17).

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 17) 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.DeleteAuthor Property

Looks up a localized string similar to Delete Author (see page 31).

C#

```
public static string DeleteAuthor;
```

1.1.1.1.1.2.64 AppResources.DeleteAuthorQuestion Property

Looks up a localized string similar to You Will Delete The Author (see page 31) With All Contents. Are You Sure?.

C#

```
public static string DeleteAuthorQuestion;
```

1.1.1.1.1.2.65 AppResources.DeleteBook Property

Looks up a localized string similar to Delete Book (see page 46).

C#

```
public static string DeleteBook;
```

1.1.1.1.1.2.66 AppResources.DeleteBookQuestion Property

Looks up a localized string similar to You Will Delete This Book (see page 46). Are You Sure?.

C#

```
public static string DeleteBookQuestion;
```

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.EnterAuthorName Property

Looks up a localized string similar to Enter Author (see page 31) Name (see page 17).

C#

```
public static string EnterAuthorName;
```

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 17).

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 27).

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 27)?.

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.Language Property

Looks up a localized string similar to Language.

C#

```
public static string Language;
```

1.1.1.1.1.2.89 AppResources.LanguageWarning Property

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

C#

```
public static string LanguageWarning;
```

1.1.1.1.1.2.90 AppResources.ModificationDate Property

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

C#

```
public static string ModificationDate;
```

1.1.1.1.1.2.91 AppResources.MostReadAuthor Property

Looks up a localized string similar to Author (see page 31) You Most Read.

C#

```
public static string MostReadAuthor;
```

1.1.1.1.1.2.92 AppResources.MostReadCategory Property

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

C#

```
public static string MostReadCategory;
```

1.1.1.1.1.2.93 AppResources.MostReadPublisher Property

Looks up a localized string similar to Publisher You Most Read.

C#

```
public static string MostReadPublisher;
```

1.1.1.1.1.2.94 AppResources.Name Property

Looks up a localized string similar to Name.

C#

```
public static string Name;
```

1.1.1.1.1.2.95 AppResources.OK Property

Looks up a localized string similar to Ok.

C#

```
public static string OK;
```

1.1.1.1.1.2.96 AppResources.OneDrive Property

Looks up a localized string similar to OneDrive.

C#

```
public static string OneDrive;
```

1.1.1.1.1.2.97 AppResources.OneDriveSyncCompleted Property

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

C#

```
public static string OneDriveSyncCompleted;
```

1.1.1.1.1.2.98 AppResources.Orange Property

Looks up a localized string similar to Orange.

C#

```
public static string Orange;
```

1.1.1.1.1.2.99 AppResources.OtherSettings Property

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

C#

```
public static string OtherSettings;
```

1.1.1.1.1.2.100 AppResources.PageNumber Property

Looks up a localized string similar to Page Number.

C#

```
public static string PageNumber;
```

1.1.1.1.1.2.101 AppResources.PagePerDay Property

Looks up a localized string similar to Average Page Read Per Day.

C#

```
public static string PagePerDay;
```

1.1.1.1.1.2.102 AppResources.Persian Property

Looks up a localized string similar to Persian.

C#

```
public static string Persian;
```

1.1.1.1.1.2.103 AppResources.Portuguese Property

Looks up a localized string similar to Portuguese.

C#

```
public static string Portuguese;
```

1.1.1.1.1.2.104 AppResources.PublisherName Property

Looks up a localized string similar to Publisher Name (see page 17).

C#

```
public static string PublisherName;
```

1.1.1.1.1.2.105 AppResources.Purple Property

Looks up a localized string similar to Purple.

C#

```
public static string Purple;
```

1.1.1.1.1.2.106 AppResources.Rate Property

Looks up a localized string similar to Rate.

C#

```
public static string Rate;
```

1.1.1.1.1.2.107 AppResources.Red Property

Looks up a localized string similar to Red.

C#

```
public static string Red;
```

1.1.1.1.1.2.108 AppResources.RemoveBackgroundImage Property

Looks up a localized string similar to Remove Background (see page 9) Image.

C#

```
public static string RemoveBackgroundImage;
```

1.1.1.1.1.2.109 AppResources.ResetSettings Property

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

C#

```
public static string ResetSettings;
```

1.1.1.1.1.2.110 AppResources.ResourceFlowDirection Property

Looks up a localized string similar to LeftToRight.

C#

```
public static string ResourceFlowDirection;
```

1.1.1.1.1.2.111 AppResources.ResourceLanguage Property

Looks up a localized string similar to en-US.

C#

```
public static string ResourceLanguage;
```

1.1.1.1.1.2.112 AppResources.ResourceManager Property

Returns the cached ResourceManager instance used by this class.

C#

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

1.1.1.1.1.2.113 AppResources.Russian Property

Looks up a localized string similar to Russian.

C#

```
public static string Russian;
```

1.1.1.1.1.2.114 AppResources.Sanskrit Property

Looks up a localized string similar to Sanskrit.

C#

```
public static string Sanskrit;
```

1.1.1.1.1.2.115 AppResources.Save Property

Looks up a localized string similar to Save.

C#

```
public static string Save;
```

1.1.1.1.1.2.116 AppResources.Search Property

Looks up a localized string similar to Search.

C#

```
public static string Search;
```

1.1.1.1.1.2.117 AppResources.SearchCompleted Property

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

C#

```
public static string SearchCompleted;
```

1.1.1.1.1.2.118 AppResources.SearchResults Property

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

C#

```
public static string SearchResults;
```

1.1.1.1.1.2.119 AppResources.SearchTrimFault Property

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

C#

```
public static string SearchTrimFault;
```

1.1.1.1.1.2.120 AppResources.Select Property

Looks up a localized string similar to Select.

C#

```
public static string Select;
```

1.1.1.1.1.2.121 AppResources.SelectBackgroundColor Property

Looks up a localized string similar to Select (see page 20) Background (see page 9) Color.

C#

```
public static string SelectBackgroundColor;
```

1.1.1.1.1.2.122 AppResources.Selected Property

Looks up a localized string similar to Selected.

C#

```
public static string Selected;
```

1.1.1.1.1.2.123 AppResources.SelectFontFamily Property

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

C#

```
public static string SelectFontFamily;
```

1.1.1.1.1.2.124 AppResources.SelectFontSize Property

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

C#

```
public static string SelectFontSize;
```

1.1.1.1.1.2.125 AppResources.SelectLanguage Property

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

C#

```
public static string SelectLanguage;
```

1.1.1.1.1.2.126 AppResources.SendWithApp Property

Looks up a localized string similar to Send With Awesome Library App (see page 27).

C#

```
public static string SendWithApp;
```

1.1.1.1.1.2.127 AppResources.SendWithMail Property

Looks up a localized string similar to Send With Mail.

C#

```
public static string SendWithMail;
```

1.1.1.1.1.2.128 AppResources.SendWithSMS Property

Looks up a localized string similar to Send With SMS.

C#

```
public static string SendWithSMS;
```

1.1.1.1.1.2.129 AppResources.Settings Property

Looks up a localized string similar to Settings.

C#

```
public static string Settings;
```

1.1.1.1.1.2.130 AppResources.ShareBook Property

Looks up a localized string similar to Share Book (see page 46).

C#

```
public static string ShareBook;
```

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 9) 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.TotalBookCount Property

Looks up a localized string similar to Book (see page 46) Count You Read.

C#

```
public static string TotalBookCount;
```

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.WorstBook Property

Looks up a localized string similar to Worst Book (see page 46).

C#

```
public static string WorstBook;
```

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 AwesomeLibrary>AboutPage.
	AddCategoryPage (see page 25)	This is class AwesomeLibrary.AddCategoryPage.
	App (see page 27)	This is class AwesomeLibrary.App.
	AppSettings (see page 29)	This is class AwesomeLibrary.AppSettings.
	Author (see page 31)	This is class AwesomeLibrary.Author.
	AuthorPage (see page 33)	This is class AwesomeLibrary.AuthorPage.
	AuthorSettingsPage (see page 38)	This is class AwesomeLibrary.AuthorSettingsPage.
	AwesomeLibraryDataContext (see page 41)	This is class AwesomeLibrary.AwesomeLibraryDataContext.
	BackgroundColorSettingsPage (see page 43)	This is class AwesomeLibrary.BackgroundColorSettingsPage.
	Book (see page 46)	This is class AwesomeLibrary.Book.
	BookAuthor (see page 49)	This is class AwesomeLibrary.BookAuthor.

	BookPage (see page 50)	This is class AwesomeLibrary.BookPage.
	Category (see page 55)	This is class AwesomeLibrary.Category.
	CategoryAuthor (see page 57)	This is class AwesomeLibrary.CategoryAuthor.
	CategoryPage (see page 58)	This is class AwesomeLibrary.CategoryPage.
	CategorySettingsPage (see page 62)	This is class AwesomeLibrary.CategorySettingsPage.
	FontFamilySettingsPage (see page 65)	This is class AwesomeLibrary.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 67)	This is class AwesomeLibrary.FontSizeSettingsPage.
	GeneralSettingsPage (see page 69)	This is class AwesomeLibrary.GeneralSettingsPage.
	LanguageSettingsPage (see page 75)	This is class AwesomeLibrary.LanguageSettingsPage.
	LocalizedStrings (see page 76)	Provides access to string resources.
	MainPage (see page 77)	This is class AwesomeLibrary.MainPage.
	OrderSettingsPage (see page 79)	This is class AwesomeLibrary.OrderSettingsPage.
	OrderStyleSettingsPage (see page 82)	This is class AwesomeLibrary.OrderStyleSettingsPage.
	PopupAddChange (see page 85)	This is class AwesomeLibrary.PopupAddChange.
	SearchPage (see page 86)	This is class AwesomeLibrary.SearchPage.
	StatisticsPage (see page 87)	This is class AwesomeLibrary.StatisticsPage.

1.1.2.1 AboutPage Class

Class Hierarchy



C#

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

File

AboutPage.xaml.cs ([see page 90](#))

Description

This is class AwesomeLibrary.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()
```

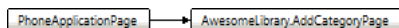
```

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 AwesomeLibrary.AddCategoryPage.

Methods

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

AddCategoryPage Fields

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

AddCategoryPage Methods

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

C#

```
public int authorId;
```

Description

This is authorId, 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:     authorId = int.Parse(e.Fragment);  
7:     using (var context = new  
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))  
8:     {  
9:         var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()  
as Author;  
10:         lstCategories.Items.Clear();  
11:         lblAuthorName.Text = author.AuthorName;  
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

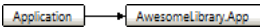
This is OnNavigatedTo, a member of class AddCategoryPage.

Body Source

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

1.1.2.3 App Class

Class Hierarchy



C#

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

File

App.xaml.cs (🔗 see page 95)

Description

This is class AwesomeLibrary.App.

Methods

	Name	Description
🔗	App (🔗 see page 28)	Constructor for the Application object.

App Fields

	Name	Description
🔗	categoryNumber (🔗 see page 28)	This is categoryNumber, a member of class App.

App Properties

	Name	Description
🔗	IsTrial (🔗 see page 29)	This is IsTrial, a member of class App.
🔗	RootFrame (🔗 see page 29)	This is RootFrame, a member of class App.

1.1.2.3.1 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: }
```

1.1.2.3.2 App Fields

1.1.2.3.2.1 App.categoryNumber Field

C#

```
public int categoryNumber;
```

Description

This is categoryNumber, a member of class App.

1.1.2.3.3 App Properties

1.1.2.3.3.1 App.IsTrial Property

C#

```
public bool IsTrial;
```

Description

This is IsTrial, a member of class App.

1.1.2.3.3.2 App.RootFrame Property

C#

```
public static PhoneApplicationFrame RootFrame;
```

Description

This is RootFrame, a member of class App.

1.1.2.4 AppSettings Class

Class Hierarchy

AwesomeLibrary.AppSettings

C#

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











File

AppSettings.cs ([see page 121](#))

Description

This is class AwesomeLibrary.AppSettings.

AppSettings Properties

	Name	Description
	AppBackgroundColor (see page 30)	This is AppBackgroundColor, a member of class AppSettings.
	AppBackgroundImage (see page 30)	This is AppBackgroundImage, a member of class AppSettings.
	AppLangName (see page 30)	This is AppLangName, a member of class AppSettings.
	AppSettingsId (see page 30)	This is AppSettingsId, a member of class AppSettings.
	CategoryOrderBy (see page 30)	This is CategoryOrderBy, a member of class AppSettings.
	CategoryOrderStyle (see page 30)	This is CategoryOrderStyle, a member of class AppSettings.
	CurrentAuthorNumber (see page 31)	This is CurrentAuthorNumber, a member of class AppSettings.
	CurrentCategoryNumber (see page 31)	This is CurrentCategoryNumber, a member of class AppSettings.
	FontFamily (see page 31)	This is FontFamily, a member of class AppSettings.
	FontSize (see page 31)	This is FontSize, a member of class AppSettings.

1.1.2.4.1 AppSettings Properties

1.1.2.4.1.1 AppSettings.AppBackgroundColor Property

C#

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

Description

This is AppBackgroundColor, a member of class AppSettings.

1.1.2.4.1.2 AppSettings.AppBackgroundImage Property

C#

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

Description

This is AppBackgroundImage, a member of class AppSettings.

1.1.2.4.1.3 AppSettings.AppLangName Property

C#

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

Description

This is AppLangName, a member of class AppSettings.

1.1.2.4.1.4 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.

1.1.2.4.1.5 AppSettings.CategoryOrderBy Property

C#

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

Description

This is CategoryOrderBy, a member of class AppSettings.

1.1.2.4.1.6 AppSettings.CategoryOrderStyle Property

C#

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

Description

This is CategoryOrderStyle, a member of class AppSettings.

1.1.2.4.1.7 AppSettings.CurrentAuthorNumber Property**C#**

```
[Column]
public int CurrentAuthorNumber;
```

Description

This is CurrentAuthorNumber, a member of class AppSettings.

1.1.2.4.1.8 AppSettings.CurrentCategoryNumber Property**C#**

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

Description

This is CurrentCategoryNumber, a member of class AppSettings.

1.1.2.4.1.9 AppSettings.FontFamily Property**C#**

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

Description

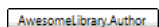
This is FontFamily, a member of class AppSettings.

1.1.2.4.1.10 AppSettings.FontSize Property**C#**

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

Description

This is FontSize, a member of class AppSettings.

1.1.2.5 Author Class**Class Hierarchy**A small rectangular box containing the text 'AwesomeLibrary.Author'.**C#**

```
[Table]
public class Author;
```









File

Author.cs ( see page 123)

Description

This is class AwesomeLibrary.Author.

Author Properties

	Name	Description
	AuthorBookCount (see page 32)	This is AuthorBookCount, a member of class Author.
	AuthorId (see page 32)	This is AuthorId, a member of class Author.
	AuthorName (see page 32)	This is AuthorName, a member of class Author.
	AuthorNameCount (see page 32)	This is AuthorNameCount, a member of class Author.
	BookOrderBy (see page 33)	This is BookOrderBy, a member of class Author.
	BookOrderStyle (see page 33)	This is BookOrderStyle, a member of class Author.
	CreationDate (see page 33)	This is CreationDate, a member of class Author.
	ModificationDate (see page 33)	This is ModificationDate, a member of class Author.

1.1.2.5.1 Author Properties

1.1.2.5.1.1 Author.AuthorBookCount Property

C#

```
[Column]
public int AuthorBookCount;
```

Description

This is AuthorBookCount, a member of class Author.

1.1.2.5.1.2 Author.AuthorId Property

C#

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

Description

This is AuthorId, a member of class Author.

1.1.2.5.1.3 Author.AuthorName Property

C#

```
[Column]
public string AuthorName;
```

Description

This is AuthorName, a member of class Author.

1.1.2.5.1.4 Author.AuthorNameCount Property

C#

```
[Column]
public string AuthorNameCount;
```

Description

This is AuthorNameCount, a member of class Author.

1.1.2.5.1.5 Author.BookOrderBy Property

C#

```
[Column]
public string BookOrderBy;
```

Description

This is BookOrderBy, a member of class Author.

1.1.2.5.1.6 Author.BookOrderStyle Property

C#

```
[Column]
public string BookOrderStyle;
```

Description

This is BookOrderStyle, a member of class Author.

1.1.2.5.1.7 Author.CreationDate Property

C#

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

Description

This is CreationDate, a member of class Author.

1.1.2.5.1.8 Author.ModificationDate Property

C#

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

Description

This is ModificationDate, a member of class Author.

1.1.2.6 AuthorPage Class

Class Hierarchy



C#

```
public class AuthorPage : PhoneApplicationPage;
```

File

AuthorPage.xaml.cs (🔗 see page 124)






Description

This is class AwesomeLibrary.AuthorPage.




Methods

	Name	Description
🔗	AuthorPage (🔗 see page 34)	This is AuthorPage, a member of class AuthorPage.

AuthorPage Fields

	Name	Description
	authorId (see page 35)	This is authorId, a member of class AuthorPage.
	bookId (see page 35)	This is bookId, a member of class AuthorPage.
	categoryId (see page 35)	This is categoryId, a member of class AuthorPage.
	oldAuthorName (see page 35)	This is oldAuthorName, a member of class AuthorPage.
	popup (see page 35)	This is popup, a member of class AuthorPage.

AuthorPage Methods

	Name	Description
	OnFragmentNavigation (see page 35)	This is OnFragmentNavigation, a member of class AuthorPage.
	OnNavigatedFrom (see page 38)	This is OnNavigatedFrom, a member of class AuthorPage.
	OnNavigatedTo (see page 38)	This is OnNavigatedTo, a member of class AuthorPage.

1.1.2.6.1 AuthorPage.AuthorPage Constructor**C#**

```
public AuthorPage();
```

Description

This is AuthorPage, a member of class AuthorPage.

Body Source

```
1: public AuthorPage()
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.AddBook;
10:    ApplicationBar.Buttons.Add(button1);
11:    button1.Click += new EventHandler(AddBookButton_Click);
12:
13:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
14:    button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
15:    button2.Text = AppResources.DeleteAuthor;
16:    ApplicationBar.Buttons.Add(button2);
17:    button2.Click += new EventHandler>DeleteAuthorButton_Click);
18:
19:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
20:    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
21:    button3.Text = AppResources.AuthorSettings;
22:    ApplicationBar.Buttons.Add(button3);
23:    button3.Click += new EventHandler(AuthorSettingsButton_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: }
```

1.1.2.6.2 AuthorPage Fields

1.1.2.6.2.1 AuthorPage.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, a member of class AuthorPage.

1.1.2.6.2.2 AuthorPage.bookId Field

C#

```
public int bookId;
```

Description

This is bookId, a member of class AuthorPage.

1.1.2.6.2.3 AuthorPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class AuthorPage.

1.1.2.6.2.4 AuthorPage.oldAuthorName Field

C#

```
public string oldAuthorName;
```

Description

This is oldAuthorName, a member of class AuthorPage.

1.1.2.6.2.5 AuthorPage.popup Field

C#

```
public Popup popup;
```

Description

This is popup, a member of class AuthorPage.

1.1.2.6.3 AuthorPage Methods

1.1.2.6.3.1 AuthorPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class AuthorPage.

Body Source

```

1: protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
2: {
3:     List<Book> books = new List<Book>();
4:     List<Book> booksOrdered = new List<Book>();
5:
6:     // displays "Fragment: Detail"
7:     //MessageBox.Show("Folder Id: " + e.Fragment);
8:     base.OnFragmentNavigation(e);
9:
10:    lstBooks.Items.Clear();
11:
12:    using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
13:    {
14:
15:        var appSettings = context.AppSettings.First();
16:        categoryId = appSettings.CurrentCategoryNumber;
17:
18:        var author = context.Authors.Where(j =>
j.AuthorId.Equals(e.Fragment)).Single() as Author;
19:        authorId = author.AuthorId;
20:
21:        var appSettings2 = context.AppSettings;
22:        foreach (var item in appSettings2)
23:        {
24:            item.CurrentAuthorNumber = authorId;
25:        }
26:        context.SubmitChanges();
27:
28:        var authorBooks = context.BookAuthors.Where(j =>
j.AuthorId.Equals(e.Fragment)).ToList() as List<BookAuthor>;
29:        if (authorBooks.Count != 0)
30:        {
31:            foreach (var item in authorBooks)
32:            {
33:                try
34:                {
35:                    var book = context.Books.Where(j =>
j.BookCategoryId.Equals(categoryId) && j.BookId.Equals(item.BookId)).Single() as Book;
36:                    books.Add(book);
37:                }
38:                catch (Exception)
39:                {
40:                }
41:            }
42:
43:        }
44:
45:
46:        string orderStyle = author.BookOrderStyle;
47:
48:        switch (author.BookOrderBy)
49:        {
50:            case "NAME":
51:                if (orderStyle == "A")
52:                {
53:                    booksOrdered = books.OrderBy(j => j.BookName).ToList();
54:                }
55:                else
56:                {
57:                    booksOrdered = books.OrderByDescending(j => j.BookName).ToList();
58:                }
59:                break;
60:            case "CDATE":
61:                if (orderStyle == "A")
62:                {

```

```

63:             booksOrdered = books.OrderBy(j => j.CreationDate).ToList();
64:         }
65:         else
66:         {
67:             booksOrdered = books.OrderByDescending(j =>
j.CreationDate).ToList();
68:         }
69:         break;
70:         case "MDATE":
71:             if (orderStyle == "A")
72:             {
73:                 booksOrdered = books.OrderBy(j => j.ModificationDate).ToList();
74:             }
75:             else
76:             {
77:                 booksOrdered = books.OrderByDescending(j =>
j.ModificationDate).ToList();
78:             }
79:             break;
80:         case "SDATE":
81:             if (orderStyle == "A")
82:             {
83:                 booksOrdered = books.OrderBy(j => j.ReadStartDate).ToList();
84:             }
85:             else
86:             {
87:                 booksOrdered = books.OrderByDescending(j =>
j.ReadStartDate).ToList();
88:             }
89:             break;
90:         case "FDATE":
91:             if (orderStyle == "A")
92:             {
93:                 booksOrdered = books.OrderBy(j => j.ReadFinishDate).ToList();
94:             }
95:             else
96:             {
97:                 booksOrdered = books.OrderByDescending(j =>
j.ReadFinishDate).ToList();
98:             }
99:             break;
100:        case "RATING":
101:            if (orderStyle == "A")
102:            {
103:                booksOrdered = books.OrderBy(j => j.BookRating).ToList();
104:            }
105:            else
106:            {
107:                booksOrdered = books.OrderByDescending(j => j.BookRating).ToList();
108:            }
109:            break;
110:        default:
111:            if (orderStyle == "A")
112:            {
113:                booksOrdered = books.OrderBy(j => j.BookName).ToList();
114:            }
115:            else
116:            {
117:                booksOrdered = books.OrderBy(j => j.BookName).ToList();
118:            }
119:            break;
120:    }
121:
122:    lblAuthorName.Text = author.AuthorName;
123:    lblBookList.Text = AppResources.BookList + " (" + author.AuthorName + ")";
124:    lstBooks.ItemsSource = booksOrdered;
125:    lstBooks.DisplayMemberPath = "BookNameRating";
126:    SetBackgroundColor();
127:    //lstNoteList.DisplayMemberPath = "NameCreation";

```

```
128:     }
129: }
```

1.1.2.6.3.2 AuthorPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class AuthorPage.

Body Source

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

1.1.2.6.3.3 AuthorPage.OnNavigatedTo Method

C#

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

Description

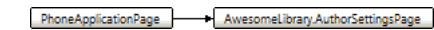
This is OnNavigatedTo, a member of class AuthorPage.

Body Source

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

1.1.2.7 AuthorSettingsPage Class

Class Hierarchy



C#

```
public class AuthorSettingsPage : PhoneApplicationPage;
```

File

AuthorSettingsPage.xaml.cs (🔗 see page 131)



Description

This is class AwesomeLibrary.AuthorSettingsPage.




Methods

	Name	Description
☰🔍	AuthorSettingsPage (🔗 see page 39)	This is AuthorSettingsPage, a member of class AuthorSettingsPage.

AuthorSettingsPage Fields

	Name	Description
	authorId (see page 39)	This is authorId, a member of class AuthorSettingsPage.
	categoryId (see page 40)	This is categoryId, a member of class AuthorSettingsPage.

AuthorSettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 40)	This is OnFragmentNavigation, a member of class AuthorSettingsPage.
	OnNavigatedFrom (see page 41)	This is OnNavigatedFrom, a member of class AuthorSettingsPage.
	OnNavigatedTo (see page 41)	This is OnNavigatedTo, a member of class AuthorSettingsPage.

1.1.2.7.1 AuthorSettingsPage.AuthorSettingsPage Constructor**C#**

```
public AuthorSettingsPage();
```

Description

This is AuthorSettingsPage, a member of class AuthorSettingsPage.

Body Source

```
1: public AuthorSettingsPage()
2: {
3:     InitializeComponent();
4:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:     pvAuthorSettings.Title = AppResources.AuthorSettings;
12:     piFont.Header = AppResources.Font;
13:     piOtherSettings.Header = AppResources.OtherSettings;
14:
15:     btnFontFamily.Content = AppResources.Select;
16:     btnFontSize.Content = AppResources.Select;
17:     btnBookOrder.Content = AppResources.Select;
18:     btnBookOrderStyle.Content = AppResources.Select;
19: }
```

1.1.2.7.2 AuthorSettingsPage Fields**1.1.2.7.2.1 AuthorSettingsPage.authorId Field****C#**

```
public int authorId;
```

Description

This is authorId, a member of class AuthorSettingsPage.

1.1.2.7.2 AuthorSettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class AuthorSettingsPage.

1.1.2.7.3 AuthorSettingsPage Methods

1.1.2.7.3.1 AuthorSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class AuthorSettingsPage.

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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
7:     {
8:         var author = context.Authors.Where(j => j.AuthorId.Equals(e.Fragment)).Single()
as Author;
9:         authorId = author.AuthorId;
10:        var appSettings = context.AppSettings.First();
11:        categoryId = appSettings.CurrentCategoryNumber;
12:        string orderStyle = author.BookOrderStyle;
13:
14:        if (author.BookOrderBy == "NAME")
15:        {
16:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.Name + ")";
17:        }
18:        if (author.BookOrderBy == "CDATE")
19:        {
20:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.CreationDate + ")";
21:        }
22:        if (author.BookOrderBy == "MDATE")
23:        {
24:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.ModificationDate + ")";
25:        }
26:        if (author.BookOrderBy == "SDATE")
27:        {
28:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.StartDate + ")";
29:        }
30:        if (author.BookOrderBy == "FDATE")
31:        {
32:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.FinishDate + ")";
33:        }
34:        if (author.BookOrderBy == "RATING")
35:        {
36:            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
```

```

+ ": " + AppResources.BookRating + ")";
37:     }
38:     if (author.BookOrderStyle == "A")
39:     {
40:         lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
41:     }
42:     if (author.BookOrderStyle == "D")
43:     {
44:         lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
45:     }
46:     //lstNoteList.DisplayMemberPath = "NameCreation";
47:     SetBackgroundColor();
48:     }
49: }

```

1.1.2.7.3.2 AuthorSettingsPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class AuthorSettingsPage.

Body Source

```

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

```

1.1.2.7.3.3 AuthorSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class AuthorSettingsPage.

Body Source

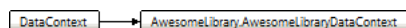
```

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

```

1.1.2.8 AwesomeLibraryDataContext Class

Class Hierarchy



C#

```
public class AwesomeLibraryDataContext : DataContext;
```


File

AwesomeLibraryDataContext.cs (📄 see page 135)








Description

This is class AwesomeLibrary.AwesomeLibraryDataContext.

Methods

	Name	Description
	AwesomeLibraryDataContext (see page 42)	This is AwesomeLibraryDataContext, a member of class AwesomeLibraryDataContext.

AwesomeLibraryDataContext Fields

	Name	Description
	AppSettings (see page 42)	This is AppSettings, a member of class AwesomeLibraryDataContext.
	Authors (see page 42)	This is Authors, a member of class AwesomeLibraryDataContext.
	BookAuthors (see page 43)	This is BookAuthors, a member of class AwesomeLibraryDataContext.
	Books (see page 43)	This is Books, a member of class AwesomeLibraryDataContext.
	Categories (see page 43)	This is Categories, a member of class AwesomeLibraryDataContext.
	CategoryAuthors (see page 43)	This is CategoryAuthors, a member of class AwesomeLibraryDataContext.
	ConnectionString (see page 43)	This is ConnectionString, a member of class AwesomeLibraryDataContext.

1.1.2.8.1 AwesomeLibraryDataContext.AwesomeLibraryDataContext Constructor

C#

```
public AwesomeLibraryDataContext(string connectionString);
```

Description

This is AwesomeLibraryDataContext, a member of class AwesomeLibraryDataContext.

Body Source

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

1.1.2.8.2 AwesomeLibraryDataContext Fields

1.1.2.8.2.1 AwesomeLibraryDataContext.AppSettings Field

C#

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

Description

This is AppSettings, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.2 AwesomeLibraryDataContext.Authors Field

C#

```
public Table<Author> Authors;
```

Description

This is Authors, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.3 AwesomeLibraryDataContext.BookAuthors Field

C#

```
public Table<BookAuthor> BookAuthors;
```

Description

This is BookAuthors, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.4 AwesomeLibraryDataContext.Books Field

C#

```
public Table<Book> Books;
```

Description

This is Books, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.5 AwesomeLibraryDataContext.Categories Field

C#

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

Description

This is Categories, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.6 AwesomeLibraryDataContext.CategoryAuthors Field

C#

```
public Table<CategoryAuthor> CategoryAuthors;
```

Description

This is CategoryAuthors, a member of class AwesomeLibraryDataContext.

1.1.2.8.2.7 AwesomeLibraryDataContext.ConnectionString Field

C#

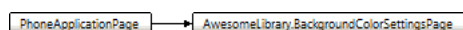
```
public const string ConnectionString = @"Data Source=isostore:/AwesomeLibrary.sdf; Max Database Size=256; Max Buffer Size=4096;";
```

Description

This is ConnectionString, a member of class AwesomeLibraryDataContext.

1.1.2.9 BackgroundColorSettingsPage Class

Class Hierarchy



C#

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


File

BackgroundColorSettingsPage.xaml.cs (📄 see page 136)

Description

This is class AwesomeLibrary.BackgroundColorSettingsPage.




Methods

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

BackgroundColorSettingsPage Fields

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

BackgroundColorSettingsPage Methods

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

1.1.2.9.1 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: }
```

1.1.2.9.2 BackgroundColorSettingsPage Fields

1.1.2.9.2.1 BackgroundColorSettingsPage.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, a member of class BackgroundColorSettingsPage.

1.1.2.9.3 BackgroundColorSettingsPage Methods

1.1.2.9.3.1 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:     authorId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
8:     {
9:         var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()
as Author;
10:         lblGeneralSettings.Text = AppResources.GeneralSettings;
11:         lblBackgroundColor.Text = AppResources.SelectFontSize;
12:     }
13: }
```

1.1.2.9.3.2 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: }
```

1.1.2.9.3.3 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: }
```

1.1.2.10 Book Class

Class Hierarchy



C#

```
[Table]
public class Book;
```

File

Book.cs (see page 139)

Description

This is class AwesomeLibrary.Book.

Book Properties

	Name	Description
	BookAuthorName (see page 46)	This is BookAuthorName, a member of class Book.
	BookCategoryId (see page 47)	This is BookCategoryId, a member of class Book.
	BookCategoryName (see page 47)	This is BookCategoryName, a member of class Book.
	BookComment (see page 47)	This is BookComment, a member of class Book.
	BookGuid (see page 47)	This is BookGuid, a member of class Book.
	BookId (see page 47)	This is BookId, a member of class Book.
	BookName (see page 47)	This is BookName, a member of class Book.
	BookNameRating (see page 48)	[Column] public string BookInformation { get; set; }
	BookPageNumber (see page 48)	This is BookPageNumber, a member of class Book.
	BookPublisherName (see page 48)	This is BookPublisherName, a member of class Book.
	BookRating (see page 48)	This is BookRating, a member of class Book.
	CreationDate (see page 48)	This is CreationDate, a member of class Book.
	ModificationDate (see page 48)	This is ModificationDate, a member of class Book.
	ReadFinishDate (see page 49)	This is ReadFinishDate, a member of class Book.
	ReadStartDate (see page 49)	This is ReadStartDate, a member of class Book.

1.1.2.10.1 Book Properties

1.1.2.10.1.1 Book.BookAuthorName Property

C#

```
[Column]
public string BookAuthorName;
```

Description

This is BookAuthorName, a member of class Book.

1.1.2.10.1.2 Book.BookCategoryId Property**C#**

```
[Column]  
public int BookCategoryId;
```

Description

This is BookCategoryId, a member of class Book.

1.1.2.10.1.3 Book.BookCategoryName Property**C#**

```
[Column]  
public string BookCategoryName;
```

Description

This is BookCategoryName, a member of class Book.

1.1.2.10.1.4 Book.BookComment Property**C#**

```
[Column]  
public string BookComment;
```

Description

This is BookComment, a member of class Book.

1.1.2.10.1.5 Book.BookGuid Property**C#**

```
[Column]  
public string BookGuid;
```

Description

This is BookGuid, a member of class Book.

1.1.2.10.1.6 Book.BookId Property**C#**

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

Description

This is BookId, a member of class Book.

1.1.2.10.1.7 Book.BookName Property**C#**

```
[Column]  
public string BookName;
```

Description

This is BookName, a member of class Book.

1.1.2.10.1.8 Book.BookNameRating Property**C#**

```
[Column]  
public string BookNameRating;
```

Description

[Column] public string BookInformation { get; set; }

1.1.2.10.1.9 Book.BookPageNumber Property**C#**

```
[Column]  
public int BookPageNumber;
```

Description

This is BookPageNumber, a member of class Book.

1.1.2.10.1.10 Book.BookPublisherName Property**C#**

```
[Column]  
public string BookPublisherName;
```

Description

This is BookPublisherName, a member of class Book.

1.1.2.10.1.11 Book.BookRating Property**C#**

```
[Column]  
public int BookRating;
```

Description

This is BookRating, a member of class Book.

1.1.2.10.1.12 Book.CreationDate Property**C#**

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

Description

This is CreationDate, a member of class Book.

1.1.2.10.1.13 Book.ModificationDate Property**C#**

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

Description

This is ModificationDate, a member of class Book.

1.1.2.10.1.14 Book.ReadFinishDate Property

C#

```
[Column]
public DateTime ReadFinishDate;
```

Description

This is ReadFinishDate, a member of class Book.

1.1.2.10.1.15 Book.ReadStartDate Property

C#

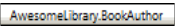
```
[Column]
public DateTime ReadStartDate;
```

Description

This is ReadStartDate, a member of class Book.

1.1.2.11 BookAuthor Class

Class Hierarchy



C#

```
[Table]
public class BookAuthor;
```

File

BookAuthor.cs (🔗 see page 140)

Description

This is class AwesomeLibrary.BookAuthor.

BookAuthor Properties

	Name	Description
🔗	AuthorId (🔗 see page 49)	This is AuthorId, a member of class BookAuthor.
🔗	BookAuthorId (🔗 see page 50)	This is BookAuthorId, a member of class BookAuthor.
🔗	BookId (🔗 see page 50)	This is BookId, a member of class BookAuthor.

1.1.2.11.1 BookAuthor Properties

1.1.2.11.1.1 BookAuthor.AuthorId Property

C#

```
[Column]
public int AuthorId;
```

Description

This is AuthorId, a member of class BookAuthor.

1.1.2.11.1.2 BookAuthor.BookAuthorId Property

C#

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

Description

This is BookAuthorId, a member of class BookAuthor.

1.1.2.11.1.3 BookAuthor.BookId Property

C#

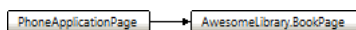
```
[Column]  
public int BookId;
```

Description

This is BookId, a member of class BookAuthor.

1.1.2.12 BookPage Class

Class Hierarchy



C#

```
public class BookPage : PhoneApplicationPage;
```

File

BookPage.xaml.cs (see page 141)

Description

This is class AwesomeLibrary.BookPage.

Methods

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

BookPage Fields

	Name	Description
	authorId (see page 52)	This is authorId, a member of class BookPage.
	authorName (see page 52)	This is authorName, a member of class BookPage.
	bookId (see page 52)	This is bookId, a member of class BookPage.
	categoryId (see page 52)	This is categoryId, a member of class BookPage.
	categoryName (see page 52)	This is categoryName, a member of class BookPage.
	flag (see page 52)	This is flag, a member of class BookPage.
	isFilled (see page 53)	This is isFilled, a member of class BookPage.
	pageName (see page 53)	This is pageName, a member of class BookPage.
	ratingValue (see page 53)	This is ratingValue, a member of class BookPage.

BookPage Methods

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

1.1.2.12.1 BookPage.BookPage Constructor**C#**

```
public BookPage();
```

Description

This is BookPage, a member of class BookPage.

Body Source

```

1: public BookPage()
2: {
3:     InitializeComponent();
4:
5:     SetBackgroundColor();
6:
7:     //pvAuthor.Title = authorName;
8:     piBookName.Header = AppResources.BookName;
9:     piComment.Header = AppResources.BookComment;
10:    piPublisherName.Header = AppResources.PublisherName;
11:    piRating.Header = AppResources.BookRating;
12:    piStartFinishDate.Header = AppResources.Date;
13:    lblStartDate.Text = AppResources.StartDate;
14:    lblFinishDate.Text = AppResources.FinishDate;
15:    piPageNumber.Header = AppResources.PageNumber;
16:
17:
18:    ApplicationBar = new ApplicationBar();
19:
20:    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
21:    button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
22:    button1.Text = AppResources.Save;
23:    ApplicationBar.Buttons.Add(button1);
24:    button1.Click += new EventHandler(SaveButton_Click);
25:
26:    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
27:    button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
28:    button2.Text = AppResources.SendWithMail;
29:    ApplicationBar.Buttons.Add(button2);
30:    button2.Click += new EventHandler(SendMailButton_Click);
31:
32:    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
33:    button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
34:    button3.Text = AppResources.SendWithSMS;
35:    ApplicationBar.Buttons.Add(button3);
36:    button3.Click += new EventHandler(SendSMSButton_Click);
37:
38:    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
39:    button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
40:    button4.Text = AppResources.ShareBook;
41:    ApplicationBar.Buttons.Add(button4);
42:    button4.Click += new EventHandler(ShareBookButton_Click);
43:
44:    isFilled = false;
45:
46:    ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
47:    menuItem1.Text = AppResources.DeleteBook;

```



```
48:     ApplicationBar.MenuItems.Add(menuItem1);
49:     menuItem1.Click += new EventHandler(DeleteBookMenuItem_Click);
50:
51: }
```

1.1.2.12.2 BookPage Fields

1.1.2.12.2.1 BookPage.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, a member of class BookPage.

1.1.2.12.2.2 BookPage.authorName Field

C#

```
public string authorName;
```

Description

This is authorName, a member of class BookPage.

1.1.2.12.2.3 BookPage.bookId Field

C#

```
public int bookId;
```

Description

This is bookId, a member of class BookPage.

1.1.2.12.2.4 BookPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class BookPage.

1.1.2.12.2.5 BookPage.categoryName Field

C#

```
public string categoryName;
```

Description

This is categoryName, a member of class BookPage.

1.1.2.12.2.6 BookPage.flag Field

C#

```
public bool flag;
```

Description

This is flag, a member of class BookPage.

1.1.2.12.2.7 BookPage.isFilled Field

C#

```
public bool isFilled;
```

Description

This is isFilled, a member of class BookPage.

1.1.2.12.2.8 BookPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class BookPage.

1.1.2.12.2.9 BookPage.ratingValue Field

C#

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

Description

This is ratingValue, a member of class BookPage.

1.1.2.12.3 BookPage Methods

1.1.2.12.3.1 BookPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class BookPage.

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:     bookId = int.Parse(e.Fragment);
7:     if (pageName.Contains("/AuthorPage.xaml"))
8:     {
9:         isFilled = true;
10:    }
11:    else
12:    {
13:        //using (var context2 = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
14:        //{
15:            //    var appSettings = context2.AppSettings; ;
16:            //    var book2 = context2.Books.Where(j => j.BookId.Equals(bookId)) as Book;
17:            //    var bookAuthor = context2.BookAuthors.Where(j =>
j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
18:            //    var bAuthor = bookAuthor.First();
19:            //    var author = context2.Authors.Where(j =>
j.AuthorId.Equals(bAuthor.AuthorId)) as Author;
20:            //    foreach (var item in appSettings)
```

```

21:         //      {
22:         //          item.CurrentAuthorNumber = author.AuthorId;
23:         //          item.CurrentCategoryNumber = book2.BookCategoryId;
24:         //      }
25:         //      context2.SubmitChanges();
26:         //      pvAuthor.Title = author.AuthorName;
27:         //  }
28:     }
29:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
30:     {
31:         var book = context.Books.Where(j => j.BookId.Equals(e.Fragment)).Single() as
Book;
32:
33:         txtBookName.Text = book.BookName == "" ? "" : book.BookName;
34:         txtPageNumber.Text = book.BookPageNumber.ToString() == "" ? "" :
book.BookPageNumber.ToString();
35:         txtPublisherName.Text = book.BookPublisherName == "" ? "" :
book.BookPublisherName;
36:         dtStart.Value = book.ReadStartDate == null ? DateTime.Now : book.ReadStartDate;
37:         dtFinish.Value = book.ReadFinishDate == null ? DateTime.Now :
book.ReadFinishDate;
38:         rtRating.Value = book.BookRating == null ? 0 : book.BookRating;
39:         txtBookComment.Text = book.BookComment == "" ? "" : book.BookComment;
40:     }
41:
42:     SetBackgroundColor();
43:     pvAuthor.SelectedIndex = 0;
44:     //pvAuthor.Name = authorName;
45:     txtBookName.Focus();
46: }

```

1.1.2.12.3.2 BookPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class BookPage.

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 BookPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class BookPage.

Body Source

```

1: protected override void OnNavigatedTo(NavigationEventArgs e)
2: {
3:     base.OnNavigatedTo(e);
4:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
5:     {
6:         var appSettings = context.AppSettings.First();

```

```

7:         categoryId = appSettings.CurrentCategoryNumber;
8:         authorId = appSettings.CurrentAuthorNumber;
9:
10:        // sayfanin font ayarlari için yapılan bir degisiklik
11:        FontFamily temp = new FontFamily(appSettings.FontFamily);
12:        double fontsize = double.Parse(appSettings.FontSize);
13:        txtBookComment.FontFamily = temp;
14:        txtBookComment.FontSize = fontsize;
15:        txtPageNumber.FontFamily = temp;
16:        txtPageNumber.FontSize = fontsize;
17:        txtBookName.FontFamily = temp;
18:        txtBookName.FontSize = fontsize;
19:        txtBookComment.FontFamily = temp;
20:        txtBookComment.FontSize = fontsize;
21:        txtPublisherName.FontFamily = temp;
22:        txtPublisherName.FontSize = fontsize;
23:        rtRating.Value = 5;
24:
25:        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()
as Author;
26:        authorName = author.AuthorName;
27:
28:        var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
29:        categoryName = category.CategoryName;
30:    }
31:
32:    var lastPage = NavigationService.BackStack.FirstOrDefault();
33:    pageName = lastPage.Source.ToString();
34:    pvAuthor.SelectedIndex = 0;
35:    txtBookName.Focus();
36:    // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
37:    pvAuthor.Title = authorName;
38:    SetBackgroundColor();
39: }

```

1.1.2.13 Category Class

Class Hierarchy

AwesomeLibrary.Category

C#

```

[Table]
public class Category;

```






File




Category.cs (see page 151)

Description

This is class AwesomeLibrary.Category.

Category Properties

	Name	Description
	AuthorOrderBy (see page 56)	This is AuthorOrderBy, a member of class Category.
	AuthorOrderStyle (see page 56)	This is AuthorOrderStyle, a member of class Category.
	CategoryBookCount (see page 56)	This is CategoryBookCount, 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 57)	This is CategoryNameCount, a member of class Category.
	CreationDate (🔗 see page 57)	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.AuthorOrderBy Property

C#

```
[Column]
public string AuthorOrderBy;
```

Description

This is AuthorOrderBy, a member of class Category.

1.1.2.13.1.2 Category.AuthorOrderStyle Property

C#

```
[Column]
public string AuthorOrderStyle;
```

Description

This is AuthorOrderStyle, a member of class Category.

1.1.2.13.1.3 Category.CategoryBookCount Property

C#

```
[Column]
public int CategoryBookCount;
```

Description

This is CategoryBookCount, 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 CategoryAuthor Class

Class Hierarchy



C#

```
[Table]
public class CategoryAuthor;
```

File

CategoryAuthor.cs (📄 see page 152)

Description

This is class AwesomeLibrary.CategoryAuthor.

CategoryAuthor Properties

	Name	Description
📄	AuthorId (📄 see page 58)	This is AuthorId, a member of class CategoryAuthor.
📄	CategoryAuthorId (📄 see page 58)	This is CategoryAuthorId, a member of class CategoryAuthor.
📄	CategoryId (📄 see page 58)	This is CategoryId, a member of class CategoryAuthor.

1.1.2.14.1 CategoryAuthor Properties

1.1.2.14.1.1 CategoryAuthor.AuthorId Property

C#

```
[Column]
public int AuthorId;
```

Description

This is AuthorId, a member of class CategoryAuthor.

1.1.2.14.1.2 CategoryAuthor.CategoryAuthorId Property

C#

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

Description

This is CategoryAuthorId, a member of class CategoryAuthor.

1.1.2.14.1.3 CategoryAuthor.CategoryId Property

C#

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

Description

This is CategoryId, a member of class CategoryAuthor.

1.1.2.15 CategoryPage Class

Class Hierarchy



C#

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

File

CategoryPage.xaml.cs (see page 153)

Description

This is class AwesomeLibrary.CategoryPage.

Methods

	Name	Description
	CategoryPage (see page 59)	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 60)	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 62)	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.AddAuthor;
10:    ApplicationBar.Buttons.Add(button1);
11:    button1.Click += new EventHandler(AddAuthorButton_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<Author> authors = new List<Author>();
4:     List<Author> authorsOrdered = new List<Author>();
5:
6:     // displays "Fragment: Detail"
7:     //MessageBox.Show("Folder Id: " + e.Fragment);
8:     base.OnFragmentNavigation(e);
9:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
10:    {
11:        var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
12:        string orderStyle = category.AuthorOrderStyle;
13:        var categoryAuthor = context.CategoryAuthors.Where(j =>
j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryAuthor>;
14:
15:        foreach (var item in categoryAuthor)
16:        {
17:            try
18:            {
19:                authors.Add(context.Authors.Where(j =>
j.AuthorId.Equals(item.AuthorId)).Single());
20:            }
21:            catch (Exception)
22:            {
23:            }
24:        }
25:    }
26:
27:    switch (category.AuthorOrderBy)
28:    {
29:        case "NAME":
30:            if (orderStyle == "A")
31:            {
32:                authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
33:            }
34:            else
35:            {
```

```

36:             authorsOrdered = authors.OrderByDescending(j =>
j.AuthorName).ToList();
37:         }
38:         break;
39:         case "BOOKCOUNT":
40:             if (orderStyle == "A")
41:             {
42:                 authorsOrdered = authors.OrderBy(j => j.AuthorBookCount).ToList();
43:             }
44:             else
45:             {
46:                 authorsOrdered = authors.OrderByDescending(j =>
j.AuthorBookCount).ToList();
47:             }
48:             break;
49:         case "CDATE":
50:             if (orderStyle == "A")
51:             {
52:                 authorsOrdered = authors.OrderBy(j => j.CreationDate).ToList();
53:             }
54:             else
55:             {
56:                 authorsOrdered = authors.OrderByDescending(j =>
j.CreationDate).ToList();
57:             }
58:             break;
59:         case "MDATE":
60:             if (orderStyle == "A")
61:             {
62:                 authorsOrdered = authors.OrderBy(j => j.ModificationDate).ToList();
63:             }
64:             else
65:             {
66:                 authorsOrdered = authors.OrderByDescending(j =>
j.ModificationDate).ToList();
67:             }
68:             break;
69:         default:
70:             if (orderStyle == "A")
71:             {
72:                 authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
73:             }
74:             else
75:             {
76:                 authorsOrdered = authors.OrderByDescending(j =>
j.AuthorName).ToList();
77:             }
78:             break;
79:     }
80:
81:     lstAuthors.Items.Clear();
82:     categoryId = category.CategoryId;
83:     lblCategoryName.Text = category.CategoryName;
84:     lblAuthorList.Text = AppResources.AuthorList + " (" + category.CategoryName +
")";
85:     lstAuthors.ItemsSource = authorsOrdered;
86:     lstAuthors.DisplayMemberPath = "AuthorNameCount";
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

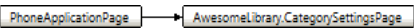
This is OnNavigatedTo, a member of class CategoryPage.

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 161)

Description

This is class AwesomeLibrary.CategorySettingsPage.

Methods



	Name	Description
🔗	CategorySettingsPage (🔗 see page 63)	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:     btnAuthorOrder.Content = AppResources.Select;
9:     btnAuthorOrderStyle.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
7:     {
8:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
9:         string orderStyle = category.AuthorOrderStyle;
```

```

10:         categoryId = category.CategoryId;
11:
12:         if (category.AuthorOrderBy == "NAME")
13:         {
14:             lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
15:         }
16:         if (category.AuthorOrderBy == "BOOKCOUNT")
17:         {
18:             lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.BookCount + ")";
19:         }
20:         if (category.AuthorOrderBy == "CDATE")
21:         {
22:             lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
23:         }
24:         if (category.AuthorOrderBy == "MDATE")
25:         {
26:             lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
27:         }
28:         if (category.AuthorOrderStyle == "A")
29:         {
30:             lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
31:         }
32:         if (category.AuthorOrderStyle == "D")
33:         {
34:             lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" +
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 164)

Description

This is class AwesomeLibrary.FontFamilySettingsPage.

Methods

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

FontFamilySettingsPage Fields

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

FontFamilySettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 66)	This is OnFragmentNavigation, a member of class FontFamilySettingsPage.
	OnNavigatedFrom (see page 67)	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("Bookman 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.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, 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:     authorId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
8:     {
9:         var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()
as Author;
10:         lblAuthorName.Text = author.AuthorName;
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 167)

Description

This is class AwesomeLibrary.FontSizeSettingsPage.

Methods



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

FontSizeSettingsPage Fields

	Name	Description
🔗	authorId (🔗 see page 68)	This is authorId, 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.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, 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:     authorId = int.Parse(e.Fragment);
7:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
8:     {
9:         var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()
as Author;
10:         lblAuthorName.Text = author.AuthorName;
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

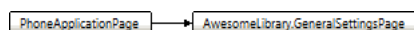
```

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 169)


Description

This is class AwesomeLibrary.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 75)	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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:
104:         if (appSettings.CategoryOrderBy == "NAME")
105:         {
106:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
107:         }
108:         if (appSettings.CategoryOrderBy == "CDATE")
109:         {
110:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
111:         }
112:         if (appSettings.CategoryOrderBy == "MDATE")
113:         {
114:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
115:         }
116:         if (appSettings.CategoryOrderBy == "BOOKCOUNT")
117:         {
118:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.BookCount + ")";
119:         }
120:         if (appSettings.CategoryOrderStyle == "A")
121:         {
122:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
123:         }
124:         if (appSettings.CategoryOrderStyle == "D")
125:         {
126:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Descending + ")";
127:         }
128:         if (appSettings.AppBackgroundColor == "BLA")
129:         {
130:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + ")";
131:         }
132:         if (appSettings.AppBackgroundColor == "BLU")
133:         {
134:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + ")";
135:         }
136:         if (appSettings.AppBackgroundColor == "BRO")
137:         {
138:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + ")";
139:         }
140:         if (appSettings.AppBackgroundColor == "RED")
141:         {
142:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + ")";
143:         }
144:         if (appSettings.AppBackgroundColor == "GRE")
145:         {
146:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +

```

```

AppResources.Selected + ": " + AppResources.Green + "));
147:     }
148:     if (appSettings.AppBackgroundColor == "YEL")
149:     {
150:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + "));
151:     }
152:     if (appSettings.AppBackgroundColor == "GRA")
153:     {
154:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + "));
155:     }
156:     if (appSettings.AppBackgroundColor == "ORA")
157:     {
158:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + "));
159:     }
160:     if (appSettings.AppBackgroundColor == "PUR")
161:     {
162:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + "));
163:     }
164: }
165: }

```

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

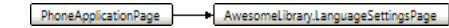
This is OnNavigatedTo, a member of class GeneralSettingsPage.

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 182)

Description

This is class AwesomeLibrary.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.Spanish);
18:    lstLanguage.Items.Add(AppResources.Sanskrit);
19:    lstLanguage.Items.Add(AppResources.Thai);
20:
21:    lstLanguage.SelectedIndex = -1;
22:    lblLanguage.Text = AppResources.SelectLanguage;
23:    lblGeneralSettings.Text = AppResources.GeneralSettings;
24:
25:    SetBackgroundColor();
26: }

```

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

```
AwesomeLibrary.LocalizedStrings
```


C#

```
public class LocalizedStrings;
```

File

LocalizedStrings.cs ([🔗](#) see page 186)

LocalizedStrings Properties

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

1.1.2.21.1 LocalizedStrings Properties

1.1.2.21.1.1 LocalizedStrings.LocalizedResources Property

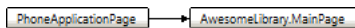
C#

```
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 186)


Description

This is class AwesomeLibrary.MainPage.



Methods

	Name	Description
	MainPage (🔗 see page 78)	Constructor

MainPage Fields

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

MainPage Methods

	Name	Description
	OnNavigatedFrom (🔗 see page 79)	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);
```

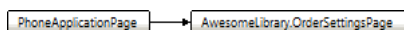
Description

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 192)




Description

This is class AwesomeLibrary.OrderSettingsPage.




Methods

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

OrderSettingsPage Fields

	Name	Description
	authorId (see page 80)	This is authorId, a member of class OrderSettingsPage.
	categoryId (see page 80)	This is categoryId, a member of class OrderSettingsPage.
	pageName (see page 81)	This is pageName, a member of class OrderSettingsPage.

OrderSettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 81)	This is OnFragmentNavigation, a member of class OrderSettingsPage.
	OnNavigatedFrom (see page 82)	This is OnNavigatedFrom, a member of class OrderSettingsPage.
	OnNavigatedTo (see page 82)	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.authorId Field

C#

```
public int authorId;
```

Description

This is authorId, 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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.AuthorOrderBy;
15:            lstOrderBy.Items.Add(AppResources.Name);
16:            lstOrderBy.Items.Add(AppResources.BookCount);
17:            lstOrderBy.Items.Add(AppResources.CreationDate);
18:            lstOrderBy.Items.Add(AppResources.ModificationDate);
19:        }
20:    }
21:    else
22:    {
23:        authorId = int.Parse(e.Fragment);
24:        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
25:        {
26:            var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
27:            lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName +
")" ;
28:            lblOrderBy.Text = AppResources.BookOrderBy;
29:            lstOrderBy.Items.Add(AppResources.Name);
30:            lstOrderBy.Items.Add(AppResources.CreationDate);
31:            lstOrderBy.Items.Add(AppResources.ModificationDate);
32:            lstOrderBy.Items.Add(AppResources.StartDate);
33:            lstOrderBy.Items.Add(AppResources.FinishDate);
34:            lstOrderBy.Items.Add(AppResources.BookRating);
35:        }
36:    }
37:    lstOrderBy.SelectedIndex = -1;
```

```
38:     SetBackgroundColor();
39: }
```

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

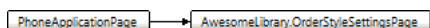
This is OnNavigatedTo, a member of class OrderSettingsPage.

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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.BookCount);
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 197)

Description

This is class AwesomeLibrary.OrderStyleSettingsPage.

Methods

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

OrderStyleSettingsPage Fields

	Name	Description
🔗	authorId (🔗 see page 83)	This is authorId, a member of class OrderStyleSettingsPage.
🔗	categoryId (🔗 see page 84)	This is categoryId, a member of class OrderStyleSettingsPage.
🔗	pageName (🔗 see page 84)	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 85)	This is OnNavigatedFrom, a member of class OrderStyleSettingsPage.
🔗	OnNavigatedTo (🔗 see page 85)	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.authorId Field****C#**

```
public int authorId;
```


Description

This is authorId, a member of class OrderStyleSettingsPage.

1.1.2.24.2 OrderStyleSettingsPage.categoryId Field**C#**

```
public int categoryId;
```

Description

This is categoryId, a member of class OrderStyleSettingsPage.

1.1.2.24.3 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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.AuthorOrderStyle;
14:         }
15:     }
16:     else
17:     {
18:         authorId = int.Parse(e.Fragment);
19:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
20:         {
21:             var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
22:             lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName +
")";
23:         }
24:     }
25: }
```

```

23:         lblOrderStyle.Text = AppResources.BookOrderStyle;
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

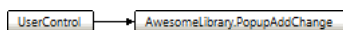
```

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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 201)

Description

This is class AwesomeLibrary.PopupAddChange.

Methods

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

1.1.2.25.1 PopupAddChange.PopupAddChange Constructor

C#

```
public PopupAddChange ( ) ;
```

Description

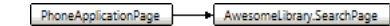
This is PopupAddChange, a member of class PopupAddChange.

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 202)



Description

This is class AwesomeLibrary.SearchPage.

Methods

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

SearchPage Methods

	Name	Description
	OnNavigatedFrom (🔗 see page 87)	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: }

```

1.1.2.26.2.2 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 AwesomeLibrary.StatisticsPage.

Methods

	Name	Description
	StatisticsPage (see page 88)	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 90)	This is file AboutPage.xaml.cs.
AddCategoryPage.xaml.cs (see page 92)	This is file AddCategoryPage.xaml.cs.
App.xaml.cs (see page 95)	This is file App.xaml.cs.
AppResources.Designer.cs (see page 101)	This code was generated by a tool. Runtime Version:4.0.30319.34209 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
AppSettings.cs (see page 121)	This is file AppSettings.cs.
AssemblyInfo.cs (see page 122)	This is file AssemblyInfo.cs.
Author.cs (see page 123)	This is file Author.cs.
AuthorPage.xaml.cs (see page 124)	This is file AuthorPage.xaml.cs.
AuthorSettingsPage.xaml.cs (see page 131)	This is file AuthorSettingsPage.xaml.cs.
AwesomeLibrary.csproj (see page 135)	This is file AwesomeLibrary.csproj.
AwesomeLibrary.sln (see page 135)	This is file AwesomeLibrary.sln.
AwesomeLibraryDataContext.cs (see page 135)	This is file AwesomeLibraryDataContext.cs.
BackgroundColorSettingsPage.xaml.cs (see page 136)	This is file BackgroundColorSettingsPage.xaml.cs.
Book.cs (see page 139)	This is file Book.cs.
BookAuthor.cs (see page 140)	This is file BookAuthor.cs.
BookPage.xaml.cs (see page 141)	This is file BookPage.xaml.cs.
Category.cs (see page 151)	This is file Category.cs.
CategoryAuthor.cs (see page 152)	This is file CategoryAuthor.cs.
CategoryPage.xaml.cs (see page 153)	This is file CategoryPage.xaml.cs.
CategorySettingsPage.xaml.cs (see page 161)	This is file CategorySettingsPage.xaml.cs.
FontFamilySettingsPage.xaml.cs (see page 164)	This is file FontFamilySettingsPage.xaml.cs.
FontSizeSettingsPage.xaml.cs (see page 167)	This is file FontSizeSettingsPage.xaml.cs.
GeneralSettingsPage.xaml.cs (see page 169)	This is file GeneralSettingsPage.xaml.cs.
LanguageSettingsPage.xaml.cs (see page 182)	This is file LanguageSettingsPage.xaml.cs.
LocalizedStrings.cs (see page 186)	This is file LocalizedStrings.cs.
MainPage.xaml.cs (see page 186)	This is file MainPage.xaml.cs.
OrderSettingsPage.xaml.cs (see page 192)	This is file OrderSettingsPage.xaml.cs.
OrderStyleSettingsPage.xaml.cs (see page 197)	This is file OrderStyleSettingsPage.xaml.cs.
PopupAddChange.xaml.cs (see page 201)	This is file PopupAddChange.xaml.cs.
SearchPage.xaml.cs (see page 202)	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 AwesomeLibrary.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeLibrary
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.AboutTheAwesomeLibrary;
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.2 AddCategoryPage.xaml.cs

1

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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
19: {
20:     public partial class AddCategoryPage : PhoneApplicationPage
21:     {
22:
23:         public int authorId;
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:         authorId = int.Parse(e.Fragment);
47:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
48:         {
49:             var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
50:             lstCategories.Items.Clear();
51:             lblAuthorName.Text = author.AuthorName;
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("/AuthorPage.xaml#" +
authorId, UriKind.Relative));
64:         }
65:     }
66:
67:     private void SetBackgroundColor()
68:     {
69:         AppSettings appSettings = new AppSettings();
70:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:     CategoryAuthor categoryAuthor2 = null;
125:     Category category = lstCategories.SelectedItem as Category;
126:     CategoryAuthor categoryAuthor = new CategoryAuthor();
127:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
128:     {
129:         categoryAuthor.AuthorId = authorId;
130:         categoryAuthor.CategoryId = category.CategoryId;
131:         try
132:         {
133:             categoryAuthor2 = context.CategoryAuthors.Where(j =>
j.CategoryId.Equals(categoryAuthor.CategoryId) &&
j.AuthorId.Equals(categoryAuthor.AuthorId)).Single() as CategoryAuthor;
134:         }
135:         catch (Exception)
136:         {
137:             context.CategoryAuthors.InsertOnSubmit(categoryAuthor);
138:             context.SubmitChanges();
139:             var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
140:             foreach (var item in author)
141:             {
142:                 item.ModificationDate = DateTime.Now;
143:             }
144:             context.SubmitChanges();
145:             MessageBox.Show(AppResources.AuthorCategoryAddSuccess);
146:         }
147:         if (categoryAuthor2 != null)
148:         {
149:             MessageBox.Show(AppResources.AuthorAlreadySameCategory);
150:         }
151:         else
152:         {

```

```

153:
154:         }
155:     }
156:     this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId,
UriKind.Relative));
157: }
158: }
159: }

```

Namespaces

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

1.2.3 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 AwesomeLibrary.Resources;
13: using System.Collections.Generic;
14: using Microsoft.Phone.Marketplace;
15:
16:
17: namespace AwesomeLibrary
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:
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:         }
94:
95:         // Code to execute when the application is activated (brought to foreground)
96:         // This code will not execute when the application is first launched
97:         private void Application_Activated(object sender, ActivatedEventArgs e)
98:         {
99:         }
100:
101:         // Code to execute when the application is deactivated (sent to background)
102:         // This code will not execute when the application is closing
103:         private void Application_Deactivated(object sender, DeactivatedEventArgs e)
104:         {
105:         }
106:
107:         // Code to execute when the application is closing (eg, user hit Back)
108:         // This code will not execute when the application is deactivated
109:         private void Application_Closing(object sender, ClosingEventArgs e)
110:         {

```

```

111:         }
112:
113:         // Code to execute if a navigation fails
114:         private void RootFrame_NavigationFailed(object sender,
NavigationFailedEventArgs e)
115:         {
116:             if (Debugger.IsAttached)
117:             {
118:                 // A navigation has failed; break into the debugger
119:                 Debugger.Break();
120:             }
121:         }
122:
123:         // Code to execute on Unhandled Exceptions
124:         private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
125:         {
126:             if (Debugger.IsAttached)
127:             {
128:                 // An unhandled exception has occurred; break into the debugger
129:                 Debugger.Break();
130:             }
131:         }
132:
133:         #region Phone application initialization
134:
135:         // Avoid double-initialization
136:         private bool phoneApplicationInitialized = false;
137:
138:         // Do not add any additional code to this method
139:         private void InitializePhoneApplication()
140:         {
141:             if (phoneApplicationInitialized)
142:                 return;
143:
144:             // Create the frame but don't set it as RootVisual yet; this allows the
splash
145:             // screen to remain active until the application is ready to render.
146:             RootFrame = new PhoneApplicationFrame();
147:             RootFrame.Navigated += CompleteInitializePhoneApplication;
148:
149:             // Handle navigation failures
150:             RootFrame.NavigationFailed += RootFrame_NavigationFailed;
151:
152:             // Handle reset requests for clearing the backstack
153:             RootFrame.Navigated += CheckForResetNavigation;
154:
155:             // Ensure we don't initialize again
156:             phoneApplicationInitialized = true;
157:         }
158:
159:         // Do not add any additional code to this method
160:         private void CompleteInitializePhoneApplication(object sender,
NavigationEventArgs e)
161:         {
162:             // Set the root visual to allow the application to render
163:             if (RootVisual != RootFrame)
164:                 RootVisual = RootFrame;
165:
166:             // Remove this handler since it is no longer needed
167:             RootFrame.Navigated -= CompleteInitializePhoneApplication;
168:         }
169:
170:         private void CheckForResetNavigation(object sender, NavigationEventArgs e)
171:         {
172:             // If the app has received a 'reset' navigation, then we need to check
// on the next navigation to see if the page stack should be reset
173:             if (e.NavigationMode == NavigationMode.Reset)
174:                 RootFrame.Navigated += ClearBackStackAfterReset;
175:

```

```

176:     }
177:
178:     private void ClearBackStackAfterReset(object sender, NavigationEventArgs e)
179:     {
180:         // Unregister the event so it doesn't get called again
181:         RootFrame.Navigated -= ClearBackStackAfterReset;
182:
183:         // Only clear the stack for 'new' (forward) and 'refresh' navigations
184:         if (e.NavigationMode != NavigationMode.New && e.NavigationMode !=
NavigationMode.Refresh)
185:             return;
186:
187:         // For UI consistency, clear the entire page stack
188:         while (RootFrame.RemoveBackEntry() != null)
189:         {
190:             ; // do nothing
191:         }
192:     }
193:
194:     #endregion
195:
196:     // Initialize the app's font and flow direction as defined in its localized
resource strings.
197:     //
198:     // To ensure that the font of your application is aligned with its supported
languages and that the
199:     // FlowDirection for each of those languages follows its traditional
direction, ResourceLanguage
200:     // and ResourceFlowDirection should be initialized in each resx file to match
these values with that
201:     // file's culture. For example:
202:     //
203:     // AppResources.es-ES.resx
204:     //     ResourceLanguage's value should be "es-ES"
205:     //     ResourceFlowDirection's value should be "LeftToRight"
206:     //
207:     // AppResources.ar-SA.resx
208:     //     ResourceLanguage's value should be "ar-SA"
209:     //     ResourceFlowDirection's value should be "RightToLeft"
210:     //
211:     // For more info on localizing Windows Phone apps see
http://go.microsoft.com/fwlink/?LinkId=262072.
212:     //
213:     private void InitializeLanguage()
214:     {
215:         try
216:         {
217:             // Set the font to match the display language defined by the
218:             // ResourceLanguage resource string for each supported language.
219:             //
220:             // Fall back to the font of the neutral language if the Display
221:             // language of the phone is not supported.
222:             //
223:             // If a compiler error is hit then ResourceLanguage is missing from
224:             // the resource file.
225:             RootFrame.Language =
XmlLanguage.GetLanguage(AppResources.ResourceLanguage);
226:
227:             // Set the FlowDirection of all elements under the root frame based
228:             // on the ResourceFlowDirection resource string for each
229:             // supported language.
230:             //
231:             // If a compiler error is hit then ResourceFlowDirection is missing
from
232:             // the resource file.
233:             FlowDirection flow = (FlowDirection)Enum.Parse(typeof(FlowDirection),
AppResources.ResourceFlowDirection);
234:             RootFrame.FlowDirection = flow;
235:         }

```

```

236:         catch
237:         {
238:             // If an exception is caught here it is most likely due to either
239:             // ResourceLanguage not being correctly set to a supported language
240:             // code or ResourceFlowDirection is set to a value other than
LeftToRight
241:             // or RightToLeft.
242:
243:             if (Debugger.IsAttached)
244:             {
245:                 Debugger.Break();
246:             }
247:
248:             throw;
249:         }
250:     }
251:
252:     private void CheckLicense()
253:     {
254:         // When debugging, we want to simulate a trial mode experience. The
following conditional allows us to set the _isTrial
255:         // property to simulate trial mode being on or off.
256:         #if DEBUG
257:             string message = "This sample demonstrates the implementation of a trial
mode in an application." +
258:                 "Press 'OK' to simulate trial mode. Press 'Cancel' to
run the application in normal mode.";
259:             if (MessageBox.Show(message, "Debug Trial",
260:                 MessageBoxButton.OKCancel) == MessageBoxResult.OK)
261:             {
262:                 _isTrial = true;
263:             }
264:             else
265:             {
266:                 _isTrial = false;
267:             }
268:         #else
269:             _isTrial = _licenseInfo.IsTrial();
270:         #endif
271:     }
272:
273:     private void Application_Startup(object sender, StartupEventArgs e)
274:     {
275:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
276:         {
277:             if (!context.DatabaseExists())
278:             {
279:                 context.CreateDatabase();
280:                 DilAyariOlustur(context);
281:             }
282:             else
283:             {
284:                 using (var context2 = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
285:                 {
286:
287:                     AppSettings lang =
288:                         context2.AppSettings.First() as AppSettings;
289:                     string culture = "";
290:                     switch (lang.AppLangName)
291:                     {
292:                         case "TR":
293:                             culture = "tr";
294:                             break;
295:                         case "EN":
296:                             culture = "en";
297:                             break;
298:                         case "DE":

```



```

299:             culture = "de";
300:             break;
301:         case "ES":
302:             culture = "es";
303:             break;
304:         case "FR":
305:             culture = "fr";
306:             break;
307:         case "IT":
308:             culture = "it";
309:             break;
310:         case "AR":
311:             culture = "ar";
312:             break;
313:         case "FA":
314:             culture = "fa-IR";
315:             break;
316:         case "ZH":
317:             culture = "zh";
318:             break;
319:         case "PT":
320:             culture = "pt";
321:             break;
322:         case "RU":
323:             culture = "ru";
324:             break;
325:         case "JA":
326:             culture = "ja";
327:             break;
328:         case "SA":
329:             culture = "sa";
330:             break;
331:         case "TH":
332:             culture = "th";
333:             break;
334:         default:
335:             culture = "tr-TR";
336:             break;
337:     }
338:     CultureInfo newCulture = new CultureInfo(culture);
339:     Thread.CurrentThread.CurrentCulture = newCulture;
340:     Thread.CurrentThread.CurrentUICulture = newCulture;
341:     }
342: }
343:
344:     // kullanıcının programla ilgili bilgilendirici notları kendi dilinde
345:     // görebilmesi için burada ekliyoruz.
346: }
347:
348: private static void DilAyariOlustur(AwesomeLibraryDataContext context)
349: {
350:     var appSettings = new AppSettings()
351:     {
352:         //AppLangId = 42,
353:         AppLangName = "EN",
354:         AppBackgroundColor = "BLA",
355:         CategoryOrderBy = "NAME",
356:         CategoryOrderStyle = "A",
357:         CurrentCategoryNumber = 0,
358:         CurrentAuthorNumber = 0,
359:         FontFamily = "Verdana",
360:         FontSize = "30",
361:         AppBackgroundImage = null
362:     };
363:
364:     context.AppSettings.InsertOnSubmit(appSettings);
365:     context.SubmitChanges();
366: }

```

```
367:         CultureInfo newCulture = new CultureInfo("en");
368:         Thread.CurrentThread.CurrentCulture = newCulture;
369:         Thread.CurrentThread.CurrentUICulture = newCulture;
370:     }
371: }
372: }
```

Namespaces

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

1.2.4 AppResources.Designer.cs

This code was generated by a tool. Runtime Version:4.0.30319.34209

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:4.0.30319.34209
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 AwesomeLibrary.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>
38:             ///     Returns the cached ResourceManager instance used by this class.
39:             /// </summary>
40:
41: [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("AwesomeLibrary.Resources.AppResources",
typeof(AppResources).Assembly);
43:                     resourceMan = temp;
44:                 }
45:                 return resourceMan;
46:             }
47:         }
48:
49:         /// <summary>
50:         /// Overrides the current thread's CurrentUICulture property for all
51:         /// resource lookups using this strongly typed resource class.
52:         /// </summary>
53:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
54:         public static global::System.Globalization.CultureInfo Culture {
55:             get {
56:                 return resourceCulture;
57:             }
58:             set {
59:                 resourceCulture = value;
60:             }
61:         }
62:
63:         /// <summary>
64:         /// Looks up a localized string similar to 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 reading book a lot.
And after Awesome Note app, i decide to create an app which has similar properties like
GoodReads. You can add categories, add authors and add books on it. You can send
information of your books 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 Library.
92:         /// </summary>
93:         public static string AboutTheAwesomeLibrary {
94:             get {
95:                 return ResourceManager.GetString("AboutTheAwesomeLibrary",
resourceCulture);
96:             }

```

```

97:         }
98:
99:         /// <summary>
100:        ///     Looks up a localized string similar to Add Author.
101:        /// </summary>
102:        public static string AddAuthor {
103:            get {
104:                return ResourceManager.GetString("AddAuthor", resourceCulture);
105:            }
106:        }
107:
108:        /// <summary>
109:        ///     Looks up a localized string similar to Add Book.
110:        /// </summary>
111:        public static string AddBook {
112:            get {
113:                return ResourceManager.GetString("AddBook", 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 Arabic.
128:        /// </summary>
129:        public static string Arabic {
130:            get {
131:                return ResourceManager.GetString("Arabic", resourceCulture);
132:            }
133:        }
134:
135:        /// <summary>
136:        ///     Looks up a localized string similar to Ascending.
137:        /// </summary>
138:        public static string Ascending {
139:            get {
140:                return ResourceManager.GetString("Ascending", resourceCulture);
141:            }
142:        }
143:
144:        /// <summary>
145:        ///     Looks up a localized string similar to Author Has Been Added
146:        Successfully.
147:        /// </summary>
148:        public static string AuthorAddSuccess {
149:            get {
150:                return ResourceManager.GetString("AuthorAddSuccess", resourceCulture);
151:            }
152:        }
153:
154:        /// <summary>
155:        ///     Looks up a localized string similar to Author Has This Category Already.
156:        /// </summary>
157:        public static string AuthorAlreadySameCategory {
158:            get {
159:                return ResourceManager.GetString("AuthorAlreadySameCategory",
160:                resourceCulture);
161:            }
162:        }
163:
164:        /// <summary>
165:        ///     Looks up a localized string similar to Category Has Been Added To

```

```

Author Successfully.
164:         /// </summary>
165:         public static string AuthorCategoryAddSuccess {
166:             get {
167:                 return ResourceManager.GetString("AuthorCategoryAddSuccess",
resourceCulture);
168:             }
169:         }
170:
171:         /// <summary>
172:         /// Looks up a localized string similar to Author Has Been Removed
Successfully.
173:         /// </summary>
174:         public static string AuthorDeleteSuccess {
175:             get {
176:                 return ResourceManager.GetString("AuthorDeleteSuccess",
resourceCulture);
177:             }
178:         }
179:
180:         /// <summary>
181:         /// Looks up a localized string similar to This Author Has Already Exists.
182:         /// </summary>
183:         public static string AuthorExists {
184:             get {
185:                 return ResourceManager.GetString("AuthorExists", resourceCulture);
186:             }
187:         }
188:
189:         /// <summary>
190:         /// Looks up a localized string similar to Author List.
191:         /// </summary>
192:         public static string AuthorList {
193:             get {
194:                 return ResourceManager.GetString("AuthorList", resourceCulture);
195:             }
196:         }
197:
198:         /// <summary>
199:         /// Looks up a localized string similar to Author Name.
200:         /// </summary>
201:         public static string AuthorName {
202:             get {
203:                 return ResourceManager.GetString("AuthorName", resourceCulture);
204:             }
205:         }
206:
207:         /// <summary>
208:         /// Looks up a localized string similar to Author Name Has Been Changed
Successfully.
209:         /// </summary>
210:         public static string AuthorNameChangeSuccess {
211:             get {
212:                 return ResourceManager.GetString("AuthorNameChangeSuccess",
resourceCulture);
213:             }
214:         }
215:
216:         /// <summary>
217:         /// Looks up a localized string similar to Author Order By.
218:         /// </summary>
219:         public static string AuthorOrderBy {
220:             get {
221:                 return ResourceManager.GetString("AuthorOrderBy", resourceCulture);
222:             }
223:         }
224:
225:         /// <summary>
226:         /// Looks up a localized string similar to Author Order Style.

```

```
227:         /// </summary>
228:         public static string AuthorOrderStyle {
229:             get {
230:                 return ResourceManager.GetString("AuthorOrderStyle", resourceCulture);
231:             }
232:         }
233:
234:         /// <summary>
235:         /// Looks up a localized string similar to Author Order Style Has Been
236:         /// </summary>
237:         public static string AuthorOrderStyleChangeSuccess {
238:             get {
239:                 return ResourceManager.GetString("AuthorOrderStyleChangeSuccess",
240: resourceCulture);
241:             }
242:         }
243:         /// <summary>
244:         /// Looks up a localized string similar to Author Order Type Has Been
245:         /// </summary>
246:         public static string AuthorOrderTypeChangeSuccess {
247:             get {
248:                 return ResourceManager.GetString("AuthorOrderTypeChangeSuccess",
249: resourceCulture);
250:             }
251:         }
252:         /// <summary>
253:         /// Looks up a localized string similar to Author Settings.
254:         /// </summary>
255:         public static string AuthorSettings {
256:             get {
257:                 return ResourceManager.GetString("AuthorSettings", resourceCulture);
258:             }
259:         }
260:
261:         /// <summary>
262:         /// Looks up a localized string similar to Background.
263:         /// </summary>
264:         public static string Background {
265:             get {
266:                 return ResourceManager.GetString("Background", resourceCulture);
267:             }
268:         }
269:
270:         /// <summary>
271:         /// Looks up a localized string similar to Background Color.
272:         /// </summary>
273:         public static string BackgroundColor {
274:             get {
275:                 return ResourceManager.GetString("BackgroundColor", resourceCulture);
276:             }
277:         }
278:
279:         /// <summary>
280:         /// Looks up a localized string similar to Background Color Has Been
281:         /// </summary>
282:         public static string BackgroundColorChangeSuccess {
283:             get {
284:                 return ResourceManager.GetString("BackgroundColorChangeSuccess",
285: resourceCulture);
286:             }
287:         }
288:         /// <summary>
289:         /// Looks up a localized string similar to Background Image.
```

```

290:         /// </summary>
291:         public static string BackgroundImage {
292:             get {
293:                 return ResourceManager.GetString("BackgroundImage", resourceCulture);
294:             }
295:         }
296:
297:         /// <summary>
298:         /// Looks up a localized string similar to Background Image Has Been
Changed Successfully.
299:         /// </summary>
300:         public static string BackgroundImageChangeSuccess {
301:             get {
302:                 return ResourceManager.GetString("BackgroundImageChangeSuccess",
resourceCulture);
303:             }
304:         }
305:
306:         /// <summary>
307:         /// Looks up a localized string similar to Background Image Has Been
Removed Successfully.
308:         /// </summary>
309:         public static string BackgroundImageRemoveSuccess {
310:             get {
311:                 return ResourceManager.GetString("BackgroundImageRemoveSuccess",
resourceCulture);
312:             }
313:         }
314:
315:         /// <summary>
316:         /// Looks up a localized string similar to Best Book.
317:         /// </summary>
318:         public static string BestBook {
319:             get {
320:                 return ResourceManager.GetString("BestBook", resourceCulture);
321:             }
322:         }
323:
324:         /// <summary>
325:         /// Looks up a localized string similar to Black.
326:         /// </summary>
327:         public static string Black {
328:             get {
329:                 return ResourceManager.GetString("Black", resourceCulture);
330:             }
331:         }
332:
333:         /// <summary>
334:         /// Looks up a localized string similar to Blue.
335:         /// </summary>
336:         public static string Blue {
337:             get {
338:                 return ResourceManager.GetString("Blue", resourceCulture);
339:             }
340:         }
341:
342:         /// <summary>
343:         /// Looks up a localized string similar to Book Comment.
344:         /// </summary>
345:         public static string BookComment {
346:             get {
347:                 return ResourceManager.GetString("BookComment", resourceCulture);
348:             }
349:         }
350:
351:         /// <summary>
352:         /// Looks up a localized string similar to Book Count.
353:         /// </summary>
354:         public static string BookCount {

```

```

355:         get {
356:             return ResourceManager.GetString("BookCount", resourceCulture);
357:         }
358:     }
359:
360:     /// <summary>
361:     /// Looks up a localized string similar to Book Has Been Removed
Successfully.
362:     /// </summary>
363:     public static string BookDeleteSuccess {
364:         get {
365:             return ResourceManager.GetString("BookDeleteSuccess",
resourceCulture);
366:         }
367:     }
368:
369:     /// <summary>
370:     /// Looks up a localized string similar to Book List.
371:     /// </summary>
372:     public static string BookList {
373:         get {
374:             return ResourceManager.GetString("BookList", resourceCulture);
375:         }
376:     }
377:
378:     /// <summary>
379:     /// Looks up a localized string similar to Book Name.
380:     /// </summary>
381:     public static string BookName {
382:         get {
383:             return ResourceManager.GetString("BookName", resourceCulture);
384:         }
385:     }
386:
387:     /// <summary>
388:     /// Looks up a localized string similar to You Have To Enter Book Name At
Least.
389:     /// </summary>
390:     public static string BookNameMustBe {
391:         get {
392:             return ResourceManager.GetString("BookNameMustBe", resourceCulture);
393:         }
394:     }
395:
396:     /// <summary>
397:     /// Looks up a localized string similar to Book Order By.
398:     /// </summary>
399:     public static string BookOrderBy {
400:         get {
401:             return ResourceManager.GetString("BookOrderBy", resourceCulture);
402:         }
403:     }
404:
405:     /// <summary>
406:     /// Looks up a localized string similar to Book Order Style.
407:     /// </summary>
408:     public static string BookOrderStyle {
409:         get {
410:             return ResourceManager.GetString("BookOrderStyle", resourceCulture);
411:         }
412:     }
413:
414:     /// <summary>
415:     /// Looks up a localized string similar to Book Order Style Has Been
Changed Successfully.
416:     /// </summary>
417:     public static string BookOrderStyleChangeSuccess {
418:         get {
419:             return ResourceManager.GetString("BookOrderStyleChangeSuccess",

```



```
resourceCulture);
420:         }
421:     }
422:
423:     /// <summary>
424:     /// Looks up a localized string similar to Book Order Type Has Been Changed
Successfully.
425:     /// </summary>
426:     public static string BookOrderTypeChangeSuccess {
427:         get {
428:             return ResourceManager.GetString("BookOrderTypeChangeSuccess",
resourceCulture);
429:         }
430:     }
431:
432:     /// <summary>
433:     /// Looks up a localized string similar to Average Book Read Per Day.
434:     /// </summary>
435:     public static string BookPerDay {
436:         get {
437:             return ResourceManager.GetString("BookPerDay", resourceCulture);
438:         }
439:     }
440:
441:     /// <summary>
442:     /// Looks up a localized string similar to Book Rating.
443:     /// </summary>
444:     public static string BookRating {
445:         get {
446:             return ResourceManager.GetString("BookRating", resourceCulture);
447:         }
448:     }
449:
450:     /// <summary>
451:     /// Looks up a localized string similar to Book Has Been Saved Successfully.
452:     /// </summary>
453:     public static string BookSaveSuccess {
454:         get {
455:             return ResourceManager.GetString("BookSaveSuccess", 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 Author.
614:         /// </summary>
615:         public static string DeleteAuthor {
616:             get {
617:                 return ResourceManager.GetString("DeleteAuthor", resourceCulture);
618:             }
619:         }
620:
621:         /// <summary>
622:         /// Looks up a localized string similar to You Will Delete The Author With
All Contents. Are You Sure?.
623:         /// </summary>
624:         public static string DeleteAuthorQuestion {
625:             get {
626:                 return ResourceManager.GetString("DeleteAuthorQuestion",
resourceCulture);
627:             }
628:         }
629:
630:         /// <summary>
631:         /// Looks up a localized string similar to Delete Book.
632:         /// </summary>
633:         public static string DeleteBook {
634:             get {
635:                 return ResourceManager.GetString("DeleteBook", resourceCulture);
636:             }
637:         }
638:
639:         /// <summary>
640:         /// Looks up a localized string similar to You Will Delete This Book. Are
You Sure?.
641:         /// </summary>
642:         public static string DeleteBookQuestion {
643:             get {
644:                 return ResourceManager.GetString("DeleteBookQuestion",
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 Author Name.
686:         /// </summary>
687:         public static string EnterAuthorName {
688:             get {
689:                 return ResourceManager.GetString("EnterAuthorName", 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 Language.
839:     /// </summary>
840:     public static string Language {
841:         get {
842:             return ResourceManager.GetString("Language", resourceCulture);
843:         }
844:     }
845:
846:     /// <summary>
847:     ///     Looks up a localized string similar to You May Restart Application To
Change Affect.
848:     /// </summary>
849:     public static string LanguageWarning {
850:         get {
851:             return ResourceManager.GetString("LanguageWarning", resourceCulture);
852:         }
853:     }
854:
855:     /// <summary>
856:     ///     Looks up a localized string similar to Last Modification Date.
857:     /// </summary>
858:     public static string ModificationDate {
859:         get {
860:             return ResourceManager.GetString("ModificationDate", resourceCulture);
861:         }
862:     }
863:
864:     /// <summary>
865:     ///     Looks up a localized string similar to Author You Most Read.
866:     /// </summary>
867:     public static string MostReadAuthor {
868:         get {
869:             return ResourceManager.GetString("MostReadAuthor", resourceCulture);
870:         }
871:     }
872:
873:     /// <summary>
874:     ///     Looks up a localized string similar to Category You Most Read.
875:     /// </summary>
876:     public static string MostReadCategory {
```

```
877:         get {
878:             return ResourceManager.GetString("MostReadCategory", resourceCulture);
879:         }
880:     }
881:
882:     /// <summary>
883:     /// Looks up a localized string similar to Publisher You Most Read.
884:     /// </summary>
885:     public static string MostReadPublisher {
886:         get {
887:             return ResourceManager.GetString("MostReadPublisher",
resourceCulture);
888:         }
889:     }
890:
891:     /// <summary>
892:     /// Looks up a localized string similar to Name.
893:     /// </summary>
894:     public static string Name {
895:         get {
896:             return ResourceManager.GetString("Name", resourceCulture);
897:         }
898:     }
899:
900:     /// <summary>
901:     /// Looks up a localized string similar to Ok.
902:     /// </summary>
903:     public static string OK {
904:         get {
905:             return ResourceManager.GetString("OK", resourceCulture);
906:         }
907:     }
908:
909:     /// <summary>
910:     /// Looks up a localized string similar to OneDrive.
911:     /// </summary>
912:     public static string OneDrive {
913:         get {
914:             return ResourceManager.GetString("OneDrive", resourceCulture);
915:         }
916:     }
917:
918:     /// <summary>
919:     /// Looks up a localized string similar to OneDrive Sync Completed.
920:     /// </summary>
921:     public static string OneDriveSyncCompleted {
922:         get {
923:             return ResourceManager.GetString("OneDriveSyncCompleted",
resourceCulture);
924:         }
925:     }
926:
927:     /// <summary>
928:     /// Looks up a localized string similar to Orange.
929:     /// </summary>
930:     public static string Orange {
931:         get {
932:             return ResourceManager.GetString("Orange", resourceCulture);
933:         }
934:     }
935:
936:     /// <summary>
937:     /// Looks up a localized string similar to Other Settings.
938:     /// </summary>
939:     public static string OtherSettings {
940:         get {
941:             return ResourceManager.GetString("OtherSettings", resourceCulture);
942:         }
943:     }
```



```
944:
945:     /// <summary>
946:     /// Looks up a localized string similar to Page Number.
947:     /// </summary>
948:     public static string PageNumber {
949:         get {
950:             return ResourceManager.GetString("PageNumber", resourceCulture);
951:         }
952:     }
953:
954:     /// <summary>
955:     /// Looks up a localized string similar to Average Page Read Per Day.
956:     /// </summary>
957:     public static string PagePerDay {
958:         get {
959:             return ResourceManager.GetString("PagePerDay", resourceCulture);
960:         }
961:     }
962:
963:     /// <summary>
964:     /// Looks up a localized string similar to Persian.
965:     /// </summary>
966:     public static string Persian {
967:         get {
968:             return ResourceManager.GetString("Persian", resourceCulture);
969:         }
970:     }
971:
972:     /// <summary>
973:     /// Looks up a localized string similar to Portuguese.
974:     /// </summary>
975:     public static string Portuguese {
976:         get {
977:             return ResourceManager.GetString("Portuguese", resourceCulture);
978:         }
979:     }
980:
981:     /// <summary>
982:     /// Looks up a localized string similar to Publisher Name.
983:     /// </summary>
984:     public static string PublisherName {
985:         get {
986:             return ResourceManager.GetString("PublisherName", resourceCulture);
987:         }
988:     }
989:
990:     /// <summary>
991:     /// Looks up a localized string similar to Purple.
992:     /// </summary>
993:     public static string Purple {
994:         get {
995:             return ResourceManager.GetString("Purple", resourceCulture);
996:         }
997:     }
998:
999:     /// <summary>
1000:    /// Looks up a localized string similar to Rate.
1001:    /// </summary>
1002:    public static string Rate {
1003:        get {
1004:            return ResourceManager.GetString("Rate", resourceCulture);
1005:        }
1006:    }
1007:
1008:    /// <summary>
1009:    /// Looks up a localized string similar to Red.
1010:    /// </summary>
1011:    public static string Red {
1012:        get {
```

```
1013:         return ResourceManager.GetString("Red", resourceCulture);
1014:     }
1015: }
1016:
1017: /// <summary>
1018: /// Looks up a localized string similar to Remove Background Image.
1019: /// </summary>
1020: public static string RemoveBackgroundImage {
1021:     get {
1022:         return ResourceManager.GetString("RemoveBackgroundImage",
resourceCulture);
1023:     }
1024: }
1025:
1026: /// <summary>
1027: /// Looks up a localized string similar to Reset Settings.
1028: /// </summary>
1029: public static string ResetSettings {
1030:     get {
1031:         return ResourceManager.GetString("ResetSettings", resourceCulture);
1032:     }
1033: }
1034:
1035: /// <summary>
1036: /// Looks up a localized string similar to LeftToRight.
1037: /// </summary>
1038: public static string ResourceFlowDirection {
1039:     get {
1040:         return ResourceManager.GetString("ResourceFlowDirection",
resourceCulture);
1041:     }
1042: }
1043:
1044: /// <summary>
1045: /// Looks up a localized string similar to en-US.
1046: /// </summary>
1047: public static string ResourceLanguage {
1048:     get {
1049:         return ResourceManager.GetString("ResourceLanguage", resourceCulture);
1050:     }
1051: }
1052:
1053: /// <summary>
1054: /// Looks up a localized string similar to Russian.
1055: /// </summary>
1056: public static string Russian {
1057:     get {
1058:         return ResourceManager.GetString("Russian", resourceCulture);
1059:     }
1060: }
1061:
1062: /// <summary>
1063: /// Looks up a localized string similar to Sanskrit.
1064: /// </summary>
1065: public static string Sanskrit {
1066:     get {
1067:         return ResourceManager.GetString("Sanskrit", resourceCulture);
1068:     }
1069: }
1070:
1071: /// <summary>
1072: /// Looks up a localized string similar to Save.
1073: /// </summary>
1074: public static string Save {
1075:     get {
1076:         return ResourceManager.GetString("Save", resourceCulture);
1077:     }
1078: }
1079:
```

```
1080:         /// <summary>
1081:         /// Looks up a localized string similar to Search.
1082:         /// </summary>
1083:         public static string Search {
1084:             get {
1085:                 return ResourceManager.GetString("Search", resourceCulture);
1086:             }
1087:         }
1088:
1089:         /// <summary>
1090:         /// Looks up a localized string similar to Search Completed.
1091:         /// </summary>
1092:         public static string SearchCompleted {
1093:             get {
1094:                 return ResourceManager.GetString("SearchCompleted", resourceCulture);
1095:             }
1096:         }
1097:
1098:         /// <summary>
1099:         /// Looks up a localized string similar to Search Results.
1100:         /// </summary>
1101:         public static string SearchResults {
1102:             get {
1103:                 return ResourceManager.GetString("SearchResults", resourceCulture);
1104:             }
1105:         }
1106:
1107:         /// <summary>
1108:         /// Looks up a localized string similar to You Have To Fill Search Criteria.
1109:         /// </summary>
1110:         public static string SearchTrimFault {
1111:             get {
1112:                 return ResourceManager.GetString("SearchTrimFault", resourceCulture);
1113:             }
1114:         }
1115:
1116:         /// <summary>
1117:         /// Looks up a localized string similar to Select.
1118:         /// </summary>
1119:         public static string Select {
1120:             get {
1121:                 return ResourceManager.GetString("Select", resourceCulture);
1122:             }
1123:         }
1124:
1125:         /// <summary>
1126:         /// Looks up a localized string similar to Select Background Color.
1127:         /// </summary>
1128:         public static string SelectBackgroundColor {
1129:             get {
1130:                 return ResourceManager.GetString("SelectBackgroundColor",
1131: resourceCulture);
1132:             }
1133:         }
1134:
1135:         /// <summary>
1136:         /// Looks up a localized string similar to Selected.
1137:         /// </summary>
1138:         public static string Selected {
1139:             get {
1140:                 return ResourceManager.GetString("Selected", resourceCulture);
1141:             }
1142:         }
1143:
1144:         /// <summary>
1145:         /// Looks up a localized string similar to Select Font Family.
1146:         /// </summary>
1147:         public static string SelectFontFamily {
1148:             get {
```

```
1148:         return ResourceManager.GetString("SelectFontFamily", resourceCulture);
1149:     }
1150: }
1151:
1152: /// <summary>
1153: /// Looks up a localized string similar to Select Font Size.
1154: /// </summary>
1155: public static string SelectFontSize {
1156:     get {
1157:         return ResourceManager.GetString("SelectFontSize", resourceCulture);
1158:     }
1159: }
1160:
1161: /// <summary>
1162: /// Looks up a localized string similar to Select Language.
1163: /// </summary>
1164: public static string SelectLanguage {
1165:     get {
1166:         return ResourceManager.GetString("SelectLanguage", resourceCulture);
1167:     }
1168: }
1169:
1170: /// <summary>
1171: /// Looks up a localized string similar to Send With Awesome Library App.
1172: /// </summary>
1173: public static string SendWithApp {
1174:     get {
1175:         return ResourceManager.GetString("SendWithApp", resourceCulture);
1176:     }
1177: }
1178:
1179: /// <summary>
1180: /// Looks up a localized string similar to Send With Mail.
1181: /// </summary>
1182: public static string SendWithMail {
1183:     get {
1184:         return ResourceManager.GetString("SendWithMail", resourceCulture);
1185:     }
1186: }
1187:
1188: /// <summary>
1189: /// Looks up a localized string similar to Send With SMS.
1190: /// </summary>
1191: public static string SendWithSMS {
1192:     get {
1193:         return ResourceManager.GetString("SendWithSMS", resourceCulture);
1194:     }
1195: }
1196:
1197: /// <summary>
1198: /// Looks up a localized string similar to Settings.
1199: /// </summary>
1200: public static string Settings {
1201:     get {
1202:         return ResourceManager.GetString("Settings", resourceCulture);
1203:     }
1204: }
1205:
1206: /// <summary>
1207: /// Looks up a localized string similar to Share Book.
1208: /// </summary>
1209: public static string ShareBook {
1210:     get {
1211:         return ResourceManager.GetString("ShareBook", 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 Book Count You Read.
1298: /// </summary>
1299: public static string TotalBookCount {
1300:     get {
1301:         return ResourceManager.GetString("TotalBookCount", 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 Book.
1316: /// </summary>
1317: public static string WorstBook {
1318:     get {
1319:         return ResourceManager.GetString("WorstBook", 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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.5 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 AwesomeLibrary
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 CurrentAuthorNumber { 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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.6 AssemblyInfo.cs

This is file AssemblyInfo.cs.

Body Source

```
1: ?using System.Reflection;
2: using System.Runtime.CompilerServices;
3: using System.Runtime.InteropServices;
4: using System.Resources;
5:
6: // General Information about an assembly is controlled through the following
7: // set of attributes. Change these attribute values to modify the information
8: // associated with an assembly.
9: [assembly: AssemblyTitle("Awesome Library")]
10: [assembly: AssemblyDescription("The Best Library Application Ever")]
```

```

11: [assembly: AssemblyConfiguration("")]
12: [assembly: AssemblyCompany("ÇMS Software")]
13: [assembly: AssemblyProduct("ÇMS Software")]
14: [assembly: AssemblyCopyright("Copyright © 2014")]
15: [assembly: AssemblyTrademark("")]
16: [assembly: AssemblyCulture("")]
17:
18: // Setting ComVisible to false makes the types in this assembly not visible
19: // to COM components. If you need to access a type in this assembly from
20: // COM, set the ComVisible attribute to true on that type.
21: [assembly: ComVisible(false)]
22:
23: // The following GUID is for the ID of the typelib if this project is exposed to COM
24: [assembly: Guid("2112005b-9c3f-4c40-912f-9da47b7d295b")]
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("1.0.2.0")]
36: [assembly: AssemblyFileVersion("1.0.2.0")]
37: [assembly: NeutralResourcesLanguageAttribute("en-US")]

```

1.2.7 Author.cs

This is file Author.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 AwesomeLibrary
10: {
11:     [Table]
12:     public class Author
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int AuthorId { get; set; }
19:
20:         [Column]
21:         public string AuthorName { get; set; }
22:
23:         [Column]
24:         public int AuthorBookCount { get; set; }
25:
26:         [Column]
27:         public string BookOrderBy { get; set; }
28:
29:         [Column]
30:         public string BookOrderStyle { get; set; }
31:
32:         [Column]

```



```
33:         public string AuthorNameCount { 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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.8 AuthorPage.xaml.cs

This is file AuthorPage.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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
19: {
20:     public partial class AuthorPage : PhoneApplicationPage
21:     {
22:
23:         public int authorId;
24:         public int categoryId;
25:         public int bookId;
26:         public Popup popup;
27:         public string oldAuthorName;
28:
29:         public AuthorPage()
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.AddBook;
38:             ApplicationBar.Buttons.Add(button1);
39:             button1.Click += new EventHandler(AddBookButton_Click);
40:
41:             ApplicationBarIconButton button2 = new ApplicationBarIconButton();
42:             button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
43:             button2.Text = AppResources.DeleteAuthor;
44:             ApplicationBar.Buttons.Add(button2);
```

```

45:         button2.Click += new EventHandler(DeleteAuthorButton_Click);
46:
47:         ApplicationBarIconButton button3 = new ApplicationBarIconButton();
48:         button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
49:         button3.Text = AppResources.AuthorSettings;
50:         ApplicationBar.Buttons.Add(button3);
51:         button3.Click += new EventHandler(AuthorSettingsButton_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:
63:     protected override void OnNavigatedTo(NavigationEventArgs e)
64:     {
65:         base.OnNavigatedTo(e);
66:         //while (NavigationService.CanGoBack)
67:         //NavigationService.RemoveBackEntry();
68:
69:     }
70:
71:     protected override void OnNavigatedFrom(NavigationEventArgs e)
72:     {
73:         base.OnNavigatedFrom(e);
74:         //while (NavigationService.CanGoBack)
75:         //NavigationService.RemoveBackEntry();
76:
77:     }
78:
79:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
80:     {
81:         List<Book> books = new List<Book>();
82:         List<Book> booksOrdered = new List<Book>();
83:
84:         // displays "Fragment: Detail"
85:         //MessageBox.Show("Folder Id: " + e.Fragment);
86:         base.OnFragmentNavigation(e);
87:
88:         lstBooks.Items.Clear();
89:
90:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
91:         {
92:
93:             var appSettings = context.AppSettings.First();
94:             categoryId = appSettings.CurrentCategoryNumber;
95:
96:             var author = context.Authors.Where(j =>
j.AuthorId.Equals(e.Fragment)).Single() as Author;
97:             authorId = author.AuthorId;
98:
99:             var appSettings2 = context.AppSettings;
100:             foreach (var item in appSettings2)
101:             {
102:                 item.CurrentAuthorNumber = authorId;
103:             }
104:             context.SubmitChanges();
105:
106:             var authorBooks = context.BookAuthors.Where(j =>
j.AuthorId.Equals(e.Fragment)).ToList() as List<BookAuthor>;
107:             if (authorBooks.Count != 0)
108:             {
109:                 foreach (var item in authorBooks)
110:                 {

```

```
111:         try
112:         {
113:             var book = context.Books.Where(j =>
114: j.BookCategoryId.Equals(categoryId) && j.BookId.Equals(item.BookId)).Single() as Book;
115:             books.Add(book);
116:         }
117:         catch (Exception)
118:         {
119:         }
120:     }
121: }
122:
123:
124: string orderStyle = author.BookOrderStyle;
125:
126: switch (author.BookOrderBy)
127: {
128:     case "NAME":
129:         if (orderStyle == "A")
130:         {
131:             booksOrdered = books.OrderBy(j => j.BookName).ToList();
132:         }
133:         else
134:         {
135:             booksOrdered = books.OrderByDescending(j =>
j.BookName).ToList();
136:         }
137:         break;
138:     case "CDATE":
139:         if (orderStyle == "A")
140:         {
141:             booksOrdered = books.OrderBy(j => j.CreationDate).ToList();
142:         }
143:         else
144:         {
145:             booksOrdered = books.OrderByDescending(j =>
j.CreationDate).ToList();
146:         }
147:         break;
148:     case "MDATE":
149:         if (orderStyle == "A")
150:         {
151:             booksOrdered = books.OrderBy(j =>
j.ModificationDate).ToList();
152:         }
153:         else
154:         {
155:             booksOrdered = books.OrderByDescending(j =>
j.ModificationDate).ToList();
156:         }
157:         break;
158:     case "SDATE":
159:         if (orderStyle == "A")
160:         {
161:             booksOrdered = books.OrderBy(j =>
j.ReadStartDate).ToList();
162:         }
163:         else
164:         {
165:             booksOrdered = books.OrderByDescending(j =>
j.ReadStartDate).ToList();
166:         }
167:         break;
168:     case "FDATE":
169:         if (orderStyle == "A")
170:         {
171:             booksOrdered = books.OrderBy(j =>
j.ReadFinishDate).ToList();
```

```

172:         }
173:         else
174:         {
175:             booksOrdered = books.OrderByDescending(j =>
j.ReadFinishDate).ToList();
176:         }
177:         break;
178:         case "RATING":
179:             if (orderStyle == "A")
180:             {
181:                 booksOrdered = books.OrderBy(j => j.BookRating).ToList();
182:             }
183:             else
184:             {
185:                 booksOrdered = books.OrderByDescending(j =>
j.BookRating).ToList();
186:             }
187:             break;
188:         default:
189:             if (orderStyle == "A")
190:             {
191:                 booksOrdered = books.OrderBy(j => j.BookName).ToList();
192:             }
193:             else
194:             {
195:                 booksOrdered = books.OrderBy(j => j.BookName).ToList();
196:             }
197:             break;
198:         }
199:
200:         lblAuthorName.Text = author.AuthorName;
201:         lblBookList.Text = AppResources.BookList + " (" + author.AuthorName +
")";
202:         lstBooks.ItemsSource = booksOrdered;
203:         lstBooks.DisplayMemberPath = "BookNameRating";
204:         SetBackgroundColor();
205:         //lstNoteList.DisplayMemberPath = "NameCreation";
206:     }
207: }
208:
209: private void SetBackgroundColor()
210: {
211:     AppSettings appSettings = new AppSettings();
212:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 lstBooks_SelectionChanged(object sender,
SelectionChangedEventArgs e)
265: {
266:     var book = (Book)lstBooks.SelectedItem;
267:     int bookId = book.BookId;
268:     NavigationService.Navigate(new Uri("/BookPage.xaml#" + bookId,
UriKind.Relative));
269: }
270:
271: private void AddBookButton_Click(object sender, EventArgs e)
272: {
273:     NavigationService.Navigate(new Uri("/BookPage.xaml", UriKind.Relative));
274: }
275:
276: private void DeleteAuthorButton_Click(object sender, EventArgs e)
277: {
278:     if (MessageBox.Show(AppResources.DeleteAuthorQuestion,
279:         AppResources.DeleteAuthor, MessageBoxButton.OKCancel)
280:         != MessageBoxResult.OK)
281:     {
282:     }
283: }
284: else
285: {
286:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
287:     {
288:         var bookAuthors = context.BookAuthors.Where(j =>
j.AuthorId.Equals(authorId)).ToList() as List<BookAuthor>;
289:         foreach (var item in bookAuthors)
290:         {
291:             var book = context.Books.Where(j =>
j.BookId.Equals(item.BookId)).Single() as Book;
292:             var bookAuthors2 = context.BookAuthors.Where(j =>
j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
293:             context.BookAuthors.DeleteAllOnSubmit(bookAuthors2);
294:             context.Books.DeleteOnSubmit(book);
295:         }
296:     }

```

```

297:         var categoryAuthors = context.CategoryAuthors.Where(j =>
j.AuthorId.Equals(authorId)).ToList() as List<CategoryAuthor>;
298:         context.CategoryAuthors.DeleteAllOnSubmit(categoryAuthors);
299:
300:         var authors = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
301:         context.Authors.DeleteOnSubmit(authors);
302:
303:         context.SubmitChanges();
304:
305:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
306:         foreach (var item in category)
307:         {
308:             item.CategoryBookCount = context.Books.Where(j =>
j.BookCategoryId.Equals(item.CategoryId)).ToList().Count;
309:             item.CategoryNameCount = item.CategoryName + " (" +
item.CategoryBookCount + ")";
310:             item.ModificationDate = DateTime.Now;
311:             context.SubmitChanges();
312:         }
313:     }
314:     MessageBox.Show(AppResources.AuthorDeleteSuccess);
315:     NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId,
UriKind.Relative));
316: }
317: //MessageBox.Show(AppResources.NoteSaved);
318: }
319:
320: private void lblAuthorName_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
321: {
322:     oldAuthorName = lblAuthorName.Text;
323:     popup = new Popup();
324:     popup.Height = 300;
325:     popup.Width = 400;
326:     popup.VerticalOffset = 20;
327:     PopupAddChange control = new PopupAddChange();
328:     control.txtLabel.Text = AppResources.EnterAuthorName;
329:     control.btnCancel.Content = AppResources.Cancel;
330:     control.btnOK.Content = AppResources.OK;
331:     popup.Child = control;
332:     popup.IsOpen = true;
333:     control.txtName.Text = lblAuthorName.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 authorName;
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:             authorName = control.txtName.Text.Remove(length - 1, 1);
347:         }
348:         else
349:         {
350:             authorName = control.txtName.Text;
351:         }
352:
353:         if (authorName != lblAuthorName.Text)
354:         {
355:             // aynı isimde bir klasörün daha önceden oluşturulup
olusturulmadigini
356:             // kontrol eden bir kod bölümü

```

```

357:         using (var contextFolder = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
358:         {
359:             isCreated =
360:                 contextFolder.Authors.Any(j =>
j.AuthorName.Equals(authorName));
361:         }
362:         if (isCreated == true)
363:         {
364:             MessageBox.Show(AppResources.AuthorExists);
365:         }
366:         // eger bu isimde bir klasör olusturulmamissa
367:         // olusturulmasi için gerekli kodlar asagidadir
368:         else
369:         {
370:             using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
371:             {
372:
373:                 // buraya kitapla ilgili bilginin güncellenecegi kod da
eklenecek
374:
375:                 var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
376:                 foreach (var item in author)
377:                 {
378:                     item.AuthorName = authorName;
379:                     item.ModificationDate = DateTime.Now;
380:                     item.AuthorNameCount = authorName + " (" +
item.AuthorBookCount.ToString() + ")";
381:                 }
382:                 context.SubmitChanges();
383:
384:                 var bookAuthors = context.BookAuthors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
385:                 foreach (var item in bookAuthors)
386:                 {
387:                     var book = context.Books.Where(j =>
j.BookId.Equals(item.BookId)).Select(j => j);
388:                     foreach (var item2 in book)
389:                     {
390:                         item2.BookAuthorName =
item2.BookAuthorName.Replace(oldAuthorName, authorName);
391:                         //item2.BookInformation =
item2.BookInformation.Replace(oldAuthorName, authorName);
392:                         item2.ModificationDate = DateTime.Now;
393:                         context.SubmitChanges();
394:                     }
395:                 }
396:
397:                 //lstFolders.ItemsSource = context.NoteFolders;
398:                 //lstAuthors.ItemsSource = context.Categories;
399:                 MessageBox.Show(AppResources.AuthorNameChangeSuccess);
400:                 popup.IsOpen = false;
401:                 CategoryAuthor categoryAuthor =
context.CategoryAuthors.Where(j => j.AuthorId.Equals(authorId) &&
j.CategoryId.Equals(categoryId)).Single() as CategoryAuthor;
402:                 Author author2 = context.Authors.Where(j =>
j.AuthorName.Equals(authorName) && j.AuthorId.Equals(categoryAuthor.AuthorId)).Single() as
Author;
403:                 NavigationService.Navigate(new Uri("/AuthorPage.xaml#" +
categoryAuthor.AuthorId, UriKind.Relative));
404:             }
405:         }
406:     }
407: };
408: control.btnCancel.Click += (s, args) =>
409: {
410:     popup.IsOpen = false;

```

```
411:         };
412:     }
413:
414:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
415:     {
416:         if (popup.IsOpen)
417:         {
418:             popup.IsOpen = false;
419:         }
420:         if (this.NavigationService.CanGoBack)
421:         {
422:             this.NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
categoryId, UriKind.Relative));
423:         }
424:     }
425:
426:     private void AuthorSettingsButton_Click(object sender, EventArgs e)
427:     {
428:         NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId,
UriKind.Relative));
429:     }
430:
431:     private void AddCategoryButton_Click(object sender, EventArgs e)
432:     {
433:         NavigationService.Navigate(new Uri("/AddCategoryPage.xaml#" + authorId,
UriKind.Relative));
434:     }
435:
436:     }
437: }
```

Namespaces

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

1.2.9 AuthorSettingsPage.xaml.cs

This is file AuthorSettingsPage.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 AwesomeLibrary.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 AwesomeLibrary
24: {
25:     public partial class AuthorSettingsPage : PhoneApplicationPage
26:     {
27:         public int authorId;
28:         public int categoryId;
29:
30:         public AuthorSettingsPage()
31:         {
32:             InitializeComponent();
33:             using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:             pvAuthorSettings.Title = AppResources.AuthorSettings;
41:             piFont.Header = AppResources.Font;
42:             piOtherSettings.Header = AppResources.OtherSettings;
43:
44:             btnFontFamily.Content = AppResources.Select;
45:             btnFontSize.Content = AppResources.Select;
46:             btnBookOrder.Content = AppResources.Select;
47:             btnBookOrderStyle.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
72:             {
73:                 var author = context.Authors.Where(j =>
j.AuthorId.Equals(e.Fragment)).Single() as Author;
74:                 authorId = author.AuthorId;
75:                 var appSettings = context.AppSettings.First();
76:                 categoryId = appSettings.CurrentCategoryNumber;
77:                 string orderStyle = author.BookOrderStyle;
78:
79:                 if (author.BookOrderBy == "NAME")
80:                 {
81:                     lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
82:                 }
83:                 if (author.BookOrderBy == "CDATE")
84:                 {
85:                     lblBookOrder.Text = AppResources.BookOrderBy + " (" +

```

```

AppResources.Selected + ": " + AppResources.CreationDate + "));
86:     }
87:     if (author.BookOrderBy == "MDATE")
88:     {
89:         lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + "));
90:     }
91:     if (author.BookOrderBy == "SDATE")
92:     {
93:         lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.StartDate + "));
94:     }
95:     if (author.BookOrderBy == "FDATE")
96:     {
97:         lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.FinishDate + "));
98:     }
99:     if (author.BookOrderBy == "RATING")
100:    {
101:        lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.BookRating + "));
102:    }
103:    if (author.BookOrderStyle == "A")
104:    {
105:        lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + "));
106:    }
107:    if (author.BookOrderStyle == "D")
108:    {
109:        lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" +
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 btnBookOrder_Click(object sender, RoutedEventArgs e)
172: {
173:     this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" +
authorId, UriKind.Relative));
174: }
175:
176: private void btnBookOrderStyle_Click(object sender, RoutedEventArgs e)
177: {
178:     this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml#" +
authorId, UriKind.Relative));
179: }
180:
181: private void btnFontSize_Click(object sender, RoutedEventArgs e)
182: {
183:     this.NavigationService.Navigate(new Uri("/FontSizeSettingsPage.xaml#" +
authorId, UriKind.Relative));
184: }
185:
186: private void btnFontFamily_Click(object sender, RoutedEventArgs e)
187: {
188:     this.NavigationService.Navigate(new Uri("/FontFamilySettingsPage.xaml#" +
authorId, UriKind.Relative));
189: }
190:
191: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
192: {
193:     //pvAuthorSettings.Title = AppResources.AuthorSettings;
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("/AuthorPage.xaml#" +
authorId, UriKind.Relative));
206:     }

```

```
207:         }
208:     }
209: }
```

Namespaces

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

1.2.10 AwesomeLibrary.csproj

This is file AwesomeLibrary.csproj.

1.2.11 AwesomeLibrary.sln

This is file AwesomeLibrary.sln.

1.2.12 AwesomeLibraryDataContext.cs

This is file AwesomeLibraryDataContext.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 AwesomeLibrary
10: {
11:     public class AwesomeLibraryDataContext : DataContext
12:     {
13:         public const string ConnectionString = @"Data
Source=isostore:/AwesomeLibrary.sdf; Max Database Size=256; Max Buffer Size=4096;";
14:         public AwesomeLibraryDataContext(string connectionString)
15:             : base(connectionString) { }
16:         public Table<Category> Categories;
17:         public Table<Author> Authors;
18:         public Table<Book> Books;
19:         public Table<AppSettings> AppSettings;
20:         public Table<BookAuthor> BookAuthors;
21:         public Table<CategoryAuthor> CategoryAuthors;
22:     }
23: }
```

Namespaces

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

1.2.13 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 AwesomeLibrary.Resources;
12: using Microsoft.Phone.Controls;
13: using Microsoft.Phone.Shell;
14:
15: namespace AwesomeLibrary
16: {
17:     public partial class BackgroundColorSettingsPage : PhoneApplicationPage
18:     {
19:         public int authorId;
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:         //MessageBox.Show("Folder Id: " + e.Fragment);
117:         base.OnFragmentNavigation(e);
118:         authorId = int.Parse(e.Fragment);
119:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
120:         {
121:             var author = context.Authors.Where(j =>

```

```

j.AuthorId.Equals(authorId)).Single() as Author;
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.14 Book.cs

This is file Book.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 AwesomeLibrary
10: {
11:     [Table]
12:     public class Book
13:     {
14:         [Column(IsPrimaryKey = true,
15:             IsDbGenerated = true,
16:             DbType = "INT NOT NULL Identity",
17:             CanBeNull = false)]
18:         public int BookId { get; set; }
19:
20:         [Column]
21:         public string BookGuid { get; set; }
22:
23:         [Column]
24:         public int BookCategoryId { get; set; }
25:
26:         [Column]
27:         public string BookCategoryName { get; set; }
28:
29:         [Column]
30:         public string BookAuthorName { get; set; }
31:     }
```



```
32:         [Column]
33:         public string BookName { get; set; }
34:
35:         [Column]
36:         public int BookPageNumber { get; set; }
37:
38:         [Column]
39:         public string BookPublisherName { get; set; }
40:
41:         [Column]
42:         public DateTime ReadStartDate { get; set; }
43:
44:         [Column]
45:         public DateTime ReadFinishDate { get; set; }
46:
47:         [Column]
48:         public int BookRating { get; set; }
49:
50:         [Column]
51:         public string BookComment { get; set; }
52:
53:         [Column]
54:         public DateTime CreationDate { get; set; }
55:
56:         [Column]
57:         public DateTime ModificationDate { get; set; }
58:
59:         //[Column]
60:         //public string BookInformation { get; set; }
61:
62:         [Column]
63:         public string BookNameRating { get; set; }
64:     }
65: }
```

Namespaces

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

1.2.15 BookAuthor.cs

This is file BookAuthor.cs.

Body Source

```
1: ?using System;
2: using System;
3: using System.Collections.Generic;
4: using System.Data.Linq;
5: using System.Data.Linq.Mapping;
6: using System.Linq;
7: using System.Text;
8: using System.Threading.Tasks;
9:
10: namespace AwesomeLibrary
11: {
12:     [Table]
13:     public class BookAuthor
14:     {
15:         [Column(IsPrimaryKey = true,
16:             IsDbGenerated = true,
17:             DbType = "INT NOT NULL Identity",
18:             CanBeNull = false)]
19:         public int BookAuthorId { get; set; }
```

```
20:
21:         [Column]
22:         public int BookId { get; set; }
23:
24:         [Column]
25:         public int AuthorId { get; set; }
26:     }
27: }
```

Namespaces

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

1.2.16 BookPage.xaml.cs

This is file BookPage.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 AwesomeLibrary.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeLibrary
20: {
21:     public partial class BookPage : PhoneApplicationPage
22:     {
23:         public int authorId;
24:         public string authorName;
25:         public string categoryName;
26:         public int categoryId;
27:         public int bookId;
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 BookPage()
35:         {
36:             InitializeComponent();
37:
38:             SetBackgroundColor();
39:
40:             //pvAuthor.Title = authorName;
41:             piBookName.Header = AppResources.BookName;
42:             piComment.Header = AppResources.BookComment;
43:             piPublisherName.Header = AppResources.PublisherName;
44:             piRating.Header = AppResources.BookRating;
45:             piStartFinishDate.Header = AppResources.Date;
```

```

46:         lblStartDate.Text = AppResources.StartDate;
47:         lblFinishDate.Text = AppResources.FinishDate;
48:         piPageNumber.Header = AppResources.PageNumber;
49:
50:
51:         ApplicationBar = new ApplicationBar();
52:
53:         ApplicationBarIconButton button1 = new ApplicationBarIconButton();
54:         button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
55:         button1.Text = AppResources.Save;
56:         ApplicationBar.Buttons.Add(button1);
57:         button1.Click += new EventHandler(SaveButton_Click);
58:
59:         ApplicationBarIconButton button2 = new ApplicationBarIconButton();
60:         button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
61:         button2.Text = AppResources.SendWithMail;
62:         ApplicationBar.Buttons.Add(button2);
63:         button2.Click += new EventHandler(SendMailButton_Click);
64:
65:         ApplicationBarIconButton button3 = new ApplicationBarIconButton();
66:         button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
67:         button3.Text = AppResources.SendWithSMS;
68:         ApplicationBar.Buttons.Add(button3);
69:         button3.Click += new EventHandler(SendSMSButton_Click);
70:
71:         ApplicationBarIconButton button4 = new ApplicationBarIconButton();
72:         button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
73:         button4.Text = AppResources.ShareBook;
74:         ApplicationBar.Buttons.Add(button4);
75:         button4.Click += new EventHandler(ShareBookButton_Click);
76:
77:         isFilled = false;
78:
79:         ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
80:         menuItem1.Text = AppResources.DeleteBook;
81:         ApplicationBar.MenuItems.Add(menuItem1);
82:         menuItem1.Click += new EventHandler(DeleteBookMenuItem_Click);
83:
84:     }
85:
86:     private void SendSMSButton_Click(object sender, EventArgs e)
87:     {
88:         SmsComposeTask smsComposeTask = new SmsComposeTask();
89:
90:         smsComposeTask.To = "";
91:         smsComposeTask.Body = CreateSendMaterial();
92:
93:         smsComposeTask.Show();
94:         //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
95:     }
96:
97:     private void ShareBookButton_Click(object sender, EventArgs e)
98:     {
99:         ShareStatusTask shareStatusTask = new ShareStatusTask();
100:
101:         shareStatusTask.Status = CreateSendMaterial();
102:
103:         shareStatusTask.Show();
104:         //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
105:     }
106:
107:     private void SendMailButton_Click(object sender, EventArgs e)
108:     {
109:         // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
110:         //EmailAddressChooserTask emailAddressChooserTask;
111:         //emailAddressChooserTask = new EmailAddressChooserTask();
112:         //emailAddressChooserTask.Completed += new
113:         EventHandler<EmailResult>(emailAddressChooserTask_Completed);
114:         //emailAddressChooserTask.Show();

```

```

114:
115:         EmailComposeTask emailComposeTask = new EmailComposeTask();
116:
117:         emailComposeTask.Subject = txtBookName.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.BookName + ": " + txtBookName.Text);
131:         sb.AppendLine(AppResources.CategoryName + ": " + categoryName);
132:         sb.AppendLine(AppResources.AuthorName + ": " + authorName);
133:         sb.AppendLine(AppResources.PageNumber + ": " + txtPageNumber.Text);
134:         sb.AppendLine(AppResources.PublisherName + ": " + txtPublisherName.Text);
135:         sb.AppendLine(AppResources.StartDate + ": " +
DateTime.Parse(dtStart.Value.ToString()).ToShortDateString());
136:         sb.AppendLine(AppResources.FinishDate + ": " +
DateTime.Parse(dtFinish.Value.ToString()).ToShortDateString());
137:         sb.AppendLine(AppResources.BookComment + ": " + txtBookComment.Text);
138:         sb.AppendLine(AppResources.BookRating + ": " + rtRating.Value.ToString() +
"/10");
139:         sb.AppendLine();
140:         sb.AppendLine();
141:         sb.AppendLine(AppResources.SendWithApp);
142:         return sb.ToString();
143:     }
144:
145:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
146:     {
147:         //SetBackgroundColor();
148:
149:         // yazarin adi sayfanin en üstünde görünsün diye yapıyor bu
150:         //pvAuthor.Title = authorName;
151:
152:         //pvAuthor.Title = authorName;
153:         //piBookName.Header = AppResources.BookName;
154:         //piComment.Header = AppResources.BookComment;
155:         //piPublisherName.Header = AppResources.PublisherName;
156:         //piRating.Header = AppResources.BookRating;
157:         //piStartFinishDate.Header = AppResources.Date;
158:         //lblStartDate.Text = AppResources.StartDate;
159:         //lblFinishDate.Text = AppResources.FinishDate;
160:         //piPageNumber.Header = AppResources.PageNumber;
161:     }
162:
163:     private void SetBackgroundColor()
164:     {
165:         AppSettings appSettings = new AppSettings();
166:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
167:         {
168:             appSettings = context.AppSettings.First() as AppSettings;
169:         }
170:
171:         if (appSettings.AppBackgroundImage != null)
172:         {
173:             MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
174:             BitmapImage image = new BitmapImage();
175:             image.SetSource(stream);
176:             ImageBrush ib = new ImageBrush();
177:             ib.ImageSource = image;
178:             this.LayoutRoot.Background = ib;

```

```

179:         }
180:     else
181:     {
182:         switch (appSettings.AppBackgroundColor)
183:         {
184:             case "BLA":
185:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
186:                 break;
187:             case "BLU":
188:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
189:                 break;
190:             case "BRO":
191:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
192:                 break;
193:             case "RED":
194:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
195:                 break;
196:             case "GRE":
197:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
198:                 break;
199:             case "GRA":
200:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
201:                 break;
202:             case "YEL":
203:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
204:                 break;
205:             case "ORA":
206:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
207:                 break;
208:             case "PUR":
209:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
210:                 break;
211:             default:
212:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
213:                 break;
214:         }
215:     }
216: }
217:
218: protected override void OnNavigatedTo(NavigationEventArgs e)
219: {
220:     base.OnNavigatedTo(e);
221:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
222:     {
223:         var appSettings = context.AppSettings.First();
224:         categoryId = appSettings.CurrentCategoryNumber;
225:         authorId = appSettings.CurrentAuthorNumber;
226:
227:         // sayfanin font ayarlari için yapılan bir degisiklik
228:         FontFamily temp = new FontFamily(appSettings.FontFamily);
229:         double fontsize = double.Parse(appSettings.FontSize);
230:         txtBookComment.FontFamily = temp;
231:         txtBookComment.FontSize = fontsize;
232:         txtPageNumber.FontFamily = temp;
233:         txtPageNumber.FontSize = fontsize;
234:         txtBookName.FontFamily = temp;
235:         txtBookName.FontSize = fontsize;
236:         txtBookComment.FontFamily = temp;
237:         txtBookComment.FontSize = fontsize;
238:         txtPublisherName.FontFamily = temp;
239:         txtPublisherName.FontSize = fontsize;
240:         rtRating.Value = 5;
241:
242:         var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;

```

```

243:             authorName = author.AuthorName;
244:
245:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
246:             categoryName = category.CategoryName;
247:         }
248:
249:         var lastPage = NavigationService.BackStack.FirstOrDefault();
250:         pageName = lastPage.Source.ToString();
251:         pvAuthor.SelectedIndex = 0;
252:         txtBookName.Focus();
253:         // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
254:         pvAuthor.Title = authorName;
255:         SetBackgroundColor();
256:     }
257:
258:     protected override void OnNavigatedFrom(NavigationEventArgs e)
259:     {
260:         base.OnNavigatedFrom(e);
261:         //while (NavigationService.CanGoBack)
262:         //NavigationService.RemoveBackEntry();
263:
264:     }
265:
266:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
267:     {
268:         // displays "Fragment: Detail"
269:         //MessageBox.Show("Folder Id: " + e.Fragment);
270:         base.OnFragmentNavigation(e);
271:         bookId = int.Parse(e.Fragment);
272:         if (pageName.Contains("/AuthorPage.xaml"))
273:         {
274:             isFilled = true;
275:         }
276:         else
277:         {
278:             //using (var context2 = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
279:             //{
280:             //    var appSettings = context2.AppSettings; ;
281:             //    var book2 = context2.Books.Where(j => j.BookId.Equals(bookId))
as Book;
282:             //    var bookAuthor = context2.BookAuthors.Where(j =>
j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
283:             //    var bAuthor = bookAuthor.First();
284:             //    var author = context2.Authors.Where(j =>
j.AuthorId.Equals(bAuthor.AuthorId)) as Author;
285:             //    foreach (var item in appSettings)
286:             //    {
287:             //        item.CurrentAuthorNumber = author.AuthorId;
288:             //        item.CurrentCategoryNumber = book2.BookCategoryId;
289:             //    }
290:             //    context2.SubmitChanges();
291:             //    pvAuthor.Title = author.AuthorName;
292:             //}
293:         }
294:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
295:         {
296:             var book = context.Books.Where(j =>
j.BookId.Equals(e.Fragment)).Single() as Book;
297:
298:             txtBookName.Text = book.BookName == "" ? "" : book.BookName;
299:             txtPageNumber.Text = book.BookPageNumber.ToString() == "" ? "" :
book.BookPageNumber.ToString();
300:             txtPublisherName.Text = book.BookPublisherName == "" ? "" :
book.BookPublisherName;
301:             dtStart.Value = book.ReadStartDate == null ? DateTime.Now :
book.ReadStartDate;

```

```

302:         dtFinish.Value = book.ReadFinishDate == null ? DateTime.Now :
book.ReadFinishDate;
303:         rtRating.Value = book.BookRating == null ? 0 : book.BookRating;
304:         txtBookComment.Text = book.BookComment == "" ? "" : book.BookComment;
305:     }
306:
307:     SetBackgroundColor();
308:     pvAuthor.SelectedIndex = 0;
309:     //pvAuthor.Name = authorName;
310:     txtBookName.Focus();
311: }
312:
313: private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
314: {
315:     //SaveButton_Click(this, new EventArgs());
316:     if (pageName.Contains("/SearchPage.xaml"))
317:     {
318:         //this.NavigationService.Navigate(new Uri("/SearchPage.xaml",
UriKind.Relative));
319:     }
320:     else
321:     {
322:         this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" +
authorId, UriKind.Relative));
323:     }
324: }
325:
326: private void SaveButton_Click(object sender, EventArgs e)
327: {
328:     txtBookComment_LostFocus(this, new RoutedEventArgs());
329:     this.pnlKeyboardPlaceHolder.Visibility = Visibility.Collapsed;
330:     if (txtBookName.Text.Trim().Length < 1)
331:     {
332:         MessageBox.Show(AppResources.BookNameMustBe);
333:     }
334:     else
335:     {
336:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
337:         {
338:             if (isFilled || pageName.Contains("/SearchPage.xaml"))
339:             {
340:                 var book = context.Books.Where(j =>
j.BookId.Equals(bookId)).Select(j => j);
341:                 foreach (var item in book)
342:                 {
343:                     item.BookCategoryId = categoryId;
344:                     item.BookName = txtBookName.Text == "" ? "" :
txtBookName.Text;
345:                     item.BookPageNumber = txtPageNumber.Text == "" ? 0 :
int.Parse(txtPageNumber.Text);
346:                     item.BookPublisherName = txtPublisherName.Text == "" ? ""
: txtPublisherName.Text;
347:                     item.ReadStartDate =
DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtStart.Value.ToString());
348:                     item.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString()); ;
349:                     //item.BookRating = rtRating.Value.ToString() == "" ? 0 :
int.Parse(rtRating.Value.ToString());
350:                     item.BookRating = int.Parse(ratingValue.ToString()) == 0 ?
0 : int.Parse(ratingValue.ToString());
351:                     item.BookComment = txtBookComment.Text == "" ? "" :
txtBookComment.Text;
352:                     item.ModificationDate = DateTime.Now;
353:
354:                     item.BookCategoryName = categoryName;

```



```

355:             item.BookAuthorName = authorName;
356:
357:             //item.BookInformation = categoryName + " " + authorName +
" " + txtBookName.Text + " " + txtPublisherName.Text + " " + txtBookComment.Text;
358:             item.BookNameRating = item.BookName + " (" +
item.BookRating + "/10)";
359:         }
360:         context.SubmitChanges();
361:     }
362:     else
363:     {
364:         Book book = new Book();
365:         book.BookCategoryId = categoryId;
366:         book.BookGuid = Guid.NewGuid().ToString();
367:         book.BookName = txtBookName.Text == "" ? "" : txtBookName.Text;
368:         book.BookPageNumber = txtPageNumber.Text == "" ? 0 :
int.Parse(txtPageNumber.Text);
369:         book.BookPublisherName = txtPublisherName.Text == "" ? "" :
txtPublisherName.Text;
370:         book.ReadStartDate = DateTime.Parse(dtStart.Value.ToString())
== null ? DateTime.Now : DateTime.Parse(dtStart.Value.ToString());
371:         book.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString()); ;
372:         //book.BookRating = rtRating.Value.ToString() == "" ? 0 :
int.Parse(rtRating.Value.ToString());
373:         book.BookRating = int.Parse(ratingValue.ToString()) == 0 ? 0 :
int.Parse(ratingValue.ToString());
374:         book.BookComment = txtBookComment.Text == "" ? "" :
txtBookComment.Text;
375:         book.ModificationDate = DateTime.Now;
376:
377:         book.BookCategoryName = categoryName;
378:         book.BookAuthorName = authorName;
379:
380:         //book.BookInformation = categoryName + " " + authorName + " "
+ book.BookName + " " + book.BookPublisherName + " " + book.BookComment;
381:         book.BookNameRating = book.BookName + " (" + book.BookRating +
"/10)";
382:         book.CreationDate = DateTime.Now;
383:         context.Books.InsertOnSubmit(book);
384:         context.SubmitChanges();
385:
386:         Book book2 = context.Books.Where(j =>
j.BookGuid.Equals(book.BookGuid)).Single() as Book;
387:
388:         BookAuthor bookAuthor = new BookAuthor();
389:         bookAuthor.AuthorId = authorId;
390:         bookAuthor.BookId = book2.BookId;
391:         context.BookAuthors.InsertOnSubmit(bookAuthor);
392:         context.SubmitChanges();
393:
394:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
395:         foreach (var item in category)
396:         {
397:             item.CategoryBookCount = item.CategoryBookCount + 1;
398:             item.CategoryNameCount = item.CategoryName + " (" +
item.CategoryBookCount + ")";
399:             item.ModificationDate = DateTime.Now;
400:         }
401:         context.SubmitChanges();
402:
403:         var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
404:         foreach (var item in author)
405:         {
406:             item.AuthorBookCount = item.AuthorBookCount + 1;
407:             item.AuthorNameCount = item.AuthorName + " (" +

```



```

item.AuthorBookCount + "));
408:         item.ModificationDate = DateTime.Now;
409:     }
410:     context.SubmitChanges();
411: }
412: }
413:     MessageBox.Show(AppResources.BookSaveSuccess);
414: }
415:     isFilled = false;
416: }
417:
418: private void txtBookComment_TextChanged(object sender, TextChangedEventArgs e)
419: {
420:     Dispatcher.BeginInvoke(() =>
421:     {
422:         double CurrentInputHeight = txtBookComment.ActualHeight;
423:
424:         if (CurrentInputHeight > InputHeight)
425:         {
426:             svBookComment.ScrollToVerticalOffset(svBookComment.VerticalOffset
+ CurrentInputHeight - InputHeight);
427:         }
428:
429:         InputHeight = CurrentInputHeight;
430:     });
431: }
432:
433: private void txtBookComment_GotFocus(object sender, RoutedEventArgs e)
434: {
435:     App.RootFrame.RenderTransform = new CompositeTransform();
436:     flag = true;
437: }
438:
439: private void txtBookComment_Tap(object sender,
System.Windows.Input.GestureEventArgs e)
440: {
441:     txtBookComment.Focus();
442:     //txtBookComment.Select(txtBookComment.Text.Length, 1);
443:     svBookComment.ScrollToVerticalOffset(e.GetPosition(txtBookComment).Y - 80);
444: }
445:
446: private void txtBookComment_LostFocus(object sender, RoutedEventArgs e)
447: {
448:     if (!flag) return;
449:     txtBookComment.Focus();
450:     flag = false;
451:     this.pnlKeyboardPlaceholder.Visibility = Visibility.Collapsed;
452: }
453:
454: private void txtBookComment_KeyDown(object sender, KeyEventArgs e)
455: {
456:     if (e.Key == Key.Enter)
457:     {
458:         svBookComment.ScrollToVerticalOffset(txtBookComment.ActualHeight);
459:     }
460: }
461:
462: private void svBookComment_GotFocus(object sender, RoutedEventArgs e)
463: {
464:     this.svBookComment.ScrollToVerticalOffset(this.txtBookComment.ActualHeight);
465:     this.svBookComment.UpdateLayout();
466: }
467:
468: private void txtBookName_KeyDown(object sender, KeyEventArgs e)
469: {
470:     if (e.Key == Key.Enter)
471:     {
472:         pvAuthor.SelectedIndex = 1;

```

```

473:         txtPageNumber.Focus();
474:     }
475: }
476:
477: private void txtPageNumber_KeyDown(object sender, KeyEventArgs e)
478: {
479:     if (e.Key == Key.Enter)
480:     {
481:         pvAuthor.SelectedIndex = 2;
482:         txtPublisherName.Focus();
483:     }
484: }
485:
486: private void txtPublisherName_KeyDown(object sender, KeyEventArgs e)
487: {
488:     if (e.Key == Key.Enter)
489:     {
490:         pvAuthor.SelectedIndex = 3;
491:         dtStart.Focus();
492:     }
493: }
494:
495: private void dtStart_KeyDown(object sender, KeyEventArgs e)
496: {
497:     if (e.Key == Key.Enter)
498:     {
499:         pvAuthor.SelectedIndex = 3;
500:         dtFinish.Focus();
501:     }
502: }
503:
504: private void dtFinish_KeyDown(object sender, KeyEventArgs e)
505: {
506:     if (e.Key == Key.Enter)
507:     {
508:         pvAuthor.SelectedIndex = 4;
509:         rtRating.Focus();
510:     }
511: }
512:
513: private void rtRating_KeyDown(object sender, KeyEventArgs e)
514: {
515:     //if (e.Key == Key.Enter)
516:     //{
517:     //    pvAuthor.SelectedIndex = 5;
518:     //    txtBookComment.Focus();
519:     //}
520: }
521:
522: private void rtRating_ValueChanged(object sender, EventArgs e)
523: {
524:     //pvAuthor.SelectedIndex = 5;
525:     ratingValue = rtRating.Value;
526:     //txtBookComment.Focus();
527: }
528:
529: private void dtFinish_ValueChanged(object sender,
DateTimeValueChangedEventArgs e)
530: {
531:     if (isFilled)
532:     {
533:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
534:         {
535:             var book = context.Books.Where(j =>
j.BookId.Equals(bookId)).Select(j => j);
536:             foreach (var item in book)
537:             {
538:                 item.ReadFinishDate =

```

```

DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now :
DateTime.Parse(dtFinish.Value.ToString());
539:         item.ModificationDate = DateTime.Now;
540:     }
541:     context.SubmitChanges();
542: }
543: }
544:     pvAuthor.SelectedIndex = 4;
545:     rtRating.Focus();
546: }
547:
548:     private void dtStart_ValueChanged(object sender, DateTimeValueChangedEventArgs
e)
549:     {
550:         if (isFilled)
551:         {
552:             using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
553:             {
554:                 var book = context.Books.Where(j =>
j.BookId.Equals(bookId)).Select(j => j);
555:                 foreach (var item in book)
556:                 {
557:                     item.ReadStartDate = DateTime.Parse(dtStart.Value.ToString())
== null ? DateTime.Now : DateTime.Parse(dtStart.Value.ToString());
558:                     item.ModificationDate = DateTime.Now;
559:                 }
560:                 context.SubmitChanges();
561:             }
562:         }
563:         pvAuthor.SelectedIndex = 3;
564:         dtFinish.Focus();
565:     }
566:
567:     private void rtRating_Tap(object sender, System.Windows.Input.GestureEventArgs
e)
568:     {
569:         //pvAuthor.SelectedIndex = 5;
570:         //ratingValue = rtRating.Value;
571:         //txtBookComment.Focus();
572:     }
573:
574:     private void DeleteBookMenuItem_Click(object sender, EventArgs e)
575:     {
576:         if (MessageBox.Show(AppResources.DeleteBookQuestion,
AppResources.DeleteBook, MessageBoxButton.OKCancel)
577:             != MessageBoxResult.OK)
578:         {
579:         }
580:     }
581: }
582: else
583: {
584:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
585:     {
586:         var book = context.Books.Where(j =>
j.BookId.Equals(bookId)).Single() as Book;
587:         var bookAuthors = context.BookAuthors.Where(j =>
j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
588:         context.BookAuthors.DeleteAllOnSubmit(bookAuthors);
589:         context.Books.DeleteOnSubmit(book);
590:
591:         var authors = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
592:         foreach (var item in authors)
593:         {
594:             item.ModificationDate = DateTime.Now;
595:             item.AuthorBookCount = item.AuthorBookCount - 1;
596:             item.AuthorNameCount = item.AuthorName + " (" +

```

```

item.AuthorBookCount + " ");
597:     }
598:     context.SubmitChanges();
599:
600:     var categories = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Select(j => j);
601:     foreach (var item in categories)
602:     {
603:         item.ModificationDate = DateTime.Now;
604:         item.CategoryBookCount = item.CategoryBookCount - 1;
605:         item.CategoryNameCount = item.CategoryName + " (" +
item.CategoryBookCount + ")";
606:     }
607:     context.SubmitChanges();
608: }
609: MessageBox.Show(AppResources.BookDeleteSuccess);
610: NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
611: }
612: //MessageBox.Show(AppResources.NoteSaved);
613: }
614:
615: private void btnIncrease_Click(object sender, RoutedEventArgs e)
616: {
617:     if (rtRating.Value != 10.0)
618:     {
619:         rtRating.Value = rtRating.Value + 1.0;
620:     }
621: }
622:
623: private void btnDecrease_Click(object sender, RoutedEventArgs e)
624: {
625:     if (rtRating.Value != 0.0)
626:     {
627:         rtRating.Value = rtRating.Value - 1.0;
628:     }
629: }
630: }
631: }

```

Namespaces

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

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 AwesomeLibrary
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 CategoryBookCount { get; set; }
25:
26:     [Column]
27:     public string AuthorOrderBy { get; set; }
28:
29:     [Column]
30:     public string AuthorOrderStyle { 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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.18 CategoryAuthor.cs

This is file CategoryAuthor.cs.

Body Source

```
1: ?using System;
2: using System;
3: using System.Collections.Generic;
4: using System.Data.Linq;
5: using System.Data.Linq.Mapping;
6: using System.Linq;
7: using System.Text;
8: using System.Threading.Tasks;
9:
10: namespace AwesomeLibrary
11: {
12:     [Table]
13:     public class CategoryAuthor
14:     {
15:         [Column(IsPrimaryKey = true,
16:             IsDbGenerated = true,
17:             DbType = "INT NOT NULL Identity",
18:             CanBeNull = false)]
19:         public int CategoryAuthorId { get; set; }
20:
21:         [Column]
22:         public int CategoryId { get; set; }
23:
24:         [Column]
25:         public int AuthorId { get; set; }
26:     }
```

```
27: }
```

Namespaces

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

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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
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.AddAuthor;
34:             ApplicationBar.Buttons.Add(button1);
35:             button1.Click += new EventHandler(AddAuthorButton_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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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<Author> authors = new List<Author>();
127:     List<Author> authorsOrdered = new List<Author>();
128:
129:     // displays "Fragment: Detail"
130:     //MessageBox.Show("Folder Id: " + e.Fragment);
131:     base.OnFragmentNavigation(e);
132:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
133:     {
134:         var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
135:         string orderStyle = category.AuthorOrderStyle;
136:         var categoryAuthor = context.CategoryAuthors.Where(j =>
j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryAuthor>;
137:
138:         foreach (var item in categoryAuthor)
139:         {
140:             try
141:             {
142:                 authors.Add(context.Authors.Where(j =>
j.AuthorId.Equals(item.AuthorId)).Single());
143:             }
144:             catch (Exception)
145:             {
146:             }
147:         }
148:     }
149:
150:     switch (category.AuthorOrderBy)
151:     {
152:         case "NAME":
153:             if (orderStyle == "A")
154:             {
155:                 authorsOrdered = authors.OrderBy(j =>
j.AuthorName).ToList();
156:             }
157:             else
158:             {
159:                 authorsOrdered = authors.OrderByDescending(j =>
j.AuthorName).ToList();
160:             }
161:             break;
162:         case "BOOKCOUNT":
163:             if (orderStyle == "A")
164:             {
165:                 authorsOrdered = authors.OrderBy(j =>
j.AuthorBookCount).ToList();
166:             }
167:             else
168:             {
169:                 authorsOrdered = authors.OrderByDescending(j =>
j.AuthorBookCount).ToList();
170:             }
171:             break;
172:         case "CDATE":
173:             if (orderStyle == "A")
174:             {
175:                 authorsOrdered = authors.OrderBy(j =>
j.CreationDate).ToList();
176:             }
177:             else

```



```

178:                {
179:                    authorsOrdered = authors.OrderByDescending(j =>
j.CreationDate).ToList();
180:                }
181:                break;
182:                case "MDATE":
183:                    if (orderStyle == "A")
184:                    {
185:                        authorsOrdered = authors.OrderBy(j =>
j.ModificationDate).ToList();
186:                    }
187:                    else
188:                    {
189:                        authorsOrdered = authors.OrderByDescending(j =>
j.ModificationDate).ToList();
190:                    }
191:                    break;
192:                default:
193:                    if (orderStyle == "A")
194:                    {
195:                        authorsOrdered = authors.OrderBy(j =>
j.AuthorName).ToList();
196:                    }
197:                    else
198:                    {
199:                        authorsOrdered = authors.OrderByDescending(j =>
j.AuthorName).ToList();
200:                    }
201:                    break;
202:            }
203:
204:            lstAuthors.Items.Clear();
205:            categoryId = category.CategoryId;
206:            lblCategoryName.Text = category.CategoryName;
207:            lblAuthorList.Text = AppResources.AuthorList + " (" +
category.CategoryName + ")";
208:            lstAuthors.ItemsSource = authorsOrdered;
209:            lstAuthors.DisplayMemberPath = "AuthorNameCount";
210:            SetBackgroundColor();
211:            //lstNoteList.DisplayMemberPath = "NameCreation";
212:        }
213:    }
214:
215:    private void AddAuthorButton_Click(object sender, EventArgs e)
216:    {
217:        popup = new Popup();
218:        popup.Height = 300;
219:        popup.Width = 400;
220:        popup.VerticalOffset = 20;
221:        PopupAddChange control = new PopupAddChange();
222:        control.txtLabel.Text = AppResources.EnterAuthorName;
223:        control.btnCancel.Content = AppResources.Cancel;
224:        control.btnOK.Content = AppResources.OK;
225:        popup.Child = control;
226:        popup.IsOpen = true;
227:        control.txtName.Focus();
228:
229:        control.btnOK.Click += (s, args) =>
230:        {
231:            bool isCreated;
232:            string authorName;
233:            popup.IsOpen = false;
234:
235:            int length = control.txtName.Text.Length;
236:            string space = control.txtName.Text.Substring(length - Math.Min(1,
length));
237:            if (space == " ")
238:            {
239:                authorName = control.txtName.Text.Remove(length - 1, 1);

```

```

240:         }
241:     }
242:     {
243:         authorName = control.txtName.Text;
244:     }
245:
246:     // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
247:     // kontrol eden bir kod bölümü
248:     using (var contextAuthor = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
249:     {
250:         isCreated =
251:             contextAuthor.Authors.Any(j =>
j.AuthorName.Equals(authorName));
252:     }
253:     if (isCreated == true)
254:     {
255:         MessageBox.Show(AppResources.AuthorExists);
256:     }
257:     // eger bu isimde bir klasör olusturulmamissa
258:     // olusturulmasi için gerekli kodlar asagidadir
259:     else
260:     {
261:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
262:         {
263:             Author author = new Author();
264:             author.AuthorName = authorName;
265:             author.CreationDate = DateTime.Now;
266:             author.ModificationDate = DateTime.Now;
267:             author.AuthorBookCount = 0;
268:             // burada yazarin kitaplarini
269:             // bitirme tarihine göre azalan bir sekilde ayarlamak için
gerekli düzenleme yapiliyor
270:             author.BookOrderBy = "FDATE";
271:             author.BookOrderStyle = "D";
272:             author.AuthorNameCount = author.AuthorName + " (" +
author.AuthorBookCount + ")";
273:             //note.NameDescriptionWithoutNewline =
note.NameDescription.Replace(Environment.NewLine, " ");
274:             //note.IsPasswordProtected = false;
275:
276:             context.Authors.InsertOnSubmit(author);
277:             context.SubmitChanges();
278:
279:             Author author3 = context.Authors.Where(j =>
j.AuthorName.Equals(authorName)).Single() as Author;
280:
281:             CategoryAuthor categoryAuthor = new CategoryAuthor();
282:             categoryAuthor.CategoryId = categoryId;
283:             categoryAuthor.AuthorId = author3.AuthorId;
284:             context.CategoryAuthors.InsertOnSubmit(categoryAuthor);
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<Author> authors = new List<Author>();
303:             var categoryAuthors = context.CategoryAuthors.Where(j =>
304: j.CategoryId.Equals(categoryId)).ToList() as List<CategoryAuthor>;
305:             foreach (var item in categoryAuthors)
306:             {
307:                 authors.Add(context.Authors.Where(j =>
308: j.AuthorId.Equals(item.AuthorId)).Single());
309:             }
310:             lstAuthors.ItemsSource = authors;
311:             MessageBox.Show(AppResources.AuthorAddSuccess);
312:             //Author author2 = context.Authors.Where(j =>
313: j.AuthorName.Equals(authorName)).Single() as Author;
314:
315:             var appSettings2 = context.AppSettings;
316:             foreach (var item in appSettings2)
317:             {
318:                 item.CurrentAuthorNumber = author3.AuthorId;
319:             }
320:             context.SubmitChanges();
321:             NavigationService.Navigate(new Uri("/AuthorPage.xaml#" +
322: author3.AuthorId, UriKind.Relative));
323:         }
324:     };
325:     control.btnCancel.Click += (s, args) =>
326:     {
327:         popup.IsOpen = false;
328:     };
329:     //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
330: }
331:
332: private void CategorySettingsButton_Click(object sender, EventArgs e)
333: {
334:     NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" +
335: categoryId, UriKind.Relative));
336: }
337:
338: private void lblCategoryName_Tap(object sender,
339: System.Windows.Input.GestureEventArgs e)
340: {
341:     oldCategoryName = lblCategoryName.Text;
342:     popup = new Popup();
343:     popup.Height = 300;
344:     popup.Width = 400;
345:     popup.VerticalOffset = 20;
346:     PopupAddChange control = new PopupAddChange();
347:     control.txtLabel.Text = AppResources.EnterCategoryName;
348:     control.btnCancel.Content = AppResources.Cancel;
349:     control.btnOK.Content = AppResources.OK;
350:     popup.Child = control;
351:     popup.IsOpen = true;
352:     control.txtName.Text = lblCategoryName.Text;
353:     control.txtName.Focus();
354:     control.txtName.Select(0, control.txtName.Text.Length);
355:
356:     control.btnOK.Click += (s, args) =>
357:     {
358:         bool isCreated;
359:         string categoryName;
360:         popup.IsOpen = false;
361:
362:         int length = control.txtName.Text.Length;
363:         string space = control.txtName.Text.Substring(length - Math.Min(1,
364: length));
365:         if (space == " ")
366:         {

```

```

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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.CategoryBookCount.ToString() + ")";
397:                 }
398:                 context.SubmitChanges();
399:
400:                 var book = context.Books.Where(j =>
j.BookCategoryId.Equals(categoryId)).Select(j => j);
401:                 foreach (var item in book)
402:                 {
403:                     item.BookCategoryName =
item.BookCategoryName.Replace(oldCategoryName, categoryName);
404:
405:                     //item.BookInformation =
item.BookInformation.Replace(oldCategoryName, categoryName);
406:                     item.ModificationDate = DateTime.Now;
407:                 }
408:                 context.SubmitChanges();
409:                 //lstFolders.ItemsSource = context.NoteFolders;
410:                 //lstAuthors.ItemsSource = context.Categories;
411:                 MessageBox.Show(AppResources.CategoryNameChangeSuccess);
412:                 popup.IsOpen = false;
413:                 Category category2 = context.Categories.Where(j =>
j.CategoryName.Equals(categoryName)).Single() as Category;
414:                 NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
category2.CategoryId, UriKind.Relative));
415:             }
416:         }
417:     }
418: };

```

```

419:         control.btnCancel.Click += (s, args) =>
420:         {
421:             popup.IsOpen = false;
422:         };
423:     }
424:
425:     private void DeleteCategoryButton_Click(object sender, EventArgs e)
426:     {
427:         if (MessageBox.Show(AppResources.DeleteCategoryQuestion,
428:             AppResources.DeleteCategory, MessageBoxButton.OKCancel)
429:             != MessageBoxResult.OK)
430:         {
431:         }
432:     }
433:     else
434:     {
435:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
436:         {
437:             var books = context.Books.Where(j =>
j.BookCategoryId.Equals(categoryId)).ToList() as List<Book>;
438:             foreach (var item in books)
439:             {
440:                 var bookAuthors = context.BookAuthors.Where(j =>
j.BookId.Equals(item.BookId)).ToList() as List<BookAuthor>;
441:                 context.BookAuthors.DeleteAllOnSubmit(bookAuthors);
442:             }
443:             context.Books.DeleteAllOnSubmit(books);
444:
445:             var authorCategories = context.CategoryAuthors.Where(j =>
j.CategoryId.Equals(categoryId)).ToList() as List<CategoryAuthor>;
446:             foreach (var item in authorCategories)
447:             {
448:                 var author = context.Authors.Where(j =>
j.AuthorId.Equals(item.AuthorId)).ToList() as List<Author>;
449:                 context.Authors.DeleteAllOnSubmit(author);
450:             }
451:             context.CategoryAuthors.DeleteAllOnSubmit(authorCategories);
452:
453:             var categories = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
454:             context.Categories.DeleteOnSubmit(categories);
455:
456:             context.SubmitChanges();
457:         }
458:         MessageBox.Show(AppResources.CategoryDeleteSuccess);
459:         NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
460:     }
461:     //MessageBox.Show(AppResources.NoteSaved);
462: }
463:
464:     private void lstAuthors_SelectionChanged(object sender,
SelectionChangedEventArgs e)
465:     {
466:         var author = (Author)lstAuthors.SelectedItem;
467:         int authorId = author.AuthorId;
468:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
469:         {
470:             var appSettings = context.AppSettings;
471:             foreach (var item in appSettings)
472:             {
473:                 item.CurrentAuthorNumber = authorId;
474:             }
475:             context.SubmitChanges();
476:         }
477:         NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId,
UriKind.Relative));

```

```
478:         }
479:
480:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
481:         {
482:             if (popup.IsOpen)
483:             {
484:                 popup.IsOpen = false;
485:             }
486:             if (this.NavigationService.CanGoBack)
487:             {
488:                 this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
489:             }
490:         }
491:
492:         private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
493:         {
494:             //SetBackgroundColor();
495:         }
496:     }
497: }
```

Namespaces

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

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 AwesomeLibrary.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21:
22:
23: namespace AwesomeLibrary
24: {
25:     public partial class CategorySettingsPage : PhoneApplicationPage
26:     {
27:         public int categoryId;
28:         public CategorySettingsPage()
29:         {
30:             InitializeComponent();
31:         }
32:     }
33: }
```

```

32:         pvCategorySettings.Title = AppResources.CategorySettings;
33:
34:         piOtherSettings.Header = AppResources.OtherSettings;
35:         btnAuthorOrder.Content = AppResources.Select;
36:         btnAuthorOrderStyle.Content = AppResources.Select;
37:         SetBackgroundColor();
38:
39:     }
40:     private void SetBackgroundColor()
41:     {
42:         AppSettings appSettings = new AppSettings();
43:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
117:         {
118:             var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
119:             string orderStyle = category.AuthorOrderStyle;
120:             categoryId = category.CategoryId;
121:
122:             if (category.AuthorOrderBy == "NAME")
123:             {
124:                 lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
125:             }
126:             if (category.AuthorOrderBy == "BOOKCOUNT")
127:             {
128:                 lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.BookCount + ")";
129:             }
130:             if (category.AuthorOrderBy == "CDATE")
131:             {
132:                 lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
133:             }
134:             if (category.AuthorOrderBy == "MDATE")
135:             {
136:                 lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
137:             }
138:             if (category.AuthorOrderStyle == "A")
139:             {
140:                 lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" +
AppResources.Selected + ": " + AppResources.Ascending + ")";
141:             }
142:             if (category.AuthorOrderStyle == "D")
143:             {
144:                 lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" +
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:         //btnAuthorOrder.Content = AppResources.Select;
157:         //btnAuthorOrderStyle.Content = AppResources.Select;

```



```
158:         //SetBackgroundColor();
159:     }
160:
161:     private void btnAuthorOrder_Click(object sender, RoutedEventArgs e)
162:     {
163:         this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" +
categoryId, UriKind.Relative));
164:     }
165:
166:     private void btnAuthorOrderStyle_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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

1.2.21 FontFamilySettingsPage.xaml.cs

1

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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
19: {
20:     public partial class FontFamilySettingsPage : PhoneApplicationPage
21:     {
22:         public int authorId;
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("Bookman 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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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("/AuthorSettingsPage.xaml#" + authorId,
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:         authorId = int.Parse(e.Fragment);
90:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
91:         {
92:             var author = context.Authors.Where(j =>

```

```

j.AuthorId.Equals(authorId)).Single() as Author;
93:         lblAuthorName.Text = author.AuthorName;
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("/AuthorSettingsPage.xaml#" +
authorId, UriKind.Relative));
104:         }
105:     }
106:
107:     private void SetBackgroundColor()
108:     {
109:         AppSettings appSettings = new AppSettings();
110:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
19: {
20:     public partial class FontSizeSettingsPage : PhoneApplicationPage
21:     {
22:         public int authorId;
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,
SelectionChangedEventArgs e)
47:         {
48:             if (lstFontSize.SelectedIndex != -1)
49:             {
50:                 using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
51:                 {
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("/AuthorSettingsPage.xaml#" + authorId,
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:             authorId = int.Parse(e.Fragment);
80:             using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
81:             {
82:                 var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
83:                 lblAuthorName.Text = author.AuthorName;
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("/AuthorSettingsPage.xaml#" +
authorId, UriKind.Relative));
94:             }
95:         }
96:
97:         private void SetBackgroundColor()
98:         {
99:             AppSettings appSettings = new AppSettings();
100:            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

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 AwesomeLibrary.Resources;
17: using Microsoft.Live;
18: using Microsoft.Phone.Controls;
19: using Microsoft.Phone.Shell;
20: using Microsoft.Phone.Tasks;
21:
22: namespace AwesomeLibrary
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 signIn;
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:
144:         if (appSettings.CategoryOrderBy == "NAME")
145:         {
146:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.Name + ")";
147:         }
148:         if (appSettings.CategoryOrderBy == "CDATE")
149:         {
150:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
151:         }
152:         if (appSettings.CategoryOrderBy == "MDATE")
153:         {
154:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.ModificationDate + ")";
155:         }
156:         if (appSettings.CategoryOrderBy == "BOOKCOUNT")
157:         {
158:             lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" +
AppResources.Selected + ": " + AppResources.BookCount + ")";
159:         }
160:         if (appSettings.CategoryOrderStyle == "A")
161:         {
162:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + "
(" + AppResources.Selected + ": " + AppResources.Ascending + ")";
163:         }
164:         if (appSettings.CategoryOrderStyle == "D")
165:         {
166:             lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + "
(" + AppResources.Selected + ": " + AppResources.Descending + ")";
167:         }
168:         if (appSettings.AppBackgroundColor == "BLA")
169:         {
170:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Black + ")";
171:         }
172:         if (appSettings.AppBackgroundColor == "BLU")
173:         {
174:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Blue + ")";
175:         }
176:         if (appSettings.AppBackgroundColor == "BRO")
177:         {
178:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Brown + ")";
179:         }
180:         if (appSettings.AppBackgroundColor == "RED")
181:         {
182:             lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Red + ")";
183:         }
184:         if (appSettings.AppBackgroundColor == "GRE")
185:         {

```

```

186:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + ")";
187:     }
188:     if (appSettings.AppBackgroundColor == "YEL")
189:     {
190:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + ")";
191:     }
192:     if (appSettings.AppBackgroundColor == "GRA")
193:     {
194:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + ")";
195:     }
196:     if (appSettings.AppBackgroundColor == "ORA")
197:     {
198:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + ")";
199:     }
200:     if (appSettings.AppBackgroundColor == "PUR")
201:     {
202:         lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + ")";
203:     }
204: }
205: }
206:
207: protected override void OnNavigatedTo(NavigationEventArgs e)
208: {
209:     base.OnNavigatedTo(e);
210:     SetBackgroundColor();
211:     //while (NavigationService.CanGoBack)
212:     //NavigationService.RemoveBackEntry();
213: }
214:
215: protected override void OnNavigatedFrom(NavigationEventArgs e)
216: {
217:     base.OnNavigatedFrom(e);
218:     //while (NavigationService.CanGoBack)
219:     //NavigationService.RemoveBackEntry();
220: }
221:
222: private async void btnOneDrive_Click(object sender, RoutedEventArgs e)
223: {
224:     try
225:     {
226:         if (this.btnOneDrive.Content.ToString() == "Sign In" ||
this.btnOneDrive.Content.ToString() == "Sign in")
227:         {
228:             LiveLoginResult loginResult = await
this.authClient.LoginAsync(scopes);
229:             if (loginResult.Status == LiveConnectSessionStatus.Connected)
230:             {
231:                 //this.btnOneDrive.Content = AppResources.SignOut;
232:                 this.btnOneDrive.Content = "Sign Out";
233:
234:                 this.liveClient = new LiveConnectClient(loginResult.Session);
235:                 this.GetMe();
236:                 btnOneDriveSync.IsEnabled = true;
237:                 cbSync.IsEnabled = true;
238:             }
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:     }

```

```

248:         cbSync.IsEnabled = true;
249:         //this.tbResponse.Text = "";
250:     }
251: }
252: catch (LiveAuthException authExp)
253: {
254:     //this.tbResponse.Text = authExp.ToString();
255: }
256: }
257:
258: private async void InitializePage()
259: {
260:     try
261:     {
262:         // bu benim uygulamama ait bir client id
263:         this.authClient = new LiveAuthClient("0000000044125951");
264:         LiveLoginResult loginResult = await
this.authClient.InitializeAsync(scopes);
265:         btnOneDrive.Content = "Sign In";
266:         if (loginResult.Status == LiveConnectSessionStatus.Connected)
267:         {
268:             //this.btnOneDrive.Content = AppResources.SignOut;
269:             this.btnOneDrive.Content = "Sign Out";
270:
271:             this.liveClient = new LiveConnectClient(loginResult.Session);
272:             //this.GetMe();
273:         }
274:     }
275:     catch (LiveAuthException authExp)
276:     {
277:         //this.tbResponse.Text = authExp.ToString();
278:     }
279: }
280:
281: private async void GetMe()
282: {
283:     try
284:     {
285:         LiveOperationResult operationResult = await
this.liveClient.GetAsync("me");
286:
287:         dynamic properties = operationResult.Result;
288:         //this.tbResponse.Text = properties.first_name + " " +
properties.last_name;
289:     }
290:     catch (LiveConnectException e)
291:     {
292:         //this.tbResponse.Text = e.ToString();
293:     }
294: }
295:
296: private void SetBackgroundColor()
297: {
298:     AppSettings appSettings = new AppSettings();
299:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
300:     {
301:         appSettings = context.AppSettings.First() as AppSettings;
302:     }
303:
304:     if (appSettings.AppBackgroundImage != null)
305:     {
306:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
307:         BitmapImage image = new BitmapImage();
308:         image.SetSource(stream);
309:         ImageBrush ib = new ImageBrush();
310:         ib.ImageSource = image;
311:         this.LayoutRoot.Background = ib;
312:     }

```

```

313:         else
314:         {
315:             switch (appSettings.AppBackgroundColor)
316:             {
317:                 case "BLA":
318:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
319:                     break;
320:                 case "BLU":
321:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
322:                     break;
323:                 case "BRO":
324:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
325:                     break;
326:                 case "RED":
327:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
328:                     break;
329:                 case "GRE":
330:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
331:                     break;
332:                 case "GRA":
333:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
334:                     break;
335:                 case "YEL":
336:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
337:                     break;
338:                 case "ORA":
339:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
340:                     break;
341:                 case "PUR":
342:                     this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
343:                     break;
344:                 default:
345:                     this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
346:                     break;
347:             }
348:         }
349:     }
350:
351:     public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
352: string folderName, string parentFolder)
353:     {
354:         string folderId = null;
355:
356:         // Retrieves all the directories.
357:         var queryFolder = parentFolder + "/files?filter=folders,albums";
358:         var opResult = await client.GetAsync(queryFolder);
359:         dynamic result = opResult.Result;
360:
361:         foreach (dynamic folder in result.data)
362:         {
363:             // Checks if current folder has the passed name.
364:             if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
365:             {
366:                 folderId = folder.id;
367:                 break;
368:             }
369:         }
370:
371:         if (folderId == null)
372:         {
373:             // Directory hasn't been found, so creates it using the PostAsync
method.
374:             var folderData = new Dictionary<string, object>();
375:             folderData.Add("name", folderName);
376:             opResult = await client.PostAsync(parentFolder, folderData);
377:             result = opResult.Result;

```

```

378:
379:         // Retrieves the id of the created folder.
380:         folderId = result.id;
381:     }
382:
383:     return folderId;
384: }
385:
386: private async void btnOneDriveSync_Click(object sender, RoutedEventArgs e)
387: {
388:
389:     IsolatedStorageFile myIsolatedStorage = null;
390:     StringBuilder sb = null;
391:
392:
393:     string folderName;
394:     try
395:     {
396:         //var folderData = new Dictionary<string, object>();
397:         folderName = "Awesome Library (" + DateTime.Now + ")";
398:         //folderName = folderName.Replace(":", ".");
399:         //folderName = folderName.Replace("/", ".");
400:         folderName = DesignFileName(folderName);
401:
402:         string skyDriveFolder = await CreateDirectoryAsync(liveClient,
403: folderName, "me/skydrive");
404:
405:         if (cbSync.IsChecked == false)
406:         {
407:             btnOneDrive.IsEnabled = false;
408:             pbSync.Visibility = Visibility.Visible;
409:             txtSynchronizing.Visibility = Visibility.Visible;
410:
411:             using (var context = new
412: AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
413:             {
414:                 //var noteFolders = context.NoteFolders.ToList() as
415: List<NoteFolder>;
416:                 var books = context.Books.ToList() as List<Book>;
417:
418:                 for (int i = 0; i < books.Count; i++)
419:                 {
420:                     var bookAuthor =
421:                         context.BookAuthors.Where(j =>
422: j.BookId.Equals(books[i].BookId)).ToList() as
423: List<BookAuthor>;
424:
425:                     var category = context.Categories.Where(j =>
426: j.CategoryId.Equals(books[i].BookCategoryId)).Single() as Category;
427:
428:                     List<Author> authors = new List<Author>();
429:
430:                     for (int k = 0; k < bookAuthor.Count; k++)
431:                     {
432:                         authors.Add(context.Authors.Where(j =>
433: j.AuthorId.Equals(bookAuthor[k].AuthorId)).Single() as Author);
434:                     }
435:
436:                     string fileName = Guid.NewGuid() + ". " +
437: books[i].BookName + " (" + category.CategoryName +
438: " ).txt";
439:
440:                     fileName = DesignFileName(fileName);
441:                     //fileName = fileName.Replace(":", ".");
442:                     //fileName = fileName.Replace("/", ".");
443:                     //StringBuilder sb = new StringBuilder();
444:                     //sb.AppendLine(AppResources.NoteName + ": " +

```

```

notes[i].NoteName);
440:                                     //sb.AppendLine(AppResources.FolderName + ": " +
noteFolder.NoteFolderName);
441:                                     //sb.AppendLine(AppResources.Password + ": " +
noteFolder.IsPasswordProtected);
442:                                     //sb.AppendLine(AppResources.CreationDate + ": " +
notes[i].CreationDate);
443:                                     //sb.AppendLine(AppResources.ModificationDate + ": " +
notes[i].ModificationDate);
444:                                     //sb.AppendLine(AppResources.Note + ": " +
notes[i].NoteDescription);
445:
446:
447:                                     myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
448:                                     //if (myIsolatedStorage.FileExists(fileName))
449:                                     //{
450:                                     //myIsolatedStorage.DeleteFile(fileName);
451:                                     //} //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
452:                                     using (StreamWriter writeFile = new StreamWriter
453:                                     (new IsolatedStorageFileStream(fileName,
FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
454:                                     {
455:                                         writeFile.WriteLine(AppResources.BookName + ": " +
books[i].BookName);
456:                                         writeFile.WriteLine(AppResources.CategoryName + ": " +
category.CategoryName);
457:                                         string authorNames = "";
458:                                         for (int l = 0; l < authors.Count; l++)
459:                                         {
460:                                             authorNames = authorNames + authors[l].AuthorName
+ ", ";
461:                                         }
462:                                         authorNames = authorNames.Substring(0,
authorNames.Length - 2);
463:                                         writeFile.WriteLine(AppResources.AuthorName + ": " +
authorNames);
464:                                         writeFile.WriteLine(AppResources.PageNumber + ": " +
books[i].BookPageNumber);
465:                                         writeFile.WriteLine(AppResources.PublisherName + ": " +
+ books[i].BookPublisherName);
466:                                         writeFile.WriteLine(AppResources.StartDate + ": " +
books[i].ReadStartDate.ToShortDateString());
467:                                         writeFile.WriteLine(AppResources.FinishDate + ": " +
books[i].ReadFinishDate.ToShortDateString());
468:                                         writeFile.WriteLine(AppResources.BookRating + ": " +
books[i].BookRating + "/10");
469:                                         writeFile.WriteLine(AppResources.BookComment + ": " +
books[i].BookComment);
470:                                         writeFile.Close();
471:                                     }
472:                                     IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.ReadWrite);
473:                                     var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
474:                                     pbSync.Value = (i + 1) * (100) / books.Count;
475:                                     //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
476:                                     }
477:                                 }
478:                             }
479:                             else
480:                             {
481:                                 using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
482:                                 {
483:                                     //var noteFolders = context.NoteFolders.ToList() as

```

```

List<NoteFolder>;
484:                                var books = context.Books.OrderBy(j =>
j.ReadFinishDate).ToList() as List<Book>;
485:                                var bookFirst = books.First();
486:                                var bookLast = books.Last();
487:
488:                                string fileName = Guid.NewGuid() + ". Awesome Library (" +
bookFirst.ReadFinishDate.ToShortDateString() + " - " +
bookLast.ReadFinishDate.ToShortDateString() + ").txt";
489:                                fileName = DesignFileName(fileName);
490:
491:                                myIsolatedStorage =
IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
492:
493:                                sb = new StringBuilder();
494:
495:                                for (int i = 0; i < books.Count; i++)
496:                                {
497:                                    var bookAuthor =
498:                                        context.BookAuthors.Where(j =>
j.BookId.Equals(books[i].BookId)).ToList() as
499:                                        List<BookAuthor>;
500:
501:                                    var category = context.Categories.Where(j =>
j.CategoryId.Equals(books[i].BookCategoryId)).Single() as Category;
502:
503:                                    List<Author> authors = new List<Author>();
504:
505:                                    for (int k = 0; k < bookAuthor.Count; k++)
506:                                    {
507:                                        authors.Add(context.Authors.Where(j =>
j.AuthorId.Equals(bookAuthor[k].AuthorId)).Single() as Author);
508:                                    }
509:
510:                                    sb.AppendLine();
511:                                    sb.AppendLine(AppResources.BookName + ": " +
books[i].BookName);
512:                                    sb.AppendLine(AppResources.CategoryName + ": " +
category.CategoryName);
513:                                    string authorNames = "";
514:                                    for (int l = 0; l < authors.Count; l++)
515:                                    {
516:                                        authorNames = authorNames + authors[l].AuthorName + ",
";
517:                                    }
518:                                    authorNames = authorNames.Substring(0, authorNames.Length
- 2);
519:                                    sb.AppendLine(AppResources.AuthorName + ": " +
authorNames);
520:                                    sb.AppendLine(AppResources.PageNumber + ": " +
books[i].BookPageNumber);
521:                                    sb.AppendLine(AppResources.PublisherName + ": " +
books[i].BookPublisherName);
522:                                    sb.AppendLine(AppResources.StartDate + ": " +
books[i].ReadStartDate.ToShortDateString());
523:                                    sb.AppendLine(AppResources.FinishDate + ": " +
books[i].ReadFinishDate.ToShortDateString());
524:                                    sb.AppendLine(AppResources.BookRating + ": " +
books[i].BookRating + "/10");
525:                                    sb.AppendLine(AppResources.BookComment + ": " +
books[i].BookComment);
526:                                    sb.AppendLine();
527:
528:                                    //if (myIsolatedStorage.FileExists(fileName))
529:                                    //{
530:                                        //myIsolatedStorage.DeleteFile(fileName);
531:                                        //}
532:                                    //now we use a StreamWriter to write inputBox.Text to
the file and save it to IsolatedStorage
                                    //pbSync.Value = (i + 1) * (100) / books.Count;

```



```

533:                                     //var res = await liveClient.UploadAsync("me/skydrive/" +
folderName, fileName, isfs, OverwriteOption.Overwrite);
534:                                     }
535:                                     using (StreamWriter writeFile = new StreamWriter
536:                                     (new IsolatedStorageFileStream(fileName,
FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
537:                                     {
538:                                         writeFile.Write(sb.ToString());
539:                                         writeFile.Close();
540:                                     }
541:                                     IsolatedStorageFileStream isfs =
myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.ReadWrite);
542:                                     var res = await liveClient.UploadAsync(skyDriveFolder,
fileName, isfs, OverwriteOption.Overwrite);
543:                                     }
544:                                     }
545:
546:                                     //this.infoTextBlock.Text = string.Join(" ", "Created folder:",
result.name, "ID:", result.id);
547:                                     MessageBox.Show(AppResources.OneDriveSyncCompleted);
548:
549:                                     pbSync.Visibility = Visibility.Collapsed;
550:                                     txtSynchronizing.Visibility = Visibility.Collapsed;
551:                                     pbSync.Value = 0;
552:                                     btnOneDrive.IsEnabled = true;
553:                                 }
554:                                 catch (Exception exception)
555:                                 {
556:                                     //this.infoTextBlock.Text = "Error creating folder: " +
exception.Message;
557:                                     MessageBox.Show(AppResources.SystemFault);
558:                                 }
559:                             }
560:
561:                             public string DesignFileName(string fileName)
562:                             {
563:                                 fileName = fileName.Replace(":", ".");
564:                                 fileName = fileName.Replace("?", ".");
565:                                 fileName = fileName.Replace("\\", ".");
566:                                 fileName = fileName.Replace("/", ".");
567:                                 fileName = fileName.Replace("<", ".");
568:                                 fileName = fileName.Replace(">", ".");
569:                                 fileName = fileName.Replace("|", ".");
570:                                 fileName = fileName.Replace("*", ".");
571:                                 return fileName;
572:                             }
573:
574:                             private void btnBackgroundColor_Click(object sender, RoutedEventArgs e)
575:                             {
576:                                 NavigationService.Navigate(new Uri("/BackgroundColorSettingsPage.xaml",
UriKind.Relative));
577:                             }
578:
579:                             private void btnBackgroundImage_Click(object sender, RoutedEventArgs e)
580:                             {
581:                                 PhotoChooserTask objPhotoChooser = new PhotoChooserTask();
582:                                 objPhotoChooser.Completed += new
EventHandler<PhotoResult>(PhotoChooseCall);
583:                                 objPhotoChooser.Show();
584:                             }
585:
586:                             private void PhotoChooseCall(object sender, PhotoResult e)
587:                             {
588:                                 switch (e.TaskResult)
589:                                 {
590:                                     case TaskResult.OK:
591:                                         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))

```



```

592:         {
593:             var appSettings = context.AppSettings;
594:             foreach (var appSetting in appSettings)
595:             {
596:                 appSetting.AppBackgroundImage = new
byte[e.ChosenPhoto.Length];
597:                 e.ChosenPhoto.Position = 0;
598:                 e.ChosenPhoto.Read(appSetting.AppBackgroundImage, 0,
appSetting.AppBackgroundImage.Length);
599:                 //noteFolder.NoteFolderPassword = "";
600:             }
601:             context.SubmitChanges();
602:             MessageBox.Show(AppResources.BackgroundImageChangeSuccess);
603:         }
604:         break;
605:     case TaskResult.Cancel:
606:         //MessageBox.Show("Cancelled");
607:         break;
608:     case TaskResult.None:
609:         //MessageBox.Show("Nothing Entered");
610:         break;
611:     }
612:     SetBackgroundColor();
613: }
614:
615: private void btnRemoveBackgroundImage_Click(object sender, RoutedEventArgs e)
616: {
617:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
618:     {
619:         var appSettings = context.AppSettings;
620:         foreach (var appSetting in appSettings)
621:         {
622:             appSetting.AppBackgroundImage = null;
623:         }
624:         context.SubmitChanges();
625:         MessageBox.Show(AppResources.BackgroundImageRemoveSuccess);
626:     }
627: }
628:
629: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
630: {
631:
632:     //pvGeneralSettings.Title = AppResources.GeneralSettings;
633:
634:     //piLanguage.Header = AppResources.Language;
635:     //piSync.Header = AppResources.Sync;
636:     //piOtherSettings.Header = AppResources.OtherSettings;
637:     //piBackground.Header = AppResources.Background;
638:
639:     ////lblOneDrive.Text = AppResources.OneDrive;
640:
641:     //btnCategoryOrder.Content = AppResources.Select;
642:     //btnCategoryOrderStyle.Content = AppResources.Select;
643:     //btnLanguage.Content = AppResources.Select;
644:     //btnBackgroundColor.Content = AppResources.Select;
645:     ////btnOneDrive.Content = AppResources.Login;
646:     ////btnOneDrive.SignInText = AppResources.SignIn;
647:     ////btnOneDrive.SignOutText = AppResources.SignOut;
648:     //btnOneDriveSync.Content = AppResources.Sync;
649:     //lblOneDrive.Text = AppResources.OneDrive;
650:     //txtSynchronizing.Text = AppResources.Synchronizing;
651:
652:     //pbSync.Visibility = Visibility.Collapsed;
653:     //txtSynchronizing.Visibility = Visibility.Collapsed;
654:     //txtSynchronizing.BorderBrush = this.LayoutRoot.Background;
655:
656:     //btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
657:     //lblBackgroundImage.Text = AppResources.BackgroundImage;

```

```

658:         //btnBackgroundImage.Content = AppResources.Select;
659:         //btnResetSettings.Content = AppResources.ResetSettings;
660:
661:         //btnOneDriveSync.IsEnabled = false;
662:         //cbSync.Content = AppResources.SyncOnOneFile;
663:         //cbSync.IsEnabled = false;
664:         //btnOneDrive.Content = "Sign In";
665:
666:         //SetBackgroundColor();
667:     }
668:
669:     private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
670:     {
671:         if (this.NavigationService.CanGoBack)
672:         {
673:             this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
674:         }
675:     }
676:
677:     private void btnOneDrive_SessionChanged(object sender,
Microsoft.Live.Controls.LiveConnectSessionChangedEventArgs e)
678:     {
679:         if (e != null && e.Status == LiveConnectSessionStatus.Connected)
680:         {
681:             //the session status is connected so we need to set this session
status to client
682:             this.liveClient = new LiveConnectClient(e.Session);
683:         }
684:         else
685:         {
686:             this.liveClient = null;
687:         }
688:     }
689:
690:     private void btnLanguage_Click(object sender, RoutedEventArgs e)
691:     {
692:         this.NavigationService.Navigate(new Uri("/LanguageSettingsPage.xaml",
UriKind.Relative));
693:     }
694:
695:     private void btnCategoryOrder_Click(object sender, RoutedEventArgs e)
696:     {
697:         this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml",
UriKind.Relative));
698:     }
699:
700:     private void btnCategoryOrderStyle_Click(object sender, RoutedEventArgs e)
701:     {
702:         this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml",
UriKind.Relative));
703:     }
704:
705:     private void btnResetSettings_Click(object sender, RoutedEventArgs e)
706:     {
707:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
708:         {
709:             var appSettings = context.AppSettings;
710:             foreach (var appSetting in appSettings)
711:             {
712:                 appSetting.AppBackgroundImage = null;
713:                 appSetting.AppBackgroundColor = "BLA";
714:             }
715:             context.SubmitChanges();
716:             this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
717:             MessageBox.Show(AppResources.SuccessfulResetSettings);
718:         }

```

```

719:         }
720:     }
721: }

```

Namespaces

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

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 AwesomeLibrary.Resources;
14: using Microsoft.Phone.Controls;
15: using Microsoft.Phone.Shell;
16:
17: namespace AwesomeLibrary
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.Spanish);
38:             lstLanguage.Items.Add(AppResources.Sanskrit);
39:             lstLanguage.Items.Add(AppResources.Thai);
40:
41:             lstLanguage.SelectedIndex = -1;
42:             lblLanguage.Text = AppResources.SelectLanguage;
43:             lblGeneralSettings.Text = AppResources.GeneralSettings;
44:
45:             SetBackgroundColor();
46:         }
47:
48:         protected override void OnNavigatedTo(NavigationEventArgs e)
49:         {
50:             base.OnNavigatedTo(e);

```

```

51:         SetBackgroundColor();
52:     }
53:
54:     protected override void OnNavigatedFrom(NavigationEventArgs e)
55:     {
56:         base.OnNavigatedFrom(e);
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("/GeneralSettingsPage.xaml",
UriKind.Relative));
64:         }
65:     }
66:
67:     private void SetBackgroundColor()
68:     {
69:         AppSettings appSettings = new AppSettings();
70:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 lstLanguage_SelectionChanged(object sender,
SelectionChangedEventArgs e)
123: {
124:     int index = lstLanguage.SelectedIndex;
125:     string culture = "";
126:     string lang = "";
127:     if (index == 0)
128:     {
129:         culture = "en";
130:         lang = "EN";
131:     }
132:     else if (index == 1)
133:     {
134:         culture = "tr";
135:         lang = "TR";
136:     }
137:     else if (index == 2)
138:     {
139:         culture = "de";
140:         lang = "DE";
141:     }
142:     else if (index == 3)
143:     {
144:         culture = "ru";
145:         lang = "RU";
146:     }
147:     else if (index == 4)
148:     {
149:         culture = "ar";
150:         lang = "AR";
151:     }
152:     else if (index == 5)
153:     {
154:         culture = "fa-IR";
155:         lang = "FA";
156:     }
157:     else if (index == 6)
158:     {
159:         culture = "zh";
160:         lang = "ZH";
161:     }
162:     else if (index == 7)
163:     {
164:         culture = "it";
165:         lang = "IT";
166:     }
167:     else if (index == 8)
168:     {
169:         culture = "fr";
170:         lang = "FR";
171:     }
172:     else if (index == 9)
173:     {
174:         culture = "ja";
175:         lang = "JA";
176:     }
177:     else if (index == 10)
178:     {
179:         culture = "es";
180:         lang = "ES";
181:     }
```

```
182:         else if (index == 11)
183:         {
184:             culture = "sa";
185:             lang = "SA";
186:         }
187:         else if (index == 12)
188:         {
189:             culture = "th";
190:             lang = "TH";
191:         }
192:
193:         //else if (index == 3)
194:         //{
195:         //    culture = "es";
196:         //    lang = "ES";
197:         //}
198:         //else if (index == 4)
199:         //{
200:         //    culture = "ru";
201:         //    lang = "RU";
202:         //}
203:         //else if (index == 5)
204:         //{
205:         //    culture = "zh";
206:         //    lang = "AR";
207:         //}
208:         //else if (index == 6)
209:         //{
210:         //    culture = "ar";
211:         //    lang = "AR";
212:         //}
213:         //else if (index == 7)
214:         //{
215:         //    culture = "fa-IR";
216:         //    lang = "FA";
217:         //}
218:         //else if (index == 8)
219:         //{
220:         //    culture = "it";
221:         //    lang = "IT";
222:         //}
223:         //else if (index == 9)
224:         //{
225:         //    culture = "fr";
226:         //    lang = "FR";
227:         //}
228:         //else if (index == 10)
229:         //{
230:         //    culture = "pt";
231:         //    lang = "PT";
232:         //}
233:         else
234:         {
235:             culture = "en";
236:             lang = "EN";
237:         }
238:
239:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
240:         {
241:             var appSettings = context.AppSettings;
242:             foreach (var appSetting in appSettings)
243:             {
244:                 appSetting.AppLangName = lang;
245:             }
246:             context.SubmitChanges();
247:         }
248:
249:         CultureInfo newCulture = new CultureInfo(culture);
```

```
250:         Thread.CurrentThread.CurrentCulture = newCulture;
251:         Thread.CurrentThread.CurrentUICulture = newCulture;
252:         MessageBox.Show(AppResources.LanguageWarning);
253:         NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
254:     }
255:
256:     private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
257:     {
258:         //SetBackgroundColor();
259:     }
260: }
261: }
```

Namespaces

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

1.2.25 LocalizedStrings.cs

This is file LocalizedStrings.cs.

Body Source

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

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 AwesomeLibrary.Resources;
17:
18: namespace AwesomeLibrary
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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,
147:         AppResources.ExitApp, MessageBoxButton.OKCancel)
148:         != MessageBoxResult.OK)
149:     {
150:         e.Cancel = true;
151:     }
152:     else
153:     {
154:         Application.Current.Terminate();
155:     }
156: }
157:
158: private void lstCategories_SelectionChanged(object sender,
159: SelectionChangedEventArgs e)
160: {
161:     var category = (Category)lstCategories.SelectedItem;
162:     int categoryId = category.CategoryId;
163:     using (var context = new
164: AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
165:     {
166:         var appSettings = context.AppSettings;
167:         foreach (var appSetting in appSettings)
168:         {
169:             appSetting.CurrentCategoryNumber = categoryId;
170:         }
171:         context.SubmitChanges();
172:     }
173:     NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId,
174: UriKind.Relative));
175: }
176: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
177: {
178:     using (var context = new
179: AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
180:     {
181:         var appSettings = context.AppSettings.First() as AppSettings;
182:         string orderStyle = appSettings.CategoryOrderStyle;
183:         //lstCategories.ItemsSource = null;
184:         lstCategories.Items.Clear();
185:         if (context.Categories.Count() != 0)
186:         {
187:             switch (appSettings.CategoryOrderBy)
188:             {
189:                 case "NAME":
190:                     if (orderStyle == "A")
191:                     {
192:                         lstCategories.ItemsSource =
193: context.Categories.OrderBy(j => j.CategoryName).ToList();
194:                     }
195:                     else
196:                     {
197:                         lstCategories.ItemsSource =
198: context.Categories.OrderByDescending(j => j.CategoryName).ToList();
199:                     }
200:                     break;
201:                 case "CDATE":
202:                     if (orderStyle == "A")
203:                     {
204:                         lstCategories.ItemsSource =
205: context.Categories.OrderBy(j => j.CreationDate).ToList();
206:                     }
207:                     else
208:                     {
209:                         lstCategories.ItemsSource =
210: context.Categories.OrderByDescending(j => j.CreationDate).ToList();
211:                     }
212:                 }
213:             }
214:         }
215:     }
216: }

```

```

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 "BOOKCOUNT":
217:         if (orderStyle == "A")
218:         {
219:             lstCategories.ItemsSource =
context.Categories.OrderBy(j => j.CategoryBookCount).ToList();
220:         }
221:         else
222:         {
223:             lstCategories.ItemsSource =
context.Categories.OrderByDescending(j => j.CategoryBookCount).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 AppBar
243: //private void BuildLocalizedAppBar()
244: //{
245: //    // Set the page's AppBar to a new instance of AppBar.
246: //    AppBar = new AppBar();
247: //
248: //    // Create a new button and set the text value to the localized
string from AppResources.
249: //    AppBarIconButton appBarButton = new
AppBarIconButton(new Uri("/Assets/AppBar/appbar.add.rest.png", UriKind.Relative));
250: //    appBarButton.Text = AppResources.AppBarButtonText;
251: //    AppBar.Buttons.Add(appBarButton);
252: //
253: //    // Create a new menu item with the localized string from
AppResources.
254: //    AppBarMenuItem appBarMenuItem = new
AppBarMenuItem(AppResources.AppBarMenuItemText);
255: //    AppBar.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:             // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
293:             // kontrol eden bir bölümü
294:             using (var contextFolder = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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:             // eger bu isimde bir klasör olusturulmamissa
304:             // olusturulmasi için gerekli kodlar asagidadir
305:             else
306:             {
307:                 using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
308:                 {
309:                     //noteFolderId = context.NoteFolders.Count() + 1;
310:                     Category category = new Category();
311:                     category.CategoryName = categoryName;
312:                     //noteFolder.NoteFolderId = noteFolderId;
313:                     category.CategoryBookCount = 0;
314:                     category.CreationDate = DateTime.Now;
315:                     category.ModificationDate = DateTime.Now;
316:                     category.AuthorOrderBy = "NAME";
317:                     category.AuthorOrderStyle = "A";
318:                     category.CategoryNameCount = category.CategoryName + " (" +
category.CategoryBookCount.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

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 AwesomeLibrary.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16: namespace AwesomeLibrary
```

```

17: {
18:     public partial class OrderSettingsPage : PhoneApplicationPage
19:     {
20:         public string pageName;
21:         public int categoryId;
22:         public int authorId;
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 = "BOOKCOUNT";
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 = "BOOKCOUNT";
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.AuthorOrderBy = orderType;
98:             }
99:             context.SubmitChanges();
100:            MessageBox.Show(AppResources.AuthorOrderTypeChangeSuccess);
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 = "SDATE";
121:        }
122:        else if (index == 4)
123:        {
124:            orderType = "FDATE";
125:        }
126:        else if (index == 5)
127:        {
128:            orderType = "RATING";
129:        }
130:        else
131:        {
132:            orderType = "NAME";
133:        }
134:
135:        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
136:        {
137:            var authors = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).ToList() as List<Author>;
138:            foreach (var author in authors)
139:            {
140:                author.BookOrderBy = orderType;
141:            }
142:            context.SubmitChanges();
143:            MessageBox.Show(AppResources.BookOrderTypeChangeSuccess);
144:        }
145:        NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" +
authorId, UriKind.Relative));

```

```

146:         }
147:     }
148:
149:     protected override void OnNavigatedTo(NavigationEventArgs e)
150:     {
151:         base.OnNavigatedTo(e);
152:         // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor
153:         bu bölüm
154:         var lastPage = NavigationService.BackStack.FirstOrDefault();
155:         pageName = lastPage.Source.ToString();
156:         lstOrderBy.Items.Clear();
157:         if (pageName.Contains("/GeneralSettingsPage.xaml"))
158:         {
159:             lblSettings.Text = AppResources.GeneralSettings;
160:             using (var context = new
161:                 AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
162:             {
163:                 var appSettings =
164:                     context.AppSettings.First();
165:                 lblOrderBy.Text = AppResources.CategoryOrderBy;
166:                 lstOrderBy.Items.Add(AppResources.Name);
167:                 lstOrderBy.Items.Add(AppResources.BookCount);
168:                 lstOrderBy.Items.Add(AppResources.CreationDate);
169:                 lstOrderBy.Items.Add(AppResources.ModificationDate);
170:             }
171:         }
172:     }
173:
174:     protected override void OnNavigatedFrom(NavigationEventArgs e)
175:     {
176:         base.OnNavigatedFrom(e);
177:     }
178:
179:     protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
180:     {
181:         // displays "Fragment: Detail"
182:         //MessageBox.Show("Folder Id: " + e.Fragment);
183:         base.OnFragmentNavigation(e);
184:         lstOrderBy.Items.Clear();
185:         if (pageName.Contains("/CategorySettingsPage.xaml"))
186:         {
187:             categoryId = int.Parse(e.Fragment);
188:             using (var context = new
189:                 AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
190:             {
191:                 var category = context.Categories.Where(j =>
192:                     j.CategoryId.Equals(categoryId)).Single() as Category;
193:                 lblSettings.Text = AppResources.CategorySettings + " (" +
194:                     category.CategoryName + ")" ;
195:                 lblOrderBy.Text = AppResources.AuthorOrderBy;
196:                 lstOrderBy.Items.Add(AppResources.Name);
197:                 lstOrderBy.Items.Add(AppResources.BookCount);
198:                 lstOrderBy.Items.Add(AppResources.CreationDate);
199:                 lstOrderBy.Items.Add(AppResources.ModificationDate);
200:             }
201:         }
202:         else
203:         {
204:             authorId = int.Parse(e.Fragment);
205:             using (var context = new
206:                 AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
207:             {
208:                 var author = context.Authors.Where(j =>
209:                     j.AuthorId.Equals(authorId)).Single() as Author;
210:                 lblSettings.Text = AppResources.AuthorSettings + " (" +
211:                     author.AuthorName + ")" ;
212:                 lblOrderBy.Text = AppResources.BookOrderBy;

```



```

207:         lstOrderBy.Items.Add(AppResources.Name);
208:         lstOrderBy.Items.Add(AppResources.CreationDate);
209:         lstOrderBy.Items.Add(AppResources.ModificationDate);
210:         lstOrderBy.Items.Add(AppResources.StartDate);
211:         lstOrderBy.Items.Add(AppResources.FinishDate);
212:         lstOrderBy.Items.Add(AppResources.BookRating);
213:     }
214: }
215: lstOrderBy.SelectedIndex = -1;
216: SetBackgroundColor();
217: }
218:
219: private void SetBackgroundColor()
220: {
221:     AppSettings appSettings = new AppSettings();
222:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
223:     {
224:         appSettings = context.AppSettings.First() as AppSettings;
225:     }
226:
227:     if (appSettings.AppBackgroundImage != null)
228:     {
229:         MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
230:         BitmapImage image = new BitmapImage();
231:         image.SetSource(stream);
232:         ImageBrush ib = new ImageBrush();
233:         ib.ImageSource = image;
234:         this.LayoutRoot.Background = ib;
235:     }
236:     else
237:     {
238:         switch (appSettings.AppBackgroundColor)
239:         {
240:             case "BLA":
241:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
242:                 break;
243:             case "BLU":
244:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
245:                 break;
246:             case "BRO":
247:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
248:                 break;
249:             case "RED":
250:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
251:                 break;
252:             case "GRE":
253:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
254:                 break;
255:             case "GRA":
256:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
257:                 break;
258:             case "YEL":
259:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
260:                 break;
261:             case "ORA":
262:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
263:                 break;
264:             case "PUR":
265:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
266:                 break;
267:             default:
268:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
269:                 break;
270:         }
271:     }

```

```
272:         }
273:
274:         private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
275:         {
276:             if (this.NavigationService.CanGoBack)
277:             {
278:                 if (pageName.Contains("/GeneralSettingsPage.xaml"))
279:                 {
280:                     this.NavigationService.Navigate(new
Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
281:                 }
282:                 else if (pageName.Contains("/CategorySettingsPage.xaml"))
283:                 {
284:                     this.NavigationService.Navigate(new
Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
285:                 }
286:                 else
287:                 {
288:                     this.NavigationService.Navigate(new
Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
289:                 }
290:             }
291:         }
292:
293:         private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
294:         {
295:             //SetBackgroundColor();
296:         }
297:     }
298: }
```

Namespaces

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

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 AwesomeLibrary.Resources;
13: using Microsoft.Phone.Controls;
14: using Microsoft.Phone.Shell;
15:
16:
17: namespace AwesomeLibrary
18: {
19:     public partial class OrderStyleSettingsPage : PhoneApplicationPage
20:     {
21:         public string pageName;
22:         public int categoryId;
```

```

23:         public int authorId;
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("/AuthorSettingsPage.xaml#" + authorId, 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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.AuthorOrderStyle = orderStyle;
103:         }
104:         context.SubmitChanges();
105:         MessageBox.Show(AppResources.AuthorOrderStyleChangeSuccess);
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
121:     {
122:         var authors = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).ToList() as List<Author>;
123:         foreach (var author in authors)
124:         {
125:             author.BookOrderStyle = orderStyle;
126:         }
127:         context.SubmitChanges();
128:         MessageBox.Show(AppResources.BookOrderStyleChangeSuccess);
129:     }
130:     NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" +
authorId, 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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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.AuthorOrderStyle;
170:         }
171:     }
172:     else
173:     {
174:         authorId = int.Parse(e.Fragment);
175:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
176:         {
177:             var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
178:             lblSettings.Text = AppResources.AuthorSettings + " (" +
author.AuthorName + ")";
179:             lblOrderStyle.Text = AppResources.BookOrderStyle;
180:         }
181:     }
182:     SetBackgroundColor();
183: }
184:
185: private void SetBackgroundColor()
186: {
187:     AppSettings appSettings = new AppSettings();
188:     using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

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 AwesomeLibrary
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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
AwesomeLibrary (see page 1)	This is namespace AwesomeLibrary.

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 AwesomeLibrary.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18: using System.Threading;
19:
20: namespace AwesomeLibrary
21: {
22:     public partial class SearchPage : PhoneApplicationPage
23:     {
24:         public SearchPage()
25:         {
26:             InitializeComponent();
27:             SetBackgroundColor();
28:
29:             txtSearchResult.Text = AppResources.SearchResults;
30:             lblSearch.Text = AppResources.Search;
31:             //btnSearch.Content = AppResources.Search;
32:             //lstSearch.SelectedIndex = -1;
33:         }
34:
35:         private void SetBackgroundColor()
36:         {
37:             AppSettings appSettings = new AppSettings();
38:             using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
39:             {
40:                 appSettings = context.AppSettings.First() as AppSettings;
41:             }
42:
43:             if (appSettings.AppBackgroundImage != null)
44:             {
45:                 MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
46:                 BitmapImage image = new BitmapImage();
47:                 image.SetSource(stream);
48:                 ImageBrush ib = new ImageBrush();
49:                 ib.ImageSource = image;
50:                 this.LayoutRoot.Background = ib;
51:             }
52:             else
53:             {
54:                 switch (appSettings.AppBackgroundColor)
55:                 {
56:                     case "BLA":
57:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
58:                         break;
59:                     case "BLU":
60:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
61:                         break;
62:                     case "BRO":
63:                         this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
64:                         break;
65:                     case "RED":

```



```

66:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
67:                 break;
68:             case "GRE":
69:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
70:                 break;
71:             case "GRA":
72:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
73:                 break;
74:             case "YEL":
75:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
76:                 break;
77:             case "ORA":
78:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
79:                 break;
80:             case "PUR":
81:                 this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
82:                 break;
83:             default:
84:                 this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
85:                 break;
86:         }
87:     }
88: }
89:
90: private void btnSearch_Click(object sender, RoutedEventArgs e)
91: {
92:     if (txtSearch.Text.TrimStart().Length < 1)
93:     {
94:         MessageBox.Show(AppResources.SearchTrimFault);
95:     }
96:     else
97:     {
98:         lstSearch.Items.Clear();
99:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
100:         {
101:             var bookList =
102:                 context.Books.Where(j =>
j.BookComment.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(T
hread.CurrentThread.CurrentCulture))
||
j.BookName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thre
ad.CurrentThread.CurrentCulture))
||
j.BookPublisherName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToL
ower(Thread.CurrentThread.CurrentCulture))
||
j.BookAuthorName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLowe
r(Thread.CurrentThread.CurrentCulture))
||
j.BookCategoryName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLo
wer(Thread.CurrentThread.CurrentCulture))).ToList()
as List<Book>;
103:             //var noteList = context.Notes.ToList() as List<Note>;
104:
105:             if (bookList != null)
106:             {
107:                 txtSearchResult.Text = AppResources.SearchResults + " (" +
bookList.Count() + ")";
108:             }
109:
110:             //lstSearch.ItemsSource = noteList;
111:             for (int i = 0; i < bookList.Count; i++)
112:             {
113:                 //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Inde

```

```

xOf(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
114:                                     //{
115:                                     lstSearch.Items.Add(bookList[i] as Book);
116:                                     //}
117:                                 }
118:                                 //lstSearch.ItemTemplate.
119:                                 //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate"
+ ")";
120:                                 lstSearch.DisplayMemberPath = "BookNameRating";
121:                                 MessageBox.Show(AppResources.SearchCompleted);
122:                             }
123:                         }
124:                     }
125:
126:     private void lstSearch_SelectionChanged(object sender,
SelectionChangedEventArgs e)
127:     {
128:         try
129:         {
130:             if (lstSearch.SelectedIndex != -1)
131:             {
132:
133:                 Book selectedBook = lstSearch.SelectedItem as Book;
134:                 using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
135:                 {
136:                     var appSettings = context.AppSettings;
137:                     var category =
138:                         context.Categories.Where(j =>
j.CategoryId.Equals(selectedBook.BookCategoryId)).Single() as
139:                         Category;
140:
141:                     var authorBook = context.BookAuthors.Where(j =>
j.BookId.Equals(selectedBook.BookId)).Single() as BookAuthor;
142:
143:                     foreach (var item in appSettings)
144:                     {
145:                         item.CurrentCategoryNumber = category.CategoryId;
146:                         item.CurrentAuthorNumber = authorBook.AuthorId;
147:                     }
148:                     context.SubmitChanges();
149:
150:                     NavigationService.Navigate(new Uri("/BookPage.xaml#" +
selectedBook.BookId, UriKind.Relative));
151:                 }
152:                 lstSearch.SelectedIndex = -1;
153:             }
154:
155:         }
156:         catch (Exception)
157:         {
158:             MessageBox.Show(AppResources.SystemFault);
159:         }
160:     }
161:
162:     protected override void OnNavigatedTo(NavigationEventArgs e)
163:     {
164:         base.OnNavigatedTo(e);
165:     }
166:     protected override void OnNavigatedFrom(NavigationEventArgs e)
167:     {
168:         base.OnNavigatedFrom(e);
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("/MainPage.xaml",
UriKind.Relative));
176:         }
177:     }
178:
179:     private void txtSearch_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
180:     {
181:         if (e.Key == Key.Enter)
182:         {
183:             if (txtSearch.Text.TrimStart().Length < 1)
184:             {
185:                 MessageBox.Show(AppResources.SearchTrimFault);
186:             }
187:             else
188:             {
189:                 lstSearch.Items.Clear();
190:                 using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
191:                 {
192:                     var bookList =
193:                         context.Books.Where(j =>
j.BookComment.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(T
hread.CurrentThread.CurrentCulture))
||
j.BookName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thre
ad.CurrentThread.CurrentCulture))
||
j.BookPublisherName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToL
ower(Thread.CurrentThread.CurrentCulture))
||
j.BookAuthorName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLowe
r(Thread.CurrentThread.CurrentCulture))
||
j.BookCategoryName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLo
wer(Thread.CurrentThread.CurrentCulture))).ToList()
194:                 as List<Book>;
195:                 //var noteList = context.Notes.ToList() as List<Note>;
196:
197:                 if (bookList != null)
198:                 {
199:                     txtSearchResult.Text = AppResources.SearchResults + " (" +
bookList.Count() + ")";
200:                 }
201:
202:                 //lstSearch.ItemsSource = noteList;
203:                 for (int i = 0; i < bookList.Count; i++)
204:                 {
205:                     //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Inde
xOf(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
206:                     //{
207:                         lstSearch.Items.Add(bookList[i] as Book);
208:                     //}
209:                 }
210:                 //lstSearch.ItemTemplate.
211:                 //lstSearch.DisplayMemberPath = "NoteName" + " (" +
"CreationDate" + ")";
212:                 lstSearch.DisplayMemberPath = "BookNameRating";
213:                 MessageBox.Show(AppResources.SearchCompleted);
214:             }
215:         }
216:     }
217: }
218:
219: private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)

```

```
220:         {
221:             txtSearch.Focus();
222:         }
223:     }
224: }
```

Namespaces

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

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 AwesomeLibrary.Resources;
16: using Microsoft.Phone.Controls;
17: using Microsoft.Phone.Shell;
18:
19: namespace AwesomeLibrary
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
AwesomeLibraryDataContext(AwesomeLibraryDataContext.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 authorName = "", categoryName = "", bookName, publisherName = "",
bestBook = "", worstBook = "";

```

```

111:         double pagePerDay, bookPerDay;
112:         int bookCount;
113:         int pageSum = 0;
114:         DateTime beginDate, endDate;
115:         using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
116:         {
117:             bookCount = context.Books.Count();
118:
119:             var categories = context.Categories.OrderByDescending(j =>
j.CategoryBookCount).ToList() as List<Category>;
120:             categoryName = categories.First().CategoryNameCount;
121:
122:             var authors = context.Authors.OrderByDescending(j =>
j.AuthorBookCount).ToList() as List<Author>;
123:             authorName = authors.First().AuthorNameCount;
124:
125:             var books = context.Books.OrderBy(j => j.ReadStartDate).ToList() as
List<Book>;
126:             beginDate = books.First().ReadStartDate;
127:             foreach (var item in books)
128:             {
129:                 pageSum = pageSum + item.BookPageNumber;
130:             }
131:
132:             // bugünün tarihi ile ilgili olarak son tarih düzenleniyor
133:             endDate = DateTime.Now;
134:
135:             var books3 = context.Books.GroupBy(j => j.BookPublisherName).Select(j
=> new { Name = j.Key, Total = j.Count() }).OrderByDescending(k => k.Total);
136:             publisherName = books3.First().Name + " (" + books3.First().Total +
")";
137:
138:             int diffDays = (endDate.Date - beginDate.Date).Days;
139:
140:             pagePerDay = pageSum / Convert.ToDouble(diffDays);
141:             bookPerDay = bookCount / Convert.ToDouble(diffDays);
142:
143:             var books4 = context.Books.OrderByDescending(j =>
j.BookRating).ToList() as List<Book>;
144:             int bestBookRating = books4.First().BookRating;
145:             int worstBookRating = books4.Last().BookRating;
146:             // en iyi ve en kötü puana sahip birden fazla albüm varsa hepsini
listelemeye yarayan
147:             // kod parçası buradadır.
148:             var bestBooks = context.Books.Where(j =>
j.BookRating.Equals(bestBookRating)).ToList() as List<Book>;
149:             if (bestBooks.Count < 2)
150:             {
151:                 bestBook = books4.First().BookNameRating;
152:             }
153:             else
154:             {
155:                 for (int i = 0; i < bestBooks.Count; i++)
156:                 {
157:                     bestBook = bestBook + bestBooks[i].BookNameRating + ", ";
158:                 }
159:                 bestBook = bestBook.Substring(0, bestBook.Length - 2);
160:             }
161:
162:             var worstBooks = context.Books.Where(j =>
j.BookRating.Equals(worstBookRating)).ToList() as List<Book>;
163:             if (worstBooks.Count < 2)
164:             {
165:                 worstBook = books4.Last().BookNameRating;
166:             }
167:             else
168:
169:

```

```
170:         {
171:             for (int i = 0; i < worstBooks.Count; i++)
172:             {
173:                 worstBook = worstBook + worstBooks[i].BookNameRating + ", ";
174:             }
175:             worstBook = worstBook.Substring(0, worstBook.Length - 2);
176:         }
177:
178:     }
179:
180:     sb.AppendLine(beginDate.ToShortDateString() + " - " +
endDate.ToShortDateString());
181:     sb.AppendLine();
182:     sb.AppendLine(AppResources.TotalBookCount + ": " + bookCount);
183:     sb.AppendLine(AppResources.MostReadCategory + ": " + categoryName);
184:     sb.AppendLine(AppResources.MostReadAuthor + ": " + authorName);
185:     sb.AppendLine(AppResources.MostReadPublisher + ": " + publisherName);
186:     sb.AppendLine(AppResources.BookPerDay + ": " + String.Format("{0:0.00}",
bookPerDay) );
187:     sb.AppendLine(AppResources.PagePerDay + ": " + String.Format("{0:0.00}",
pagePerDay) );
188:     sb.AppendLine(AppResources.BestBook + ": " + bestBook);
189:     sb.AppendLine(AppResources.WorstBook + ": " + worstBook);
190:
191:     txtStatistics.Text = sb.ToString();
192:     txtStatistics.IsReadOnly = true;
193: }
194: }
195: }
```

Namespaces

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

Index

A

AboutPage class 24

about AboutPage class 24

AboutPage 24

AboutPage.xaml.cs 90

AddCategoryPage class 25

about AddCategoryPage class 25

AddCategoryPage 26

authorId 26

OnFragmentNavigation 26

OnNavigatedFrom 27

OnNavigatedTo 27

AddCategoryPage.xaml.cs 92

App class 27

about App class 27

App 28

categoryNumber 28

IsTrial 29

RootFrame 29

App.xaml.cs 95

AppResources class 2

About 7

about AppResources class 2

AboutTheApp 7

AboutTheAppText 7

AboutTheAwesomeLibrary 7

AddAuthor 7

AddBook 8

AddCategory 8

AppResources 7

Arabic 8

Ascending 8

AuthorAddSuccess 8

AuthorAlreadySameCategory 8

AuthorCategoryAddSuccess 8

AuthorDeleteSuccess 8

AuthorExists 8

AuthorList 9

AuthorName 9

AuthorNameChangeSuccess 9

AuthorOrderBy 9

AuthorOrderStyle 9

AuthorOrderStyleChangeSuccess 9

AuthorOrderTypeChangeSuccess 9

AuthorSettings 9

Background 9

BackgroundColor 10

BackgroundColorChangeSuccess 10

BackgroundImage 10

BackgroundImageChangeSuccess 10

BackgroundImageRemoveSuccess 10

BestBook 10

Black 10

Blue 10

BookComment 10

BookCount 11

BookDeleteSuccess 11

BookList 11

BookName 11

BookNameMustBe 11

BookOrderBy 11

BookOrderStyle 11

BookOrderStyleChangeSuccess 11

BookOrderTypeChangeSuccess 11

BookPerDay 12

BookRating 12

BookSaveSuccess 12

Brown 12

Cancel 12

Categories 12

CategoryAddSuccess 12

CategoryDeleteSuccess 12

CategoryExists 12

CategoryName 13

CategoryNameChangeSuccess 13

CategoryOrderBy 13

CategoryOrderStyle 13

CategoryOrderStyleChangeSuccess 13

CategoryOrderTypeChangeSuccess 13

CategorySettings 13	Orange 18
Chinese 13	OtherSettings 18
ContactWithUs 13	PageNumber 18
CreationDate 14	PagePerDay 18
Culture 14	Persian 18
Date 14	Portuguese 18
DeleteAuthor 14	PublisherName 19
DeleteAuthorQuestion 14	Purple 19
DeleteBook 14	Rate 19
DeleteBookQuestion 14	Red 19
DeleteCategory 14	RemoveBackgroundImage 19
DeleteCategoryQuestion 15	ResetSettings 19
Descending 15	ResourceFlowDirection 19
English 15	ResourceLanguage 19
EnterAuthorName 15	ResourceManager 19
EnterCategoryName 15	Russian 20
ExitApp 15	Sanskrit 20
ExitAppQuestion 15	Save 20
FinishDate 15	Search 20
Font 15	SearchCompleted 20
FontFamily 16	SearchResults 20
FontFamilyChangeSuccess 16	SearchTrimFault 20
FontSize 16	Select 20
FontSizeChangeSuccess 16	SelectBackgroundColor 21
French 16	Selected 21
GeneralSettings 16	SelectFontFamily 21
German 16	SelectFontSize 21
Gray 16	SelectLanguage 21
Green 16	SendWithApp 21
Italian 17	SendWithMail 21
Japanese 17	SendWithSMS 21
Language 17	Settings 21
LanguageWarning 17	ShareBook 22
ModificationDate 17	Spanish 22
MostReadAuthor 17	StartDate 22
MostReadCategory 17	Statistics 22
MostReadPublisher 17	SuccessfulResetSettings 22
Name 17	Sync 22
OK 18	Synchronizing 22
OneDrive 18	SyncOnOneFile 22
OneDriveSyncCompleted 18	SystemFault 22

Thai 23	popup 35
TotalBookCount 23	AuthorPage.xaml.cs 124
Turkish 23	AuthorSettingsPage class 38
WorstBook 23	about AuthorSettingsPage class 38
Yellow 23	authorId 39
AppResources.Designer.cs 101	AuthorSettingsPage 39
AppSettings class 29	categoryId 40
about AppSettings class 29	OnFragmentNavigation 40
AppBackgroundColor 30	OnNavigatedFrom 41
AppBackgroundImage 30	OnNavigatedTo 41
AppLangName 30	AuthorSettingsPage.xaml.cs 131
AppSettingsId 30	AwesomeLibrary 1
CategoryOrderBy 30	AwesomeLibrary namespace 1
CategoryOrderStyle 30	Classes 23
CurrentAuthorNumber 31	AwesomeLibrary.AboutPage 24
CurrentCategoryNumber 31	AwesomeLibrary.AboutPage.AboutPage 24
FontFamily 31	AwesomeLibrary.AddCategoryPage 25
FontSize 31	AwesomeLibrary.AddCategoryPage.AddCategoryPage 26
AppSettings.cs 121	AwesomeLibrary.AddCategoryPage.authorId 26
AssemblyInfo.cs 122	AwesomeLibrary.AddCategoryPage.OnFragmentNavigation 26
Author class 31	AwesomeLibrary.AddCategoryPage.OnNavigatedFrom 27
about Author class 31	AwesomeLibrary.AddCategoryPage.OnNavigatedTo 27
AuthorBookCount 32	AwesomeLibrary.App 27
AuthorId 32	AwesomeLibrary.App.App 28
AuthorName 32	AwesomeLibrary.App.categoryNumber 28
AuthorNameCount 32	AwesomeLibrary.App.IsTrial 29
BookOrderBy 33	AwesomeLibrary.App.RootFrame 29
BookOrderStyle 33	AwesomeLibrary.AppSettings 29
CreationDate 33	AwesomeLibrary.AppSettings.AppBackgroundColor 30
ModificationDate 33	AwesomeLibrary.AppSettings.AppBackgroundImage 30
Author.cs 123	AwesomeLibrary.AppSettings.AppLangName 30
AuthorPage class 33	AwesomeLibrary.AppSettings.AppSettingsId 30
about AuthorPage class 33	AwesomeLibrary.AppSettings.CategoryOrderBy 30
authorId 35	AwesomeLibrary.AppSettings.CategoryOrderStyle 30
AuthorPage 34	AwesomeLibrary.AppSettings.CurrentAuthorNumber 31
bookId 35	AwesomeLibrary.AppSettings.CurrentCategoryNumber 31
categoryId 35	AwesomeLibrary.AppSettings.FontFamily 31
oldAuthorName 35	AwesomeLibrary.AppSettings.FontSize 31
OnFragmentNavigation 35	AwesomeLibrary.Author 31
OnNavigatedFrom 38	AwesomeLibrary.Author.AuthorBookCount 32
OnNavigatedTo 38	

AwesomeLibrary.Author.AuthorId 32	AwesomeLibrary.BackgroundColorSettingsPage.BackgroundColorSettingsPage 44
AwesomeLibrary.Author.AuthorName 32	
AwesomeLibrary.Author.AuthorNameCount 32	AwesomeLibrary.BackgroundColorSettingsPage.OnFragmentNavigation 45
AwesomeLibrary.Author.BookOrderBy 33	
AwesomeLibrary.Author.BookOrderStyle 33	AwesomeLibrary.BackgroundColorSettingsPage.OnNavigatedFrom 45
AwesomeLibrary.Author.CreationDate 33	
AwesomeLibrary.Author.ModificationDate 33	AwesomeLibrary.BackgroundColorSettingsPage.OnNavigatedTo 45
AwesomeLibrary.AuthorPage 33	
AwesomeLibrary.AuthorPage.authorId 35	AwesomeLibrary.Book 46
AwesomeLibrary.AuthorPage.AuthorPage 34	AwesomeLibrary.Book.BookAuthorName 46
AwesomeLibrary.AuthorPage.bookId 35	AwesomeLibrary.Book.BookCategoryId 47
AwesomeLibrary.AuthorPage.categoryId 35	AwesomeLibrary.Book.BookCategoryName 47
AwesomeLibrary.AuthorPage.oldAuthorName 35	AwesomeLibrary.Book.BookComment 47
AwesomeLibrary.AuthorPage.OnFragmentNavigation 35	AwesomeLibrary.Book.BookGuid 47
AwesomeLibrary.AuthorPage.OnNavigatedFrom 38	AwesomeLibrary.Book.BookId 47
AwesomeLibrary.AuthorPage.OnNavigatedTo 38	AwesomeLibrary.Book.BookName 47
AwesomeLibrary.AuthorPage.popup 35	AwesomeLibrary.Book.BookNameRating 48
AwesomeLibrary.AuthorSettingsPage 38	AwesomeLibrary.Book.BookPageNumber 48
AwesomeLibrary.AuthorSettingsPage.authorId 39	AwesomeLibrary.Book.BookPublisherName 48
AwesomeLibrary.AuthorSettingsPage.AuthorSettingsPage 39	AwesomeLibrary.Book.BookRating 48
AwesomeLibrary.AuthorSettingsPage.categoryId 40	AwesomeLibrary.Book.CreationDate 48
AwesomeLibrary.AuthorSettingsPage.OnFragmentNavigation 40	AwesomeLibrary.Book.ModificationDate 48
AwesomeLibrary.AuthorSettingsPage.OnNavigatedFrom 41	AwesomeLibrary.Book.ReadFinishDate 49
AwesomeLibrary.AuthorSettingsPage.OnNavigatedTo 41	AwesomeLibrary.Book.ReadStartDate 49
AwesomeLibrary.AwesomeLibraryDataContext 41	AwesomeLibrary.BookAuthor 49
AwesomeLibrary.AwesomeLibraryDataContext.AppSettings 42	AwesomeLibrary.BookAuthor.AuthorId 49
AwesomeLibrary.AwesomeLibraryDataContext.Authors 42	AwesomeLibrary.BookAuthor.BookAuthorId 50
AwesomeLibrary.AwesomeLibraryDataContext.AwesomeLibraryDataContext 42	AwesomeLibrary.BookAuthor.BookId 50
AwesomeLibrary.AwesomeLibraryDataContext.BookAuthors 43	AwesomeLibrary.BookPage 50
AwesomeLibrary.AwesomeLibraryDataContext.Books 43	AwesomeLibrary.BookPage.authorId 52
AwesomeLibrary.AwesomeLibraryDataContext.Categories 43	AwesomeLibrary.BookPage.authorName 52
AwesomeLibrary.AwesomeLibraryDataContext.CategoryAuthors 43	AwesomeLibrary.BookPage.bookId 52
AwesomeLibrary.AwesomeLibraryDataContext.ConnectionString 43	AwesomeLibrary.BookPage.BookPage 51
AwesomeLibrary.BackgroundColorSettingsPage 43	AwesomeLibrary.BookPage.categoryId 52
AwesomeLibrary.BackgroundColorSettingsPage.authorId 45	AwesomeLibrary.BookPage.categoryName 52
	AwesomeLibrary.BookPage.flag 52
	AwesomeLibrary.BookPage.isFilled 53
	AwesomeLibrary.BookPage.OnFragmentNavigation 53
	AwesomeLibrary.BookPage.OnNavigatedFrom 54
	AwesomeLibrary.BookPage.OnNavigatedTo 54

AwesomeLibrary.BookPage.pageName 53	AwesomeLibrary.FontFamilySettingsPage.OnNavigatedTo 67
AwesomeLibrary.BookPage.ratingValue 53	AwesomeLibrary.FontSizeSettingsPage 67
AwesomeLibrary.Category 55	AwesomeLibrary.FontSizeSettingsPage.authorId 68
AwesomeLibrary.Category.AuthorOrderBy 56	AwesomeLibrary.FontSizeSettingsPage.FontSizeSettingsPage 68
AwesomeLibrary.Category.AuthorOrderStyle 56	AwesomeLibrary.FontSizeSettingsPage.OnFragmentNavigation 68
AwesomeLibrary.Category.CategoryBookCount 56	AwesomeLibrary.FontSizeSettingsPage.OnNavigatedFrom 69
AwesomeLibrary.Category.CategoryId 56	AwesomeLibrary.FontSizeSettingsPage.OnNavigatedTo 69
AwesomeLibrary.Category.CategoryName 56	AwesomeLibrary.GeneralSettingsPage 69
AwesomeLibrary.Category.CategoryNameCount 57	AwesomeLibrary.GeneralSettingsPage.CreateDirectoryAsync 73
AwesomeLibrary.Category.CreationDate 57	AwesomeLibrary.GeneralSettingsPage.DesignFileName 74
AwesomeLibrary.Category.ModificationDate 57	AwesomeLibrary.GeneralSettingsPage.GeneralSettingsPage 70
AwesomeLibrary.CategoryAuthor 57	AwesomeLibrary.GeneralSettingsPage.OnNavigatedFrom 74
AwesomeLibrary.CategoryAuthor.AuthorId 58	AwesomeLibrary.GeneralSettingsPage.OnNavigatedTo 75
AwesomeLibrary.CategoryAuthor.CategoryAuthorId 58	AwesomeLibrary.GeneralSettingsPage.signIn 73
AwesomeLibrary.CategoryAuthor.CategoryId 58	AwesomeLibrary.LanguageSettingsPage 75
AwesomeLibrary.CategoryPage 58	AwesomeLibrary.LanguageSettingsPage.LanguageSettingsPage 75
AwesomeLibrary.CategoryPage.categoryId 59	AwesomeLibrary.LanguageSettingsPage.OnNavigatedFrom 76
AwesomeLibrary.CategoryPage.CategoryPage 59	AwesomeLibrary.LanguageSettingsPage.OnNavigatedTo 76
AwesomeLibrary.CategoryPage.oldCategoryName 59	AwesomeLibrary.LocalizedStrings 76
AwesomeLibrary.CategoryPage.OnFragmentNavigation 60	AwesomeLibrary.LocalizedStrings.LocalizedResources 77
AwesomeLibrary.CategoryPage.OnNavigatedFrom 61	AwesomeLibrary.MainPage 77
AwesomeLibrary.CategoryPage.OnNavigatedTo 62	AwesomeLibrary.MainPage.MainPage 78
AwesomeLibrary.CategoryPage.popup 60	AwesomeLibrary.MainPage.OnNavigatedFrom 79
AwesomeLibrary.CategorySettingsPage 62	AwesomeLibrary.MainPage.OnNavigatedTo 79
AwesomeLibrary.CategorySettingsPage.categoryId 63	AwesomeLibrary.MainPage.popup 78
AwesomeLibrary.CategorySettingsPage.CategorySettingsPage 63	AwesomeLibrary.OrderSettingsPage 79
AwesomeLibrary.CategorySettingsPage.OnFragmentNavigation 63	AwesomeLibrary.OrderSettingsPage.authorId 80
AwesomeLibrary.CategorySettingsPage.OnNavigatedFrom 64	AwesomeLibrary.OrderSettingsPage.categoryId 80
AwesomeLibrary.CategorySettingsPage.OnNavigatedTo 64	AwesomeLibrary.OrderSettingsPage.OnFragmentNavigation 81
AwesomeLibrary.csproj 135	AwesomeLibrary.OrderSettingsPage.OnNavigatedFrom 82
AwesomeLibrary.FontFamilySettingsPage 65	AwesomeLibrary.OrderSettingsPage.OnNavigatedTo 82
AwesomeLibrary.FontFamilySettingsPage.authorId 66	AwesomeLibrary.OrderSettingsPage.OrderSettingsPage 80
AwesomeLibrary.FontFamilySettingsPage.FontFamilySettingsPage 65	AwesomeLibrary.OrderSettingsPage.pageName 81
AwesomeLibrary.FontFamilySettingsPage.OnFragmentNavigation 66	AwesomeLibrary.OrderStyleSettingsPage 82
AwesomeLibrary.FontFamilySettingsPage.OnNavigatedFrom 67	

AwesomeLibrary.OrderStyleSettingsPage.authorId 83	AwesomeLibrary.Resources.AppResources.AuthorNameChangeSuccess 9
AwesomeLibrary.OrderStyleSettingsPage.categoryId 84	AwesomeLibrary.Resources.AppResources.AuthorOrderBy 9
AwesomeLibrary.OrderStyleSettingsPage.OnFragmentNavigation 84	AwesomeLibrary.Resources.AppResources.AuthorOrderStyle 9
AwesomeLibrary.OrderStyleSettingsPage.OnNavigatedFrom 85	AwesomeLibrary.Resources.AppResources.AuthorOrderStyleChangeSuccess 9
AwesomeLibrary.OrderStyleSettingsPage.OnNavigatedTo 85	AwesomeLibrary.Resources.AppResources.AuthorOrderTypeChangeSuccess 9
AwesomeLibrary.OrderStyleSettingsPage.OrderStyleSettingsPage 83	AwesomeLibrary.Resources.AppResources.AuthorSettings 9
AwesomeLibrary.OrderStyleSettingsPage.pageName 84	AwesomeLibrary.Resources.AppResources.Background 9
AwesomeLibrary.PopupAddChange 85	AwesomeLibrary.Resources.AppResources.BackgroundColor 10
AwesomeLibrary.PopupAddChange.PopupAddChange 86	AwesomeLibrary.Resources.AppResources.BackgroundColorChangeSuccess 10
AwesomeLibrary.Resources 2	AwesomeLibrary.Resources.AppResources.BackgroundImage 10
AwesomeLibrary.Resources namespace 2	AwesomeLibrary.Resources.AppResources.BackgroundImageChangeSuccess 10
Classes 2	AwesomeLibrary.Resources.AppResources.BackgroundImageRemoveSuccess 10
AwesomeLibrary.Resources.AppResources 2	AwesomeLibrary.Resources.AppResources.BestBook 10
AwesomeLibrary.Resources.AppResources.About 7	AwesomeLibrary.Resources.AppResources.Black 10
AwesomeLibrary.Resources.AppResources.AboutTheApp 7	AwesomeLibrary.Resources.AppResources.Blue 10
AwesomeLibrary.Resources.AppResources.AboutTheAppText 7	AwesomeLibrary.Resources.AppResources.BookComment 10
AwesomeLibrary.Resources.AppResources.AboutTheAwesomeLibrary 7	AwesomeLibrary.Resources.AppResources.BookCount 11
AwesomeLibrary.Resources.AppResources.AddAuthor 7	AwesomeLibrary.Resources.AppResources.BookDeleteSuccess 11
AwesomeLibrary.Resources.AppResources.AddBook 8	AwesomeLibrary.Resources.AppResources.BookList 11
AwesomeLibrary.Resources.AppResources.AddCategory 8	AwesomeLibrary.Resources.AppResources.BookName 11
AwesomeLibrary.Resources.AppResources.AppResources 7	AwesomeLibrary.Resources.AppResources.BookNameMustBe 11
AwesomeLibrary.Resources.AppResources.Arabic 8	AwesomeLibrary.Resources.AppResources.BookOrderBy 11
AwesomeLibrary.Resources.AppResources.Ascending 8	AwesomeLibrary.Resources.AppResources.BookOrderStyle 11
AwesomeLibrary.Resources.AppResources.AuthorAddSuccess 8	AwesomeLibrary.Resources.AppResources.BookOrderStyleChangeSuccess 11
AwesomeLibrary.Resources.AppResources.AuthorAlreadySameCategory 8	AwesomeLibrary.Resources.AppResources.BookOrderTypeChangeSuccess 11
AwesomeLibrary.Resources.AppResources.AuthorCategoryAddedSuccess 8	
AwesomeLibrary.Resources.AppResources.AuthorDeleteSuccess 8	
AwesomeLibrary.Resources.AppResources.AuthorExists 8	
AwesomeLibrary.Resources.AppResources.AuthorList 9	
AwesomeLibrary.Resources.AppResources.AuthorName 9	

AwesomeLibrary.Resources.AppResources.BookPerDay 12	AwesomeLibrary.Resources.AppResources.DeleteCategoryQ uestion 15
AwesomeLibrary.Resources.AppResources.BookRating 12	AwesomeLibrary.Resources.AppResources.Descending 15
AwesomeLibrary.Resources.AppResources.BookSaveSucces s 12	AwesomeLibrary.Resources.AppResources.English 15
AwesomeLibrary.Resources.AppResources.Brown 12	AwesomeLibrary.Resources.AppResources.EnterAuthorName 15
AwesomeLibrary.Resources.AppResources.Cancel 12	AwesomeLibrary.Resources.AppResources.EnterCategoryNa me 15
AwesomeLibrary.Resources.AppResources.Categories 12	AwesomeLibrary.Resources.AppResources.ExitApp 15
AwesomeLibrary.Resources.AppResources.CategoryAddSucc ess 12	AwesomeLibrary.Resources.AppResources.ExitAppQuestion 15
AwesomeLibrary.Resources.AppResources.CategoryDeleteS uccess 12	AwesomeLibrary.Resources.AppResources.FinishDate 15
AwesomeLibrary.Resources.AppResources.CategoryExists 12	AwesomeLibrary.Resources.AppResources.Font 15
AwesomeLibrary.Resources.AppResources.CategoryName 13	AwesomeLibrary.Resources.AppResources.FontFamily 16
AwesomeLibrary.Resources.AppResources.CategoryNameCh angeSuccess 13	AwesomeLibrary.Resources.AppResources.FontFamilyChang eSuccess 16
AwesomeLibrary.Resources.AppResources.CategoryOrderBy 13	AwesomeLibrary.Resources.AppResources.FontSize 16
AwesomeLibrary.Resources.AppResources.CategoryOrderSty le 13	AwesomeLibrary.Resources.AppResources.FontSizeChange Success 16
AwesomeLibrary.Resources.AppResources.CategoryOrderSty leChangeSuccess 13	AwesomeLibrary.Resources.AppResources.French 16
AwesomeLibrary.Resources.AppResources.CategoryOrderTy peChangeSuccess 13	AwesomeLibrary.Resources.AppResources.GeneralSettings 16
AwesomeLibrary.Resources.AppResources.CategorySettings 13	AwesomeLibrary.Resources.AppResources.German 16
AwesomeLibrary.Resources.AppResources.Chinese 13	AwesomeLibrary.Resources.AppResources.Gray 16
AwesomeLibrary.Resources.AppResources.ContactWithUs 13	AwesomeLibrary.Resources.AppResources.Green 16
AwesomeLibrary.Resources.AppResources.CreationDate 14	AwesomeLibrary.Resources.AppResources.Italian 17
AwesomeLibrary.Resources.AppResources.Culture 14	AwesomeLibrary.Resources.AppResources.Japanese 17
AwesomeLibrary.Resources.AppResources.Date 14	AwesomeLibrary.Resources.AppResources.Language 17
AwesomeLibrary.Resources.AppResources.DeleteAuthor 14	AwesomeLibrary.Resources.AppResources.LanguageWarnin g 17
AwesomeLibrary.Resources.AppResources.DeleteAuthorQue stion 14	AwesomeLibrary.Resources.AppResources.ModificationDate 17
AwesomeLibrary.Resources.AppResources.DeleteBook 14	AwesomeLibrary.Resources.AppResources.MostReadAuthor 17
AwesomeLibrary.Resources.AppResources.DeleteBookQuesti on 14	AwesomeLibrary.Resources.AppResources.MostReadCatego ry 17
AwesomeLibrary.Resources.AppResources.DeleteCategory 14	AwesomeLibrary.Resources.AppResources.MostReadPublish er 17
	AwesomeLibrary.Resources.AppResources.Name 17
	AwesomeLibrary.Resources.AppResources.OK 18

AwesomeLibrary.Resources.AppResources.OneDrive 18	AwesomeLibrary.Resources.AppResources.SelectLanguage 21
AwesomeLibrary.Resources.AppResources.OneDriveSyncCompleted 18	AwesomeLibrary.Resources.AppResources.SendWithApp 21
AwesomeLibrary.Resources.AppResources.Orange 18	AwesomeLibrary.Resources.AppResources.SendWithMail 21
AwesomeLibrary.Resources.AppResources.OtherSettings 18	AwesomeLibrary.Resources.AppResources.SendWithSMS 21
AwesomeLibrary.Resources.AppResources.PageNumber 18	AwesomeLibrary.Resources.AppResources.Settings 21
AwesomeLibrary.Resources.AppResources.PagePerDay 18	AwesomeLibrary.Resources.AppResources.ShareBook 22
AwesomeLibrary.Resources.AppResources.Persian 18	AwesomeLibrary.Resources.AppResources.Spanish 22
AwesomeLibrary.Resources.AppResources.Portuguese 18	AwesomeLibrary.Resources.AppResources.StartDate 22
AwesomeLibrary.Resources.AppResources.PublisherName 19	AwesomeLibrary.Resources.AppResources.Statistics 22
AwesomeLibrary.Resources.AppResources.Purple 19	AwesomeLibrary.Resources.AppResources.SuccessfulResetSettings 22
AwesomeLibrary.Resources.AppResources.Rate 19	AwesomeLibrary.Resources.AppResources.Sync 22
AwesomeLibrary.Resources.AppResources.Red 19	AwesomeLibrary.Resources.AppResources.Synchronizing 22
AwesomeLibrary.Resources.AppResources.RemoveBackgroundImage 19	AwesomeLibrary.Resources.AppResources.SyncOnOneFile 22
AwesomeLibrary.Resources.AppResources.ResetSettings 19	AwesomeLibrary.Resources.AppResources.SystemFault 22
AwesomeLibrary.Resources.AppResources.ResourceFlowDirection 19	AwesomeLibrary.Resources.AppResources.Thai 23
AwesomeLibrary.Resources.AppResources.ResourceLanguage 19	AwesomeLibrary.Resources.AppResources.TotalBookCount 23
AwesomeLibrary.Resources.AppResources.ResourceManager 19	AwesomeLibrary.Resources.AppResources.Turkish 23
AwesomeLibrary.Resources.AppResources.Russian 20	AwesomeLibrary.Resources.AppResources.WorstBook 23
AwesomeLibrary.Resources.AppResources.Sanskrit 20	AwesomeLibrary.Resources.AppResources.Yellow 23
AwesomeLibrary.Resources.AppResources.Save 20	AwesomeLibrary.SearchPage 86
AwesomeLibrary.Resources.AppResources.Search 20	AwesomeLibrary.SearchPage.OnNavigatedFrom 87
AwesomeLibrary.Resources.AppResources.SearchCompleted 20	AwesomeLibrary.SearchPage.OnNavigatedTo 87
AwesomeLibrary.Resources.AppResources.SearchResults 20	AwesomeLibrary.SearchPage.SearchPage 86
AwesomeLibrary.Resources.AppResources.SearchTrimFault 20	AwesomeLibrary.Sln 135
AwesomeLibrary.Resources.AppResources.Select 20	AwesomeLibrary.StatisticsPage 87
AwesomeLibrary.Resources.AppResources.SelectBackgroundColor 21	AwesomeLibrary.StatisticsPage.OnNavigatedFrom 88
AwesomeLibrary.Resources.AppResources.Selected 21	AwesomeLibrary.StatisticsPage.OnNavigatedTo 88
AwesomeLibrary.Resources.AppResources.SelectFontFamily 21	AwesomeLibrary.StatisticsPage.StatisticsPage 88
AwesomeLibrary.Resources.AppResources.SelectFontSize 21	AwesomeLibraryDataContext class 41
	about AwesomeLibraryDataContext class 41
	AppSettings 42
	Authors 42
	AwesomeLibraryDataContext 42
	BookAuthors 43
	Books 43
	Categories 43
	CategoryAuthors 43

ConnectionString 43
 AwesomeLibraryDataContext.cs 135

B

BackgroundColorSettingsPage class 43
 about BackgroundColorSettingsPage class 43
 authorId 45
 BackgroundColorSettingsPage 44
 OnFragmentNavigation 45
 OnNavigatedFrom 45
 OnNavigatedTo 45
 BackgroundColorSettingsPage.xaml.cs 136
 Book class 46
 about Book class 46
 BookAuthorName 46
 BookCategoryId 47
 BookCategoryName 47
 BookComment 47
 BookGuid 47
 BookId 47
 BookName 47
 BookNameRating 48
 BookPageNumber 48
 BookPublisherName 48
 BookRating 48
 CreationDate 48
 ModificationDate 48
 ReadFinishDate 49
 ReadStartDate 49
 Book.cs 139
 BookAuthor class 49
 about BookAuthor class 49
 AuthorId 49
 BookAuthorId 50
 BookId 50
 BookAuthor.cs 140
 BookPage class 50
 about BookPage class 50
 authorId 52
 authorName 52

bookId 52
 BookPage 51
 categoryId 52
 categoryName 52
 flag 52
 isFilled 53
 OnFragmentNavigation 53
 OnNavigatedFrom 54
 OnNavigatedTo 54
 pageName 53
 ratingValue 53
 BookPage.xaml.cs 141

C

Category class 55
 about Category class 55
 AuthorOrderBy 56
 AuthorOrderStyle 56
 CategoryBookCount 56
 CategoryId 56
 CategoryName 56
 CategoryNameCount 57
 CreationDate 57
 ModificationDate 57
 Category.cs 151
 CategoryAuthor class 57
 about CategoryAuthor class 57
 AuthorId 58
 CategoryAuthorId 58
 CategoryId 58
 CategoryAuthor.cs 152
 CategoryPage class 58
 about CategoryPage class 58
 categoryId 59
 CategoryPage 59
 oldCategoryName 59
 OnFragmentNavigation 60
 OnNavigatedFrom 61
 OnNavigatedTo 62
 popup 60

CategoryPage.xaml.cs 153

CategorySettingsPage class 62

about CategorySettingsPage class 62

categoryId 63

CategorySettingsPage 63

OnFragmentNavigation 63

OnNavigatedFrom 64

OnNavigatedTo 64

CategorySettingsPage.xaml.cs 161

F

Files 89

FontFamilySettingsPage class 65

about FontFamilySettingsPage class 65

authorId 66

FontFamilySettingsPage 65

OnFragmentNavigation 66

OnNavigatedFrom 67

OnNavigatedTo 67

FontFamilySettingsPage.xaml.cs 164

FontSizeSettingsPage class 67

about FontSizeSettingsPage class 67

authorId 68

FontSizeSettingsPage 68

OnFragmentNavigation 68

OnNavigatedFrom 69

OnNavigatedTo 69

FontSizeSettingsPage.xaml.cs 167

G

GeneralSettingsPage class 69

about GeneralSettingsPage class 69

CreateDirectoryAsync 73

DesignFileName 74

GeneralSettingsPage 70

OnNavigatedFrom 74

OnNavigatedTo 75

signIn 73

GeneralSettingsPage.xaml.cs 169

L

LanguageSettingsPage class 75

about LanguageSettingsPage class 75

LanguageSettingsPage 75

OnNavigatedFrom 76

OnNavigatedTo 76

LanguageSettingsPage.xaml.cs 182

LocalizedStrings class 76

about LocalizedStrings class 76

LocalizedResources 77

LocalizedStrings.cs 186

M

MainPage class 77

about MainPage class 77

MainPage 78

OnNavigatedFrom 79

OnNavigatedTo 79

popup 78

MainPage.xaml.cs 186

O

OrderSettingsPage class 79

about OrderSettingsPage class 79

authorId 80

categoryId 80

OnFragmentNavigation 81

OnNavigatedFrom 82

OnNavigatedTo 82

OrderSettingsPage 80

pageName 81

OrderSettingsPage.xaml.cs 192

OrderStyleSettingsPage class 82

about OrderStyleSettingsPage class 82

authorId 83

categoryId 84

OnFragmentNavigation 84

OnNavigatedFrom 85

OnNavigatedTo 85

OrderStyleSettingsPage 83

pageName 84

OrderStyleSettingsPage.xaml.cs 197

P

PopupAddChange class 85

about PopupAddChange class 85

PopupAddChange 86

PopupAddChange.xaml.cs 201

S

SearchPage class 86

about SearchPage class 86

OnNavigatedFrom 87

OnNavigatedTo 87

SearchPage 86

SearchPage.xaml.cs 202

StatisticsPage class 87

about StatisticsPage class 87

OnNavigatedFrom 88

OnNavigatedTo 88

StatisticsPage 88

StatisticsPage.xaml.cs 207