Awesome Library

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

Table of Contents

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

C#

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder",
"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.Editor
BrowsableState.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.Editor
BrowsableState.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

PhoneApplicationPage
AwesomeLibrary.AboutPage

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

```
public AboutPage()
```

```
{
    InitializeComponent();
    SetBackgroundColor();

    ApplicationBar = new ApplicationBar();

    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
    button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
    button2.Text = AppResources.ContactWithUs;
    ApplicationBar.Buttons.Add(button2);
    button2.Click += new EventHandler(SendMailButton_Click);

    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
    button3.IconUri = new Uri("/Assets/Rate.png", UriKind.Relative);
    button3.Text = AppResources.Rate;
    ApplicationBar.Buttons.Add(button3);
    button3.Click += new EventHandler(RateButton_Click);

    lblAboutTheApp.Text = AppResources.AboutTheApp;
    //txtAbout2.Text = AppResources.AboutTheAppText;
    //var paragraph = new Paragraph();
    //paragraph.Inlines.Add(AppResources.AboutTheAppText);
    //txtAbout.Blocks.Add(paragraph);
    txtAbout.Text = AppResources.AboutTheAppText;
    //txtAbout.IsEnabled = false;
    txtAbout.IsReadOnly = true;
    //this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
}
```

1.1.2.2 AddCategoryPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.AddCategoryPage

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

```
public AddCategoryPage()
{
    InitializeComponent();
    SetBackgroundColor();
}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    authorId = int.Parse(e.Fragment);
    using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single()
as Author;
        lstCategories.Items.Clear();
        lblAuthorName.Text = author.AuthorName;
        lblCategories.Text = AppResources.Categories;
        var categories = context.Categories;
        lstCategories.ItemsSource = categories;
        lstCategories.DisplayMemberPath = "CategoryName";
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

1.1.2.3 App Class

Class Hierarchy

Application
AwesomeLibrary.App

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.
▤ S	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

```
public App()
{
    // Global handler for uncaught exceptions.
    UnhandledException += Application_UnhandledException;

    // Standard XAML initialization
    InitializeComponent();

    // ayarlardan temasi açik renk bile olsa
    // kapali gibi çalismasini saglayacak bir nuget paketi yüklendi
    // bu sorunu gideriyor
    ThemeManager.ToDarkTheme();

    // Phone-specific initialization
    InitializePhoneApplication();

    // Language display initialization
    InitializeLanguage();

    // Show graphics profiling information while debugging.
    if (Debugger.IsAttached)
    {
        // Display the current frame rate counters.
        Application.Current.Host.Settings.EnableFrameRateCounter = true;

        // Show the areas of the app that are being redrawn in each frame.
        //Application.Current.Host.Settings.EnableRedrawRegions = true;

        // Enable non-production analysis visualization mode,
        // which shows areas of a page that are handed off to GPU with a colored
overlay.
        //Application.Current.Host.Settings.EnableCacheVisualization = true;

        // Prevent the screen from turning off while under the debugger by disabling
        // the application's idle detection.
        // Caution:- Use this under debug mode only. Application that disables user
idle detection will continue to run
        // and consume battery power when the user is not using the phone.
        PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
    }

}
```

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

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

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

PhoneApplicationPage
AwesomeLibrary.AuthorPage

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

```
public AuthorPage()
{
    InitializeComponent();

    ApplicationBar = new ApplicationBar();

    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
    button1.Text = AppResources.AddBook;
    ApplicationBar.Buttons.Add(button1);
    button1.Click += new EventHandler(AddBookButton_Click);

    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
    button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
    button2.Text = AppResources.DeleteAuthor;
    ApplicationBar.Buttons.Add(button2);
    button2.Click += new EventHandler(DeleteAuthorButton_Click);

    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
    button3.Text = AppResources.AuthorSettings;
    ApplicationBar.Buttons.Add(button3);
    button3.Click += new EventHandler(AuthorSettingsButton_Click);

    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
    button4.IconUri = new Uri("/Assets/AddCategory.png", UriKind.Relative);
    button4.Text = AppResources.AddCategory;
    ApplicationBar.Buttons.Add(button4);
    button4.Click += new EventHandler(AddCategoryButton_Click);

    popup = new Popup();

}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    List<Book> books = new List<Book>();
    List<Book> booksOrdered = new List<Book>();

    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);

    lstBooks.Items.Clear();

    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {

        var appSettings = context.AppSettings.First();
        categoryId = appSettings.CurrentCategoryNumber;

        var author = context.Authors.Where(j => j.AuthorId.Equals(e.Fragment)).Single() as Author;
        authorId = author.AuthorId;

        var appSettings2 = context.AppSettings;
        foreach (var item in appSettings2)
        {
            item.CurrentAuthorNumber = authorId;
        }
        context.SubmitChanges();

        var authorBooks = context.BookAuthors.Where(j => j.AuthorId.Equals(e.Fragment)).ToList() as List<BookAuthor>;
        if (authorBooks.Count != 0)
        {
            foreach (var item in authorBooks)
            {
                try
                {
                    var book = context.Books.Where(j => j.BookCategoryId.Equals(categoryId) && j.BookId.Equals(item.BookId)).Single() as Book;
                    books.Add(book);
                }
                catch (Exception)
                {
                }
            }

        }


        string orderStyle = author.BookOrderStyle;

        switch (author.BookOrderBy)
        {
            case "NAME":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.BookName).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.BookName).ToList();
                }
                break;
            case "CDATE":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.CreationDate).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.CreationDate).ToList();
                }
                break;
            case "MDATE":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.ModificationDate).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.ModificationDate).ToList();
                }
                break;
            case "SDATE":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.ReadStartDate).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.ReadStartDate).ToList();
                }
                break;
            case "FDATE":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.ReadFinishDate).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.ReadFinishDate).ToList();
                }
                break;
            case "RATING":
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.BookRating).ToList();
                }
                else
                {
                    booksOrdered = books.OrderByDescending(j => j.BookRating).ToList();
                }
                break;
            default:
                if (orderStyle == "A")
                {
                    booksOrdered = books.OrderBy(j => j.BookName).ToList();
                }
                else
                {
                    booksOrdered = books.OrderBy(j => j.BookName).ToList();
                }
                break;
        }

        lblAuthorName.Text = author.AuthorName;
        lblBookList.Text = AppResources.BookList + " (" + author.AuthorName + ")";
        lstBooks.ItemsSource = booksOrdered;
        lstBooks.DisplayMemberPath = "BookNameRating";
        SetBackgroundColor();
        //lstNoteList.DisplayMemberPath = "NameCreation";
```

```
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

1.1.2.7 AuthorSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.AuthorSettingsPage

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

```
public AuthorSettingsPage()
{
    InitializeComponent();
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var appSettings = context.AppSettings.First();
        lblFontFamily.Text = AppResources.FontFamily + " (" + AppResources.Selected + ": " + appSettings.FontFamily + ")";
        lblFontSize.Text = AppResources.FontSize + " (" + AppResources.Selected + ": " + appSettings.FontSize + ")";
    }

    pvAuthorSettings.Title = AppResources.AuthorSettings;
    piFont.Header = AppResources.Font;
    piOtherSettings.Header = AppResources.OtherSettings;

    btnFontFamily.Content = AppResources.Select;
    btnFontSize.Content = AppResources.Select;
    btnBookOrder.Content = AppResources.Select;
    btnBookOrderStyle.Content = AppResources.Select;
}
```

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.

1.1.2.7.2.2 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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    using (var context = new 
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var author = context.Authors.Where(j => j.AuthorId.Equals(e.Fragment)).Single() 
as Author;
        authorId = author.AuthorId;
        var appSettings = context.AppSettings.First();
        categoryId = appSettings.CurrentCategoryNumber;
        string orderStyle = author.BookOrderStyle;

        if (author.BookOrderBy == "NAME")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected 
+ ": " + AppResources.Name + ")";
        }
        if (author.BookOrderBy == "CDATE")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected 
+ ": " + AppResources.CreationDate + ")";
        }
        if (author.BookOrderBy == "MDATE")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected 
+ ": " + AppResources.ModificationDate + ")";
        }
        if (author.BookOrderBy == "SDATE")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected 
+ ": " + AppResources.StartDate + ")";
        }
        if (author.BookOrderBy == "FDATE")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected 
+ ": " + AppResources.FinishDate + ")";
        }
        if (author.BookOrderBy == "RATING")
        {
            lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected
+ ": " + AppResources.BookRating + ")";
        }
        if (author.BookOrderStyle == "A")
        {
            lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
        }
        if (author.BookOrderStyle == "D")
        {
            lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
        }
        //lstNoteList.DisplayMemberPath = "NameCreation";
        SetBackgroundColor();
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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.

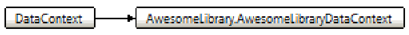
Body Source

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
public AwesomeLibraryDataContext(string connectionString)
: 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

PhoneApplicationPage
AwesomeLibrary.BackgroundColorSettingsPage

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

```
public BackgroundColorSettingsPage()
{
    InitializeComponent();

    lstBackgroundColor.Items.Clear();
    lstBackgroundColor.Items.Add(AppResources.Black);
    lstBackgroundColor.Items.Add(AppResources.Blue);
    lstBackgroundColor.Items.Add(AppResources.Brown);
    lstBackgroundColor.Items.Add(AppResources.Gray);
    lstBackgroundColor.Items.Add(AppResources.Green);
    lstBackgroundColor.Items.Add(AppResources.Orange);
    lstBackgroundColor.Items.Add(AppResources.Purple);
    lstBackgroundColor.Items.Add(AppResources.Red);
    lstBackgroundColor.Items.Add(AppResources.Yellow);
    lstBackgroundColor.SelectedIndex = -1;

    lblBackgroundColor.Text = AppResources.SelectBackgroundColor;
    lblGeneralSettings.Text = AppResources.GeneralSettings;

    SetBackgroundColor();
}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    authorId = int.Parse(e.Fragment);
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
        lblGeneralSettings.Text = AppResources.GeneralSettings;
        lblBackgroundColor.Text = AppResources.SelectFontSize;
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //SetBackgroundColor();
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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#

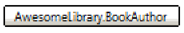
```
[Column]
public DateTime ReadStartDate;
```

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#

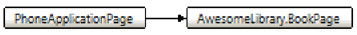
```
[Column]
public int BookId;
```

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

```
public BookPage()
{
    InitializeComponent();

    SetBackgroundColor();

    //pvAuthor.Title = authorName;
    piBookName.Header = AppResources.BookName;
    piComment.Header = AppResources.BookComment;
    piPublisherName.Header = AppResources.PublisherName;
    piRating.Header = AppResources.BookRating;
    piStartFinishDate.Header = AppResources.Date;
    lblStartDate.Text = AppResources.StartDate;
    lblFinishDate.Text = AppResources.FinishDate;
    piPageNumber.Header = AppResources.PageNumber;


    ApplicationBar = new ApplicationBar();

    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
    button1.Text = AppResources.Save;
    ApplicationBar.Buttons.Add(button1);
    button1.Click += new EventHandler(SaveButton_Click);

    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
    button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
    button2.Text = AppResources.SendWithMail;
    ApplicationBar.Buttons.Add(button2);
    button2.Click += new EventHandler(SendMailButton_Click);

    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
    button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
    button3.Text = AppResources.SendWithSMS;
    ApplicationBar.Buttons.Add(button3);
    button3.Click += new EventHandler(SendSMSButton_Click);

    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
    button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
    button4.Text = AppResources.ShareBook;
    ApplicationBar.Buttons.Add(button4);
    button4.Click += new EventHandler(ShareBookButton_Click);

    isFilled = false;

    ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
    menuItem1.Text = AppResources.DeleteBook;
    ApplicationBar.MenuItems.Add(menuItem1);
    menuItem1.Click += new EventHandler(DeleteBookMenuItem_Click);

}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    bookId = int.Parse(e.Fragment);
    if (pageName.Contains("/AuthorPage.xaml"))
    {
        isFilled = true;
    }
    else
    {
        //using (var context2 = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        //{
        //    var appSettings = context2.AppSettings; ;
        //    var book2 = context2.Books.Where(j => j.BookId.Equals(bookId)) as Book;
        //    var bookAuthor = context2.BookAuthors.Where(j =>
j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
        //    var bAuthor = bookAuthor.First();
        //    var author = context2.Authors.Where(j =>
j.AuthorId.Equals(bAuthor.AuthorId)) as Author;
        //    foreach (var item in appSettings)
        //    {
        //        item.CurrentAuthorNumber = author.AuthorId;
        //        item.CurrentCategoryNumber = book2.BookCategoryId;
        //    }
        //    context2.SubmitChanges();
        //    pvAuthor.Title = author.AuthorName;
        //}
    }
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var book = context.Books.Where(j => j.BookId.Equals(e.Fragment)).Single() as Book;

        txtBookName.Text = book.BookName == "" ? "" : book.BookName;
        txtPageNumber.Text = book.BookPageNumber.ToString() == "" ? "" : book.BookPageNumber.ToString();
        txtPublisherName.Text = book.BookPublisherName == "" ? "" : book.BookPublisherName;
        dtStart.Value = book.ReadStartDate == null ? DateTime.Now : book.ReadStartDate;
        dtFinish.Value = book.ReadFinishDate == null ? DateTime.Now : book.ReadFinishDate;
        rtRating.Value = book.BookRating == null ? 0 : book.BookRating;
        txtBookComment.Text = book.BookComment == "" ? "" : book.BookComment;
    }

    SetBackgroundColor();
    pvAuthor.SelectedIndex = 0;
    //pvAuthor.Name = authorName;
    txtBookName.Focus();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var appSettings = context.AppSettings.First();
        categoryId = appSettings.CurrentCategoryNumber;
        authorId = appSettings.CurrentAuthorNumber;

        // sayfanin font ayarlari için yapilan bir degisiklik
        FontFamily temp = new FontFamily(appSettings.FontFamily);
        double fontsize = double.Parse(appSettings.FontSize);
        txtBookComment.FontFamily = temp;
        txtBookComment.FontSize = fontsize;
        txtPageNumber.FontFamily = temp;
        txtPageNumber.FontSize = fontsize;
        txtBookName.FontFamily = temp;
        txtBookName.FontSize = fontsize;
        txtBookComment.FontFamily = temp;
        txtBookComment.FontSize = fontsize;
        txtPublisherName.FontFamily = temp;
        txtPublisherName.FontSize = fontsize;
        rtRating.Value = 5;

        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
        authorName = author.AuthorName;

        var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
        categoryName = category.CategoryName;
    }

    var lastPage = NavigationService.BackStack.FirstOrDefault();
    pageName = lastPage.Source.ToString();
    pvAuthor.SelectedIndex = 0;
    txtBookName.Focus();
    // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
    pvAuthor.Title = authorName;
    SetBackgroundColor();
}
```

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

PhoneApplicationPage
AwesomeLibrary.CategoryPage

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

```
public CategoryPage()
{
    InitializeComponent();

    ApplicationBar = new ApplicationBar();

    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
    button1.Text = AppResources.AddAuthor;
    ApplicationBar.Buttons.Add(button1);
    button1.Click += new EventHandler(AddAuthorButton_Click);

    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
    button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
    button2.Text = AppResources.DeleteCategory;
    ApplicationBar.Buttons.Add(button2);
    button2.Click += new EventHandler(DeleteCategoryButton_Click);

    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
    button3.Text = AppResources.CategorySettings;
    ApplicationBar.Buttons.Add(button3);
    button3.Click += new EventHandler(CategorySettingsButton_Click);

    SetBackgroundColor();
    popup = new Popup();
}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    List<Author> authors = new List<Author>();
    List<Author> authorsOrdered = new List<Author>();

    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var category = context.Categories.Where(j => j.CategoryId.Equals(e.Fragment)).Single() as Category;
        string orderStyle = category.AuthorOrderStyle;
        var categoryAuthor = context.CategoryAuthors.Where(j => j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryAuthor>;

        foreach (var item in categoryAuthor)
        {
            try
            {
                authors.Add(context.Authors.Where(j => j.AuthorId.Equals(item.AuthorId)).Single());
            }
            catch (Exception)
            {
            }

        }

        switch (category.AuthorOrderBy)
        {
            case "NAME":
                if (orderStyle == "A")
                {
                    authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
                }
                else
                {
                    authorsOrdered = authors.OrderByDescending(j => j.AuthorName).ToList();
                }
                break;
            case "BOOKCOUNT":
                if (orderStyle == "A")
                {
                    authorsOrdered = authors.OrderBy(j => j.AuthorBookCount).ToList();
                }
                else
                {
                    authorsOrdered = authors.OrderByDescending(j => j.AuthorBookCount).ToList();
                }
                break;
            case "CDATE":
                if (orderStyle == "A")
                {
                    authorsOrdered = authors.OrderBy(j => j.CreationDate).ToList();
                }
                else
                {
                    authorsOrdered = authors.OrderByDescending(j => j.CreationDate).ToList();
                }
                break;
            case "MDATE":
                if (orderStyle == "A")
                {
                    authorsOrdered = authors.OrderBy(j => j.ModificationDate).ToList();
                }
                else
                {
                    authorsOrdered = authors.OrderByDescending(j => j.ModificationDate).ToList();
                }
                break;
            default:
                if (orderStyle == "A")
                {
                    authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
                }
                else
                {
                    authorsOrdered = authors.OrderByDescending(j => j.AuthorName).ToList();
                }
                break;
        }

        lstAuthors.Items.Clear();
        categoryId = category.CategoryId;
        lblCategoryName.Text = category.CategoryName;
        lblAuthorList.Text = AppResources.AuthorList + " (" + category.CategoryName + ")";
        lstAuthors.ItemsSource = authorsOrdered;
        lstAuthors.DisplayMemberPath = "AuthorNameCount";
        SetBackgroundColor();
        //lstNoteList.DisplayMemberPath = "NameCreation";
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

1.1.2.16 CategorySettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.CategorySettingsPage

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

```
public CategorySettingsPage()
{
    InitializeComponent();

    pvCategorySettings.Title = AppResources.CategorySettings;

    piOtherSettings.Header = AppResources.OtherSettings;
    btnAuthorOrder.Content = AppResources.Select;
    btnAuthorOrderStyle.Content = AppResources.Select;
    SetBackgroundColor();

}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var category = context.Categories.Where(j =>
j.CategoryId.Equals(e.Fragment)).Single() as Category;
        string orderStyle = category.AuthorOrderStyle;
        categoryId = category.CategoryId;

        if (category.AuthorOrderBy == "NAME")
        {
            lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.Name + ")";
        }
        if (category.AuthorOrderBy == "BOOKCOUNT")
        {
            lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.BookCount + ")";
        }
        if (category.AuthorOrderBy == "CDATE")
        {
            lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.CreationDate + ")";
        }
        if (category.AuthorOrderBy == "MDATE")
        {
            lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.ModificationDate + ")";
        }
        if (category.AuthorOrderStyle == "A")
        {
            lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
        }
        if (category.AuthorOrderStyle == "D")
        {
            lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
        }
        //lstNoteList.DisplayMemberPath = "NameCreation";
        SetBackgroundColor();
    }
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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.

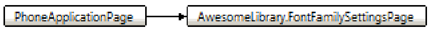
Body Source

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
public FontFamilySettingsPage()
{
    InitializeComponent();

    lstFontFamily.Items.Clear();
    lstFontFamily.Items.Add("Arial");
    lstFontFamily.Items.Add("Arial Black");
    lstFontFamily.Items.Add("Baskerville Old Face");
    lstFontFamily.Items.Add("Berlin Sans FB");
    lstFontFamily.Items.Add("Bookman Old Style");
    lstFontFamily.Items.Add("Calibri");
    lstFontFamily.Items.Add("Cambria");
    lstFontFamily.Items.Add("Candara");
    lstFontFamily.Items.Add("Comic Sans MS");
    lstFontFamily.Items.Add("Consolas");
    lstFontFamily.Items.Add("Constantia");
    lstFontFamily.Items.Add("Courier New");
    lstFontFamily.Items.Add("DokChampa");
    lstFontFamily.Items.Add("Ebrima");
    lstFontFamily.Items.Add("Georgia");
    lstFontFamily.Items.Add("Lucida Sans Unicode");
    lstFontFamily.Items.Add("Meiryo UI");
    lstFontFamily.Items.Add("Microsoft YaHei");
    lstFontFamily.Items.Add("Malgun Gothic");
    lstFontFamily.Items.Add("Segoe UI");
    lstFontFamily.Items.Add("Segoe WP");
    lstFontFamily.Items.Add("Tahoma");
    lstFontFamily.Items.Add("Trebuchet MS");
    lstFontFamily.Items.Add("Times New Roman");
    lstFontFamily.Items.Add("Verdana");
    lstFontFamily.SelectedIndex = -1;
}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    authorId = int.Parse(e.Fragment);
    using (var context = new 
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() 
as Author;
        lblAuthorName.Text = author.AuthorName;
        lblFontFamily.Text = AppResources.SelectFontFamily;
    }
    SetBackgroundColor();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

1.1.2.18 FontSizeSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.FontSizeSettingsPage

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

```
public FontSizeSettingsPage()
{
    InitializeComponent();

    lstFontSize.Items.Clear();
    lstFontSize.Items.Add("14");
    lstFontSize.Items.Add("18");
    lstFontSize.Items.Add("22");
    lstFontSize.Items.Add("26");
    lstFontSize.Items.Add("28");
    lstFontSize.Items.Add("30");
    lstFontSize.Items.Add("32");
    lstFontSize.Items.Add("34");
    lstFontSize.Items.Add("36");
    lstFontSize.Items.Add("38");
    lstFontSize.Items.Add("40");
    lstFontSize.Items.Add("42");
    lstFontSize.Items.Add("44");
    lstFontSize.Items.Add("64");
    lstFontSize.Items.Add("72");
    lstFontSize.SelectedIndex = -1;
}
```

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

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

```
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    authorId = int.Parse(e.Fragment);
    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
    {
        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
        lblAuthorName.Text = author.AuthorName;
        lblFontSize.Text = AppResources.SelectFontSize;
    }
    SetBackgroundColor();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

1.1.2.19 GeneralSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.GeneralSettingsPage

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

```
public GeneralSettingsPage()
{
    InitializeComponent();
    InitializePage();

    pvGeneralSettings.Title = AppResources.GeneralSettings;

    piLanguage.Header = AppResources.Language;
    piSync.Header = AppResources.Sync;
    piOtherSettings.Header = AppResources.OtherSettings;
    piBackground.Header = AppResources.Background;

    //lblOneDrive.Text = AppResources.OneDrive;

    btnCategoryOrder.Content = AppResources.Select;
    btnCategoryOrderStyle.Content = AppResources.Select;
    btnLanguage.Content = AppResources.Select;
    btnBackgroundColor.Content = AppResources.Select;
    //btnOneDrive.Content = AppResources.Login;
    //btnOneDrive.SignInText = AppResources.SignIn;
    //btnOneDrive.SignOutText = AppResources.SignOut;
    btnOneDriveSync.Content = AppResources.Sync;
    lblOneDrive.Text = AppResources.OneDrive;
    txtSyncronizing.Text = AppResources.Synchronizing;

    pbSync.Visibility = Visibility.Collapsed;
    txtSyncronizing.Visibility = Visibility.Collapsed;
    txtSyncronizing.BorderBrush = this.LayoutRoot.Background;

    btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
    lblBackgroundImage.Text = AppResources.BackgroundImage;
    btnBackgroundImage.Content = AppResources.Select;
    btnResetSettings.Content = AppResources.ResetSettings;
```

```

        btnOneDriveSync.IsEnabled = false;
        cbSync.Content = AppResources.SyncOnOneFile;
        cbSync.IsEnabled = false;
        btnOneDrive.Content = "Sign In";

        SetBackgroundColor();

        using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var appSettings = context.AppSettings.First() as AppSettings;
            if (appSettings.AppLangName == "EN")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.English + ")";
            }
            if (appSettings.AppLangName == "TR")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Turkish + ")";
            }
            if (appSettings.AppLangName == "DE")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.German + ")";
            }
            if (appSettings.AppLangName == "ES")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Spanish + ")";
            }

            if (appSettings.AppLangName == "PT")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Portuguese + ")";
            }
            if (appSettings.AppLangName == "AR")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Arabic + ")";
            }
            if (appSettings.AppLangName == "FA")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Persian + ")";
            }
            if (appSettings.AppLangName == "IT")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Italian + ")";
            }
            if (appSettings.AppLangName == "FR")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.French + ")";
            }
            if (appSettings.AppLangName == "RU")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Russian + ")";
            }
            if (appSettings.AppLangName == "ZH")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Chinese + ")";
            }
            if (appSettings.AppLangName == "JA")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Japanese + ")";
            }
            if (appSettings.AppLangName == "SA")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Sanskrit + ")";
            }
            if (appSettings.AppLangName == "TH")
            {
                lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Thai + ")";
            }


            if (appSettings.CategoryOrderBy == "NAME")
            {
                lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.Name + ")";
            }
            if (appSettings.CategoryOrderBy == "CDATE")
            {
                lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.CreationDate + ")";
            }
            if (appSettings.CategoryOrderBy == "MDATE")
            {
                lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.ModificationDate + ")";
            }
            if (appSettings.CategoryOrderBy == "BOOKCOUNT")
            {
                lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.BookCount + ")";
            }
            if (appSettings.CategoryOrderStyle == "A")
            {
                lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
            }
            if (appSettings.CategoryOrderStyle == "D")
            {
                lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
            }
            if (appSettings.AppBackgroundColor == "BLA")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Black + ")";
            }
            if (appSettings.AppBackgroundColor == "BLU")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Blue + ")";
            }
            if (appSettings.AppBackgroundColor == "BRO")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Brown + ")";
            }
            if (appSettings.AppBackgroundColor == "RED")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Red + ")";
            }
            if (appSettings.AppBackgroundColor == "GRE")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Green + ")";
            }
            if (appSettings.AppBackgroundColor == "YEL")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Yellow + ")";
            }
            if (appSettings.AppBackgroundColor == "GRA")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Gray + ")";
            }
            if (appSettings.AppBackgroundColor == "ORA")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Orange + ")";
            }
            if (appSettings.AppBackgroundColor == "PUR")
            {
                lblBackgroundColor.Text = AppResources.BackgroundColor + " (" +
AppResources.Selected + ": " + AppResources.Purple + ")";
            }
        }
}
```

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 asyncstaticTask<string> CreateDirectoryAsync(LiveConnectClient client, string
folderName, string parentFolder);
```

Description

This is CreateDirectoryAsync, a member of class GeneralSettingsPage.

Body Source

```
public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
string folderName, string parentFolder)
        {
            string folderId = null;

            // Retrieves all the directories.
            var queryFolder = parentFolder + "/files?filter=folders,albums";
            var opResult = await client.GetAsync(queryFolder);
            dynamic result = opResult.Result;

            foreach (dynamic folder in result.data)
            {
                // Checks if current folder has the passed name.
                if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
                {
                    folderId = folder.id;
                    break;
                }
            }

            if (folderId == null)
            {
                // Directory hasn't been found, so creates it using the PostAsync method.
                var folderData = new Dictionary<string, object>();
                folderData.Add("name", folderName);
                opResult = await client.PostAsync(parentFolder, folderData);
                result = opResult.Result;

                // Retrieves the id of the created folder.
                folderId = result.id;
            }

            return folderId;
        }
```

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

```
public string DesignFileName(string fileName)
{
    fileName = fileName.Replace(":", ".");
    fileName = fileName.Replace("?", ".");
    fileName = fileName.Replace("\"", ".");
    fileName = fileName.Replace("/", ".");
    fileName = fileName.Replace("<", ".");
    fileName = fileName.Replace(">", ".");
    fileName = fileName.Replace("|", ".");
    fileName = fileName.Replace("*", ".");
    return fileName;
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    SetBackgroundColor();
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

1.1.2.20 LanguageSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.LanguageSettingsPage

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

```
public LanguageSettingsPage()
{
    InitializeComponent();

    lstLanguage.Items.Clear();
    lstLanguage.Items.Add(AppResources.English);
    lstLanguage.Items.Add(AppResources.Turkish);
    lstLanguage.Items.Add(AppResources.German);
    //lstLanguage.Items.Add(AppResources.Spanish);
    lstLanguage.Items.Add(AppResources.Russian);
    lstLanguage.Items.Add(AppResources.Arabic);
    lstLanguage.Items.Add(AppResources.Persian);
    lstLanguage.Items.Add(AppResources.Chinese);
    lstLanguage.Items.Add(AppResources.Italian);
    lstLanguage.Items.Add(AppResources.French);
    lstLanguage.Items.Add(AppResources.Japanese);
    lstLanguage.Items.Add(AppResources.Spanish);
    lstLanguage.Items.Add(AppResources.Sanskrit);
    lstLanguage.Items.Add(AppResources.Thai);

    lstLanguage.SelectedIndex = -1;
    lblLanguage.Text = AppResources.SelectLanguage;
    lblGeneralSettings.Text = AppResources.GeneralSettings;

    SetBackgroundColor();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    SetBackgroundColor();
}
```

1.1.2.21 LocalizedStrings Class

Provides access to string resources.

Class Hierarchy

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#

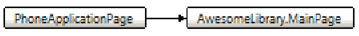
```
public AppResources LocalizedResources;
```

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

```
public MainPage()
{
    InitializeComponent();


    ApplicationBar = new ApplicationBar();

    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
    button1.Text = AppResources.AddCategory;
    ApplicationBar.Buttons.Add(button1);
    button1.Click += new EventHandler(AddCategoryButton_Click);

    ApplicationBarIconButton button2 = new ApplicationBarIconButton();
    button2.IconUri = new Uri("/Assets/Search.png", UriKind.Relative);
    button2.Text = AppResources.Search;
    ApplicationBar.Buttons.Add(button2);
    button2.Click += new EventHandler(SearchButton_Click);

    ApplicationBarIconButton button3 = new ApplicationBarIconButton();
    button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
    button3.Text = AppResources.Settings;
    ApplicationBar.Buttons.Add(button3);
    button3.Click += new EventHandler(SettingsButton_Click);

    ApplicationBarIconButton button4 = new ApplicationBarIconButton();
    button4.IconUri = new Uri("/Assets/Statistics.png", UriKind.Relative);
    button4.Text = AppResources.Statistics;
    ApplicationBar.Buttons.Add(button4);
    button4.Click += new EventHandler(StatisticsButton_Click);

    ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
    menuItem1.Text = AppResources.About;
    ApplicationBar.MenuItems.Add(menuItem1);
    menuItem1.Click += new EventHandler(AboutMenuItem_Click);

    lblCategories.Text = AppResources.Categories;
    // Sample code to localize the ApplicationBar
    //BuildLocalizedApplicationBar();

    SetBackgroundColor();

    popup = new Popup();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

1.1.2.23 OrderSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.OrderSettingsPage

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

```
public OrderSettingsPage()
{
    InitializeComponent();
    SetBackgroundColor();
}
```

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

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    lstOrderBy.Items.Clear();
    if (pageName.Contains("/CategorySettingsPage.xaml"))
    {
        categoryId = int.Parse(e.Fragment);
        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var category = context.Categories.Where(j =>
j.CategoryId.Equals(categoryId)).Single() as Category;
            lblSettings.Text = AppResources.CategorySettings + " (" +
category.CategoryName + ")" ;
            lblOrderBy.Text = AppResources.AuthorOrderBy;
            lstOrderBy.Items.Add(AppResources.Name);
            lstOrderBy.Items.Add(AppResources.BookCount);
            lstOrderBy.Items.Add(AppResources.CreationDate);
            lstOrderBy.Items.Add(AppResources.ModificationDate);
        }
    }
    else
    {
        authorId = int.Parse(e.Fragment);
        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
            lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName +
")" ;
            lblOrderBy.Text = AppResources.BookOrderBy;
            lstOrderBy.Items.Add(AppResources.Name);
            lstOrderBy.Items.Add(AppResources.CreationDate);
            lstOrderBy.Items.Add(AppResources.ModificationDate);
            lstOrderBy.Items.Add(AppResources.StartDate);
            lstOrderBy.Items.Add(AppResources.FinishDate);
            lstOrderBy.Items.Add(AppResources.BookRating);
        }
    }
    lstOrderBy.SelectedIndex = -1;
    SetBackgroundColor();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor bu bölüm
    var lastPage = NavigationService.BackStack.FirstOrDefault();
    pageName = lastPage.Source.ToString();
    lstOrderBy.Items.Clear();
    if (pageName.Contains("/GeneralSettingsPage.xaml"))
    {
        lblSettings.Text = AppResources.GeneralSettings;
        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var appSettings =
                context.AppSettings.First();
            lblOrderBy.Text = AppResources.CategoryOrderBy;

            lstOrderBy.Items.Add(AppResources.Name);
            lstOrderBy.Items.Add(AppResources.BookCount);
            lstOrderBy.Items.Add(AppResources.CreationDate);
            lstOrderBy.Items.Add(AppResources.ModificationDate);

        }
    }
}
```

1.1.2.24 OrderStyleSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeLibrary.OrderStyleSettingsPage

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

```
public OrderStyleSettingsPage()
{
    InitializeComponent();

    lstOrderStyle.Items.Clear();

    lstOrderStyle.Items.Add(AppResources.Ascending);
    lstOrderStyle.Items.Add(AppResources.Descending);

    lstOrderStyle.SelectedIndex = -1;

    SetBackgroundColor();
}
```

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.

1.1.2.24.2.2 OrderStyleSettingsPage.categoryId Field

C#

```
public int categoryId;
```

Description

This is categoryId, a member of class OrderStyleSettingsPage.

1.1.2.24.2.3 OrderStyleSettingsPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class OrderStyleSettingsPage.

1.1.2.24.3 OrderStyleSettingsPage Methods

1.1.2.24.3.1 OrderStyleSettingsPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class OrderStyleSettingsPage.

Body Source

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    if (pageName.Contains("/CategorySettingsPage.xaml"))
    {
        categoryId = int.Parse(e.Fragment);
        using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
            lblSettings.Text = AppResources.CategorySettings + " (" + category.CategoryName + ")";
            lblOrderStyle.Text = AppResources.AuthorOrderStyle;
        }
    }
    else
    {
        authorId = int.Parse(e.Fragment);
        using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
            lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName + ")";
            lblOrderStyle.Text = AppResources.BookOrderStyle;
        }
    }
    SetBackgroundColor();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

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

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor bu bölüm
    var lastPage = NavigationService.BackStack.FirstOrDefault();
    pageName = lastPage.Source.ToString();
    if (pageName.Contains("/GeneralSettingsPage.xaml"))
    {
        lblSettings.Text = AppResources.GeneralSettings;
        using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
        {
            var appSettings =
                context.AppSettings.First();
            lblOrderStyle.Text = AppResources.CategoryOrderStyle;
        }
    }
}
```

1.1.2.25 PopupAddChange Class

Class Hierarchy

UserControl
AwesomeLibrary.PopupAddChange

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

```
public PopupAddChange()
{
    InitializeComponent();
    SetPopupBackgroundColor();
}
```

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

```
public SearchPage()
{
    InitializeComponent();
    SetBackgroundColor();

    txtSearchResult.Text = AppResources.SearchResults;
    lblSearch.Text = AppResources.Search;
    //btnSearch.Content = AppResources.Search;
    //lstSearch.SelectedIndex = -1;
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
}
```

1.1.2.26.2.2 SearchPage.OnNavigatedTo Method

C#

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

Description

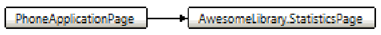
This is OnNavigatedTo, a member of class SearchPage.

Body Source

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

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

```
public StatisticsPage()
{
    InitializeComponent();
    lblStatistics.Text = AppResources.Statistics;
    SetBackgroundColor();
    SetStatistic();
}
```

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

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

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

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

```
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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using Microsoft.Phone.Tasks;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
{
    public partial class AboutPage : PhoneApplicationPage
    {
        public AboutPage()
        {
            InitializeComponent();
            SetBackgroundColor();

            ApplicationBar = new ApplicationBar();

            ApplicationBarIconButton button2 = new ApplicationBarIconButton();
            button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
            button2.Text = AppResources.ContactWithUs;
            ApplicationBar.Buttons.Add(button2);
            button2.Click += new EventHandler(SendMailButton_Click);

            ApplicationBarIconButton button3 = new ApplicationBarIconButton();
            button3.IconUri = new Uri("/Assets/Rate.png", UriKind.Relative);
            button3.Text = AppResources.Rate;
            ApplicationBar.Buttons.Add(button3);
            button3.Click += new EventHandler(RateButton_Click);

            lblAboutTheApp.Text = AppResources.AboutTheApp;
            //txtAbout2.Text = AppResources.AboutTheAppText;
            //var paragraph = new Paragraph();
            //paragraph.Inlines.Add(AppResources.AboutTheAppText);
            //txtAbout.Blocks.Add(paragraph);
            txtAbout.Text = AppResources.AboutTheAppText;
            //txtAbout.IsEnabled = false;
            txtAbout.IsReadOnly = true;
            //this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
            }
        }

        private void SendMailButton_Click(object sender, EventArgs e)
        {
            // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
            //EmailAddressChooserTask emailAddressChooserTask;
            //emailAddressChooserTask = new EmailAddressChooserTask();
            //emailAddressChooserTask.Completed += new EventHandler<EmailResult>(emailAddressChooserTask_Completed);
            //emailAddressChooserTask.Show();
            StringBuilder sb = new StringBuilder();
            EmailComposeTask emailComposeTask = new EmailComposeTask();


            sb.AppendLine();
            sb.AppendLine();
            sb.AppendLine(AppResources.SendWithApp);

            emailComposeTask.Subject = AppResources.AboutTheAwesomeLibrary;
            emailComposeTask.Body = sb.ToString();
            emailComposeTask.To = "coderserdar@outlook.com";
            emailComposeTask.Cc = "";
            emailComposeTask.Bcc = "";

            emailComposeTask.Show();
            //MessageBox.Show(AppResources.SuccessfulSendWithMail);
        }

        private void RateButton_Click(object sender, EventArgs e)
        {
            MarketplaceReviewTask marketplaceReviewTask = new MarketplaceReviewTask();
            marketplaceReviewTask.Show();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }
    }
}
```

Namespaces

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

1.2.2 AddCategoryPage.xaml.cs

This is file AddCategoryPage.xaml.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class AddCategoryPage : PhoneApplicationPage
    {

        public int authorId;

        public AddCategoryPage()
        {
            InitializeComponent();
            SetBackgroundColor();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            authorId = int.Parse(e.Fragment);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
                lstCategories.Items.Clear();
                lblAuthorName.Text = author.AuthorName;
                lblCategories.Text = AppResources.Categories;
                var categories = context.Categories;
                lstCategories.ItemsSource = categories;
                lstCategories.DisplayMemberPath = "CategoryName";
            }
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId, UriKind.Relative));
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void lstCategories_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            CategoryAuthor categoryAuthor2 = null;
            Category category = lstCategories.SelectedItem as Category;
            CategoryAuthor categoryAuthor = new CategoryAuthor();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                categoryAuthor.AuthorId = authorId;
                categoryAuthor.CategoryId = category.CategoryId;
                try
                {
                    categoryAuthor2 = context.CategoryAuthors.Where(j =>
j.CategoryId.Equals(categoryAuthor.CategoryId) &&
j.AuthorId.Equals(categoryAuthor.AuthorId)).Single() as CategoryAuthor;
                }
                catch (Exception)
                {
                    context.CategoryAuthors.InsertOnSubmit(categoryAuthor);
                    context.SubmitChanges();
                    var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Select(j => j);
                    foreach (var item in author)
                    {
                        item.ModificationDate = DateTime.Now;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.AuthorCategoryAddSuccess);
                }
                if (categoryAuthor2 != null)
                {
                    MessageBox.Show(AppResources.AuthorAlreadySameCategory);
                }
                else
                {

                }
            }
            this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId, UriKind.Relative));
        }
    }
}
```

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

```
?using System;
using System.Diagnostics;
using System.Globalization;
using System.Linq;
using System.Resources;
using System.Threading;
using System.Windows;
using System.Windows.Markup;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;
using System.Collections.Generic;
using Microsoft.Phone.Marketplace;


namespace AwesomeLibrary
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        ///


        // lisans bilgisi için gerekli olan bir sey
        private static bool _isTrial = true;

        private static LicenseInformation _licenseInfo = new LicenseInformation();
        public bool IsTrial
        {
            get
            {
                return _isTrial;
            }
        }


        public int categoryNumber;

        public static PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard XAML initialization
            InitializeComponent();

            // ayarlardan temasi açik renk bile olsa
            // kapali gibi çalismasini saglayacak bir nuget paketi yüklendi
            // bu sorunu gideriyor
            ThemeManager.ToDarkTheme();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Language display initialization
            InitializeLanguage();

            // Show graphics profiling information while debugging.
            if (Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Prevent the screen from turning off while under the debugger by disabling
                // the application's idle detection.
                // Caution:- Use this under debug mode only. Application that disables user idle detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode = IdleDetectionMode.Disabled;
            }

        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
        }

        // Code to execute when the application is activated (brought to foreground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {
        }

        // Code to execute when the application is deactivated (sent to background)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs e)
        {
            if (Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender, ApplicationUnhandledExceptionEventArgs e)
        {
            if (Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Handle reset requests for clearing the backstack
            RootFrame.Navigated += CheckForResetNavigation;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        private void CheckForResetNavigation(object sender, NavigationEventArgs e)
        {
            // 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
            if (e.NavigationMode == NavigationMode.Reset)
                RootFrame.Navigated += ClearBackStackAfterReset;
        }

        private void ClearBackStackAfterReset(object sender, NavigationEventArgs e)
        {
            // Unregister the event so it doesn't get called again
            RootFrame.Navigated -= ClearBackStackAfterReset;

            // Only clear the stack for 'new' (forward) and 'refresh' navigations
            if (e.NavigationMode != NavigationMode.New && e.NavigationMode != NavigationMode.Refresh)
                return;

            // For UI consistency, clear the entire page stack
            while (RootFrame.RemoveBackEntry() != null)
            {
                ; // do nothing
            }
        }

        #endregion

        // Initialize the app's font and flow direction as defined in its localized resource strings.
        //
        // To ensure that the font of your application is aligned with its supported languages and that the
        // FlowDirection for each of those languages follows its traditional direction, ResourceLanguage
        // and ResourceFlowDirection should be initialized in each resx file to match these values with that
        // file's culture. For example:
        //
        // AppResources.es-ES.resx
        //    ResourceLanguage's value should be "es-ES"
        //    ResourceFlowDirection's value should be "LeftToRight"
        //
        // AppResources.ar-SA.resx
        //     ResourceLanguage's value should be "ar-SA"
        //     ResourceFlowDirection's value should be "RightToLeft"
        //
        // For more info on localizing Windows Phone apps see http://go.microsoft.com/fwlink/?LinkId=262072.
        //
        private void InitializeLanguage()
        {
            try
            {
                // Set the font to match the display language defined by the
                // ResourceLanguage resource string for each supported language.
                //
                // Fall back to the font of the neutral language if the Display
                // language of the phone is not supported.
                //
                // If a compiler error is hit then ResourceLanguage is missing from
                // the resource file.
                RootFrame.Language = XmlLanguage.GetLanguage(AppResources.ResourceLanguage);

                // Set the FlowDirection of all elements under the root frame based
                // on the ResourceFlowDirection resource string for each
                // supported language.
                //
                // If a compiler error is hit then ResourceFlowDirection is missing from
                // the resource file.
                FlowDirection flow = (FlowDirection)Enum.Parse(typeof(FlowDirection), AppResources.ResourceFlowDirection);
                RootFrame.FlowDirection = flow;
            }
```

1

```
            catch
            {
                // If an exception is caught here it is most likely due to either
                // ResourceLangauge not being correctly set to a supported language
                // code or ResourceFlowDirection is set to a value other than LeftToRight
                // or RightToLeft.

                if (Debugger.IsAttached)
                {
                    Debugger.Break();
                }

                throw;
            }
        }

        private void CheckLicense()
        {
            // When debugging, we want to simulate a trial mode experience. The following conditional allows us to set the _isTrial
            // property to simulate trial mode being on or off.
#if DEBUG
            string message = "This sample demonstrates the implementation of a trial mode in an application." +
                               "Press 'OK' to simulate trial mode. Press 'Cancel' to run the application in normal mode.";
            if (MessageBox.Show(message, "Debug Trial",
                 MessageBoxButton.OKCancel) == MessageBoxResult.OK)
            {
                _isTrial = true;
            }
            else
            {
                _isTrial = false;
            }
#else
            _isTrial = _licenseInfo.IsTrial();
#endif
        }

        private void Application_Startup(object sender, StartupEventArgs e)
        {
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                if (!context.DatabaseExists())
                {
                    context.CreateDatabase();
                    DilAyariOlustur(context);
                }
                else
                {
                    using (var context2 = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {

                        AppSettings lang =
                            context2.AppSettings.First() as AppSettings;
                        string culture = "";
                        switch (lang.AppLangName)
                        {
                            case "TR":
                                culture = "tr";
                                break;
                            case "EN":
                                culture = "en";
                                break;
                            case "DE":
                                culture = "de";
                                break;
                            case "ES":
                                culture = "es";
                                break;
                            case "FR":
                                culture = "fr";
                                break;
                            case "IT":
                                culture = "it";
                                break;
                            case "AR":
                                culture = "ar";
                                break;
                            case "FA":
                                culture = "fa-IR";
                                break;
                            case "ZH":
                                culture = "zh";
                                break;
                            case "PT":
                                culture = "pt";
                                break;
                            case "RU":
                                culture = "ru";
                                break;
                            case "JA":
                                culture = "ja";
                                break;
                            case "SA":
                                culture = "sa";
                                break;
                            case "TH":
                                culture = "th";
                                break;
                            default:
                                culture = "tr-TR";
                                break;
                        }
                        CultureInfo newCulture = new CultureInfo(culture);
                        Thread.CurrentThread.CurrentCulture = newCulture;
                        Thread.CurrentThread.CurrentUICulture = newCulture;
                    }
                }

                // kullanicinin programla ilgili bilgilendirici notlari kendi dilinde görebilmesi için burada ekliyoruz.
            }
        }

        private static void DilAyariOlustur(AwesomeLibraryDataContext context)
        {
            var appSettings = new AppSettings()
            {
                //AppLangId = 42,
                AppLangName = "EN",
                AppBackgroundColor = "BLA",
                CategoryOrderBy = "NAME",
                CategoryOrderStyle = "A",
                CurrentCategoryNumber = 0,
                CurrentAuthorNumber = 0,
                FontFamily = "Verdana",
                FontSize = "30",
                AppBackgroundImage = null
            };

            context.AppSettings.InsertOnSubmit(appSettings);
            context.SubmitChanges();

            CultureInfo newCulture = new CultureInfo("en");
            Thread.CurrentThread.CurrentCulture = newCulture;
            Thread.CurrentThread.CurrentUICulture = newCulture;
        }
    }
}
```

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

```
?//------------------------------------------------------------------------------
// <auto-generated>
//     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.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AwesomeLibrary.Resources {
    using System;


    /// <summary>
    ///   A strongly-typed resource class, for looking up localized strings, etc.
    /// </summary>
    // 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.
    [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 {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;

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

        /// <summary>
        ///   Returns the cached ResourceManager instance used by this class.
        /// </summary>
        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        public static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("AwesomeLibrary.Resources.AppResources",
typeof(AppResources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }
        
        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property for all
        ///   resource lookups using this strongly typed resource class.
        /// </summary>
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
        public static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to About.
        /// </summary>
        public static string About {
            get {
                return ResourceManager.GetString("About", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to About The App.
        /// </summary>
        public static string AboutTheApp {
            get {
                return ResourceManager.GetString("AboutTheApp", resourceCulture);
            }
        }
        
        /// <summary>
        ///   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..
        /// </summary>
        public static string AboutTheAppText {
            get {
                return ResourceManager.GetString("AboutTheAppText", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to About The Awesome Library.
        /// </summary>
        public static string AboutTheAwesomeLibrary {
            get {
                return ResourceManager.GetString("AboutTheAwesomeLibrary",
resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Add Author.
        /// </summary>
        public static string AddAuthor {
            get {
                return ResourceManager.GetString("AddAuthor", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Add Book.
        /// </summary>
        public static string AddBook {
            get {
                return ResourceManager.GetString("AddBook", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Add Category.
        /// </summary>
        public static string AddCategory {
            get {
                return ResourceManager.GetString("AddCategory", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Arabic.
        /// </summary>
        public static string Arabic {
            get {
                return ResourceManager.GetString("Arabic", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Ascending.
        /// </summary>
        public static string Ascending {
            get {
                return ResourceManager.GetString("Ascending", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Has Been Added Successfully.
        /// </summary>
        public static string AuthorAddSuccess {
            get {
                return ResourceManager.GetString("AuthorAddSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Has This Category Already.
        /// </summary>
        public static string AuthorAlreadySameCategory {
            get {
                return ResourceManager.GetString("AuthorAlreadySameCategory", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Has Been Added To
```

```
Author Successfully.
        /// </summary>
        public static string AuthorCategoryAddSuccess {
            get {
                return ResourceManager.GetString("AuthorCategoryAddSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Has Been Removed Successfully.
        /// </summary>
        public static string AuthorDeleteSuccess {
            get {
                return ResourceManager.GetString("AuthorDeleteSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to This Author Has Already Exists.
        /// </summary>
        public static string AuthorExists {
            get {
                return ResourceManager.GetString("AuthorExists", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author List.
        /// </summary>
        public static string AuthorList {
            get {
                return ResourceManager.GetString("AuthorList", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Name.
        /// </summary>
        public static string AuthorName {
            get {
                return ResourceManager.GetString("AuthorName", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Name Has Been Changed Successfully.
        /// </summary>
        public static string AuthorNameChangeSuccess {
            get {
                return ResourceManager.GetString("AuthorNameChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Order By.
        /// </summary>
        public static string AuthorOrderBy {
            get {
                return ResourceManager.GetString("AuthorOrderBy", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Order Style.
        /// </summary>
        public static string AuthorOrderStyle {
            get {
                return ResourceManager.GetString("AuthorOrderStyle", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Order Style Has Been Changed Successfully.
        /// </summary>
        public static string AuthorOrderStyleChangeSuccess {
            get {
                return ResourceManager.GetString("AuthorOrderStyleChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Order Type Has Been Changed Successfully.
        /// </summary>
        public static string AuthorOrderTypeChangeSuccess {
            get {
                return ResourceManager.GetString("AuthorOrderTypeChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author Settings.
        /// </summary>
        public static string AuthorSettings {
            get {
                return ResourceManager.GetString("AuthorSettings", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background.
        /// </summary>
        public static string Background {
            get {
                return ResourceManager.GetString("Background", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background Color.
        /// </summary>
        public static string BackgroundColor {
            get {
                return ResourceManager.GetString("BackgroundColor", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background Color Has Been Changed Successfully.
        /// </summary>
        public static string BackgroundColorChangeSuccess {
            get {
                return ResourceManager.GetString("BackgroundColorChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background Image.
        /// </summary>
        public static string BackgroundImage {
            get {
                return ResourceManager.GetString("BackgroundImage", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background Image Has Been
Changed Successfully.
        /// </summary>
        public static string BackgroundImageChangeSuccess {
            get {
                return ResourceManager.GetString("BackgroundImageChangeSuccess",
resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Background Image Has Been
Removed Successfully.
        /// </summary>
        public static string BackgroundImageRemoveSuccess {
            get {
                return ResourceManager.GetString("BackgroundImageRemoveSuccess",
resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Best Book.
        /// </summary>
        public static string BestBook {
            get {
                return ResourceManager.GetString("BestBook", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Black.
        /// </summary>
        public static string Black {
            get {
                return ResourceManager.GetString("Black", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Blue.
        /// </summary>
        public static string Blue {
            get {
                return ResourceManager.GetString("Blue", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Book Comment.
        /// </summary>
        public static string BookComment {
            get {
                return ResourceManager.GetString("BookComment", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Book Count.
        /// </summary>
        public static string BookCount {
            get {
                return ResourceManager.GetString("BookCount", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book Has Been Removed 
Successfully.
        /// </summary>
        public static string BookDeleteSuccess {
            get {
                return ResourceManager.GetString("BookDeleteSuccess", 
resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book List.
        /// </summary>
        public static string BookList {
            get {
                return ResourceManager.GetString("BookList", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book Name.
        /// </summary>
        public static string BookName {
            get {
                return ResourceManager.GetString("BookName", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to You Have To Enter Book Name At 
Least.
        /// </summary>
        public static string BookNameMustBe {
            get {
                return ResourceManager.GetString("BookNameMustBe", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book Order By.
        /// </summary>
        public static string BookOrderBy {
            get {
                return ResourceManager.GetString("BookOrderBy", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book Order Style.
        /// </summary>
        public static string BookOrderStyle {
            get {
                return ResourceManager.GetString("BookOrderStyle", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Book Order Style Has Been 
Changed Successfully.
        /// </summary>
        public static string BookOrderStyleChangeSuccess {
            get {
                return ResourceManager.GetString("BookOrderStyleChangeSuccess", 
resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Book Order Type Has Been Changed
Successfully.
            /// </summary>
            public static string BookOrderTypeChangeSuccess {
                get {
                    return ResourceManager.GetString("BookOrderTypeChangeSuccess",
resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Average Book Read Per Day.
            /// </summary>
            public static string BookPerDay {
                get {
                    return ResourceManager.GetString("BookPerDay", resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Book Rating.
            /// </summary>
            public static string BookRating {
                get {
                    return ResourceManager.GetString("BookRating", resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Book Has Been Saved Successfully.
            /// </summary>
            public static string BookSaveSuccess {
                get {
                    return ResourceManager.GetString("BookSaveSuccess", resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Brown.
            /// </summary>
            public static string Brown {
                get {
                    return ResourceManager.GetString("Brown", resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Cancel.
            /// </summary>
            public static string Cancel {
                get {
                    return ResourceManager.GetString("Cancel", resourceCulture);
                }
            }

            /// <summary>
            ///   Looks up a localized string similar to Categories.
            /// </summary>
            public static string Categories {
                get {
                    return ResourceManager.GetString("Categories", resourceCulture);
                }
            }

```

```
        /// <summary>
        ///   Looks up a localized string similar to Category Has Been Added Successfully.
        /// </summary>
        public static string CategoryAddSuccess {
            get {
                return ResourceManager.GetString("CategoryAddSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Has Been Removed Successfully.
        /// </summary>
        public static string CategoryDeleteSuccess {
            get {
                return ResourceManager.GetString("CategoryDeleteSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to This Category Has Already Exists.
        /// </summary>
        public static string CategoryExists {
            get {
                return ResourceManager.GetString("CategoryExists", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Name.
        /// </summary>
        public static string CategoryName {
            get {
                return ResourceManager.GetString("CategoryName", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Name Has Been Changed Successfully.
        /// </summary>
        public static string CategoryNameChangeSuccess {
            get {
                return ResourceManager.GetString("CategoryNameChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Order By.
        /// </summary>
        public static string CategoryOrderBy {
            get {
                return ResourceManager.GetString("CategoryOrderBy", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category Order Style.
        /// </summary>
        public static string CategoryOrderStyle {
            get {
                return ResourceManager.GetString("CategoryOrderStyle", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Category Order Style Has Been Changed Successfully.
        /// </summary>
        public static string CategoryOrderStyleChangeSuccess {
            get {
                return ResourceManager.GetString("CategoryOrderStyleChangeSuccess", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Category Order Type Has Been Changed Successfully.
        /// </summary>
        public static string CategoryOrderTypeChangeSuccess {
            get {
                return ResourceManager.GetString("CategoryOrderTypeChangeSuccess", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Category Settings.
        /// </summary>
        public static string CategorySettings {
            get {
                return ResourceManager.GetString("CategorySettings", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Chinese.
        /// </summary>
        public static string Chinese {
            get {
                return ResourceManager.GetString("Chinese", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Contact With Us.
        /// </summary>
        public static string ContactWithUs {
            get {
                return ResourceManager.GetString("ContactWithUs", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Creation Date.
        /// </summary>
        public static string CreationDate {
            get {
                return ResourceManager.GetString("CreationDate", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Date.
        /// </summary>
        public static string Date {
            get {
                return ResourceManager.GetString("Date", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Delete Author.
        /// </summary>
        public static string DeleteAuthor {
            get {
                return ResourceManager.GetString("DeleteAuthor", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to You Will Delete The Author With All Contents. Are You Sure?.
        /// </summary>
        public static string DeleteAuthorQuestion {
            get {
                return ResourceManager.GetString("DeleteAuthorQuestion", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Delete Book.
        /// </summary>
        public static string DeleteBook {
            get {
                return ResourceManager.GetString("DeleteBook", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to You Will Delete This Book. Are You Sure?.
        /// </summary>
        public static string DeleteBookQuestion {
            get {
                return ResourceManager.GetString("DeleteBookQuestion", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Delete Category.
        /// </summary>
        public static string DeleteCategory {
            get {
                return ResourceManager.GetString("DeleteCategory", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to You Will Delete Category With All Contents. Do You Agree?.
        /// </summary>
        public static string DeleteCategoryQuestion {
            get {
                return ResourceManager.GetString("DeleteCategoryQuestion", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Descending.
        /// </summary>
        public static string Descending {
            get {
                return ResourceManager.GetString("Descending", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to English.
        /// </summary>
        public static string English {
            get {
                return ResourceManager.GetString("English", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Enter Author Name.
        /// </summary>
        public static string EnterAuthorName {
            get {
                return ResourceManager.GetString("EnterAuthorName", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Enter Category Name.
        /// </summary>
        public static string EnterCategoryName {
            get {
                return ResourceManager.GetString("EnterCategoryName",
resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Exit App.
        /// </summary>
        public static string ExitApp {
            get {
                return ResourceManager.GetString("ExitApp", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Do You Want To Exit App?.
        /// </summary>
        public static string ExitAppQuestion {
            get {
                return ResourceManager.GetString("ExitAppQuestion", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Finish Date.
        /// </summary>
        public static string FinishDate {
            get {
                return ResourceManager.GetString("FinishDate", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Font.
        /// </summary>
        public static string Font {
            get {
                return ResourceManager.GetString("Font", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Font Family.
        /// </summary>
        public static string FontFamily {
            get {
                return ResourceManager.GetString("FontFamily", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Font Family Has Been Changed Successfully.
        /// </summary>
        public static string FontFamilyChangeSuccess {
            get {
                return ResourceManager.GetString("FontFamilyChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Font Size.
        /// </summary>
        public static string FontSize {
            get {
                return ResourceManager.GetString("FontSize", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Font Size Has Been Changed Successfully.
        /// </summary>
        public static string FontSizeChangeSuccess {
            get {
                return ResourceManager.GetString("FontSizeChangeSuccess", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to French.
        /// </summary>
        public static string French {
            get {
                return ResourceManager.GetString("French", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to General Settings.
        /// </summary>
        public static string GeneralSettings {
            get {
                return ResourceManager.GetString("GeneralSettings", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to German.
        /// </summary>
        public static string German {
            get {
                return ResourceManager.GetString("German", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Gray.
        /// </summary>
        public static string Gray {
            get {
                return ResourceManager.GetString("Gray", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Green.
        /// </summary>
        public static string Green {
            get {
                return ResourceManager.GetString("Green", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Italian.
        /// </summary>
        public static string Italian {
            get {
                return ResourceManager.GetString("Italian", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Japanese.
        /// </summary>
        public static string Japanese {
            get {
                return ResourceManager.GetString("Japanese", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Language.
        /// </summary>
        public static string Language {
            get {
                return ResourceManager.GetString("Language", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to You May Restart Application To Change Affect.
        /// </summary>
        public static string LanguageWarning {
            get {
                return ResourceManager.GetString("LanguageWarning", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Last Modification Date.
        /// </summary>
        public static string ModificationDate {
            get {
                return ResourceManager.GetString("ModificationDate", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Author You Most Read.
        /// </summary>
        public static string MostReadAuthor {
            get {
                return ResourceManager.GetString("MostReadAuthor", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Category You Most Read.
        /// </summary>
        public static string MostReadCategory {
            get {
                return ResourceManager.GetString("MostReadCategory", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Publisher You Most Read.
        /// </summary>
        public static string MostReadPublisher {
            get {
                return ResourceManager.GetString("MostReadPublisher", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Name.
        /// </summary>
        public static string Name {
            get {
                return ResourceManager.GetString("Name", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Ok.
        /// </summary>
        public static string OK {
            get {
                return ResourceManager.GetString("OK", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to OneDrive.
        /// </summary>
        public static string OneDrive {
            get {
                return ResourceManager.GetString("OneDrive", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to OneDrive Sync Completed.
        /// </summary>
        public static string OneDriveSyncCompleted {
            get {
                return ResourceManager.GetString("OneDriveSyncCompleted", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Orange.
        /// </summary>
        public static string Orange {
            get {
                return ResourceManager.GetString("Orange", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Other Settings.
        /// </summary>
        public static string OtherSettings {
            get {
                return ResourceManager.GetString("OtherSettings", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Page Number.
        /// </summary>
        public static string PageNumber {
            get {
                return ResourceManager.GetString("PageNumber", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Average Page Read Per Day.
        /// </summary>
        public static string PagePerDay {
            get {
                return ResourceManager.GetString("PagePerDay", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Persian.
        /// </summary>
        public static string Persian {
            get {
                return ResourceManager.GetString("Persian", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Portuguese.
        /// </summary>
        public static string Portuguese {
            get {
                return ResourceManager.GetString("Portuguese", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Publisher Name.
        /// </summary>
        public static string PublisherName {
            get {
                return ResourceManager.GetString("PublisherName", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Purple.
        /// </summary>
        public static string Purple {
            get {
                return ResourceManager.GetString("Purple", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Rate.
        /// </summary>
        public static string Rate {
            get {
                return ResourceManager.GetString("Rate", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Red.
        /// </summary>
        public static string Red {
            get {
                return ResourceManager.GetString("Red", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Remove Background Image.
        /// </summary>
        public static string RemoveBackgroundImage {
            get {
                return ResourceManager.GetString("RemoveBackgroundImage",
resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Reset Settings.
        /// </summary>
        public static string ResetSettings {
            get {
                return ResourceManager.GetString("ResetSettings", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to LeftToRight.
        /// </summary>
        public static string ResourceFlowDirection {
            get {
                return ResourceManager.GetString("ResourceFlowDirection",
resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to en-US.
        /// </summary>
        public static string ResourceLanguage {
            get {
                return ResourceManager.GetString("ResourceLanguage", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Russian.
        /// </summary>
        public static string Russian {
            get {
                return ResourceManager.GetString("Russian", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Sanskrit.
        /// </summary>
        public static string Sanskrit {
            get {
                return ResourceManager.GetString("Sanskrit", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Save.
        /// </summary>
        public static string Save {
            get {
                return ResourceManager.GetString("Save", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Search.
        /// </summary>
        public static string Search {
            get {
                return ResourceManager.GetString("Search", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Search Completed.
        /// </summary>
        public static string SearchCompleted {
            get {
                return ResourceManager.GetString("SearchCompleted", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Search Results.
        /// </summary>
        public static string SearchResults {
            get {
                return ResourceManager.GetString("SearchResults", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to You Have To Fill Search Criteria.
        /// </summary>
        public static string SearchTrimFault {
            get {
                return ResourceManager.GetString("SearchTrimFault", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Select.
        /// </summary>
        public static string Select {
            get {
                return ResourceManager.GetString("Select", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Select Background Color.
        /// </summary>
        public static string SelectBackgroundColor {
            get {
                return ResourceManager.GetString("SelectBackgroundColor",
resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Selected.
        /// </summary>
        public static string Selected {
            get {
                return ResourceManager.GetString("Selected", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Select Font Family.
        /// </summary>
        public static string SelectFontFamily {
            get {
                return ResourceManager.GetString("SelectFontFamily", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Select Font Size.
        /// </summary>
        public static string SelectFontSize {
            get {
                return ResourceManager.GetString("SelectFontSize", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Select Language.
        /// </summary>
        public static string SelectLanguage {
            get {
                return ResourceManager.GetString("SelectLanguage", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Send With Awesome Library App.
        /// </summary>
        public static string SendWithApp {
            get {
                return ResourceManager.GetString("SendWithApp", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Send With Mail.
        /// </summary>
        public static string SendWithMail {
            get {
                return ResourceManager.GetString("SendWithMail", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Send With SMS.
        /// </summary>
        public static string SendWithSMS {
            get {
                return ResourceManager.GetString("SendWithSMS", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Settings.
        /// </summary>
        public static string Settings {
            get {
                return ResourceManager.GetString("Settings", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Share Book.
        /// </summary>
        public static string ShareBook {
            get {
                return ResourceManager.GetString("ShareBook", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Spanish.
        /// </summary>
        public static string Spanish {
            get {
                return ResourceManager.GetString("Spanish", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Start Date.
        /// </summary>
        public static string StartDate {
            get {
                return ResourceManager.GetString("StartDate", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Statistics.
        /// </summary>
        public static string Statistics {
            get {
                return ResourceManager.GetString("Statistics", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Background Settings Has Been Reset Successfully.
        /// </summary>
        public static string SuccessfulResetSettings {
            get {
                return ResourceManager.GetString("SuccessfulResetSettings", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Sync.
        /// </summary>
        public static string Sync {
            get {
                return ResourceManager.GetString("Sync", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Synchronizing.
        /// </summary>
        public static string Synchronizing {
            get {
                return ResourceManager.GetString("Synchronizing", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Sync All In One File.
        /// </summary>
        public static string SyncOnOneFile {
            get {
                return ResourceManager.GetString("SyncOnOneFile", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to System Has A Fault. Please Try Again Later.
        /// </summary>
        public static string SystemFault {
            get {
                return ResourceManager.GetString("SystemFault", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Thai.
        /// </summary>
        public static string Thai {
            get {
                return ResourceManager.GetString("Thai", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Book Count You Read.
        /// </summary>
        public static string TotalBookCount {
            get {
                return ResourceManager.GetString("TotalBookCount", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Turkish.
        /// </summary>
        public static string Turkish {
            get {
                return ResourceManager.GetString("Turkish", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Worst Book.
        /// </summary>
        public static string WorstBook {
            get {
                return ResourceManager.GetString("WorstBook", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Yellow.
        /// </summary>
        public static string Yellow {
            get {
                return ResourceManager.GetString("Yellow", resourceCulture);
            }
        }
    }
}
```

Namespaces

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

1.2.5 AppSettings.cs

This is file AppSettings.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class AppSettings
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int AppSettingsId { get; set; }

        [Column]
        public int CurrentCategoryNumber { get; set; }

        [Column]
        public int CurrentAuthorNumber { get; set; }

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

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

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

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

        [Column]
        public string FontFamily { get; set; }
        [Column]
        public string FontSize { get; set; }

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

Namespaces

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

1.2.6 AssemblyInfo.cs

This is file AssemblyInfo.cs.

Body Source

```
?using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Resources;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Awesome Library")]
[assembly: AssemblyDescription("The Best Library Application Ever")]
```

```
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("ÇMS Software")]
[assembly: AssemblyProduct("ÇMS Software")]
[assembly: AssemblyCopyright("Copyright ©  2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("2112005b-9c3f-4c40-912f-9da47b7d295b")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.2.0")]
[assembly: AssemblyFileVersion("1.0.2.0")]
[assembly: NeutralResourcesLanguageAttribute("en-US")]
```

1.2.7 Author.cs

This is file Author.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class Author
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int AuthorId { get; set; }

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

        [Column]
        public int AuthorBookCount { get; set; }

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

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

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

        [Column]
        public DateTime CreationDate { get; set; }

        [Column]
        public DateTime ModificationDate { get; set; }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class AuthorPage : PhoneApplicationPage
    {

        public int authorId;
        public int categoryId;
        public int bookId;
        public Popup popup;
        public string oldAuthorName;

        public AuthorPage()
        {
            InitializeComponent();

            ApplicationBar = new ApplicationBar();

            ApplicationBarIconButton button1 = new ApplicationBarIconButton();
            button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
            button1.Text = AppResources.AddBook;
            ApplicationBar.Buttons.Add(button1);
            button1.Click += new EventHandler(AddBookButton_Click);

            ApplicationBarIconButton button2 = new ApplicationBarIconButton();
            button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
            button2.Text = AppResources.DeleteAuthor;
            ApplicationBar.Buttons.Add(button2);
            button2.Click += new EventHandler(DeleteAuthorButton_Click);

            ApplicationBarIconButton button3 = new ApplicationBarIconButton();
            button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
            button3.Text = AppResources.AuthorSettings;
            ApplicationBar.Buttons.Add(button3);
            button3.Click += new EventHandler(AuthorSettingsButton_Click);

            ApplicationBarIconButton button4 = new ApplicationBarIconButton();
            button4.IconUri = new Uri("/Assets/AddCategory.png", UriKind.Relative);
            button4.Text = AppResources.AddCategory;
            ApplicationBar.Buttons.Add(button4);
            button4.Click += new EventHandler(AddCategoryButton_Click);

            popup = new Popup();

        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            List<Book> books = new List<Book>();
            List<Book> booksOrdered = new List<Book>();

            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);

            lstBooks.Items.Clear();

            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {

                var appSettings = context.AppSettings.First();
                categoryId = appSettings.CurrentCategoryNumber;

                var author = context.Authors.Where(j =>
j.AuthorId.Equals(e.Fragment)).Single() as Author;
                authorId = author.AuthorId;

                var appSettings2 = context.AppSettings;
                foreach (var item in appSettings2)
                {
                    item.CurrentAuthorNumber = authorId;
                }
                context.SubmitChanges();

                var authorBooks = context.BookAuthors.Where(j =>
j.AuthorId.Equals(e.Fragment)).ToList() as List<BookAuthor>;
                if (authorBooks.Count != 0)
                {
                    foreach (var item in authorBooks)
                    {
                        try
                        {
                            var book = context.Books.Where(j => j.BookCategoryId.Equals(categoryId) && j.BookId.Equals(item.BookId)).Single() as Book;
                            books.Add(book);
                        }
                        catch (Exception)
                        {
                        }
                    }

                }


                string orderStyle = author.BookOrderStyle;

                switch (author.BookOrderBy)
                {
                    case "NAME":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.BookName).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j => j.BookName).ToList();
                        }
                        break;
                    case "CDATE":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.CreationDate).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j => j.CreationDate).ToList();
                        }
                        break;
                    case "MDATE":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.ModificationDate).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j => j.ModificationDate).ToList();
                        }
                        break;
                    case "SDATE":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.ReadStartDate).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j => j.ReadStartDate).ToList();
                        }
                        break;
                    case "FDATE":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.ReadFinishDate).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j =>
j.ReadFinishDate).ToList();
                        }
                        break;
                    case "RATING":
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.BookRating).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderByDescending(j =>
j.BookRating).ToList();
                        }
                        break;
                    default:
                        if (orderStyle == "A")
                        {
                            booksOrdered = books.OrderBy(j => j.BookName).ToList();
                        }
                        else
                        {
                            booksOrdered = books.OrderBy(j => j.BookName).ToList();
                        }
                        break;
                }

                lblAuthorName.Text = author.AuthorName;
                lblBookList.Text = AppResources.BookList + " (" + author.AuthorName +
")";
                lstBooks.ItemsSource = booksOrdered;
                lstBooks.DisplayMemberPath = "BookNameRating";
                SetBackgroundColor();
                //lstNoteList.DisplayMemberPath = "NameCreation";
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void lstBooks_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var book = (Book)lstBooks.SelectedItem;
            int bookId = book.BookId;
            NavigationService.Navigate(new Uri("/BookPage.xaml#" + bookId, UriKind.Relative));
        }

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

        private void DeleteAuthorButton_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show(AppResources.DeleteAuthorQuestion,
                AppResources.DeleteAuthor, MessageBoxButton.OKCancel)
                != MessageBoxResult.OK)
            {

            }
            else
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var bookAuthors = context.BookAuthors.Where(j => j.AuthorId.Equals(authorId)).ToList() as List<BookAuthor>;
                    foreach (var item in bookAuthors)
                    {
                        var book = context.Books.Where(j => j.BookId.Equals(item.BookId)).Single() as Book;
                        var bookAuthors2 = context.BookAuthors.Where(j => j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
                        context.BookAuthors.DeleteAllOnSubmit(bookAuthors2);
                        context.Books.DeleteOnSubmit(book);
                    }

                    var categoryAuthors = context.CategoryAuthors.Where(j => j.AuthorId.Equals(authorId)).ToList() as List<CategoryAuthor>;
                    context.CategoryAuthors.DeleteAllOnSubmit(categoryAuthors);

                    var authors = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
                    context.Authors.DeleteOnSubmit(authors);

                    context.SubmitChanges();

                    var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Select(j => j);
                    foreach (var item in category)
                    {
                        item.CategoryBookCount = context.Books.Where(j => j.BookCategoryId.Equals(item.CategoryId)).ToList().Count;
                        item.CategoryNameCount = item.CategoryName + " (" + item.CategoryBookCount + ")";
                        item.ModificationDate = DateTime.Now;
                        context.SubmitChanges();
                    }
                }
                MessageBox.Show(AppResources.AuthorDeleteSuccess);
                NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId, UriKind.Relative));
            }
            //MessageBox.Show(AppResources.NoteSaved);
        }

        private void lblAuthorName_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            oldAuthorName = lblAuthorName.Text;
            popup = new Popup();
            popup.Height = 300;
            popup.Width = 400;
            popup.VerticalOffset = 20;
            PopupAddChange control = new PopupAddChange();
            control.txtLabel.Text = AppResources.EnterAuthorName;
            control.btnCancel.Content = AppResources.Cancel;
            control.btnOK.Content = AppResources.OK;
            popup.Child = control;
            popup.IsOpen = true;
            control.txtName.Text = lblAuthorName.Text;
            control.txtName.Focus();
            control.txtName.Select(0, control.txtName.Text.Length);

            control.btnOK.Click += (s, args) =>
            {
                bool isCreated;
                string authorName;
                popup.IsOpen = false;
                int length = control.txtName.Text.Length;
                string space = control.txtName.Text.Substring(length - Math.Min(1, length));
                if (space == " ")
                {
                    authorName = control.txtName.Text.Remove(length - 1, 1);
                }
                else
                {
                    authorName = control.txtName.Text;
                }

                if (authorName != lblAuthorName.Text)
                {
                    // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
                    // kontrol eden bir kod bölümü
                    using (var contextFolder = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        isCreated =
                            contextFolder.Authors.Any(j => j.AuthorName.Equals(authorName));
                    }
                    if (isCreated == true)
                    {
                        MessageBox.Show(AppResources.AuthorExists);
                    }
                    // eger bu isimde bir klasör olusturulmamissa
                    // olusturulmasi için gerekli kodlar asagidadir
                    else
                    {
                        using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                        {

                            // buraya kitapla ilgili bilginin güncellenecegi kod da eklenecek

                            var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Select(j => j);
                            foreach (var item in author)
                            {
                                item.AuthorName = authorName;
                                item.ModificationDate = DateTime.Now;
                                item.AuthorNameCount = authorName + " (" + item.AuthorBookCount.ToString() + ")";
                            }
                            context.SubmitChanges();

                            var bookAuthors = context.BookAuthors.Where(j => j.AuthorId.Equals(authorId)).Select(j => j);
                            foreach (var item in bookAuthors)
                            {
                                var book = context.Books.Where(j => j.BookId.Equals(item.BookId)).Select(j => j);
                                foreach (var item2 in book)
                                {
                                    item2.BookAuthorName = item2.BookAuthorName.Replace(oldAuthorName, authorName);
                                    //item2.BookInformation = item2.BookInformation.Replace(oldAuthorName, authorName);
                                    item2.ModificationDate = DateTime.Now;
                                    context.SubmitChanges();
                                }
                            }

                            //lstFolders.ItemsSource = context.NoteFolders;
                            //lstAuthors.ItemsSource = context.Categories;
                            MessageBox.Show(AppResources.AuthorNameChangeSuccess);
                            popup.IsOpen = false;
                            CategoryAuthor categoryAuthor = context.CategoryAuthors.Where(j => j.AuthorId.Equals(authorId) && j.CategoryId.Equals(categoryId)).Single() as CategoryAuthor;
                            Author author2 = context.Authors.Where(j => j.AuthorName.Equals(authorName) && j.AuthorId.Equals(categoryAuthor.AuthorId)).Single() as Author;
                            NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + categoryAuthor.AuthorId, UriKind.Relative));
                        }
                    }
                }
            };
            control.btnCancel.Click += (s, args) =>
            {
                popup.IsOpen = false;
                };
            }

            private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
            {
                if (popup.IsOpen)
                {
                    popup.IsOpen = false;
                }
                if (this.NavigationService.CanGoBack)
                {
                    this.NavigationService.Navigate(new Uri("/CategoryPage.xaml#" +
categoryId, UriKind.Relative));
                }
            }

            private void AuthorSettingsButton_Click(object sender, EventArgs e)
            {
                NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId,
UriKind.Relative));
            }

            private void AddCategoryButton_Click(object sender, EventArgs e)
            {
                NavigationService.Navigate(new Uri("/AddCategoryPage.xaml#" + authorId,
UriKind.Relative));
            }

        }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.IO.IsolatedStorage;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Live;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Phone.Tasks;
using Microsoft.Phone.Marketplace;

```

```
namespace AwesomeLibrary
{
    public partial class AuthorSettingsPage : PhoneApplicationPage
    {
        public int authorId;
        public int categoryId;

        public AuthorSettingsPage()
        {
            InitializeComponent();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings.First();
                lblFontFamily.Text = AppResources.FontFamily + " (" + AppResources.Selected + ": " + appSettings.FontFamily + ")";
                lblFontSize.Text = AppResources.FontSize + " (" + AppResources.Selected + ": " + appSettings.FontSize + ")";
            }

            pvAuthorSettings.Title = AppResources.AuthorSettings;
            piFont.Header = AppResources.Font;
            piOtherSettings.Header = AppResources.OtherSettings;

            btnFontFamily.Content = AppResources.Select;
            btnFontSize.Content = AppResources.Select;
            btnBookOrder.Content = AppResources.Select;
            btnBookOrderStyle.Content = AppResources.Select;
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var author = context.Authors.Where(j => j.AuthorId.Equals(e.Fragment)).Single() as Author;
                authorId = author.AuthorId;
                var appSettings = context.AppSettings.First();
                categoryId = appSettings.CurrentCategoryNumber;
                string orderStyle = author.BookOrderStyle;

                if (author.BookOrderBy == "NAME")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected + ": " + AppResources.Name + ")";
                }
                if (author.BookOrderBy == "CDATE")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" +
AppResources.Selected + ": " + AppResources.CreationDate + ")";
                }
                if (author.BookOrderBy == "MDATE")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected + ": " + AppResources.ModificationDate + ")";
                }
                if (author.BookOrderBy == "SDATE")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected + ": " + AppResources.StartDate + ")";
                }
                if (author.BookOrderBy == "FDATE")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected + ": " + AppResources.FinishDate + ")";
                }
                if (author.BookOrderBy == "RATING")
                {
                    lblBookOrder.Text = AppResources.BookOrderBy + " (" + AppResources.Selected + ": " + AppResources.BookRating + ")";
                }
                if (author.BookOrderStyle == "A")
                {
                    lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
                }
                if (author.BookOrderStyle == "D")
                {
                    lblBookOrderStyle.Text = AppResources.BookOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
                }
                //lstNoteList.DisplayMemberPath = "NameCreation";
                SetBackgroundColor();
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void btnBookOrder_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        private void btnBookOrderStyle_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        private void btnFontSize_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/FontSizeSettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        private void btnFontFamily_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/FontFamilySettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //pvAuthorSettings.Title = AppResources.AuthorSettings;
            //piFont.Header = AppResources.Font;
            //piOtherSettings.Header = AppResources.OtherSettings;

            //btnFontFamily.Content = AppResources.Select;
            //btnFontSize.Content = AppResources.Select;
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId, UriKind.Relative));
            }
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    public class AwesomeLibraryDataContext : DataContext
    {
        public const string ConnectionString = @"Data Source=isostore:/AwesomeLibrary.sdf; Max Database Size=256; Max Buffer Size=4096;";
        public AwesomeLibraryDataContext(string connectionString)
            : base(connectionString) { }
        public Table<Category> Categories;
        public Table<Author> Authors;
        public Table<Book> Books;
        public Table<AppSettings> AppSettings;
        public Table<BookAuthor> BookAuthors;
        public Table<CategoryAuthor> CategoryAuthors;
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
{
    public partial class BackgroundColorSettingsPage : PhoneApplicationPage
    {
        public int authorId;

        public BackgroundColorSettingsPage()
        {
            InitializeComponent();

            lstBackgroundColor.Items.Clear();
            lstBackgroundColor.Items.Add(AppResources.Black);
            lstBackgroundColor.Items.Add(AppResources.Blue);
            lstBackgroundColor.Items.Add(AppResources.Brown);
            lstBackgroundColor.Items.Add(AppResources.Gray);
            lstBackgroundColor.Items.Add(AppResources.Green);
            lstBackgroundColor.Items.Add(AppResources.Orange);
            lstBackgroundColor.Items.Add(AppResources.Purple);
            lstBackgroundColor.Items.Add(AppResources.Red);
            lstBackgroundColor.Items.Add(AppResources.Yellow);
            lstBackgroundColor.SelectedIndex = -1;

            lblBackgroundColor.Text = AppResources.SelectBackgroundColor;
            lblGeneralSettings.Text = AppResources.GeneralSettings;

            SetBackgroundColor();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //SetBackgroundColor();
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            authorId = int.Parse(e.Fragment);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
                    lblGeneralSettings.Text = AppResources.GeneralSettings;
                    lblBackgroundColor.Text = AppResources.SelectFontSize;
                }
            }

            private void lstBackgroundColor_SelectionChanged(object sender,
SelectionChangedEventArgs e)
            {
                int index = lstBackgroundColor.SelectedIndex;
                string backgroundColor = "";
                if (index == 0)
                {
                    backgroundColor = "BLA";
                }
                else if (index == 1)
                {
                    backgroundColor = "BLU";
                }
                else if (index == 2)
                {
                    backgroundColor = "BRO";
                }
                else if (index == 3)
                {
                    backgroundColor = "GRA";
                }
                else if (index == 4)
                {
                    backgroundColor = "GRE";
                }
                else if (index == 5)
                {
                    backgroundColor = "ORA";
                }
                else if (index == 6)
                {
                    backgroundColor = "PUR";
                }
                else if (index == 7)
                {
                    backgroundColor = "RED";
                }
                else if (index == 8)
                {
                    backgroundColor = "YEL";
                }
                else
                {
                    backgroundColor = "BLA";
                }

                using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var appSetting in appSettings)
                    {
                        appSetting.AppBackgroundColor = backgroundColor;
                    }
                    context.SubmitChanges();
                    //CustomMessageBox messageBox = new CustomMessageBox()
                    //{
                    //    Caption = AppResources.BackgroundColor,
                    //    Message = AppResources.SuccessfulBackgroundColorChanged,
                    //    Background = messageBackGround
                    //};
                    //messageBox.Show();
                    MessageBox.Show(AppResources.BackgroundColorChangeSuccess);
                }
                SetBackgroundColor();
                NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
            }

            private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
            {
                if (this.NavigationService.CanGoBack)
                {
                    this.NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
                }
            }

            private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
            {
                //SetBackgroundColor();
            }
        }
    }
```

Namespaces

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

1.2.14 Book.cs

This is file Book.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class Book
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int BookId { get; set; }

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

        [Column]
        public int BookCategoryId { get; set; }

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

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

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

        [Column]
        public int BookPageNumber { get; set; }

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

        [Column]
        public DateTime ReadStartDate { get; set; }

        [Column]
        public DateTime ReadFinishDate { get; set; }

        [Column]
        public int BookRating { get; set; }

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

        [Column]
        public DateTime CreationDate { get; set; }

        [Column]
        public DateTime ModificationDate { get; set; }

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

        [Column]
        public string BookNameRating { get; set; }
    }
}
```

Namespaces

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

1.2.15 BookAuthor.cs

This is file BookAuthor.cs.

Body Source

```
?using System;
using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class BookAuthor
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int BookAuthorId { get; set; }

        [Column]
        public int BookId { get; set; }

        [Column]
        public int AuthorId { get; set; }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using Microsoft.Phone.Tasks;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
{
    public partial class BookPage : PhoneApplicationPage
    {
        public int authorId;
        public string authorName;
        public string categoryName;
        public int categoryId;
        public int bookId;
        public string pageName;
        double InputHeight = 0.0;
        public bool flag;
        public bool isFilled;
        public double ratingValue = 0;

        public BookPage()
        {
            InitializeComponent();

            SetBackgroundColor();

            //pvAuthor.Title = authorName;
            piBookName.Header = AppResources.BookName;
            piComment.Header = AppResources.BookComment;
            piPublisherName.Header = AppResources.PublisherName;
            piRating.Header = AppResources.BookRating;
            piStartFinishDate.Header = AppResources.Date;
            lblStartDate.Text = AppResources.StartDate;
            lblFinishDate.Text = AppResources.FinishDate;
            piPageNumber.Header = AppResources.PageNumber;


            ApplicationBar = new ApplicationBar();

            ApplicationBarIconButton button1 = new ApplicationBarIconButton();
            button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
            button1.Text = AppResources.Save;
            ApplicationBar.Buttons.Add(button1);
            button1.Click += new EventHandler(SaveButton_Click);

            ApplicationBarIconButton button2 = new ApplicationBarIconButton();
            button2.IconUri = new Uri("/Assets/SendWithMail.png", UriKind.Relative);
            button2.Text = AppResources.SendWithMail;
            ApplicationBar.Buttons.Add(button2);
            button2.Click += new EventHandler(SendMailButton_Click);

            ApplicationBarIconButton button3 = new ApplicationBarIconButton();
            button3.IconUri = new Uri("/Assets/SendWithSMS.png", UriKind.Relative);
            button3.Text = AppResources.SendWithSMS;
            ApplicationBar.Buttons.Add(button3);
            button3.Click += new EventHandler(SendSMSButton_Click);

            ApplicationBarIconButton button4 = new ApplicationBarIconButton();
            button4.IconUri = new Uri("/Assets/Share.png", UriKind.Relative);
            button4.Text = AppResources.ShareBook;
            ApplicationBar.Buttons.Add(button4);
            button4.Click += new EventHandler(ShareBookButton_Click);

            isFilled = false;

            ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
            menuItem1.Text = AppResources.DeleteBook;
            ApplicationBar.MenuItems.Add(menuItem1);
            menuItem1.Click += new EventHandler(DeleteBookMenuItem_Click);

        }

        private void SendSMSButton_Click(object sender, EventArgs e)
        {
            SmsComposeTask smsComposeTask = new SmsComposeTask();

            smsComposeTask.To = "";
            smsComposeTask.Body = CreateSendMaterial();

            smsComposeTask.Show();
            //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
        }

        private void ShareBookButton_Click(object sender, EventArgs e)
        {
            ShareStatusTask shareStatusTask = new ShareStatusTask();

            shareStatusTask.Status = CreateSendMaterial();

            shareStatusTask.Show();
            //MessageBox.Show(AppResources.SuccessfulSendWithSMS);
        }

        private void SendMailButton_Click(object sender, EventArgs e)
        {
            // burada birden fazla e-posta hesabi varsa birini seçmesi söyleniyor
            //EmailAddressChooserTask emailAddressChooserTask;
            //emailAddressChooserTask = new EmailAddressChooserTask();
            //emailAddressChooserTask.Completed += new
EventHandler<EmailResult>(emailAddressChooserTask_Completed);
            //emailAddressChooserTask.Show();
```

1

```
            EmailComposeTask emailComposeTask = new EmailComposeTask();

            emailComposeTask.Subject = txtBookName.Text;
            emailComposeTask.Body = CreateSendMaterial();
            emailComposeTask.To = "";
            emailComposeTask.Cc = "";
            emailComposeTask.Bcc = "";

            emailComposeTask.Show();
            //MessageBox.Show(AppResources.SuccessfulSendWithMail);
        }

        private string CreateSendMaterial()
        {
            StringBuilder sb = new StringBuilder();
            sb.AppendLine(AppResources.BookName + ": " + txtBookName.Text);
            sb.AppendLine(AppResources.CategoryName + ": " + categoryName);
            sb.AppendLine(AppResources.AuthorName + ": " + authorName);
            sb.AppendLine(AppResources.PageNumber + ": " + txtPageNumber.Text);
            sb.AppendLine(AppResources.PublisherName + ": " + txtPublisherName.Text);
            sb.AppendLine(AppResources.StartDate + ": " +
DateTime.Parse(dtStart.Value.ToString()).ToShortDateString());
            sb.AppendLine(AppResources.FinishDate + ": " +
DateTime.Parse(dtFinish.Value.ToString()).ToShortDateString());
            sb.AppendLine(AppResources.BookComment + ": " + txtBookComment.Text);
            sb.AppendLine(AppResources.BookRating + ": " + rtRating.Value.ToString() +
"/10");
            sb.AppendLine();
            sb.AppendLine();
            sb.AppendLine(AppResources.SendWithApp);
            return sb.ToString();
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //SetBackgroundColor();

            // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
            //pvAuthor.Title = authorName;

            //pvAuthor.Title = authorName;
            //piBookName.Header = AppResources.BookName;
            //piComment.Header = AppResources.BookComment;
            //piPublisherName.Header = AppResources.PublisherName;
            //piRating.Header = AppResources.BookRating;
            //piStartFinishDate.Header = AppResources.Date;
            //lblStartDate.Text = AppResources.StartDate;
            //lblFinishDate.Text = AppResources.FinishDate;
            //piPageNumber.Header = AppResources.PageNumber;
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
                }
                else
                {
                    switch (appSettings.AppBackgroundColor)
                    {
                        case "BLA":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                            break;
                        case "BLU":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                            break;
                        case "BRO":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                            break;
                        case "RED":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                            break;
                        case "GRE":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                            break;
                        case "GRA":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                            break;
                        case "YEL":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                            break;
                        case "ORA":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                            break;
                        case "PUR":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                            break;
                        default:
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                            break;
                    }
                }
            }

            protected override void OnNavigatedTo(NavigationEventArgs e)
            {
                base.OnNavigatedTo(e);
                using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings.First();
                    categoryId = appSettings.CurrentCategoryNumber;
                    authorId = appSettings.CurrentAuthorNumber;

                    // sayfanin font ayarlari için yapilan bir degisiklik
                    FontFamily temp = new FontFamily(appSettings.FontFamily);
                    double fontsize = double.Parse(appSettings.FontSize);
                    txtBookComment.FontFamily = temp;
                    txtBookComment.FontSize = fontsize;
                    txtPageNumber.FontFamily = temp;
                    txtPageNumber.FontSize = fontsize;
                    txtBookName.FontFamily = temp;
                    txtBookName.FontSize = fontsize;
                    txtBookComment.FontFamily = temp;
                    txtBookComment.FontSize = fontsize;
                    txtPublisherName.FontFamily = temp;
                    txtPublisherName.FontSize = fontsize;
                    rtRating.Value = 5;

                    var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
```

```
                authorName = author.AuthorName;

                var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
                categoryName = category.CategoryName;
            }

            var lastPage = NavigationService.BackStack.FirstOrDefault();
            pageName = lastPage.Source.ToString();
            pvAuthor.SelectedIndex = 0;
            txtBookName.Focus();
            // yazarin adi sayfanin en üstünde görünsün diye yapiliyor bu
            pvAuthor.Title = authorName;
            SetBackgroundColor();
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            bookId = int.Parse(e.Fragment);
            if (pageName.Contains("/AuthorPage.xaml"))
            {
                isFilled = true;
            }
            else
            {
                //using (var context2 = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                //{
                //    var appSettings = context2.AppSettings; ;
                //    var book2 = context2.Books.Where(j => j.BookId.Equals(bookId)) as Book;
                //    var bookAuthor = context2.BookAuthors.Where(j => j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
                //    var bAuthor = bookAuthor.First();
                //    var author = context2.Authors.Where(j => j.AuthorId.Equals(bAuthor.AuthorId)) as Author;
                //    foreach (var item in appSettings)
                //    {
                //        item.CurrentAuthorNumber = author.AuthorId;
                //        item.CurrentCategoryNumber = book2.BookCategoryId;
                //    }
                //    context2.SubmitChanges();
                //    pvAuthor.Title = author.AuthorName;
                //}
            }
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var book = context.Books.Where(j => j.BookId.Equals(e.Fragment)).Single() as Book;

                txtBookName.Text = book.BookName == "" ? "" : book.BookName;
                txtPageNumber.Text = book.BookPageNumber.ToString() == "" ? "" : book.BookPageNumber.ToString();
                txtPublisherName.Text = book.BookPublisherName == "" ? "" : book.BookPublisherName;
                dtStart.Value = book.ReadStartDate == null ? DateTime.Now : book.ReadStartDate;
                dtFinish.Value = book.ReadFinishDate == null ? DateTime.Now : book.ReadFinishDate;
                rtRating.Value = book.BookRating == null ? 0 : book.BookRating;
                txtBookComment.Text = book.BookComment == "" ? "" : book.BookComment;
            }

            SetBackgroundColor();
            pvAuthor.SelectedIndex = 0;
            //pvAuthor.Name = authorName;
            txtBookName.Focus();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            //SaveButton_Click(this, new EventArgs());
            if (pageName.Contains("/SearchPage.xaml"))
            {
                //this.NavigationService.Navigate(new Uri("/SearchPage.xaml", UriKind.Relative));
            }
            else
            {
                this.NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId, UriKind.Relative));
            }
        }

        private void SaveButton_Click(object sender, EventArgs e)
        {
            txtBookComment_LostFocus(this, new RoutedEventArgs());
            this.pnlKeyboardPlaceHolder.Visibility = Visibility.Collapsed;
            if (txtBookName.Text.Trim().Length < 1)
            {
                MessageBox.Show(AppResources.BookNameMustBe);
            }
            else
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    if (isFilled || pageName.Contains("/SearchPage.xaml"))
                    {
                        var book = context.Books.Where(j => j.BookId.Equals(bookId)).Select(j => j);
                        foreach (var item in book)
                        {
                            item.BookCategoryId = categoryId;
                            item.BookName = txtBookName.Text == "" ? "" : txtBookName.Text;
                            item.BookPageNumber = txtPageNumber.Text == "" ? 0 : int.Parse(txtPageNumber.Text);
                            item.BookPublisherName = txtPublisherName.Text == "" ? "" : txtPublisherName.Text;
                            item.ReadStartDate = DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtStart.Value.ToString());
                            item.ReadFinishDate = DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtFinish.Value.ToString()); ;
                            //item.BookRating = rtRating.Value.ToString() == "" ? 0 : int.Parse(rtRating.Value.ToString());
                            item.BookRating = int.Parse(ratingValue.ToString()) == 0 ? 0 : int.Parse(ratingValue.ToString());
                            item.BookComment = txtBookComment.Text == "" ? "" : txtBookComment.Text;
                            item.ModificationDate = DateTime.Now;

                            item.BookCategoryName = categoryName;
                            item.BookAuthorName = authorName;

                            //item.BookInformation = categoryName + " " + authorName + " " + txtBookName.Text + " " + txtPublisherName.Text + " " + txtBookComment.Text;
                            item.BookNameRating = item.BookName + " (" + item.BookRating + "/10)";
                        }
                        context.SubmitChanges();
                    }
                    else
                    {
                        Book book = new Book();
                        book.BookCategoryId = categoryId;
                        book.BookGuid = Guid.NewGuid().ToString();
                        book.BookName = txtBookName.Text == "" ? "" : txtBookName.Text;
                        book.BookPageNumber = txtPageNumber.Text == "" ? 0 : int.Parse(txtPageNumber.Text);
                        book.BookPublisherName = txtPublisherName.Text == "" ? "" : txtPublisherName.Text;
                        book.ReadStartDate = DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtStart.Value.ToString());
                        book.ReadFinishDate = DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtFinish.Value.ToString()); ;
                        //book.BookRating = rtRating.Value.ToString() == "" ? 0 : int.Parse(rtRating.Value.ToString());
                        book.BookRating = int.Parse(ratingValue.ToString()) == 0 ? 0 : int.Parse(ratingValue.ToString());
                        book.BookComment = txtBookComment.Text == "" ? "" : txtBookComment.Text;
                        book.ModificationDate = DateTime.Now;

                        book.BookCategoryName = categoryName;
                        book.BookAuthorName = authorName;

                        //book.BookInformation = categoryName + " " + authorName + " " + book.BookName + " " + book.BookPublisherName + " " + book.BookComment;
                        book.BookNameRating = book.BookName + " (" + book.BookRating + "/10)";
                        book.CreationDate = DateTime.Now;
                        context.Books.InsertOnSubmit(book);
                        context.SubmitChanges();

                        Book book2 = context.Books.Where(j => j.BookGuid.Equals(book.BookGuid)).Single() as Book;

                        BookAuthor bookAuthor = new BookAuthor();
                        bookAuthor.AuthorId = authorId;
                        bookAuthor.BookId = book2.BookId;
                        context.BookAuthors.InsertOnSubmit(bookAuthor);
                        context.SubmitChanges();

                        var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Select(j => j);
                        foreach (var item in category)
                        {
                            item.CategoryBookCount = item.CategoryBookCount + 1;
                            item.CategoryNameCount = item.CategoryName + " (" + item.CategoryBookCount + ")";
                            item.ModificationDate = DateTime.Now;
                        }
                        context.SubmitChanges();

                        var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Select(j => j);
                        foreach (var item in author)
                        {
                            item.AuthorBookCount = item.AuthorBookCount + 1;
                            item.AuthorNameCount = item.AuthorName + " (" +
item.AuthorBookCount + ")";
                            item.ModificationDate = DateTime.Now;
                        }
                        context.SubmitChanges();
                    }
                }
                MessageBox.Show(AppResources.BookSaveSuccess);
            }
            isFilled = false;
        }

        private void txtBookComment_TextChanged(object sender, TextChangedEventArgs e)
        {
            Dispatcher.BeginInvoke(() =>
            {
                double CurrentInputHeight = txtBookComment.ActualHeight;

                if (CurrentInputHeight > InputHeight)
                {
                    svBookComment.ScrollToVerticalOffset(svBookComment.VerticalOffset + CurrentInputHeight - InputHeight);
                }

                InputHeight = CurrentInputHeight;
            });
        }

        private void txtBookComment_GotFocus(object sender, RoutedEventArgs e)
        {
            App.RootFrame.RenderTransform = new CompositeTransform();
            flag = true;
        }

        private void txtBookComment_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            txtBookComment.Focus();
            //txtBookComment.Select(txtBookComment.Text.Length, 1);
            svBookComment.ScrollToVerticalOffset(e.GetPosition(txtBookComment).Y - 80);
        }

        private void txtBookComment_LostFocus(object sender, RoutedEventArgs e)
        {
            if (!flag) return;
            txtBookComment.Focus();
            flag = false;
            this.pnlKeyboardPlaceHolder.Visibility = Visibility.Collapsed;
        }

        private void txtBookComment_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                svBookComment.ScrollToVerticalOffset(txtBookComment.ActualHeight);
            }
        }

        private void svBookComment_GotFocus(object sender, RoutedEventArgs e)
        {
            this.svBookComment.ScrollToVerticalOffset(this.txtBookComment.ActualHeight);
            this.svBookComment.UpdateLayout();
        }

        private void txtBookName_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                pvAuthor.SelectedIndex = 1;
                txtPageNumber.Focus();
            }
        }

        private void txtPageNumber_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                pvAuthor.SelectedIndex = 2;
                txtPublisherName.Focus();
            }
        }

        private void txtPublisherName_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                pvAuthor.SelectedIndex = 3;
                dtStart.Focus();
            }
        }

        private void dtStart_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                pvAuthor.SelectedIndex = 3;
                dtFinish.Focus();
            }
        }

        private void dtFinish_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                pvAuthor.SelectedIndex = 4;
                rtRating.Focus();
            }
        }

        private void rtRating_KeyDown(object sender, KeyEventArgs e)
        {
            //if (e.Key == Key.Enter)
            //{
            //    pvAuthor.SelectedIndex = 5;
            //    txtBookComment.Focus();
            //}
        }

        private void rtRating_ValueChanged(object sender, EventArgs e)
        {
            //pvAuthor.SelectedIndex = 5;
            ratingValue = rtRating.Value;
            //txtBookComment.Focus();
        }

        private void dtFinish_ValueChanged(object sender, DateTimeValueChangedEventArgs e)
        {
            if (isFilled)
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var book = context.Books.Where(j => j.BookId.Equals(bookId)).Select(j => j);
                    foreach (var item in book)
                    {
                        item.ReadFinishDate =
DateTime.Parse(dtFinish.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtFinish.Value.ToString());
                        item.ModificationDate = DateTime.Now;
                    }
                    context.SubmitChanges();
                }
            }
            pvAuthor.SelectedIndex = 4;
            rtRating.Focus();
        }

        private void dtStart_ValueChanged(object sender, DateTimeValueChangedEventArgs e)
        {
            if (isFilled)
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var book = context.Books.Where(j => j.BookId.Equals(bookId)).Select(j => j);
                    foreach (var item in book)
                    {
                        item.ReadStartDate = DateTime.Parse(dtStart.Value.ToString()) == null ? DateTime.Now : DateTime.Parse(dtStart.Value.ToString());
                        item.ModificationDate = DateTime.Now;
                    }
                    context.SubmitChanges();
                }
            }
            pvAuthor.SelectedIndex = 3;
            dtFinish.Focus();
        }

        private void rtRating_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            //pvAuthor.SelectedIndex = 5;
            //ratingValue = rtRating.Value;
            //txtBookComment.Focus();
        }

        private void DeleteBookMenuItem_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show(AppResources.DeleteBookQuestion,
                AppResources.DeleteBook, MessageBoxButton.OKCancel)
                != MessageBoxResult.OK)
            {

            }
            else
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var book = context.Books.Where(j => j.BookId.Equals(bookId)).Single() as Book;
                    var bookAuthors = context.BookAuthors.Where(j => j.BookId.Equals(bookId)).ToList() as List<BookAuthor>;
                    context.BookAuthors.DeleteAllOnSubmit(bookAuthors);
                    context.Books.DeleteOnSubmit(book);

                    var authors = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Select(j => j);
                    foreach (var item in authors)
                    {
                        item.ModificationDate = DateTime.Now;
                        item.AuthorBookCount = item.AuthorBookCount - 1;
                        item.AuthorNameCount = item.AuthorName + " (" +
item.AuthorBookCount + ")";
                    }
                    context.SubmitChanges();

                    var categories = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Select(j => j);
                    foreach (var item in categories)
                    {
                        item.ModificationDate = DateTime.Now;
                        item.CategoryBookCount = item.CategoryBookCount - 1;
                        item.CategoryNameCount = item.CategoryName + " (" + item.CategoryBookCount + ")";
                    }
                    context.SubmitChanges();
                }
                MessageBox.Show(AppResources.BookDeleteSuccess);
                NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
            }
            //MessageBox.Show(AppResources.NoteSaved);
        }

        private void btnIncrease_Click(object sender, RoutedEventArgs e)
        {
            if (rtRating.Value != 10.0)
            {
                rtRating.Value = rtRating.Value + 1.0;
            }
        }

        private void btnDecrease_Click(object sender, RoutedEventArgs e)
        {
            if (rtRating.Value != 0.0)
            {
                rtRating.Value = rtRating.Value - 1.0;
            }
        }
    }
}
```

Namespaces

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

1.2.17 Category.cs

This is file Category.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class Category
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int CategoryId { get; set; }

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

        [Column]
        public int CategoryBookCount { get; set; }

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

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

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

        [Column]
        public DateTime CreationDate { get; set; }

        [Column]
        public DateTime ModificationDate { get; set; }
    }
}
```

Namespaces

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

1.2.18 CategoryAuthor.cs

This is file CategoryAuthor.cs.

Body Source

```
?using System;
using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AwesomeLibrary
{
    [Table]
    public class CategoryAuthor
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int CategoryAuthorId { get; set; }

        [Column]
        public int CategoryId { get; set; }

        [Column]
        public int AuthorId { get; set; }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class CategoryPage : PhoneApplicationPage
    {
        public Popup popup;
        public int categoryId;
        public string oldCategoryName;
        public CategoryPage()
        {
            InitializeComponent();

            ApplicationBar = new ApplicationBar();

            ApplicationBarIconButton button1 = new ApplicationBarIconButton();
            button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
            button1.Text = AppResources.AddAuthor;
            ApplicationBar.Buttons.Add(button1);
            button1.Click += new EventHandler(AddAuthorButton_Click);

            ApplicationBarIconButton button2 = new ApplicationBarIconButton();
            button2.IconUri = new Uri("/Assets/Delete.png", UriKind.Relative);
            button2.Text = AppResources.DeleteCategory;
            ApplicationBar.Buttons.Add(button2);
            button2.Click += new EventHandler(DeleteCategoryButton_Click);

            ApplicationBarIconButton button3 = new ApplicationBarIconButton();
            button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
            button3.Text = AppResources.CategorySettings;
            ApplicationBar.Buttons.Add(button3);
            button3.Click += new EventHandler(CategorySettingsButton_Click);

            SetBackgroundColor();
            popup = new Popup();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            List<Author> authors = new List<Author>();
            List<Author> authorsOrdered = new List<Author>();

            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var category = context.Categories.Where(j => j.CategoryId.Equals(e.Fragment)).Single() as Category;
                string orderStyle = category.AuthorOrderStyle;
                var categoryAuthor = context.CategoryAuthors.Where(j => j.CategoryId.Equals(e.Fragment)).ToList() as List<CategoryAuthor>;

                foreach (var item in categoryAuthor)
                {
                    try
                    {
                        authors.Add(context.Authors.Where(j => j.AuthorId.Equals(item.AuthorId)).Single());
                    }
                    catch (Exception)
                    {
                    }

                }

                switch (category.AuthorOrderBy)
                {
                    case "NAME":
                        if (orderStyle == "A")
                        {
                            authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
                        }
                        else
                        {
                            authorsOrdered = authors.OrderByDescending(j => j.AuthorName).ToList();
                        }
                        break;
                    case "BOOKCOUNT":
                        if (orderStyle == "A")
                        {
                            authorsOrdered = authors.OrderBy(j => j.AuthorBookCount).ToList();
                        }
                        else
                        {
                            authorsOrdered = authors.OrderByDescending(j => j.AuthorBookCount).ToList();
                        }
                        break;
                    case "CDATE":
                        if (orderStyle == "A")
                        {
                            authorsOrdered = authors.OrderBy(j => j.CreationDate).ToList();
                        }
                        else
                        {
                            authorsOrdered = authors.OrderByDescending(j => j.CreationDate).ToList();
                        }
                        break;
                    case "MDATE":
                        if (orderStyle == "A")
                        {
                            authorsOrdered = authors.OrderBy(j => j.ModificationDate).ToList();
                        }
                        else
                        {
                            authorsOrdered = authors.OrderByDescending(j => j.ModificationDate).ToList();
                        }
                        break;
                    default:
                        if (orderStyle == "A")
                        {
                            authorsOrdered = authors.OrderBy(j => j.AuthorName).ToList();
                        }
                        else
                        {
                            authorsOrdered = authors.OrderByDescending(j => j.AuthorName).ToList();
                        }
                        break;
                }

                lstAuthors.Items.Clear();
                categoryId = category.CategoryId;
                lblCategoryName.Text = category.CategoryName;
                lblAuthorList.Text = AppResources.AuthorList + " (" + category.CategoryName + ")";
                lstAuthors.ItemsSource = authorsOrdered;
                lstAuthors.DisplayMemberPath = "AuthorNameCount";
                SetBackgroundColor();
                //lstNoteList.DisplayMemberPath = "NameCreation";
            }
        }

        private void AddAuthorButton_Click(object sender, EventArgs e)
        {
            popup = new Popup();
            popup.Height = 300;
            popup.Width = 400;
            popup.VerticalOffset = 20;
            PopupAddChange control = new PopupAddChange();
            control.txtLabel.Text = AppResources.EnterAuthorName;
            control.btnCancel.Content = AppResources.Cancel;
            control.btnOK.Content = AppResources.OK;
            popup.Child = control;
            popup.IsOpen = true;
            control.txtName.Focus();

            control.btnOK.Click += (s, args) =>
            {
                bool isCreated;
                string authorName;
                popup.IsOpen = false;

                int length = control.txtName.Text.Length;
                string space = control.txtName.Text.Substring(length - Math.Min(1, length));
                if (space == " ")
                {
                    authorName = control.txtName.Text.Remove(length - 1, 1);
                }
                else
                {
                    authorName = control.txtName.Text;
                }

                // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
                // kontrol eden bir kod bölümü
                using (var contextAuthor = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    isCreated =
                        contextAuthor.Authors.Any(j => j.AuthorName.Equals(authorName));
                }
                if (isCreated == true)
                {
                    MessageBox.Show(AppResources.AuthorExists);
                }
                // eger bu isimde bir klasör olusturulmamissa
                // olusturulmasi için gerekli kodlar asagidadir
                else
                {
                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        Author author = new Author();
                        author.AuthorName = authorName;
                        author.CreationDate = DateTime.Now;
                        author.ModificationDate = DateTime.Now;
                        author.AuthorBookCount = 0;
                        // burada yazarin kitaplarini
                        // bitirme tarihine göre azalan bir sekilde ayarlamak için gerekli düzenleme yapiliyor
                        author.BookOrderBy = "FDATE";
                        author.BookOrderStyle = "D";
                        author.AuthorNameCount = author.AuthorName + " (" + author.AuthorBookCount + ")";
                        //note.NameDescriptionWithoutNewline = note.NameDescription.Replace(Environment.NewLine," ");
                        //note.IsPasswordProtected = false;

                        context.Authors.InsertOnSubmit(author);
                        context.SubmitChanges();

                        Author author3 = context.Authors.Where(j => j.AuthorName.Equals(authorName)).Single() as Author;

                        CategoryAuthor categoryAuthor = new CategoryAuthor();
                        categoryAuthor.CategoryId = categoryId;
                        categoryAuthor.AuthorId = author3.AuthorId;
                        context.CategoryAuthors.InsertOnSubmit(categoryAuthor);
                        context.SubmitChanges();

                        var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Select(j => j);
                        foreach (var item in category)
                        {
                            item.ModificationDate = DateTime.Now;
                            //item.CategoryNameCount = item.CategoryName + " (" + item.auth + ")";
                        }
                        context.SubmitChanges();

                        var appSettings = context.AppSettings;
                        foreach (var appSetting in appSettings)
                        {
                            appSetting.CurrentCategoryNumber = categoryId;
                        }
                        context.SubmitChanges();

                        List<Author> authors = new List<Author>();
                        var categoryAuthors = context.CategoryAuthors.Where(j => j.CategoryId.Equals(categoryId)).ToList() as List<CategoryAuthor>;
                        foreach (var item in categoryAuthors)
                        {
                            authors.Add(context.Authors.Where(j => j.AuthorId.Equals(item.AuthorId)).Single());
                        }
                        lstAuthors.ItemsSource = authors;
                        MessageBox.Show(AppResources.AuthorAddSuccess);
                        //Author author2 = context.Authors.Where(j => j.AuthorName.Equals(authorName)).Single() as Author;

                        var appSettings2 = context.AppSettings;
                        foreach (var item in appSettings2)
                        {
                            item.CurrentAuthorNumber = author3.AuthorId;
                        }
                        context.SubmitChanges();
                        NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + author3.AuthorId, UriKind.Relative));
                    }
                }
            };
            control.btnCancel.Click += (s, args) =>
            {
                popup.IsOpen = false;
            };

            //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
        }

        private void CategorySettingsButton_Click(object sender, EventArgs e)
        {
            NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
        }

        private void lblCategoryName_Tap(object sender, System.Windows.Input.GestureEventArgs e)
        {
            oldCategoryName = lblCategoryName.Text;
            popup = new Popup();
            popup.Height = 300;
            popup.Width = 400;
            popup.VerticalOffset = 20;
            PopupAddChange control = new PopupAddChange();
            control.txtLabel.Text = AppResources.EnterCategoryName;
            control.btnCancel.Content = AppResources.Cancel;
            control.btnOK.Content = AppResources.OK;
            popup.Child = control;
            popup.IsOpen = true;
            control.txtName.Text = lblCategoryName.Text;
            control.txtName.Focus();
            control.txtName.Select(0, control.txtName.Text.Length);

            control.btnOK.Click += (s, args) =>
            {
                bool isCreated;
                string categoryName;
                popup.IsOpen = false;

                int length = control.txtName.Text.Length;
                string space = control.txtName.Text.Substring(length - Math.Min(1, length));
                if (space == " ")
                {
```

1

```
                    categoryName = control.txtName.Text.Remove(length - 1, 1);
                }
                else
                {
                    categoryName = control.txtName.Text;
                }

                if (categoryName != lblCategoryName.Text)
                {
                    // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
                    // kontrol eden bir kod bölümü
                    using (var contextFolder = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        isCreated =
                            contextFolder.Categories.Any(j => j.CategoryName.Equals(categoryName));
                    }
                    if (isCreated == true)
                    {
                        MessageBox.Show(AppResources.CategoryExists);
                    }
                    // eger bu isimde bir klasör olusturulmamissa
                    // olusturulmasi için gerekli kodlar asagidadir
                    else
                    {
                        using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                        {

                            // buraya kitapla ilgili bilginin güncellenecegi kod da eklenecek

                            var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Select(j => j);
                            foreach (var item in category)
                            {
                                item.CategoryName = categoryName;
                                item.ModificationDate = DateTime.Now;
                                item.CategoryNameCount = categoryName + " (" + item.CategoryBookCount.ToString() + ")";
                            }
                            context.SubmitChanges();

                            var book = context.Books.Where(j => j.BookCategoryId.Equals(categoryId)).Select(j => j);
                            foreach (var item in book)
                            {
                                item.BookCategoryName = item.BookCategoryName.Replace(oldCategoryName, categoryName);

                                //item.BookInformation = item.BookInformation.Replace(oldCategoryName, categoryName);
                                item.ModificationDate = DateTime.Now;
                            }
                            context.SubmitChanges();
                            //lstFolders.ItemsSource = context.NoteFolders;
                            //lstAuthors.ItemsSource = context.Categories;
                            MessageBox.Show(AppResources.CategoryNameChangeSuccess);
                            popup.IsOpen = false;
                            Category category2 = context.Categories.Where(j => j.CategoryName.Equals(categoryName)).Single() as Category;
                            NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + category2.CategoryId, UriKind.Relative));
                        }
                    }
                }
            };
            control.btnCancel.Click += (s, args) =>
            {
                popup.IsOpen = false;
            };
        }

        private void DeleteCategoryButton_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show(AppResources.DeleteCategoryQuestion,
                AppResources.DeleteCategory, MessageBoxButton.OKCancel)
                != MessageBoxResult.OK)
            {

            }
            else
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var books = context.Books.Where(j => j.BookCategoryId.Equals(categoryId)).ToList() as List<Book>;
                    foreach (var item in books)
                    {
                        var bookAuthors = context.BookAuthors.Where(j => j.BookId.Equals(item.BookId)).ToList() as List<BookAuthor>;
                        context.BookAuthors.DeleteAllOnSubmit(bookAuthors);
                    }
                    context.Books.DeleteAllOnSubmit(books);

                    var authorCategories = context.CategoryAuthors.Where(j => j.CategoryId.Equals(categoryId)).ToList() as List<CategoryAuthor>;
                    foreach (var item in authorCategories)
                    {
                        var author = context.Authors.Where(j => j.AuthorId.Equals(item.AuthorId)).ToList() as List<Author>;
                        context.Authors.DeleteAllOnSubmit(author);
                    }
                    context.CategoryAuthors.DeleteAllOnSubmit(authorCategories);

                    var categories = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
                    context.Categories.DeleteOnSubmit(categories);

                    context.SubmitChanges();
                }
                MessageBox.Show(AppResources.CategoryDeleteSuccess);
                NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
            }
            //MessageBox.Show(AppResources.NoteSaved);
        }

        private void lstAuthors_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var author = (Author)lstAuthors.SelectedItem;
            int authorId = author.AuthorId;
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings;
                foreach (var item in appSettings)
                {
                    item.CurrentAuthorNumber = authorId;
                }
                context.SubmitChanges();
            }
            NavigationService.Navigate(new Uri("/AuthorPage.xaml#" + authorId, UriKind.Relative));
        }

        private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (popup.IsOpen)
            {
                popup.IsOpen = false;
            }
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //SetBackgroundColor();
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.IO.IsolatedStorage;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Live;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Phone.Tasks;


namespace AwesomeLibrary
{
    public partial class CategorySettingsPage : PhoneApplicationPage
    {
        public int categoryId;
        public CategorySettingsPage()
        {
            InitializeComponent();

            pvCategorySettings.Title = AppResources.CategorySettings;

            piOtherSettings.Header = AppResources.OtherSettings;
            btnAuthorOrder.Content = AppResources.Select;
            btnAuthorOrderStyle.Content = AppResources.Select;
            SetBackgroundColor();

        }
        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var category = context.Categories.Where(j => j.CategoryId.Equals(e.Fragment)).Single() as Category;
                string orderStyle = category.AuthorOrderStyle;
                categoryId = category.CategoryId;

                if (category.AuthorOrderBy == "NAME")
                {
                    lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.Name + ")";
                }
                if (category.AuthorOrderBy == "BOOKCOUNT")
                {
                    lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.BookCount + ")";
                }
                if (category.AuthorOrderBy == "CDATE")
                {
                    lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.CreationDate + ")";
                }
                if (category.AuthorOrderBy == "MDATE")
                {
                    lblAuthorOrder.Text = AppResources.AuthorOrderBy + " (" + AppResources.Selected + ": " + AppResources.ModificationDate + ")";
                }
                if (category.AuthorOrderStyle == "A")
                {
                    lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
                }
                if (category.AuthorOrderStyle == "D")
                {
                    lblAuthorOrderStyle.Text = AppResources.AuthorOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
                }
                //lstNoteList.DisplayMemberPath = "NameCreation";
                SetBackgroundColor();
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //pvCategorySettings.Title = AppResources.CategorySettings;

            //piOtherSettings.Header = AppResources.OtherSettings;
            //btnAuthorOrder.Content = AppResources.Select;
            //btnAuthorOrderStyle.Content = AppResources.Select;
            //SetBackgroundColor();
        }

        private void btnAuthorOrder_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/OrderSettingsPage.xaml#" + categoryId, UriKind.Relative));
        }

        private void btnAuthorOrderStyle_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/OrderStyleSettingsPage.xaml#" + categoryId, UriKind.Relative));
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId, UriKind.Relative));
            }
        }
    }
}
```

Namespaces

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

1.2.21 FontFamilySettingsPage.xaml.cs

This is file FontFamilySettingsPage.xaml.cs.

Body Source

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class FontFamilySettingsPage : PhoneApplicationPage
    {
        public int authorId;
        public FontFamilySettingsPage()
        {
            InitializeComponent();

            lstFontFamily.Items.Clear();
            lstFontFamily.Items.Add("Arial");
            lstFontFamily.Items.Add("Arial Black");
            lstFontFamily.Items.Add("Baskerville Old Face");
            lstFontFamily.Items.Add("Berlin Sans FB");
            lstFontFamily.Items.Add("Bookman Old Style");
            lstFontFamily.Items.Add("Calibri");
            lstFontFamily.Items.Add("Cambria");
            lstFontFamily.Items.Add("Candara");
            lstFontFamily.Items.Add("Comic Sans MS");
            lstFontFamily.Items.Add("Consolas");
            lstFontFamily.Items.Add("Constantia");
            lstFontFamily.Items.Add("Courier New");
            lstFontFamily.Items.Add("DokChampa");
            lstFontFamily.Items.Add("Ebrima");
            lstFontFamily.Items.Add("Georgia");
            lstFontFamily.Items.Add("Lucida Sans Unicode");
            lstFontFamily.Items.Add("Meiryo UI");
            lstFontFamily.Items.Add("Microsoft YaHei");
            lstFontFamily.Items.Add("Malgun Gothic");
            lstFontFamily.Items.Add("Segoe UI");
            lstFontFamily.Items.Add("Segoe WP");
            lstFontFamily.Items.Add("Tahoma");
            lstFontFamily.Items.Add("Trebuchet MS");
            lstFontFamily.Items.Add("Times New Roman");
            lstFontFamily.Items.Add("Verdana");
            lstFontFamily.SelectedIndex = -1;
        }

        private void lstFontFamily_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (lstFontFamily.SelectedIndex != -1)
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var item in appSettings)
                    {
                        item.FontFamily = lstFontFamily.SelectedItem.ToString();
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.FontFamilyChangeSuccess);
                }
            }
            NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            authorId = int.Parse(e.Fragment);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var author = context.Authors.Where(j =>
j.AuthorId.Equals(authorId)).Single() as Author;
                lblAuthorName.Text = author.AuthorName;
                lblFontFamily.Text = AppResources.SelectFontFamily;
            }
            SetBackgroundColor();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" +
authorId, UriKind.Relative));
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class FontSizeSettingsPage : PhoneApplicationPage
    {
        public int authorId;
        public FontSizeSettingsPage()
        {
            InitializeComponent();

            lstFontSize.Items.Clear();
            lstFontSize.Items.Add("14");
            lstFontSize.Items.Add("18");
            lstFontSize.Items.Add("22");
            lstFontSize.Items.Add("26");
            lstFontSize.Items.Add("28");
            lstFontSize.Items.Add("30");
            lstFontSize.Items.Add("32");
            lstFontSize.Items.Add("34");
            lstFontSize.Items.Add("36");
            lstFontSize.Items.Add("38");
            lstFontSize.Items.Add("40");
            lstFontSize.Items.Add("42");
            lstFontSize.Items.Add("44");
            lstFontSize.Items.Add("64");
            lstFontSize.Items.Add("72");
            lstFontSize.SelectedIndex = -1;
        }

        private void lstFontSize_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (lstFontSize.SelectedIndex != -1)
            {
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var item in appSettings)
                    {
                        item.FontSize = lstFontSize.SelectedItem.ToString();
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.FontSizeChangeSuccess);
                }
            }
            NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            authorId = int.Parse(e.Fragment);
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
                lblAuthorName.Text = author.AuthorName;
                lblFontSize.Text = AppResources.SelectFontSize;
            }
            SetBackgroundColor();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.IO.IsolatedStorage;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Live;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Phone.Tasks;

namespace AwesomeLibrary
{
    public partial class GeneralSettingsPage : PhoneApplicationPage
    {

        private static readonly string[] scopes = new string[] { "wl.signin", 
"wl.basic", "wl.offline_access", "wl.skydrive", "wl.skydrive_update" };

        /// <summary>
        ///     Stores the LiveAuthClient instance.
        /// </summary>
        private LiveAuthClient authClient;

        /// <summary>
        ///     Stores the LiveConnectClient instance.
        /// </summary>
        private LiveConnectClient liveClient;

        public int signIn;

        public GeneralSettingsPage()
        {
            InitializeComponent();
            InitializePage();

            pvGeneralSettings.Title = AppResources.GeneralSettings;

            piLanguage.Header = AppResources.Language;
            piSync.Header = AppResources.Sync;
            piOtherSettings.Header = AppResources.OtherSettings;
            piBackground.Header = AppResources.Background;

            //lblOneDrive.Text = AppResources.OneDrive;

            btnCategoryOrder.Content = AppResources.Select;
            btnCategoryOrderStyle.Content = AppResources.Select;
            btnLanguage.Content = AppResources.Select;
            btnBackgroundColor.Content = AppResources.Select;
            //btnOneDrive.Content = AppResources.Login;
            //btnOneDrive.SignInText = AppResources.SignIn;
            //btnOneDrive.SignOutText = AppResources.SignOut;
            btnOneDriveSync.Content = AppResources.Sync;
            lblOneDrive.Text = AppResources.OneDrive;
            txtSyncronizing.Text = AppResources.Synchronizing;

            pbSync.Visibility = Visibility.Collapsed;
            txtSyncronizing.Visibility = Visibility.Collapsed;
            txtSyncronizing.BorderBrush = this.LayoutRoot.Background;

            btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
            lblBackgroundImage.Text = AppResources.BackgroundImage;
            btnBackgroundImage.Content = AppResources.Select;
            btnResetSettings.Content = AppResources.ResetSettings;

            btnOneDriveSync.IsEnabled = false;
            cbSync.Content = AppResources.SyncOnOneFile;
            cbSync.IsEnabled = false;
            btnOneDrive.Content = "Sign In";

            SetBackgroundColor();

            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings.First() as AppSettings;
                if (appSettings.AppLangName == "EN")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.English + ")";
                }
                if (appSettings.AppLangName == "TR")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Turkish + ")";
                }
                if (appSettings.AppLangName == "DE")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.German + ")";
                }
                if (appSettings.AppLangName == "ES")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Spanish + ")";
                }

                if (appSettings.AppLangName == "PT")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Portuguese + ")";
                }
                if (appSettings.AppLangName == "AR")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Arabic + ")";
                }
                if (appSettings.AppLangName == "FA")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Persian + ")";
                }
                if (appSettings.AppLangName == "IT")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Italian + ")";
                }
                if (appSettings.AppLangName == "FR")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.French + ")";
                }
                if (appSettings.AppLangName == "RU")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Russian + ")";
                }
                if (appSettings.AppLangName == "ZH")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Chinese + ")";
                }
                if (appSettings.AppLangName == "JA")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Japanese + ")";
                }
                if (appSettings.AppLangName == "SA")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Sanskrit + ")";
                }
                if (appSettings.AppLangName == "TH")
                {
                    lblLanguage.Text = AppResources.Language + " (" + AppResources.Selected + ": " + AppResources.Thai + ")";
                }


                if (appSettings.CategoryOrderBy == "NAME")
                {
                    lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.Name + ")";
                }
                if (appSettings.CategoryOrderBy == "CDATE")
                {
                    lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.CreationDate + ")";
                }
                if (appSettings.CategoryOrderBy == "MDATE")
                {
                    lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.ModificationDate + ")";
                }
                if (appSettings.CategoryOrderBy == "BOOKCOUNT")
                {
                    lblCategoryOrder.Text = AppResources.CategoryOrderBy + " (" + AppResources.Selected + ": " + AppResources.BookCount + ")";
                }
                if (appSettings.CategoryOrderStyle == "A")
                {
                    lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Ascending + ")";
                }
                if (appSettings.CategoryOrderStyle == "D")
                {
                    lblCategoryOrderStyle.Text = AppResources.CategoryOrderStyle + " (" + AppResources.Selected + ": " + AppResources.Descending + ")";
                }
                if (appSettings.AppBackgroundColor == "BLA")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Black + ")";
                }
                if (appSettings.AppBackgroundColor == "BLU")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Blue + ")";
                }
                if (appSettings.AppBackgroundColor == "BRO")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Brown + ")";
                }
                if (appSettings.AppBackgroundColor == "RED")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Red + ")";
                }
                if (appSettings.AppBackgroundColor == "GRE")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Green + ")";
                }
                if (appSettings.AppBackgroundColor == "YEL")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Yellow + ")";
                }
                if (appSettings.AppBackgroundColor == "GRA")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Gray + ")";
                }
                if (appSettings.AppBackgroundColor == "ORA")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Orange + ")";
                }
                if (appSettings.AppBackgroundColor == "PUR")
                {
                    lblBackgroundColor.Text = AppResources.BackgroundColor + " (" + AppResources.Selected + ": " + AppResources.Purple + ")";
                }
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            SetBackgroundColor();
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        private async void btnOneDrive_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (this.btnOneDrive.Content.ToString() == "Sign In" || this.btnOneDrive.Content.ToString() == "Sign in")
                {
                    LiveLoginResult loginResult = await this.authClient.LoginAsync(scopes);
                    if (loginResult.Status == LiveConnectSessionStatus.Connected)
                    {
                        //this.btnOneDrive.Content = AppResources.SignOut;
                        this.btnOneDrive.Content = "Sign Out";

                        this.liveClient = new LiveConnectClient(loginResult.Session);
                        this.GetMe();
                        btnOneDriveSync.IsEnabled = true;
                        cbSync.IsEnabled = true;
                    }
                }
                else
                {
                    this.authClient.Logout();
                    //this.btnOneDrive.Content = AppResources.SignIn;
                    this.btnOneDrive.Content = "Sign Out";
                    btnOneDriveSync.IsEnabled = true;
                    cbSync.IsEnabled = true;
                    //this.tbResponse.Text = "";
                }
            }
            catch (LiveAuthException authExp)
            {
                //this.tbResponse.Text = authExp.ToString();
            }
        }

        private async void InitializePage()
        {
            try
            {
                // bu benim uygulamama ait bir client id
                this.authClient = new LiveAuthClient("0000000044125951");
                LiveLoginResult loginResult = await this.authClient.InitializeAsync(scopes);
                btnOneDrive.Content = "Sign In";
                if (loginResult.Status == LiveConnectSessionStatus.Connected)
                {
                    //this.btnOneDrive.Content = AppResources.SignOut;
                    this.btnOneDrive.Content = "Sign Out";

                    this.liveClient = new LiveConnectClient(loginResult.Session);
                    //this.GetMe();
                }
            }
            catch (LiveAuthException authExp)
            {
                //this.tbResponse.Text = authExp.ToString();
            }
        }

        private async void GetMe()
        {
            try
            {
                LiveOperationResult operationResult = await this.liveClient.GetAsync("me");

                dynamic properties = operationResult.Result;
                //this.tbResponse.Text = properties.first_name + " " + properties.last_name;
            }
            catch (LiveConnectException e)
            {
                //this.tbResponse.Text = e.ToString();
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        public async static Task<string> CreateDirectoryAsync(LiveConnectClient client,
string folderName, string parentFolder)
        {
            string folderId = null;

            // Retrieves all the directories.
            var queryFolder = parentFolder + "/files?filter=folders,albums";
            var opResult = await client.GetAsync(queryFolder);
            dynamic result = opResult.Result;

            foreach (dynamic folder in result.data)
            {
                // Checks if current folder has the passed name.
                if (folder.name.ToLowerInvariant() == folderName.ToLowerInvariant())
                {
                    folderId = folder.id;
                    break;
                }
            }

            if (folderId == null)
            {
                // Directory hasn't been found, so creates it using the PostAsync method.
                var folderData = new Dictionary<string, object>();
                folderData.Add("name", folderName);
                opResult = await client.PostAsync(parentFolder, folderData);
                result = opResult.Result;

                // Retrieves the id of the created folder.
                folderId = result.id;
            }

            return folderId;
        }

        private async void btnOneDriveSync_Click(object sender, RoutedEventArgs e)
        {

            IsolatedStorageFile myIsolatedStorage = null;
            StringBuilder sb = null;


            string folderName;
            try
            {
                //var folderData = new Dictionary<string, object>();
                folderName = "Awesome Library (" + DateTime.Now + ")";
                //folderName = folderName.Replace(":", ".");
                //folderName = folderName.Replace("/", ".");
                folderName = DesignFileName(folderName);

                string skyDriveFolder = await CreateDirectoryAsync(liveClient, folderName, "me/skydrive");

                if (cbSync.IsChecked == false)
                {

                    btnOneDrive.IsEnabled = false;
                    pbSync.Visibility = Visibility.Visible;
                    txtSyncronizing.Visibility = Visibility.Visible;

                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        //var noteFolders = context.NoteFolders.ToList() as List<NoteFolder>;
                        var books = context.Books.ToList() as List<Book>;

                        for (int i = 0; i < books.Count; i++)
                        {
                            var bookAuthor =
                                context.BookAuthors.Where(j => j.BookId.Equals(books[i].BookId)).ToList() as
                                    List<BookAuthor>;

                            var category = context.Categories.Where(j => j.CategoryId.Equals(books[i].BookCategoryId)).Single() as Category;

                            List<Author> authors = new List<Author>();

                            for (int k = 0; k < bookAuthor.Count; k++)
                            {
                                authors.Add(context.Authors.Where(j => j.AuthorId.Equals(bookAuthor[k].AuthorId)).Single() as Author);
                            }


                            string fileName = Guid.NewGuid() + ". " + books[i].BookName + " (" + category.CategoryName +
                                              ").txt";

                            fileName = DesignFileName(fileName);
                            //fileName = fileName.Replace(":", ".");
                            //fileName = fileName.Replace("/", ".");
                            //StringBuilder sb = new StringBuilder();
                            //sb.AppendLine(AppResources.NoteName + ": " +
notes[i].NoteName);
                            //sb.AppendLine(AppResources.FolderName + ": " + noteFolder.NoteFolderName);
                            //sb.AppendLine(AppResources.Password + ": " + noteFolder.IsPasswordProtected);
                            //sb.AppendLine(AppResources.CreationDate + ": " + notes[i].CreationDate);
                            //sb.AppendLine(AppResources.ModificationDate + ": " + notes[i].ModificationDate);
                            //sb.AppendLine(AppResources.Note + ": " + notes[i].NoteDescription);


                            myIsolatedStorage = IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists
                            //if (myIsolatedStorage.FileExists(fileName))
                            //{
                            //myIsolatedStorage.DeleteFile(fileName);
                            //}//now we use a StreamWriter to write inputBox.Text to the file and save it to IsolatedStorage
                            using (StreamWriter writeFile = new StreamWriter
                            (new IsolatedStorageFileStream(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
                            {
                                writeFile.WriteLine(AppResources.BookName + ": " + books[i].BookName);
                                writeFile.WriteLine(AppResources.CategoryName + ": " + category.CategoryName);
                                string authorNames = "";
                                for (int l = 0; l < authors.Count; l++)
                                {
                                    authorNames = authorNames + authors[l].AuthorName + ", ";
                                }
                                authorNames = authorNames.Substring(0, authorNames.Length - 2);
                                writeFile.WriteLine(AppResources.AuthorName + ": " + authorNames);
                                writeFile.WriteLine(AppResources.PageNumber + ": " + books[i].BookPageNumber);
                                writeFile.WriteLine(AppResources.PublisherName + ": " + books[i].BookPublisherName);
                                writeFile.WriteLine(AppResources.StartDate + ": " + books[i].ReadStartDate.ToShortDateString());
                                writeFile.WriteLine(AppResources.FinishDate + ": " + books[i].ReadFinishDate.ToShortDateString());
                                writeFile.WriteLine(AppResources.BookRating + ": " + books[i].BookRating + "/10");
                                writeFile.WriteLine(AppResources.BookComment + ": " + books[i].BookComment);
                                writeFile.Close();
                            }
                            IsolatedStorageFileStream isfs = myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite);
                            var res = await liveClient.UploadAsync(skyDriveFolder, fileName, isfs, OverwriteOption.Overwrite);
                            pbSync.Value = (i + 1) * (100) / books.Count;
                            //var res = await liveClient.UploadAsync("me/skydrive/" + folderName, fileName, isfs, OverwriteOption.Overwrite);
                        }
                    }
                }
                else
                {
                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        //var noteFolders = context.NoteFolders.ToList() as
```

```
List<NoteFolder>;
            var books = context.Books.OrderBy(j => j.ReadFinishDate).ToList() as List<Book>;
            var bookFirst = books.First();
            var bookLast = books.Last();

            string fileName = Guid.NewGuid() + ". Awesome Library (" + bookFirst.ReadFinishDate.ToShortDateString() + " - " + bookLast.ReadFinishDate.ToShortDateString() + ").txt";
            fileName = DesignFileName(fileName);

            myIsolatedStorage = IsolatedStorageFile.GetUserStoreForApplication();//deletes the file if it already exists

            sb = new StringBuilder();

            for (int i = 0; i < books.Count; i++)
            {
                var bookAuthor =
                    context.BookAuthors.Where(j => j.BookId.Equals(books[i].BookId)).ToList() as
                        List<BookAuthor>;

                var category = context.Categories.Where(j => j.CategoryId.Equals(books[i].BookCategoryId)).Single() as Category;

                List<Author> authors = new List<Author>();

                for (int k = 0; k < bookAuthor.Count; k++)
                {
                    authors.Add(context.Authors.Where(j => j.AuthorId.Equals(bookAuthor[k].AuthorId)).Single() as Author);
                }

                sb.AppendLine();
                sb.AppendLine(AppResources.BookName + ": " + books[i].BookName);
                sb.AppendLine(AppResources.CategoryName + ": " + category.CategoryName);
                string authorNames = "";
                for (int l = 0; l < authors.Count; l++)
                {
                    authorNames = authorNames + authors[l].AuthorName + ", ";
                }
                authorNames = authorNames.Substring(0, authorNames.Length - 2);
                sb.AppendLine(AppResources.AuthorName + ": " + authorNames);
                sb.AppendLine(AppResources.PageNumber + ": " + books[i].BookPageNumber);
                sb.AppendLine(AppResources.PublisherName + ": " + books[i].BookPublisherName);
                sb.AppendLine(AppResources.StartDate + ": " + books[i].ReadStartDate.ToShortDateString());
                sb.AppendLine(AppResources.FinishDate + ": " + books[i].ReadFinishDate.ToShortDateString());
                sb.AppendLine(AppResources.BookRating + ": " + books[i].BookRating + "/10");
                sb.AppendLine(AppResources.BookComment + ": " + books[i].BookComment);
                sb.AppendLine();

                //if (myIsolatedStorage.FileExists(fileName))
                //{
                //myIsolatedStorage.DeleteFile(fileName);
                //}//now we use a StreamWriter to write inputBox.Text to the file and save it to IsolatedStorage
                //pbSync.Value = (i + 1) * (100) / books.Count;
```

```
                            //var res = await liveClient.UploadAsync("me/skydrive/" + folderName, fileName, isfs, OverwriteOption.Overwrite);
                        }
                        using (StreamWriter writeFile = new StreamWriter
                            (new IsolatedStorageFileStream(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite, myIsolatedStorage)))
                        {
                            writeFile.Write(sb.ToString());
                            writeFile.Close();
                        }
                        IsolatedStorageFileStream isfs = myIsolatedStorage.OpenFile(fileName, FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.ReadWrite);
                        var res = await liveClient.UploadAsync(skyDriveFolder, fileName, isfs, OverwriteOption.Overwrite);
                    }
                }

                //this.infoTextBlock.Text = string.Join(" ", "Created folder:", result.name, "ID:", result.id);
                MessageBox.Show(AppResources.OneDriveSyncCompleted);

                pbSync.Visibility = Visibility.Collapsed;
                txtSyncronizing.Visibility = Visibility.Collapsed;
                pbSync.Value = 0;
                btnOneDrive.IsEnabled = true;
            }
            catch (Exception exception)
            {
                //this.infoTextBlock.Text = "Error creating folder: " + exception.Message;
                MessageBox.Show(AppResources.SystemFault);
            }
        }

        public string DesignFileName(string fileName)
        {
            fileName = fileName.Replace(":", ".");
            fileName = fileName.Replace("?", ".");
            fileName = fileName.Replace("\"", ".");
            fileName = fileName.Replace("/", ".");
            fileName = fileName.Replace("<", ".");
            fileName = fileName.Replace(">", ".");
            fileName = fileName.Replace("|", ".");
            fileName = fileName.Replace("*", ".");
            return fileName;
        }

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

        private void btnBackgroundImage_Click(object sender, RoutedEventArgs e)
        {
            PhotoChooserTask objPhotoChooser = new PhotoChooserTask();
            objPhotoChooser.Completed += new EventHandler<PhotoResult>(PhotoChooseCall);
            objPhotoChooser.Show();
        }

        private void PhotoChooseCall(object sender, PhotoResult e)
        {
            switch (e.TaskResult)
            {
                case TaskResult.OK:
                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        var appSettings = context.AppSettings;
                        foreach (var appSetting in appSettings)
                        {
                            appSetting.AppBackgroundImage = new byte[e.ChosenPhoto.Length];
                            e.ChosenPhoto.Position = 0;
                            e.ChosenPhoto.Read(appSetting.AppBackgroundImage, 0, appSetting.AppBackgroundImage.Length);
                            //noteFolder.NoteFolderPassword = "";
                        }
                        context.SubmitChanges();
                        MessageBox.Show(AppResources.BackgroundImageChangeSuccess);
                    }
                    break;
                case TaskResult.Cancel:
                    //MessageBox.Show("Cancelled");
                    break;
                case TaskResult.None:
                    //MessageBox.Show("Nothing Entered");
                    break;
            }
            SetBackgroundColor();
        }

        private void btnRemoveBackgroundImage_Click(object sender, RoutedEventArgs e)
        {
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings;
                foreach (var appSetting in appSettings)
                {
                    appSetting.AppBackgroundImage = null;
                }
                context.SubmitChanges();
                MessageBox.Show(AppResources.BackgroundImageRemoveSuccess);
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {

            //pvGeneralSettings.Title = AppResources.GeneralSettings;

            //piLanguage.Header = AppResources.Language;
            //piSync.Header = AppResources.Sync;
            //piOtherSettings.Header = AppResources.OtherSettings;
            //piBackground.Header = AppResources.Background;

            ////lblOneDrive.Text = AppResources.OneDrive;

            //btnCategoryOrder.Content = AppResources.Select;
            //btnCategoryOrderStyle.Content = AppResources.Select;
            //btnLanguage.Content = AppResources.Select;
            //btnBackgroundColor.Content = AppResources.Select;
            ////btnOneDrive.Content = AppResources.Login;
            ////btnOneDrive.SignInText = AppResources.SignIn;
            ////btnOneDrive.SignOutText = AppResources.SignOut;
            //btnOneDriveSync.Content = AppResources.Sync;
            //lblOneDrive.Text = AppResources.OneDrive;
            //txtSyncronizing.Text = AppResources.Synchronizing;

            //pbSync.Visibility = Visibility.Collapsed;
            //txtSyncronizing.Visibility = Visibility.Collapsed;
            //txtSyncronizing.BorderBrush = this.LayoutRoot.Background;

            //btnRemoveBackgroundImage.Content = AppResources.RemoveBackgroundImage;
            //lblBackgroundImage.Text = AppResources.BackgroundImage;
            //btnBackgroundImage.Content = AppResources.Select;
            //btnResetSettings.Content = AppResources.ResetSettings;

            //btnOneDriveSync.IsEnabled = false;
            //cbSync.Content = AppResources.SyncOnOneFile;
            //cbSync.IsEnabled = false;
            //btnOneDrive.Content = "Sign In";

            //SetBackgroundColor();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
            }
        }

        private void btnOneDrive_SessionChanged(object sender, Microsoft.Live.Controls.LiveConnectSessionChangedEventArgs e)
        {
            if (e != null && e.Status == LiveConnectSessionStatus.Connected)
            {
                //the session status is connected so we need to set this session status to client
                this.liveClient = new LiveConnectClient(e.Session);
            }
            else
            {
                this.liveClient = null;
            }
        }

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

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

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

        private void btnResetSettings_Click(object sender, RoutedEventArgs e)
        {
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings;
                foreach (var appSetting in appSettings)
                {
                    appSetting.AppBackgroundImage = null;
                    appSetting.AppBackgroundColor = "BLA";
                }
                context.SubmitChanges();
                this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                MessageBox.Show(AppResources.SuccessfulResetSettings);
            }
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Net;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
{
    public partial class LanguageSettingsPage : PhoneApplicationPage
    {
        public LanguageSettingsPage()
        {
            InitializeComponent();

            lstLanguage.Items.Clear();
            lstLanguage.Items.Add(AppResources.English);
            lstLanguage.Items.Add(AppResources.Turkish);
            lstLanguage.Items.Add(AppResources.German);
            //lstLanguage.Items.Add(AppResources.Spanish);
            lstLanguage.Items.Add(AppResources.Russian);
            lstLanguage.Items.Add(AppResources.Arabic);
            lstLanguage.Items.Add(AppResources.Persian);
            lstLanguage.Items.Add(AppResources.Chinese);
            lstLanguage.Items.Add(AppResources.Italian);
            lstLanguage.Items.Add(AppResources.French);
            lstLanguage.Items.Add(AppResources.Japanese);
            lstLanguage.Items.Add(AppResources.Spanish);
            lstLanguage.Items.Add(AppResources.Sanskrit);
            lstLanguage.Items.Add(AppResources.Thai);

            lstLanguage.SelectedIndex = -1;
            lblLanguage.Text = AppResources.SelectLanguage;
            lblGeneralSettings.Text = AppResources.GeneralSettings;

            SetBackgroundColor();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            SetBackgroundColor();
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void lstLanguage_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            int index = lstLanguage.SelectedIndex;
            string culture = "";
            string lang = "";
            if (index == 0)
            {
                culture = "en";
                lang = "EN";
            }
            else if (index == 1)
            {
                culture = "tr";
                lang = "TR";
            }
            else if (index == 2)
            {
                culture = "de";
                lang = "DE";
            }
            else if (index == 3)
            {
                culture = "ru";
                lang = "RU";
            }
            else if (index == 4)
            {
                culture = "ar";
                lang = "AR";
            }
            else if (index == 5)
            {
                culture = "fa-IR";
                lang = "FA";
            }
            else if (index == 6)
            {
                culture = "zh";
                lang = "ZH";
            }
            else if (index == 7)
            {
                culture = "it";
                lang = "IT";
            }
            else if (index == 8)
            {
                culture = "fr";
                lang = "FR";
            }
            else if (index == 9)
            {
                culture = "ja";
                lang = "JA";
            }
            else if (index == 10)
            {
                culture = "es";
                lang = "ES";
            }
```

```
                else if (index == 11)
                {
                    culture = "sa";
                    lang = "SA";
                }
                else if (index == 12)
                {
                    culture = "th";
                    lang = "TH";
                }

                //else if (index == 3)
                //{
                //    culture = "es";
                //    lang = "ES";
                //}
                //else if (index == 4)
                //{
                //    culture = "ru";
                //    lang = "RU";
                //}
                //else if (index == 5)
                //{
                //    culture = "zh";
                //    lang = "AR";
                //}
                //else if (index == 6)
                //{
                //    culture = "ar";
                //    lang = "AR";
                //}
                //else if (index == 7)
                //{
                //    culture = "fa-IR";
                //    lang = "FA";
                //}
                //else if (index == 8)
                //{
                //    culture = "it";
                //    lang = "IT";
                //}
                //else if (index == 9)
                //{
                //    culture = "fr";
                //    lang = "FR";
                //}
                //else if (index == 10)
                //{
                //    culture = "pt";
                //    lang = "PT";
                //}
                else
                {
                    culture = "en";
                    lang = "EN";
                }

                using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var appSetting in appSettings)
                    {
                        appSetting.AppLangName = lang;
                    }
                    context.SubmitChanges();
                }

                CultureInfo newCulture = new CultureInfo(culture);
```

```
            Thread.CurrentThread.CurrentCulture = newCulture;
            Thread.CurrentThread.CurrentUICulture = newCulture;
            MessageBox.Show(AppResources.LanguageWarning);
            NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //SetBackgroundColor();
        }
    }
}
```

Namespaces

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

1.2.25 LocalizedStrings.cs

This is file LocalizedStrings.cs.

Body Source

```
?using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    /// <summary>
    /// Provides access to string resources.
    /// </summary>
    public class LocalizedStrings
    {
        private static AppResources _localizedResources = new AppResources();

        public AppResources LocalizedResources { get { return _localizedResources; } }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Data.Common;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Controls.Primitives;
using Microsoft.Phone.Shell;
using AwesomeLibrary.Resources;

namespace AwesomeLibrary
{
    public partial class MainPage : PhoneApplicationPage
    {
        public Popup popup;
        // Constructor
        public MainPage()
        {
            InitializeComponent();


            ApplicationBar = new ApplicationBar();

            ApplicationBarIconButton button1 = new ApplicationBarIconButton();
            button1.IconUri = new Uri("/Assets/Add.png", UriKind.Relative);
            button1.Text = AppResources.AddCategory;
            ApplicationBar.Buttons.Add(button1);
            button1.Click += new EventHandler(AddCategoryButton_Click);

            ApplicationBarIconButton button2 = new ApplicationBarIconButton();
            button2.IconUri = new Uri("/Assets/Search.png", UriKind.Relative);
            button2.Text = AppResources.Search;
            ApplicationBar.Buttons.Add(button2);
            button2.Click += new EventHandler(SearchButton_Click);

            ApplicationBarIconButton button3 = new ApplicationBarIconButton();
            button3.IconUri = new Uri("/Assets/Settings.png", UriKind.Relative);
            button3.Text = AppResources.Settings;
            ApplicationBar.Buttons.Add(button3);
            button3.Click += new EventHandler(SettingsButton_Click);

            ApplicationBarIconButton button4 = new ApplicationBarIconButton();
            button4.IconUri = new Uri("/Assets/Statistics.png", UriKind.Relative);
            button4.Text = AppResources.Statistics;
            ApplicationBar.Buttons.Add(button4);
            button4.Click += new EventHandler(StatisticsButton_Click);

            ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
            menuItem1.Text = AppResources.About;
            ApplicationBar.MenuItems.Add(menuItem1);
            menuItem1.Click += new EventHandler(AboutMenuItem_Click);

            lblCategories.Text = AppResources.Categories;
            // Sample code to localize the ApplicationBar
            //BuildLocalizedApplicationBar();

            SetBackgroundColor();

            popup = new Popup();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new 
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            //while (NavigationService.CanGoBack)
            //NavigationService.RemoveBackEntry();

        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (popup.IsOpen)
            {
                popup.IsOpen = false;
            }
            if (MessageBox.Show(AppResources.ExitAppQuestion,
                AppResources.ExitApp, MessageBoxButton.OKCancel)
                != MessageBoxResult.OK)
            {
                e.Cancel = true;
            }
            else
            {
                Application.Current.Terminate();
            }
        }

        private void lstCategories_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var category = (Category)lstCategories.SelectedItem;
            int categoryId = category.CategoryId;
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings;
                foreach (var appSetting in appSettings)
                {
                    appSetting.CurrentCategoryNumber = categoryId;
                }
                context.SubmitChanges();
            }
            NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + categoryId, UriKind.Relative));
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings.First() as AppSettings;
                string orderStyle = appSettings.CategoryOrderStyle;
                //lstCategories.ItemsSource = null;
                lstCategories.Items.Clear();
                if (context.Categories.Count() != 0)
                {
                    switch (appSettings.CategoryOrderBy)
                    {
                        case "NAME":
                            if (orderStyle == "A")
                            {
                                lstCategories.ItemsSource = context.Categories.OrderBy(j => j.CategoryName).ToList();
                            }
                            else
                            {
                                lstCategories.ItemsSource = context.Categories.OrderByDescending(j => j.CategoryName).ToList();
                            }
                            break;
                        case "CDATE":
                            if (orderStyle == "A")
                            {
                                lstCategories.ItemsSource = context.Categories.OrderBy(j => j.CreationDate).ToList();
                            }
                            else
                            {
                                lstCategories.ItemsSource = context.Categories.OrderByDescending(j => j.CreationDate).ToList();
                            }
                            break;
                        case "MDATE":
                            if (orderStyle == "A")
                            {
                                lstCategories.ItemsSource = context.Categories.OrderBy(j => j.ModificationDate).ToList();
                            }
                            else
                            {
                                lstCategories.ItemsSource = context.Categories.OrderByDescending(j => j.ModificationDate).ToList();
                            }
                            break;
                        case "BOOKCOUNT":
                            if (orderStyle == "A")
                            {
                                lstCategories.ItemsSource = context.Categories.OrderBy(j => j.CategoryBookCount).ToList();
                            }
                            else
                            {
                                lstCategories.ItemsSource = context.Categories.OrderByDescending(j => j.CategoryBookCount).ToList();
                            }
                            break;
                        default:
                            if (orderStyle == "A")
                            {
                                lstCategories.ItemsSource = context.Categories.OrderBy(j => j.CategoryName).ToList();
                            }
                            else
                            {
                                lstCategories.ItemsSource = context.Categories.OrderByDescending(j => j.CategoryName).ToList();
                            }
                            break;
                    }
                    lstCategories.DisplayMemberPath = "CategoryNameCount";
                    SetBackgroundColor();
                }
            }

            // Sample code for building a localized ApplicationBar
            //private void BuildLocalizedApplicationBar()
            //{
            //    // Set the page's ApplicationBar to a new instance of ApplicationBar.
            //    ApplicationBar = new ApplicationBar();

            //    // Create a new button and set the text value to the localized string from AppResources.
            //    ApplicationBarIconButton appBarButton = new ApplicationBarIconButton(new Uri("/Assets/AppBar/appbar.add.rest.png", UriKind.Relative));
            //    appBarButton.Text = AppResources.AppBarButtonText;
            //    ApplicationBar.Buttons.Add(appBarButton);

            //    // Create a new menu item with the localized string from AppResources.
            //    ApplicationBarMenuItem appBarMenuItem = new ApplicationBarMenuItem(AppResources.AppBarMenuItemText);
            //    ApplicationBar.MenuItems.Add(appBarMenuItem);
            //}
        }

        private void AddCategoryButton_Click(object sender, EventArgs e)
        {
            int categoryId;

            popup = new Popup();
            popup.Height = 300;
            popup.Width = 400;
            popup.VerticalOffset = 20;
            PopupAddChange control = new PopupAddChange();
            control.txtLabel.Text = AppResources.EnterCategoryName;
            control.btnCancel.Content = AppResources.Cancel;
            control.btnOK.Content = AppResources.OK;
            popup.Child = control;
            popup.IsOpen = true;
            control.txtName.Focus();

            control.btnOK.Click += (s, args) =>
            {
                bool folder = false;
                string categoryName;
                popup.IsOpen = false;

                int length = control.txtName.Text.Length;
                string space = control.txtName.Text.Substring(length - Math.Min(1, length));
                if (space == " ")
                {
                    categoryName = control.txtName.Text.Remove(length - 1, 1);
                }
                else
                {
                    categoryName = control.txtName.Text;
                }

                // ayni isimde bir klasörün daha önceden olusturulup olusturulmadigini
                // kontrol eden bir kod bölümü
                using (var contextFolder = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    folder =
                        contextFolder.Categories.Any(j => j.CategoryName.Equals(categoryName));
                }
                if (folder == true)
                {
                    MessageBox.Show(AppResources.CategoryExists);
                }
                // eger bu isimde bir klasör olusturulmamissa
                // olusturulmasi için gerekli kodlar asagidadir
                else
                {
                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        //noteFolderId = context.NoteFolders.Count() + 1;
                        Category category = new Category();
                        category.CategoryName = categoryName;
                        //noteFolder.NoteFolderId = noteFolderId;
                        category.CategoryBookCount = 0;
                        category.CreationDate = DateTime.Now;
                        category.ModificationDate = DateTime.Now;
                        category.AuthorOrderBy = "NAME";
                        category.AuthorOrderStyle = "A";
                        category.CategoryNameCount = category.CategoryName + " (" + category.CategoryBookCount.ToString() + ")";

                        context.Categories.InsertOnSubmit(category);
                        context.SubmitChanges();

                        //lstCategories.ItemsSource = context.Categories;
                        MessageBox.Show(AppResources.CategoryAddSuccess);
                        popup.IsOpen = false;
                        Category category2 = context.Categories.Where(j => j.CategoryName.Equals(categoryName)).Single() as Category;
                        NavigationService.Navigate(new Uri("/CategoryPage.xaml#" + category2.CategoryId, UriKind.Relative));
                    }

                }
            };
            control.btnCancel.Click += (s, args) =>
            {
                popup.IsOpen = false;
            };

            //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
        }

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

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

        private void AboutMenuItem_Click(object sender, EventArgs e)
        {
            NavigationService.Navigate(new Uri("/AboutPage.xaml", UriKind.Relative));
        }
        private void StatisticsButton_Click(object sender, EventArgs e)
        {
            NavigationService.Navigate(new Uri("/StatisticsPage.xaml", UriKind.Relative));
        }

    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
```

```
{
    public partial class OrderSettingsPage : PhoneApplicationPage
    {
        public string pageName;
        public int categoryId;
        public int authorId;
        public OrderSettingsPage()
        {
            InitializeComponent();
            SetBackgroundColor();
        }

        private void lstOrderBy_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            int index = lstOrderBy.SelectedIndex;
            string orderType = "";

            if (pageName.Contains("/GeneralSettingsPage.xaml"))
            {
                if (index == 0)
                {
                    orderType = "NAME";
                }
                else if (index == 1)
                {
                    orderType = "BOOKCOUNT";
                }
                else if (index == 2)
                {
                    orderType = "CDATE";
                }
                else if (index == 3)
                {
                    orderType = "MDATE";
                }
                else
                {
                    orderType = "NAME";
                }

                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var appSetting in appSettings)
                    {
                        appSetting.CategoryOrderBy = orderType;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.CategoryOrderTypeChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
            }
            else if (pageName.Contains("/CategorySettingsPage.xaml"))
            {
                if (index == 0)
                {
                    orderType = "NAME";
                }
                else if (index == 1)
                {
                    orderType = "BOOKCOUNT";
                }
                else if (index == 2)
                {
                    orderType = "CDATE";
                }
                else if (index == 3)
                {
                    orderType = "MDATE";
                }
                else
                {
                    orderType = "NAME";
                }

                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var categories = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).ToList() as List<Category>;
                    foreach (var category in categories)
                    {
                        category.AuthorOrderBy = orderType;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.AuthorOrderTypeChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
            }
            else
            {
                if (index == 0)
                {
                    orderType = "NAME";
                }
                else if (index == 1)
                {
                    orderType = "CDATE";
                }
                else if (index == 2)
                {
                    orderType = "MDATE";
                }
                else if (index == 3)
                {
                    orderType = "SDATE";
                }
                else if (index == 4)
                {
                    orderType = "FDATE";
                }
                else if (index == 5)
                {
                    orderType = "RATING";
                }
                else
                {
                    orderType = "NAME";
                }

                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var authors = context.Authors.Where(j => j.AuthorId.Equals(authorId)).ToList() as List<Author>;
                    foreach (var author in authors)
                    {
                        author.BookOrderBy = orderType;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.BookOrderTypeChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor bu bölüm
            var lastPage = NavigationService.BackStack.FirstOrDefault();
            pageName = lastPage.Source.ToString();
            lstOrderBy.Items.Clear();
            if (pageName.Contains("/GeneralSettingsPage.xaml"))
            {
                lblSettings.Text = AppResources.GeneralSettings;
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings =
                        context.AppSettings.First();
                    lblOrderBy.Text = AppResources.CategoryOrderBy;

                    lstOrderBy.Items.Add(AppResources.Name);
                    lstOrderBy.Items.Add(AppResources.BookCount);
                    lstOrderBy.Items.Add(AppResources.CreationDate);
                    lstOrderBy.Items.Add(AppResources.ModificationDate);

                }
            }
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            lstOrderBy.Items.Clear();
            if (pageName.Contains("/CategorySettingsPage.xaml"))
            {
                categoryId = int.Parse(e.Fragment);
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
                    lblSettings.Text = AppResources.CategorySettings + " (" + category.CategoryName + ")" ;
                    lblOrderBy.Text = AppResources.AuthorOrderBy;
                    lstOrderBy.Items.Add(AppResources.Name);
                    lstOrderBy.Items.Add(AppResources.BookCount);
                    lstOrderBy.Items.Add(AppResources.CreationDate);
                    lstOrderBy.Items.Add(AppResources.ModificationDate);
                }
            }
            else
            {
                authorId = int.Parse(e.Fragment);
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
                    lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName + ")" ;
                    lblOrderBy.Text = AppResources.BookOrderBy;
                    lstOrderBy.Items.Add(AppResources.Name);
                    lstOrderBy.Items.Add(AppResources.CreationDate);
                    lstOrderBy.Items.Add(AppResources.ModificationDate);
                    lstOrderBy.Items.Add(AppResources.StartDate);
                    lstOrderBy.Items.Add(AppResources.FinishDate);
                    lstOrderBy.Items.Add(AppResources.BookRating);
                }
            }
            lstOrderBy.SelectedIndex = -1;
            SetBackgroundColor();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, 
System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                if (pageName.Contains("/GeneralSettingsPage.xaml"))
                {
                    this.NavigationService.Navigate(new 
Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
                }
                else if (pageName.Contains("/CategorySettingsPage.xaml"))
                {
                    this.NavigationService.Navigate(new 
Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
                }
                else
                {
                    this.NavigationService.Navigate(new 
Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
                }
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //SetBackgroundColor();
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;


namespace AwesomeLibrary
{
    public partial class OrderStyleSettingsPage : PhoneApplicationPage
    {
        public string pageName;
        public int categoryId;
        public int authorId;

        public OrderStyleSettingsPage()
        {
            InitializeComponent();

            lstOrderStyle.Items.Clear();

            lstOrderStyle.Items.Add(AppResources.Ascending);
            lstOrderStyle.Items.Add(AppResources.Descending);

            lstOrderStyle.SelectedIndex = -1;

            SetBackgroundColor();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                if (pageName.Contains("/GeneralSettingsPage.xaml"))
                {
                    this.NavigationService.Navigate(new
Uri("/GeneralSettingsPage.xaml", UriKind.Relative));
                }
                else if (pageName.Contains("/CategorySettingsPage.xaml"))
                {
                    this.NavigationService.Navigate(new
Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
                }
                else
                {
                    this.NavigationService.Navigate(new
Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
                }
            }
        }

        private void lstOrderStyle_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            int index = lstOrderStyle.SelectedIndex;
            string orderStyle = "";

            if (pageName.Contains("/GeneralSettingsPage.xaml"))
            {
                if (index == 0)
                {
                    orderStyle = "A";
                }
                else
                {
                    orderStyle = "D";
                }

                using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var appSetting in appSettings)
                    {
                        appSetting.CategoryOrderStyle = orderStyle;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.CategoryOrderStyleChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml",
UriKind.Relative));
```

```
            }
            else if (pageName.Contains("/CategorySettingsPage.xaml"))
            {
                if (index == 0)
                {
                    orderStyle = "A";
                }
                else
                {
                    orderStyle = "D";
                }

                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var categories = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).ToList() as List<Category>;
                    foreach (var category in categories)
                    {
                        category.AuthorOrderStyle = orderStyle;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.AuthorOrderStyleChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/CategorySettingsPage.xaml#" + categoryId, UriKind.Relative));
            }
            else
            {
                if (index == 0)
                {
                    orderStyle = "A";
                }
                else
                {
                    orderStyle = "D";
                }

                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var authors = context.Authors.Where(j => j.AuthorId.Equals(authorId)).ToList() as List<Author>;
                    foreach (var author in authors)
                    {
                        author.BookOrderStyle = orderStyle;
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.BookOrderStyleChangeSuccess);
                }
                NavigationService.Navigate(new Uri("/AuthorSettingsPage.xaml#" + authorId, UriKind.Relative));
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            // hangi sayfadan buraya yönlendirme yapilmissa onun adini almaya yariyor bu bölüm
            var lastPage = NavigationService.BackStack.FirstOrDefault();
            pageName = lastPage.Source.ToString();
            if (pageName.Contains("/GeneralSettingsPage.xaml"))
            {
                lblSettings.Text = AppResources.GeneralSettings;
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var appSettings =
```

```csharp
                        context.AppSettings.First();
                    lblOrderStyle.Text = AppResources.CategoryOrderStyle;
                }
            }
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
        {
            // displays "Fragment: Detail"
            //MessageBox.Show("Folder Id: " + e.Fragment);
            base.OnFragmentNavigation(e);
            if (pageName.Contains("/CategorySettingsPage.xaml"))
            {
                categoryId = int.Parse(e.Fragment);
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var category = context.Categories.Where(j => j.CategoryId.Equals(categoryId)).Single() as Category;
                    lblSettings.Text = AppResources.CategorySettings + " (" + category.CategoryName + ")";
                    lblOrderStyle.Text = AppResources.AuthorOrderStyle;
                }
            }
            else
            {
                authorId = int.Parse(e.Fragment);
                using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var author = context.Authors.Where(j => j.AuthorId.Equals(authorId)).Single() as Author;
                    lblSettings.Text = AppResources.AuthorSettings + " (" + author.AuthorName + ")";
                    lblOrderStyle.Text = AppResources.BookOrderStyle;
                }
            }
            SetBackgroundColor();
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //SetBackgroundColor();
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;


```

```
namespace AwesomeLibrary
{
    public partial class PopupAddChange : UserControl
    {
        public PopupAddChange()
        {
            InitializeComponent();
            SetPopupBackgroundColor();
        }
        private void SetPopupBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            switch (appSettings.AppBackgroundColor)
            {
                case "BLA":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                    break;
                case "BLU":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                    break;
                case "BRO":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                    break;
                case "RED":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                    break;
                case "GRE":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                    break;
                case "GRA":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                    break;
                case "YEL":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Yellow);
                    break;
                case "ORA":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Orange);
                    break;
                case "PUR":
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Purple);
                    break;
                default:
                    this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                    break;
            }
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using Microsoft.Phone.Tasks;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using System.Threading;

namespace AwesomeLibrary
{
    public partial class SearchPage : PhoneApplicationPage
    {
        public SearchPage()
        {
            InitializeComponent();
            SetBackgroundColor();

            txtSearchResult.Text = AppResources.SearchResults;
            lblSearch.Text = AppResources.Search;
            //btnSearch.Content = AppResources.Search;
            //lstSearch.SelectedIndex = -1;
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }

            if (appSettings.AppBackgroundImage != null)
            {
                MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                BitmapImage image = new BitmapImage();
                image.SetSource(stream);
                ImageBrush ib = new ImageBrush();
                ib.ImageSource = image;
                this.LayoutRoot.Background = ib;
            }
            else
            {
                switch (appSettings.AppBackgroundColor)
                {
                    case "BLA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                    case "BLU":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                        break;
                    case "BRO":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                        break;
                    case "RED":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                        break;
                    case "GRE":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                        break;
                    case "GRA":
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                        break;
                    case "YEL":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                        break;
                    case "ORA":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                        break;
                    case "PUR":
                        this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                        break;
                    default:
                        this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                        break;
                }
            }
        }

        private void btnSearch_Click(object sender, RoutedEventArgs e)
        {
            if (txtSearch.Text.TrimStart().Length < 1)
            {
                MessageBox.Show(AppResources.SearchTrimFault);
            }
            else
            {
                lstSearch.Items.Clear();
                using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                {
                    var bookList =
                        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>;
                    //var noteList = context.Notes.ToList() as List<Note>;

                    if (bookList != null)
                    {
                        txtSearchResult.Text = AppResources.SearchResults + " (" +
bookList.Count() + ")";
                    }

                    //lstSearch.ItemsSource = noteList;
                    for (int i = 0; i < bookList.Count; i++)
                    {
                        //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).Inde
xOf(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture)) != -1)
                        //{
                        lstSearch.Items.Add(bookList[i] as Book);
                        //}
                    }
                    //lstSearch.ItemTemplate.
                    //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate" + ")";
                    lstSearch.DisplayMemberPath = "BookNameRating";
                    MessageBox.Show(AppResources.SearchCompleted);
                }
            }
        }

        private void lstSearch_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
                if (lstSearch.SelectedIndex != -1)
                {

                    Book selectedBook = lstSearch.SelectedItem as Book;
                    using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        var appSettings = context.AppSettings;
                        var category =
                            context.Categories.Where(j => j.CategoryId.Equals(selectedBook.BookCategoryId)).Single() as
                                Category;

                        var authorBook = context.BookAuthors.Where(j => j.BookId.Equals(selectedBook.BookId)).Single() as BookAuthor;

                        foreach (var item in appSettings)
                        {
                            item.CurrentCategoryNumber = category.CategoryId;
                            item.CurrentAuthorNumber = authorBook.AuthorId;
                        }
                        context.SubmitChanges();

                        NavigationService.Navigate(new Uri("/BookPage.xaml#" + selectedBook.BookId, UriKind.Relative));
                    }
                    lstSearch.SelectedIndex = -1;
                }

            }
            catch (Exception)
            {
                MessageBox.Show(AppResources.SystemFault);
            }
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
        }
        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
            }
        }

        private void txtSearch_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                if (txtSearch.Text.TrimStart().Length < 1)
                {
                    MessageBox.Show(AppResources.SearchTrimFault);
                }
                else
                {
                    lstSearch.Items.Clear();
                    using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
                    {
                        var bookList =
                        context.Books.Where(j =>
j.BookComment.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
||
j.BookName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
||
j.BookPublisherName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
||
j.BookAuthorName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
||
j.BookCategoryName.ToLower(Thread.CurrentThread.CurrentCulture).Contains(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))).ToList()
as List<Book>;

                        //var noteList = context.Notes.ToList() as List<Note>;

                        if (bookList != null)
                        {
                            txtSearchResult.Text = AppResources.SearchResults + " (" +
bookList.Count() + ")";
                        }

                        //lstSearch.ItemsSource = noteList;
                        for (int i = 0; i < bookList.Count; i++)
                        {
                            //if
(noteList[i].NameDescriptionWithoutNewline.ToLower(Thread.CurrentThread.CurrentCulture).IndexOf(txtSearch.Text.ToLower(Thread.CurrentThread.CurrentCulture))
!= -1)
                            //{
                            lstSearch.Items.Add(bookList[i] as Book);
                            //}
                        }
                        //lstSearch.ItemTemplate.
                        //lstSearch.DisplayMemberPath = "NoteName" + " (" +
"CreationDate" + ")";
                        lstSearch.DisplayMemberPath = "BookNameRating";
                        MessageBox.Show(AppResources.SearchCompleted);
                    }
                }
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            txtSearch.Focus();
        }
    }
}
```

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

```
?using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using Microsoft.Phone.Tasks;
using AwesomeLibrary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeLibrary
{
    public partial class StatisticsPage : PhoneApplicationPage
    {
        public StatisticsPage()
        {
            InitializeComponent();
            lblStatistics.Text = AppResources.Statistics;
            SetBackgroundColor();
            SetStatistic();
        }

        private void PhoneApplicationPage_BackKeyPress(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                appSettings = context.AppSettings.First() as AppSettings;
            }
```

```

                if (appSettings.AppBackgroundImage != null)
                {
                    MemoryStream stream = new MemoryStream(appSettings.AppBackgroundImage);
                    BitmapImage image = new BitmapImage();
                    image.SetSource(stream);
                    ImageBrush ib = new ImageBrush();
                    ib.ImageSource = image;
                    this.LayoutRoot.Background = ib;
                }
                else
                {
                    switch (appSettings.AppBackgroundColor)
                    {
                        case "BLA":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                            break;
                        case "BLU":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Blue);
                            break;
                        case "BRO":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Brown);
                            break;
                        case "RED":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Red);
                            break;
                        case "GRE":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Green);
                            break;
                        case "GRA":
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Gray);
                            break;
                        case "YEL":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Yellow);
                            break;
                        case "ORA":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Orange);
                            break;
                        case "PUR":
                            this.LayoutRoot.Background = new
SolidColorBrush(Colors.Purple);
                            break;
                        default:
                            this.LayoutRoot.Background = new SolidColorBrush(Colors.Black);
                            break;
                    }
                }
            }

            protected override void OnNavigatedTo(NavigationEventArgs e)
            {
                base.OnNavigatedTo(e);
            }

            protected override void OnNavigatedFrom(NavigationEventArgs e)
            {
                base.OnNavigatedFrom(e);
                //while (NavigationService.CanGoBack)
                //NavigationService.RemoveBackEntry();

            }

            private void SetStatistic()
            {
                StringBuilder sb = new StringBuilder();
                string authorName = "", categoryName = "", bookName, publisherName = "",
bestBook = "", worstBook = "";
```

```
            double pagePerDay, bookPerDay;
            int bookCount;
            int pageSum = 0;
            DateTime beginDate, endDate;
            using (var context = new AwesomeLibraryDataContext(AwesomeLibraryDataContext.ConnectionString))
            {
                bookCount = context.Books.Count();

                var categories = context.Categories.OrderByDescending(j => j.CategoryBookCount).ToList() as List<Category>;
                categoryName = categories.First().CategoryNameCount;

                var authors = context.Authors.OrderByDescending(j => j.AuthorBookCount).ToList() as List<Author>;
                authorName = authors.First().AuthorNameCount;

                var books = context.Books.OrderBy(j => j.ReadStartDate).ToList() as List<Book>;
                beginDate = books.First().ReadStartDate;
                foreach (var item in books)
                {
                    pageSum = pageSum + item.BookPageNumber;
                }

                // bugünün tarihi ile ilgili olarak son tarih düzenleniyor
                endDate = DateTime.Now;

                var books3 = context.Books.GroupBy(j => j.BookPublisherName).Select(j => new { Name = j.Key, Total = j.Count()}).OrderByDescending(k => k.Total);
                publisherName = books3.First().Name + " (" + books3.First().Total + ")";

                int diffDays = (endDate.Date - beginDate.Date).Days;

                pagePerDay = pageSum / Convert.ToDouble(diffDays);
                bookPerDay = bookCount / Convert.ToDouble(diffDays);

                var books4 = context.Books.OrderByDescending(j => j.BookRating).ToList() as List<Book>;
                int bestBookRating = books4.First().BookRating;
                int worstBookRating = books4.Last().BookRating;
                // en iyi ve en kötü puana sahip birden fazla albüm varsa hepsini listelemeye yarayan
                // kod parçasi buradadir.
                var bestBooks = context.Books.Where(j => j.BookRating.Equals(bestBookRating)).ToList() as List<Book>;
                if (bestBooks.Count < 2)
                {
                    bestBook = books4.First().BookNameRating;

                }
                else
                {
                    for (int i = 0; i < bestBooks.Count; i++)
                    {
                        bestBook = bestBook + bestBooks[i].BookNameRating + ", ";
                    }
                    bestBook = bestBook.Substring(0, bestBook.Length - 2);
                }

                var worstBooks = context.Books.Where(j => j.BookRating.Equals(worstBookRating)).ToList() as List<Book>;
                if (worstBooks.Count < 2)
                {
                    worstBook = books4.Last().BookNameRating;

                }
                else
                {
                    for (int i = 0; i < worstBooks.Count; i++)
                    {
                        worstBook = worstBook + worstBooks[i].BookNameRating + ", ";
                    }
                    worstBook = worstBook.Substring(0, worstBook.Length - 2);
                }

            }

            sb.AppendLine(beginDate.ToShortDateString() + " - " + endDate.ToShortDateString());
            sb.AppendLine();
            sb.AppendLine(AppResources.TotalBookCount + ": " + bookCount);
            sb.AppendLine(AppResources.MostReadCategory + ": " + categoryName);
            sb.AppendLine(AppResources.MostReadAuthor + ": " + authorName);
            sb.AppendLine(AppResources.MostReadPublisher + ": " + publisherName);
            sb.AppendLine(AppResources.BookPerDay + ": " + String.Format("{0:0.00}", bookPerDay) );
            sb.AppendLine(AppResources.PagePerDay + ": " + String.Format("{0:0.00}", pagePerDay) );
            sb.AppendLine(AppResources.BestBook + ": " + bestBook);
            sb.AppendLine(AppResources.WorstBook + ": " + worstBook);

            txtStatistics.Text = sb.ToString();
            txtStatistics.IsReadOnly = true;
        }
    }
}
```

Namespaces

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

Index