Awesome Dictionary

Bir veya birden çok sözlügü dahil edip kelimelerin anlamlarini bulmaya yarayan bir uygulama

Table of Contents

1 Symbol Reference

1.1 AwesomeDictionary Namespace

This is namespace AwesomeDictionary.

Namespaces

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

Classes

	Name	Description
	AboutPage (see page 17)	This is class AwesomeDictionary.AboutPage.
	All (see page 18)	This is class AwesomeDictionary.All.
	AlmancaTurkce (see page 19)	This is class AwesomeDictionary.AlmancaTurkce.
	AlphaKeyGroup (see page 21)	This is class AwesomeDictionary.AlphaKeyGroup.
	App (see page 23)	This is class AwesomeDictionary.App.
	AppSettings (see page 25)	This is class AwesomeDictionary.AppSettings.
	AwesomeDictionaryDataContext (see page 26)	This is class AwesomeDictionary.AwesomeDictionaryDataContext.
	BackgroundColorSettingsPage (see page 29)	This is class AwesomeDictionary.BackgroundColorSettingsPage.
	BilisimSozlugu (see page 31)	This is class AwesomeDictionary.BilisimSozlugu.
	BuyukLugat (see page 33)	This is class AwesomeDictionary.BuyukLugat.
	EnglishTurkishVol1 (see page 34)	This is class AwesomeDictionary.EnglishTurkishVol1.
	EnglishTurkishVol2 (see page 35)	This is class AwesomeDictionary.EnglishTurkishVol2.
	Favourite (see page 36)	This is class AwesomeDictionary.Favourite.
	FontFamilySettingsPage (see page 37)	This is class AwesomeDictionary.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 39)	This is class AwesomeDictionary.FontSizeSettingsPage.
	GeneralSettingsPage (see page 41)	This is class AwesomeDictionary.GeneralSettingsPage.
	KelimeAnlamlari (see page 46)	This is class AwesomeDictionary.KelimeAnlamlari.
	LanguageSettingsPage (see page 47)	This is class AwesomeDictionary.LanguageSettingsPage.
	LocalizedStrings (see page 49)	Provides access to string resources.
	MainPage (see page 49)	This is class AwesomeDictionary.MainPage.
	NameDetailPage (see page 50)	This is class AwesomeDictionary.NameDetailPage.
	OxfordEnglishEnglish (see page 54)	This is class AwesomeDictionary.OxfordEnglishEnglish.
	RisaleNur (see page 56)	This is class AwesomeDictionary.RisaleNur.
	SearchPage (see page 57)	This is class AwesomeDictionary.SearchPage.
	StatisticsPage (see page 58)	This is class AwesomeDictionary.StatisticsPage.

1.1.1 AwesomeDictionary.Resources Namespace

This is namespace AwesomeDictionary.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.

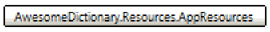
Classes

	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 73)

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

AppResources Properties

	Name	Description
	About (see page 6)	Looks up a localized string similar to About.
	AboutAwesomeDictionary (see page 6)	Looks up a localized string similar to About (see page 6) Awesome Dictionary.
	AboutTheApp (see page 6)	Looks up a localized string similar to About (see page 6) The App (see page 23).

	AboutTheAppText (see page 6)	Looks up a localized string similar to Hi everybody. I am with you with a new app. I investigate a lot of dictionary applications in Windows Phone and i tried to make a simple app which you will like it. If you rate the app and send your thoughts to coderserdar@outlook.com, I will be so appreciated to you. With my best regards. CoderSerdar.
	AddToFavourites (see page 6)	Looks up a localized string similar to Add To Favourites.
	AtLeastOneDictionary (see page 6)	Looks up a localized string similar to You Should Select (see page 13) At Least One Dictionary.
	Background (see page 6)	Looks up a localized string similar to Background.
	BackgroundColor (see page 6)	Looks up a localized string similar to Background (see page 6) Color.
	BackgroundColorChangedSuccessfully (see page 6)	Looks up a localized string similar to Background (see page 6) Color Has Been Changed Successfully.
	BackgroundImage (see page 7)	Looks up a localized string similar to Background (see page 6) Image.
	BackgroundImageChangedSuccessfully (see page 7)	Looks up a localized string similar to Background (see page 6) Image Has Been Changed Successfully.
	BackgroundImageRemovedSuccessfully (see page 7)	Looks up a localized string similar to Background (see page 6) Image Has Been Removed Successfully.
	BackgroundSettingsResetSuccessfully (see page 7)	Looks up a localized string similar to Background (see page 6) Settings (see page 14) Has Been Reset Successfully.
	Black (see page 7)	Looks up a localized string similar to Black.
	Blue (see page 7)	Looks up a localized string similar to Blue.
	Brown (see page 7)	Looks up a localized string similar to Brown.
	BuyukLugat (see page 7)	Looks up a localized string similar to Buyuk Lugat (Turkish (see page 15)->Turkish (see page 15)).
	Cancel (see page 7)	Looks up a localized string similar to Cancel.
	ComputerDictionary (see page 8)	Looks up a localized string similar to Computer Dictionary (English (see page 8)->Turkish (see page 15)).
	ContactWithUs (see page 8)	Looks up a localized string similar to Contact With Us.
	Culture (see page 8)	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
	DictionariesInstalledSuccessfully (see page 8)	Looks up a localized string similar to Dictionaries Have Been Installed Successfully.
	DictionariesUninstalledSuccessfully (see page 8)	Looks up a localized string similar to Dictionaries Have Been Uninstalled Successfully.
	DictionaryInstall (see page 8)	Looks up a localized string similar to Dictionary Install (see page 10).
	English (see page 8)	Looks up a localized string similar to English.
	EnglishTurkishVol1 (see page 8)	Looks up a localized string similar to English (see page 8)->Turkish (see page 15) Dictionary Vol. 1.
	EnglishTurkishVol2 (see page 9)	Looks up a localized string similar to English (see page 8)->Turkish (see page 15) Dictionary Vol. 2.
	ExitApp (see page 9)	Looks up a localized string similar to Exit App (see page 23).
	ExitAppQuestion (see page 9)	Looks up a localized string similar to Are You Sure To Exit The Application?.
	FavouriteNameCount (see page 9)	Looks up a localized string similar to Favourite (see page 36) Word (see page 15) Count.
	Font (see page 9)	Looks up a localized string similar to Font.
	FontFamily (see page 9)	Looks up a localized string similar to Font (see page 9) Family.
	FontFamilyChangedSuccessfully (see page 9)	Looks up a localized string similar to Font (see page 9) Family Has Been Changed Successfully.

	FontSize (see page 9)	Looks up a localized string similar to Font (see page 9) Size.
	FontSizeChangedSuccessfully (see page 9)	Looks up a localized string similar to Font (see page 9) Size Has Been Changed Successfully.
	GeneralSettings (see page 10)	Looks up a localized string similar to General Settings (see page 14).
	German (see page 10)	Looks up a localized string similar to German.
	GermanTurkish (see page 10)	Looks up a localized string similar to German (see page 10)->Turkish (see page 15) Dictionary.
	Gray (see page 10)	Looks up a localized string similar to Gray.
	Green (see page 10)	Looks up a localized string similar to Green.
	Install (see page 10)	Looks up a localized string similar to Install.
	Installing (see page 10)	Looks up a localized string similar to Installing.
	Language (see page 10)	Looks up a localized string similar to Language.
	LanguageWarning (see page 10)	Looks up a localized string similar to You may restart the application for changes will be effect..
	Meaning (see page 11)	Looks up a localized string similar to Meaning.
	MyFavourites (see page 11)	Looks up a localized string similar to My Favourites.
	Ok (see page 11)	Looks up a localized string similar to Ok.
	Orange (see page 11)	Looks up a localized string similar to Orange.
	OxfordDictionary (see page 11)	Looks up a localized string similar to Oxford (English (see page 8)->English (see page 8)).
	Purple (see page 11)	Looks up a localized string similar to Purple.
	RandomWords (see page 11)	Looks up a localized string similar to Random 10 Words.
	Rate (see page 11)	Looks up a localized string similar to Rate.
	Red (see page 11)	Looks up a localized string similar to Red.
	RemoveBackgroundImage (see page 12)	Looks up a localized string similar to Remove Background (see page 6) Image.
	RemoveFromFavourite (see page 12)	Looks up a localized string similar to Remove From Favourites.
	RemoveFromFavouriteQuestion (see page 12)	Looks up a localized string similar to You Will Remove The Word (see page 15) From Favourites. Are You Sure?.
	RemoveFromFavourites (see page 12)	Looks up a localized string similar to Remove From Favourites.
	ResetSettings (see page 12)	Looks up a localized string similar to Reset Settings (see page 14).
	ResourceFlowDirection (see page 12)	Looks up a localized string similar to LeftToRight.
	ResourceLanguage (see page 12)	Looks up a localized string similar to en-US.
	ResourceManager (see page 12)	Returns the cached ResourceManager instance used by this class.
	RisaleNur (see page 12)	Looks up a localized string similar to Risale Nur (Turkish (see page 15)->Turkish (see page 15)).
	Search (see page 13)	Looks up a localized string similar to Search.
	SearchCompleted (see page 13)	Looks up a localized string similar to Search (see page 13) Completed.
	SearchInMeanings (see page 13)	Looks up a localized string similar to Search (see page 13) In Meanings.
	SearchResults (see page 13)	Looks up a localized string similar to Search (see page 13) Results.
	SearchTrimFault (see page 13)	Looks up a localized string similar to Search (see page 13) Criteria Can Not Be Empty.
	Select (see page 13)	Looks up a localized string similar to Select.
	SelectBackgroundColor (see page 13)	Looks up a localized string similar to Select (see page 13) Background (see page 6) Color.

	Selected (see page 13)	Looks up a localized string similar to Selected.
	SelectFontFamily (see page 13)	Looks up a localized string similar to Select (see page 13) Font (see page 9) Family.
	SelectFontSize (see page 14)	Looks up a localized string similar to Select (see page 13) Font (see page 9) Size.
	SelectLanguage (see page 14)	Looks up a localized string similar to Select (see page 13) Language (see page 10).
	SendWithAwesomeDictionaryApp (see page 14)	Looks up a localized string similar to Send With Awesome Dictionary App (see page 23).
	SendWithEmail (see page 14)	Looks up a localized string similar to Send With E-Mail.
	SendWithSMS (see page 14)	Looks up a localized string similar to Send With SMS.
	Settings (see page 14)	Looks up a localized string similar to Settings.
	Share (see page 14)	Looks up a localized string similar to Share.
	Source (see page 14)	Looks up a localized string similar to Source.
	Statistics (see page 14)	Looks up a localized string similar to Statistics.
	Synchronizing (see page 15)	Looks up a localized string similar to Synchronizing.
	SystemFault (see page 15)	Looks up a localized string similar to System Has A Fault. Please Try Again Later..
	TotalNameCount (see page 15)	Looks up a localized string similar to Total Word (see page 15) Count.
	Turkish (see page 15)	Looks up a localized string similar to Turkish.
	Uninstall (see page 15)	Looks up a localized string similar to Uninstall.
	Word (see page 15)	Looks up a localized string similar to Word.
	WordAddedFavouriteSuccessfully (see page 15)	Looks up a localized string similar to Word (see page 15) Has Been Added To Favourites Successfully.
	WordAlreadyFavourite (see page 15)	Looks up a localized string similar to Word (see page 15) Is Already Favourite (see page 36).
	WordAndMeaning (see page 15)	Looks up a localized string similar to Word (see page 15) And Meaning (see page 11).
	WordMeaning (see page 16)	Looks up a localized string similar to Word (see page 15) Meaning (see page 11) Dictionary.
	WordRemovedFavouriteSuccessfully (see page 16)	Looks up a localized string similar to Word (see page 15) Has Been Removed From Favourites Successfully.
	Yellow (see page 16)	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.AboutAwesomeDictionary Property

Looks up a localized string similar to About (see page 6) Awesome Dictionary.

C#

```
public static string AboutAwesomeDictionary;
```

1.1.1.1.1.2.3 AppResources.AboutTheApp Property

Looks up a localized string similar to About (see page 6) The App (see page 23).

C#

```
public static string AboutTheApp;
```

1.1.1.1.1.2.4 AppResources.AboutTheAppText Property

Looks up a localized string similar to Hi everybody. I am with you with a new app. I investigate a lot of dictionary applications in Windows Phone and i tried to make a simple app which you will like it. If you rate the app and send your thoughts to coderserdar@outlook.com, I will be so appreciated to you. With my best regards. CoderSerdar.

C#

```
public static string AboutTheAppText;
```

1.1.1.1.1.2.5 AppResources.AddToFavourites Property

Looks up a localized string similar to Add To Favourites.

C#

```
public static string AddToFavourites;
```

1.1.1.1.1.2.6 AppResources.AtLeastOneDictionary Property

Looks up a localized string similar to You Should Select (see page 13) At Least One Dictionary.

C#

```
public static string AtLeastOneDictionary;
```

1.1.1.1.1.2.7 AppResources.Background Property

Looks up a localized string similar to Background.

C#

```
public static string Background;
```

1.1.1.1.1.2.8 AppResources.BackgroundColor Property

Looks up a localized string similar to Background (see page 6) Color.

C#

```
public static string BackgroundColor;
```

1.1.1.1.1.2.9 AppResources.BackgroundColorChangedSuccessfully Property

Looks up a localized string similar to Background (see page 6) Color Has Been Changed Successfully.

C#

```
public static string BackgroundColorChangedSuccessfully;
```

1.1.1.1.1.2.10 AppResources.BackgroundImage Property

Looks up a localized string similar to Background (see page 6) Image.

C#

```
public static string BackgroundImage;
```

1.1.1.1.1.2.11 AppResources.BackgroundImageChangedSuccessfully Property

Looks up a localized string similar to Background (see page 6) Image Has Been Changed Successfully.

C#

```
public static string BackgroundImageChangedSuccessfully;
```

1.1.1.1.1.2.12 AppResources.BackgroundImageRemovedSuccessfully Property

Looks up a localized string similar to Background (see page 6) Image Has Been Removed Successfully.

C#

```
public static string BackgroundImageRemovedSuccessfully;
```

1.1.1.1.1.2.13 AppResources.BackgroundSettingsResetSuccessfully Property

Looks up a localized string similar to Background (see page 6) Settings (see page 14) Has Been Reset Successfully.

C#

```
public static string BackgroundSettingsResetSuccessfully;
```

1.1.1.1.1.2.14 AppResources.Black Property

Looks up a localized string similar to Black.

C#

```
public static string Black;
```

1.1.1.1.1.2.15 AppResources.Blue Property

Looks up a localized string similar to Blue.

C#

```
public static string Blue;
```

1.1.1.1.1.2.16 AppResources.Brown Property

Looks up a localized string similar to Brown.

C#

```
public static string Brown;
```

1.1.1.1.1.2.17 AppResources.BuyukLugat Property

Looks up a localized string similar to Buyuk Lugat (Turkish (see page 15)->Turkish (see page 15)).

C#

```
public static string BuyukLugat;
```

1.1.1.1.1.2.18 AppResources.Cancel Property

Looks up a localized string similar to Cancel.

C#

```
public static string Cancel;
```

1.1.1.1.1.2.19 AppResources.ComputerDictionary Property

Looks up a localized string similar to Computer Dictionary (English (see page 8)->Turkish (see page 15)).

C#

```
public static string ComputerDictionary;
```

1.1.1.1.1.2.20 AppResources.ContactWithUs Property

Looks up a localized string similar to Contact With Us.

C#

```
public static string ContactWithUs;
```

1.1.1.1.1.2.21 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;
```

1.1.1.1.1.2.22 AppResources.DictionariesInstalledSuccessfully Property

Looks up a localized string similar to Dictionaries Have Been Installed Successfully.

C#

```
public static string DictionariesInstalledSuccessfully;
```

1.1.1.1.1.2.23 AppResources.DictionariesUninstalledSuccessfully Property

Looks up a localized string similar to Dictionaries Have Been Uninstalled Successfully.

C#

```
public static string DictionariesUninstalledSuccessfully;
```

1.1.1.1.1.2.24 AppResources.DictionaryInstall Property

Looks up a localized string similar to Dictionary Install (see page 10).

C#

```
public static string DictionaryInstall;
```

1.1.1.1.1.2.25 AppResources.English Property

Looks up a localized string similar to English.

C#

```
public static string English;
```

1.1.1.1.1.2.26 AppResources.EnglishTurkishVol1 Property

Looks up a localized string similar to English (see page 8)->Turkish (see page 15) Dictionary Vol. 1.

C#

```
public static string EnglishTurkishVol1;
```

1.1.1.1.1.2.27 AppResources.EnglishTurkishVol2 Property

Looks up a localized string similar to English (see page 8)->Turkish (see page 15) Dictionary Vol. 2.

C#

```
public static string EnglishTurkishVol2;
```

1.1.1.1.1.2.28 AppResources.ExitApp Property

Looks up a localized string similar to Exit App (see page 23).

C#

```
public static string ExitApp;
```

1.1.1.1.1.2.29 AppResources.ExitAppQuestion Property

Looks up a localized string similar to Are You Sure To Exit The Application?.

C#

```
public static string ExitAppQuestion;
```

1.1.1.1.1.2.30 AppResources.FavouriteNameCount Property

Looks up a localized string similar to Favourite (see page 36) Word (see page 15) Count.

C#

```
public static string FavouriteNameCount;
```

1.1.1.1.1.2.31 AppResources.Font Property

Looks up a localized string similar to Font.

C#

```
public static string Font;
```

1.1.1.1.1.2.32 AppResources.FontFamily Property

Looks up a localized string similar to Font (see page 9) Family.

C#

```
public static string FontFamily;
```

1.1.1.1.1.2.33 AppResources.FontFamilyChangedSuccessfully Property

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

C#

```
public static string FontFamilyChangedSuccessfully;
```

1.1.1.1.1.2.34 AppResources.FontSize Property

Looks up a localized string similar to Font (see page 9) Size.

C#

```
public static string FontSize;
```

1.1.1.1.1.2.35 AppResources.FontSizeChangedSuccessfully Property

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

C#

```
public static string FontSizeChangedSuccessfully;
```

1.1.1.1.1.2.36 AppResources.GeneralSettings Property

Looks up a localized string similar to General Settings (see page 14).

C#

```
public static string GeneralSettings;
```

1.1.1.1.1.2.37 AppResources.German Property

Looks up a localized string similar to German.

C#

```
public static string German;
```

1.1.1.1.1.2.38 AppResources.GermanTurkish Property

Looks up a localized string similar to German (see page 10)->Turkish (see page 15) Dictionary.

C#

```
public static string GermanTurkish;
```

1.1.1.1.1.2.39 AppResources.Gray Property

Looks up a localized string similar to Gray.

C#

```
public static string Gray;
```

1.1.1.1.1.2.40 AppResources.Green Property

Looks up a localized string similar to Green.

C#

```
public static string Green;
```

1.1.1.1.1.2.41 AppResources.Install Property

Looks up a localized string similar to Install.

C#

```
public static string Install;
```

1.1.1.1.1.2.42 AppResources.Installing Property

Looks up a localized string similar to Installing.

C#

```
public static string Installing;
```

1.1.1.1.1.2.43 AppResources.Language Property

Looks up a localized string similar to Language.

C#

```
public static string Language;
```

1.1.1.1.1.2.44 AppResources.LanguageWarning Property

Looks up a localized string similar to You may restart the application for changes will be effect..

C#

```
public static string LanguageWarning;
```

1.1.1.1.1.2.45 AppResources.Meaning Property

Looks up a localized string similar to Meaning.

C#

```
public static string Meaning;
```

1.1.1.1.1.2.46 AppResources.MyFavourites Property

Looks up a localized string similar to My Favourites.

C#

```
public static string MyFavourites;
```

1.1.1.1.1.2.47 AppResources.Ok Property

Looks up a localized string similar to Ok.

C#

```
public static string Ok;
```

1.1.1.1.1.2.48 AppResources.Orange Property

Looks up a localized string similar to Orange.

C#

```
public static string Orange;
```

1.1.1.1.1.2.49 AppResources.OxfordDictionary Property

Looks up a localized string similar to Oxford (English (see page 8)->English (see page 8)).

C#

```
public static string OxfordDictionary;
```

1.1.1.1.1.2.50 AppResources.Purple Property

Looks up a localized string similar to Purple.

C#

```
public static string Purple;
```

1.1.1.1.1.2.51 AppResources.RandomWords Property

Looks up a localized string similar to Random 10 Words.

C#

```
public static string RandomWords;
```

1.1.1.1.1.2.52 AppResources.Rate Property

Looks up a localized string similar to Rate.

C#

```
public static string Rate;
```

1.1.1.1.1.2.53 AppResources.Red Property

Looks up a localized string similar to Red.

C#

```
public static string Red;
```

1.1.1.1.1.2.54 AppResources.RemoveBackgroundImage Property

Looks up a localized string similar to Remove Background (see page 6) Image.

C#

```
public static string RemoveBackgroundImage;
```

1.1.1.1.1.2.55 AppResources.RemoveFromFavourite Property

Looks up a localized string similar to Remove From Favourites.

C#

```
public static string RemoveFromFavourite;
```

1.1.1.1.1.2.56 AppResources.RemoveFromFavouriteQuestion Property

Looks up a localized string similar to You Will Remove The Word (see page 15) From Favourites. Are You Sure?.

C#

```
public static string RemoveFromFavouriteQuestion;
```

1.1.1.1.1.2.57 AppResources.RemoveFromFavourites Property

Looks up a localized string similar to Remove From Favourites.

C#

```
public static string RemoveFromFavourites;
```

1.1.1.1.1.2.58 AppResources.ResetSettings Property

Looks up a localized string similar to Reset Settings (see page 14).

C#

```
public static string ResetSettings;
```

1.1.1.1.1.2.59 AppResources.ResourceFlowDirection Property

Looks up a localized string similar to LeftToRight.

C#

```
public static string ResourceFlowDirection;
```

1.1.1.1.1.2.60 AppResources.ResourceLanguage Property

Looks up a localized string similar to en-US.

C#

```
public static string ResourceLanguage;
```

1.1.1.1.1.2.61 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;
```

1.1.1.1.1.2.62 AppResources.RisaleNur Property

Looks up a localized string similar to Risale Nur (Turkish (see page 15)->Turkish (see page 15)).

C#

```
public static string RisaleNur;
```

1.1.1.1.1.2.63 AppResources.Search Property

Looks up a localized string similar to Search.

C#

```
public static string Search;
```

1.1.1.1.1.2.64 AppResources.SearchCompleted Property

Looks up a localized string similar to Search (see page 13) Completed.

C#

```
public static string SearchCompleted;
```

1.1.1.1.1.2.65 AppResources.SearchInMeanings Property

Looks up a localized string similar to Search (see page 13) In Meanings.

C#

```
public static string SearchInMeanings;
```

1.1.1.1.1.2.66 AppResources.SearchResults Property

Looks up a localized string similar to Search (see page 13) Results.

C#

```
public static string SearchResults;
```

1.1.1.1.1.2.67 AppResources.SearchTrimFault Property

Looks up a localized string similar to Search (see page 13) Criteria Can Not Be Empty.

C#

```
public static string SearchTrimFault;
```

1.1.1.1.1.2.68 AppResources.Select Property

Looks up a localized string similar to Select.

C#

```
public static string Select;
```

1.1.1.1.1.2.69 AppResources.SelectBackgroundColor Property

Looks up a localized string similar to Select (see page 13) Background (see page 6) Color.

C#

```
public static string SelectBackgroundColor;
```

1.1.1.1.1.2.70 AppResources.Selected Property

Looks up a localized string similar to Selected.

C#

```
public static string Selected;
```

1.1.1.1.1.2.71 AppResources.SelectFontFamily Property

Looks up a localized string similar to Select (see page 13) Font (see page 9) Family.

C#

```
public static string SelectFontFamily;
```

1.1.1.1.1.2.72 AppResources.SelectFontSize Property

Looks up a localized string similar to Select (see page 13) Font (see page 9) Size.

C#

```
public static string SelectFontSize;
```

1.1.1.1.1.2.73 AppResources.SelectLanguage Property

Looks up a localized string similar to Select (see page 13) Language (see page 10).

C#

```
public static string SelectLanguage;
```

1.1.1.1.1.2.74 AppResources.SendWithAwesomeDictionaryApp Property

Looks up a localized string similar to Send With Awesome Dictionary App (see page 23).

C#

```
public static string SendWithAwesomeDictionaryApp;
```

1.1.1.1.1.2.75 AppResources.SendWithEmail Property

Looks up a localized string similar to Send With E-Mail.

C#

```
public static string SendWithEmail;
```

1.1.1.1.1.2.76 AppResources.SendWithSMS Property

Looks up a localized string similar to Send With SMS.

C#

```
public static string SendWithSMS;
```

1.1.1.1.1.2.77 AppResources.Settings Property

Looks up a localized string similar to Settings.

C#

```
public static string Settings;
```

1.1.1.1.1.2.78 AppResources.Share Property

Looks up a localized string similar to Share.

C#

```
public static string Share;
```

1.1.1.1.1.2.79 AppResources.Source Property

Looks up a localized string similar to Source.

C#

```
public static string Source;
```

1.1.1.1.1.2.80 AppResources.Statistics Property

Looks up a localized string similar to Statistics.

C#

```
public static string Statistics;
```

1.1.1.1.1.2.81 AppResources.Synchronizing Property

Looks up a localized string similar to Synchronizing.

C#

```
public static string Synchronizing;
```

1.1.1.1.1.2.82 AppResources.SystemFault Property

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

C#

```
public static string SystemFault;
```

1.1.1.1.1.2.83 AppResources.TotalNameCount Property

Looks up a localized string similar to Total Word (see page 15) Count.

C#

```
public static string TotalNameCount;
```

1.1.1.1.1.2.84 AppResources.Turkish Property

Looks up a localized string similar to Turkish.

C#

```
public static string Turkish;
```

1.1.1.1.1.2.85 AppResources.Uninstall Property

Looks up a localized string similar to Uninstall.

C#

```
public static string Uninstall;
```

1.1.1.1.1.2.86 AppResources.Word Property

Looks up a localized string similar to Word.

C#

```
public static string Word;
```

1.1.1.1.1.2.87 AppResources.WordAddedFavouriteSuccessfully Property

Looks up a localized string similar to Word (see page 15) Has Been Added To Favourites Successfully.

C#

```
public static string WordAddedFavouriteSuccessfully;
```

1.1.1.1.1.2.88 AppResources.WordAlreadyFavourite Property

Looks up a localized string similar to Word (see page 15) Is Already Favourite (see page 36).

C#

```
public static string WordAlreadyFavourite;
```

1.1.1.1.1.2.89 AppResources.WordAndMeaning Property

Looks up a localized string similar to Word (see page 15) And Meaning (see page 11).

C#

```
public static string WordAndMeaning;
```

1.1.1.1.1.2.90 AppResources.WordMeaning Property

Looks up a localized string similar to Word (see page 15) Meaning (see page 11) Dictionary.

C#

```
public static string WordMeaning;
```

1.1.1.1.1.2.91 AppResources.WordRemovedFavouriteSuccessfully Property

Looks up a localized string similar to Word (see page 15) Has Been Removed From Favourites Successfully.

C#

```
public static string WordRemovedFavouriteSuccessfully;
```

1.1.1.1.1.2.92 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 17)	This is class AwesomeDictionary.AboutPage.
	All (see page 18)	This is class AwesomeDictionary.All.
	AlmancaTurkce (see page 19)	This is class AwesomeDictionary.AlmancaTurkce.
	AlphaKeyGroup (see page 21)	This is class AwesomeDictionary.AlphaKeyGroup.
	App (see page 23)	This is class AwesomeDictionary.App.
	AppSettings (see page 25)	This is class AwesomeDictionary.AppSettings.
	AwesomeDictionaryDataContext (see page 26)	This is class AwesomeDictionary.AwesomeDictionaryDataContext.
	BackgroundColorSettingsPage (see page 29)	This is class AwesomeDictionary.BackgroundColorSettingsPage.
	BilisimSozlugu (see page 31)	This is class AwesomeDictionary.BilisimSozlugu.
	BuyukLugat (see page 33)	This is class AwesomeDictionary.BuyukLugat.
	EnglishTurkishVol1 (see page 34)	This is class AwesomeDictionary.EnglishTurkishVol1.
	EnglishTurkishVol2 (see page 35)	This is class AwesomeDictionary.EnglishTurkishVol2.
	Favourite (see page 36)	This is class AwesomeDictionary.Favourite.
	FontFamilySettingsPage (see page 37)	This is class AwesomeDictionary.FontFamilySettingsPage.
	FontSizeSettingsPage (see page 39)	This is class AwesomeDictionary.FontSizeSettingsPage.
	GeneralSettingsPage (see page 41)	This is class AwesomeDictionary.GeneralSettingsPage.
	KelimeAnlamlari (see page 46)	This is class AwesomeDictionary.KelimeAnlamlari.

	LanguageSettingsPage (see page 47)	This is class AwesomeDictionary.LanguageSettingsPage.
	LocalizedStrings (see page 49)	Provides access to string resources.
	MainPage (see page 49)	This is class AwesomeDictionary.MainPage.
	NameDetailPage (see page 50)	This is class AwesomeDictionary.NameDetailPage.
	OxfordEnglishEnglish (see page 54)	This is class AwesomeDictionary.OxfordEnglishEnglish.
	RisaleNur (see page 56)	This is class AwesomeDictionary.RisaleNur.
	SearchPage (see page 57)	This is class AwesomeDictionary.SearchPage.
	StatisticsPage (see page 58)	This is class AwesomeDictionary.StatisticsPage.

1.1.2.1 AboutPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.AboutPage

C#

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

File

AboutPage.xaml.cs (see page 60)

Description

This is class AwesomeDictionary.AboutPage.

Methods

	Name	Description
	AboutPage (see page 17)	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;
    txtAbout.Text = AppResources.AboutTheAppText;
    txtAbout.IsEnabled = false;
    //var paragraph = new Paragraph();
    //paragraph.Inlines.Add(AppResources.AboutTheAppText);
    //txtAbout.Blocks.Add(paragraph);
}
```

1.1.2.2 All Class

Class Hierarchy

AwesomeDictionary.All

C#

```
[Index(Columns = "AllName, AllMeaning, AllNameMeaning, AllNameSource ASC", IsUnique =
false, Name = "indAllNames")]
[Table]
public class All;
```

File

All.cs (see page 63)

Description

This is class AwesomeDictionary.All.

All Properties

	Name	Description
▣	AllId (see page 18)	This is AllId, a member of class All.
▣	AllMeaning (see page 18)	This is AllMeaning, a member of class All.
▣	AllName (see page 19)	This is AllName, a member of class All.
▣	AllNameMeaning (see page 19)	This is AllNameMeaning, a member of class All.
▣	AllNameSource (see page 19)	This is AllNameSource, a member of class All.
▣	AllSource (see page 19)	This is AllSource, a member of class All.

1.1.2.2.1 All Properties

1.1.2.2.1.1 All.AllId Property

C#

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

Description

This is AllId, a member of class All.

1.1.2.2.1.2 All.AllMeaning Property

C#

```
[Column]
public string AllMeaning;
```

Description

This is AllMeaning, a member of class All.

1.1.2.2.1.3 All.AllName Property

C#

```
[Column]
public string AllName;
```

Description

This is AllName, a member of class All.

1.1.2.2.1.4 All.AllNameMeaning Property

C#

```
[Column]
public string AllNameMeaning;
```

Description

This is AllNameMeaning, a member of class All.

1.1.2.2.1.5 All.AllNameSource Property

C#

```
[Column]
public string AllNameSource;
```

Description

This is AllNameSource, a member of class All.

1.1.2.2.1.6 All.AllSource Property

C#

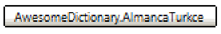
```
[Column]
public string AllSource;
```

Description

This is AllSource, a member of class All.

1.1.2.3 AlmancaTurkce Class

Class Hierarchy



C#

```
[Table]
public class AlmancaTurkce;
```

File

AlmancaTurkce.cs (see page 64)

Description

This is class AwesomeDictionary.AlmancaTurkce.

AlmancaTurkce Properties

	Name	Description
	AlmancaTurkceId (see page 20)	This is AlmancaTurkceId, a member of class AlmancaTurkce.
	AlmancaTurkceMeaning (see page 20)	This is AlmancaTurkceMeaning, a member of class AlmancaTurkce.
	AlmancaTurkceName (see page 20)	This is AlmancaTurkceName, a member of class AlmancaTurkce.
	AlmancaTurkceNameMeaning (see page 20)	This is AlmancaTurkceNameMeaning, a member of class AlmancaTurkce.

1.1.2.3.1 AlmancaTurkce Properties

1.1.2.3.1.1 AlmancaTurkce.AlmancaTurkceId Property

C#

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

Description

This is AlmancaTurkceId, a member of class AlmancaTurkce.

1.1.2.3.1.2 AlmancaTurkce.AlmancaTurkceMeaning Property

C#

```
[Column]
public string AlmancaTurkceMeaning;
```

Description

This is AlmancaTurkceMeaning, a member of class AlmancaTurkce.

1.1.2.3.1.3 AlmancaTurkce.AlmancaTurkceName Property

C#

```
[Column]
public string AlmancaTurkceName;
```

Description

This is AlmancaTurkceName, a member of class AlmancaTurkce.

1.1.2.3.1.4 AlmancaTurkce.AlmancaTurkceNameMeaning Property

C#

```
[Column]
public string AlmancaTurkceNameMeaning;
```

Description

This is AlmancaTurkceNameMeaning, a member of class AlmancaTurkce.

1.1.2.4 AlphaKeyGroup Class

Class Hierarchy

C#

```
public class AlphaKeyGroup<T> : List<T>;
```

File

AlphaKeyGroup.cs (see page 64)

Description

This is class AwesomeDictionary.AlphaKeyGroup.

Methods

	Name	Description
	AlphaKeyGroup (see page 21)	Public constructor.

AlphaKeyGroup Delegates

Name	Description
GetKeyDelegate (see page 23)	The delegate that is used to get the key information.

AlphaKeyGroup Methods

	Name	Description
	CreateGroups (see page 21)	Create a list of AlphaGroupwith keys set by a SortedLocaleGrouping.

AlphaKeyGroup Properties

	Name	Description
	Key (see page 22)	The Key of this group.

1.1.2.4.1 AlphaKeyGroup.AlphaKeyGroup Constructor

Public constructor.

C#

```
public AlphaKeyGroup(string key);
```

Parameters

Parameters	Description
string key	The key for this group.

Body Source

```
public AlphaKeyGroup(string key)
{
    Key = key;
}
```

1.1.2.4.2 AlphaKeyGroup Methods

1.1.2.4.2.1 AlphaKeyGroup.CreateGroups Method

Create a list of AlphaGroupwith keys set by a SortedLocaleGrouping.

C#

```
public static List<AlphaKeyGroup<T>> CreateGroups(IEnumerable<T> items, CultureInfo ci,
GetKeyDelegate getKey, bool sort);
```

Parameters

Parameters	Description
IEnumerable<T> items	The items to place in the groups.
CultureInfo ci	The CultureInfo to group and sort by.
GetKeyDelegate getKey	A delegate to get the key from an item.
bool sort	Will sort the data if true.

Returns

An items source for a LongListSelector

Body Source

```
public static List<AlphaKeyGroup<T>> CreateGroups(IEnumerable<T> items, CultureInfo ci,
GetKeyDelegate getKey, bool sort)
{
    SortedLocaleGrouping slg = new SortedLocaleGrouping(ci);
    List<AlphaKeyGroup<T>> list = CreateGroups(slg);

    foreach (T item in items)
    {
        int index = 0;
        if (slg.SupportsPhonetics)
        {
            //check if your database has yomi string for item
            //if it does not, then do you want to generate Yomi or ask the user for
this item.
            //index = slg.GetGroupIndex(getKey(Yomiof(item)));
        }
        else
        {
            index = slg.GetGroupIndex(getKey(item));
        }
        if (index >= 0 && index < list.Count)
        {
            list[index].Add(item);
        }
    }

    if (sort)
    {
        foreach (AlphaKeyGroup<T> group in list)
        {
            group.Sort((c0, c1) => { return ci.CompareInfo.Compare(getKey(c0),
getKey(c1)); });
        }
    }

    return list;
}
```

1.1.2.4.3 AlphaKeyGroup Properties

1.1.2.4.3.1 AlphaKeyGroup.Key Property

The Key of this group.

C#

```
public string Key;
```

1.1.2.4.4 AlphaKeyGroup Delegates

1.1.2.4.4.1 AlphaKeyGroup.GetKeyDelegate Delegate

The delegate that is used to get the key information.

C#

```
public delegate string GetKeyDelegate(T item);
```

Parameters

Parameters	Description
item	An object of type T

Returns

The key value to use for this object

1.1.2.5 App Class

Class Hierarchy

Application
AwesomeDictionary.App

C#

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

File

App.xaml.cs (see page 66)

Description

This is class AwesomeDictionary.App.

Methods

	Name	Description
	App (see page 23)	Constructor for the Application object.

App Methods

	Name	Description
	ReadFile (see page 24)	This is ReadFile, a member of class App.

App Properties

	Name	Description
	RootFrame (see page 25)	Provides easy access to the root frame of the Phone Application.

1.1.2.5.1 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;
    }

}
```

1.1.2.5.2 App Methods

1.1.2.5.2.1 App.ReadFile Method

C#

```
public string ReadFile(string filePath);
```

Description

This is ReadFile, a member of class App.

Body Source

```
public string ReadFile(string filePath)
{
    var ResrouceStream = Application.GetResourceStream(new Uri(filePath, UriKind.Relative));
    if (ResrouceStream != null)
    {
        Stream myFileStream = ResrouceStream.Stream;
        if (myFileStream.CanRead)
        {
            StreamReader myStreamReader = new StreamReader(myFileStream);

            return myStreamReader.ReadToEnd();
        }
    }
    return "";
}
```

1.1.2.5.3 App Properties

1.1.2.5.3.1 App.RootFrame Property

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

C#

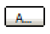
```
public static PhoneApplicationFrame RootFrame;
```

Returns

The root frame of the Phone Application.

1.1.2.6 AppSettings Class

Class Hierarchy



C#

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

File

AppSettings.cs (see page 87)

Description

This is class AwesomeDictionary.AppSettings.

AppSettings Properties

	Name	Description
	AppBackgroundColor (see page 25)	This is AppBackgroundColor, a member of class AppSettings.
	AppBackgroundImage (see page 26)	This is AppBackgroundImage, a member of class AppSettings.
	AppLangName (see page 26)	This is AppLangName, a member of class AppSettings.
	AppSettingsId (see page 26)	This is AppSettingsId, a member of class AppSettings.
	FontFamily (see page 26)	This is FontFamily, a member of class AppSettings.
	FontSize (see page 26)	This is FontSize, a member of class AppSettings.

1.1.2.6.1 AppSettings Properties

1.1.2.6.1.1 AppSettings.AppBackgroundColor Property

C#

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

Description

This is AppBackgroundColor, a member of class AppSettings.

1.1.2.6.1.2 AppSettings.AppBackgroundImage Property

C#

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

Description

This is AppBackgroundImage, a member of class AppSettings.

1.1.2.6.1.3 AppSettings.AppLangName Property

C#

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

Description

This is AppLangName, a member of class AppSettings.

1.1.2.6.1.4 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.

1.1.2.6.1.5 AppSettings.FontFamily Property

C#

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

Description

This is FontFamily, a member of class AppSettings.

1.1.2.6.1.6 AppSettings.FontSize Property

C#

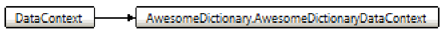
```
[Column]
public string FontSize;
```

Description

This is FontSize, a member of class AppSettings.

1.1.2.7 AwesomeDictionaryDataContext Class

Class Hierarchy



C#

```
public class AwesomeDictionaryDataContext : DataContext;
```

File

AwesomeDictionaryDataContext.cs (see page 88)

Description

This is class AwesomeDictionary.AwesomeDictionaryDataContext.

Methods

	Name	Description
	AwesomeDictionaryDataContext (see page 27)	This is AwesomeDictionaryDataContext, a member of class AwesomeDictionaryDataContext.

AwesomeDictionaryDataContext Fields

	Name	Description
	AllNames (see page 27)	This is AllNames, a member of class AwesomeDictionaryDataContext.
	AlmancaTurkces (see page 28)	This is AlmancaTurkces, a member of class AwesomeDictionaryDataContext.
	AppSettings (see page 28)	This is AppSettings, a member of class AwesomeDictionaryDataContext.
	Bilisims (see page 28)	This is Bilisims, a member of class AwesomeDictionaryDataContext.
	BuyukLugats (see page 28)	This is BuyukLugats, a member of class AwesomeDictionaryDataContext.
	ConnectionString (see page 28)	This is ConnectionString, a member of class AwesomeDictionaryDataContext.
	EnglishTurkishVol1s (see page 28)	This is EnglishTurkishVol1s, a member of class AwesomeDictionaryDataContext.
	EnglishTurkishVol2s (see page 28)	This is EnglishTurkishVol2s, a member of class AwesomeDictionaryDataContext.
	Favourites (see page 29)	This is Favourites, a member of class AwesomeDictionaryDataContext.
	Kelimes (see page 29)	This is Kelimes, a member of class AwesomeDictionaryDataContext.
	Oxfords (see page 29)	This is Oxfords, a member of class AwesomeDictionaryDataContext.
	RisaleNurs (see page 29)	This is RisaleNurs, a member of class AwesomeDictionaryDataContext.

1.1.2.7.1 AwesomeDictionaryDataContext.AwesomeDictionaryDataContext Constructor

C#

```
public AwesomeDictionaryDataContext(string connectionString);
```

Description

This is AwesomeDictionaryDataContext, a member of class AwesomeDictionaryDataContext.

Body Source

```
public AwesomeDictionaryDataContext(string connectionString)
: base(connectionString) { }
```

1.1.2.7.2 AwesomeDictionaryDataContext Fields

1.1.2.7.2.1 AwesomeDictionaryDataContext.AllNames Field

C#

```
public Table<All> AllNames;
```

Description

This is AllNames, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.2 AwesomeDictionaryDataContext.AlmancaTurkces Field

C#

```
public Table<AlmancaTurkce> AlmancaTurkces;
```

Description

This is AlmancaTurkces, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.3 AwesomeDictionaryDataContext.AppSettings Field

C#

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

Description

This is AppSettings, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.4 AwesomeDictionaryDataContext.Bilisims Field

C#

```
public Table<BilisimSozlugu> Bilisims;
```

Description

This is Bilisims, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.5 AwesomeDictionaryDataContext.BuyukLugats Field

C#

```
public Table<BuyukLugat> BuyukLugats;
```

Description

This is BuyukLugats, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.6 AwesomeDictionaryDataContext.ConnectionString Field

C#

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

Description

This is ConnectionString, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.7 AwesomeDictionaryDataContext.EnglishTurkishVol1s Field

C#

```
public Table<EnglishTurkishVol1> EnglishTurkishVol1s;
```

Description

This is EnglishTurkishVol1s, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.8 AwesomeDictionaryDataContext.EnglishTurkishVol2s Field

C#

```
public Table<EnglishTurkishVol2> EnglishTurkishVol2s;
```

Description

This is EnglishTurkishVol2s, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.9 AwesomeDictionaryDataContext.Favourites Field

C#

```
public Table<Favourite> Favourites;
```

Description

This is Favourites, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.10 AwesomeDictionaryDataContext.Kelimes Field

C#

```
public Table<KelimeAnlamlari> Kelimes;
```

Description

This is Kelimes, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.11 AwesomeDictionaryDataContext.Oxfords Field

C#

```
public Table<OxfordEnglishEnglish> Oxfords;
```

Description

This is Oxfords, a member of class AwesomeDictionaryDataContext.

1.1.2.7.2.12 AwesomeDictionaryDataContext.RisaleNurs Field

C#

```
public Table<RisaleNur> RisaleNurs;
```

Description

This is RisaleNurs, a member of class AwesomeDictionaryDataContext.

1.1.2.8 BackgroundColorSettingsPage Class

Class Hierarchy

C#

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

File

BackgroundColorSettingsPage.xaml.cs (see page 89)

Description

This is class AwesomeDictionary.BackgroundColorSettingsPage.

Methods

	Name	Description
	BackgroundColorSettingsPage (see page 30)	This is BackgroundColorSettingsPage, a member of class BackgroundColorSettingsPage.

BackgroundColorSettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 30)	This is OnFragmentNavigation, a member of class BackgroundColorSettingsPage.
	OnNavigatedFrom (see page 31)	This is OnNavigatedFrom, a member of class BackgroundColorSettingsPage.
	OnNavigatedTo (see page 31)	This is OnNavigatedTo, a member of class BackgroundColorSettingsPage.

1.1.2.8.1 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();
}
```

1.1.2.8.2 BackgroundColorSettingsPage Methods

1.1.2.8.2.1 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);
//    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
//    {
//        lblGeneralSettings.Text = AppResources.GeneralSettings;
//        lblBackgroundColor.Text = AppResources.SelectFontSize;
//    }
}
```

1.1.2.8.2.2 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();

}
```

1.1.2.8.2.3 BackgroundColorSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class BackgroundColorSettingsPage.

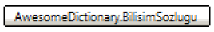
Body Source

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

}
```

1.1.2.9 BilisimSozlugu Class

Class Hierarchy



C#

```
[Table]
public class BilisimSozlugu;
```

File

BilisimSozlugu.cs (see page 92)

Description

This is class AwesomeDictionary.BilisimSozlugu.

BilisimSozlugu Properties

	Name	Description
	BilisimSozluguId (see page 32)	This is BilisimSozluguId, a member of class BilisimSozlugu.
	BilisimSozluguMeaning (see page 32)	This is BilisimSozluguMeaning, a member of class BilisimSozlugu.
	BilisimSozluguName (see page 32)	This is BilisimSozluguName, a member of class BilisimSozlugu.
	BilisimSozluguNameMeaning (see page 32)	This is BilisimSozluguNameMeaning, a member of class BilisimSozlugu.

1.1.2.9.1 BilisimSozlugu Properties

1.1.2.9.1.1 BilisimSozlugu.BilisimSozluguId Property

C#

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

Description

This is BilisimSozluguId, a member of class BilisimSozlugu.

1.1.2.9.1.2 BilisimSozlugu.BilisimSozluguMeaning Property

C#

```
[Column]
public string BilisimSozluguMeaning;
```

Description

This is BilisimSozluguMeaning, a member of class BilisimSozlugu.

1.1.2.9.1.3 BilisimSozlugu.BilisimSozluguName Property

C#

```
[Column]
public string BilisimSozluguName;
```

Description

This is BilisimSozluguName, a member of class BilisimSozlugu.

1.1.2.9.1.4 BilisimSozlugu.BilisimSozluguNameMeaning Property

C#

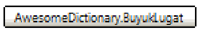
```
[Column]
public string BilisimSozluguNameMeaning;
```

Description

This is BilisimSozluguNameMeaning, a member of class BilisimSozlugu.

1.1.2.10 BuyukLugat Class

Class Hierarchy



C#

```
[Table]
public class BuyukLugat;
```

File

BuyukLugat.cs (↗ see page 93)

Description

This is class AwesomeDictionary.BuyukLugat.

BuyukLugat Properties

	Name	Description
	BuyukLugatId (↗ see page 33)	This is BuyukLugatId, a member of class BuyukLugat.
	BuyukLugatMeaning (↗ see page 33)	This is BuyukLugatMeaning, a member of class BuyukLugat.
	BuyukLugatName (↗ see page 33)	This is BuyukLugatName, a member of class BuyukLugat.
	BuyukLugatNameMeaning (↗ see page 34)	This is BuyukLugatNameMeaning, a member of class BuyukLugat.

1.1.2.10.1 BuyukLugat Properties

1.1.2.10.1.1 BuyukLugat.BuyukLugatId Property

C#

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

Description

This is BuyukLugatId, a member of class BuyukLugat.

1.1.2.10.1.2 BuyukLugat.BuyukLugatMeaning Property

C#

```
[Column]
public string BuyukLugatMeaning;
```

Description

This is BuyukLugatMeaning, a member of class BuyukLugat.

1.1.2.10.1.3 BuyukLugat.BuyukLugatName Property

C#

```
[Column]
public string BuyukLugatName;
```

Description

This is BuyukLugatName, a member of class BuyukLugat.

1.1.2.10.1.4 BuyukLugat.BuyukLugatNameMeaning Property

C#

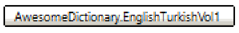
```
[Column]
public string BuyukLugatNameMeaning;
```

Description

This is BuyukLugatNameMeaning, a member of class BuyukLugat.

1.1.2.11 EnglishTurkishVol1 Class

Class Hierarchy



C#

```
[Table]
public class EnglishTurkishVol1;
```

File

EnglishTurkishVol1.cs (see page 94)

Description

This is class AwesomeDictionary.EnglishTurkishVol1.

EnglishTurkishVol1 Properties

	Name	Description
▤	EnglishVol1Id (see page 34)	This is EnglishVol1Id, a member of class EnglishTurkishVol1.
▤	EnglishVol1Meaning (see page 34)	This is EnglishVol1Meaning, a member of class EnglishTurkishVol1.
▤	EnglishVol1Name (see page 35)	This is EnglishVol1Name, a member of class EnglishTurkishVol1.
▤	EnglishVol1NameMeaning (see page 35)	This is EnglishVol1NameMeaning, a member of class EnglishTurkishVol1.

1.1.2.11.1 EnglishTurkishVol1 Properties

1.1.2.11.1.1 EnglishTurkishVol1.EnglishVol1Id Property

C#

```
[Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
CanBeNull = false)]
public int EnglishVol1Id;
```

Description

This is EnglishVol1Id, a member of class EnglishTurkishVol1.

1.1.2.11.1.2 EnglishTurkishVol1.EnglishVol1Meaning Property

C#

```
[Column]
```

```
public string EnglishVol1Meaning;
```

Description

This is EnglishVol1Meaning, a member of class EnglishTurkishVol1.

1.1.2.11.1.3 EnglishTurkishVol1.EnglishVol1Name Property

C#

```
[Column]
public string EnglishVol1Name;
```

Description

This is EnglishVol1Name, a member of class EnglishTurkishVol1.

1.1.2.11.1.4 EnglishTurkishVol1.EnglishVol1NameMeaning Property

C#

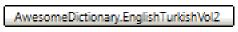
```
[Column]
public string EnglishVol1NameMeaning;
```

Description

This is EnglishVol1NameMeaning, a member of class EnglishTurkishVol1.

1.1.2.12 EnglishTurkishVol2 Class

Class Hierarchy



C#

```
[Table]
public class EnglishTurkishVol2;
```

File

EnglishTurkishVol2.cs (see page 94)

Description

This is class AwesomeDictionary.EnglishTurkishVol2.

EnglishTurkishVol2 Properties

	Name	Description
▤	EnglishVol2Id (see page 35)	This is EnglishVol2Id, a member of class EnglishTurkishVol2.
▤	EnglishVol2Meaning (see page 36)	This is EnglishVol2Meaning, a member of class EnglishTurkishVol2.
▤	EnglishVol2Name (see page 36)	This is EnglishVol2Name, a member of class EnglishTurkishVol2.
▤	EnglishVol2NameMeaning (see page 36)	This is EnglishVol2NameMeaning, a member of class EnglishTurkishVol2.

1.1.2.12.1 EnglishTurkishVol2 Properties

1.1.2.12.1.1 EnglishTurkishVol2.EnglishVol2Id Property

C#

```
[Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
CanBeNull = false)]
public int EnglishVol2Id;
```

Description

This is EnglishVol2Id, a member of class EnglishTurkishVol2.

1.1.2.12.1.2 EnglishTurkishVol2.EnglishVol2Meaning Property

C#

```
[Column]
public string EnglishVol2Meaning;
```

Description

This is EnglishVol2Meaning, a member of class EnglishTurkishVol2.

1.1.2.12.1.3 EnglishTurkishVol2.EnglishVol2Name Property

C#

```
[Column]
public string EnglishVol2Name;
```

Description

This is EnglishVol2Name, a member of class EnglishTurkishVol2.

1.1.2.12.1.4 EnglishTurkishVol2.EnglishVol2NameMeaning Property

C#

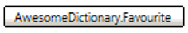
```
[Column]
public string EnglishVol2NameMeaning;
```

Description

This is EnglishVol2NameMeaning, a member of class EnglishTurkishVol2.

1.1.2.13 Favourite Class

Class Hierarchy



C#

```
[Table]
public class Favourite;
```

File

Favourite.cs (see page 95)

Description

This is class AwesomeDictionary.Favourite.

Favourite Properties

	Name	Description
	FavouriteAllId (see page 37)	This is FavouriteAllId, a member of class Favourite.
	FavouriteId (see page 37)	This is FavouriteId, a member of class Favourite.
	FavouriteName (see page 37)	This is FavouriteName, a member of class Favourite.

1.1.2.13.1 Favourite Properties

1.1.2.13.1.1 Favourite.FavouriteAllId Property

C#

```
[Column]
public int FavouriteAllId;
```

Description

This is FavouriteAllId, a member of class Favourite.

1.1.2.13.1.2 Favourite.FavouriteId Property

C#

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

Description

This is FavouriteId, a member of class Favourite.

1.1.2.13.1.3 Favourite.FavouriteName Property

C#

```
[Column]
public string FavouriteName;
```

Description

This is FavouriteName, a member of class Favourite.

1.1.2.14 FontFamilySettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.FontFamilySettingsPage

C#

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

File

FontFamilySettingsPage.xaml.cs (see page 96)

Description

This is class AwesomeDictionary.FontFamilySettingsPage.

Methods

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

FontFamilySettingsPage Methods

	Name	Description
	OnFragmentNavigation (see page 38)	This is OnFragmentNavigation, a member of class FontFamilySettingsPage.
	OnNavigatedFrom (see page 39)	This is OnNavigatedFrom, a member of class FontFamilySettingsPage.
	OnNavigatedTo (see page 39)	This is OnNavigatedTo, a member of class FontFamilySettingsPage.

1.1.2.14.1 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("Albumman 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;

    lblFontFamily.Text = AppResources.SelectFontFamily;
    lblGeneralSettings.Text = AppResources.GeneralSettings;

    SetBackgroundColor();
}
```

1.1.2.14.2 FontFamilySettingsPage Methods

1.1.2.14.2.1 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);
    //artistId = int.Parse(e.Fragment);
    //using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
    //{
    //    var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single() as Artist;
    //    lblArtistName.Text = artist.ArtistName;
    //    lblFontFamily.Text = AppResources.SelectFontFamily;
    //}

}
```

1.1.2.14.2.2 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);
}
```

1.1.2.14.2.3 FontFamilySettingsPage.OnNavigatedTo Method

C#

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

Description

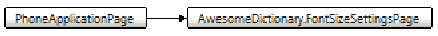
This is OnNavigatedTo, a member of class FontFamilySettingsPage.

Body Source

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

1.1.2.15 FontSizeSettingsPage Class

Class Hierarchy



C#

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

File

FontSizeSettingsPage.xaml.cs (see page 98)

Description

This is class AwesomeDictionary.FontSizeSettingsPage.

Methods

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

FontSizeSettingsPage Methods

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

1.1.2.15.1 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;

    lblGeneralSettings.Text = AppResources.GeneralSettings;
    lblFontSize.Text = AppResources.SelectFontSize;

    SetBackgroundColor();
}
```

1.1.2.15.2 FontSizeSettingsPage Methods

1.1.2.15.2.1 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);
    //artistId = int.Parse(e.Fragment);
    //using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
    //{
    //    var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single() as Artist;
    //    lblArtistName.Text = artist.ArtistName;
    //    lblFontSize.Text = AppResources.SelectFontSize;
    //}

}
```

1.1.2.15.2.2 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);
}
```

1.1.2.15.2.3 FontSizeSettingsPage.OnNavigatedTo Method

C#

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

Description

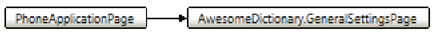
This is OnNavigatedTo, a member of class FontSizeSettingsPage.

Body Source

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

1.1.2.16 GeneralSettingsPage Class

Class Hierarchy



C#

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

File

GeneralSettingsPage.xaml.cs (see page 101)

Description

This is class AwesomeDictionary.GeneralSettingsPage.

Methods

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

GeneralSettingsPage Methods

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

1.1.2.16.1 GeneralSettingsPage.GeneralSettingsPage Constructor

C#

```
public GeneralSettingsPage();
```

Description

This is GeneralSettingsPage, a member of class GeneralSettingsPage.

Body Source

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

    pvGeneralSettings.Title = AppResources.GeneralSettings;

    piLanguage.Header = AppResources.Language;
    piDictionaryInstall.Header = AppResources.DictionaryInstall;
    //piOtherSettings.Header = AppResources.OtherSettings;
    piBackground.Header = AppResources.Background;

    //lblOneDrive.Text = AppResources.OneDrive;

    btnLanguage.Content = AppResources.Select;
    btnBackgroundColor.Content = AppResources.Select;
    //btnOneDrive.Content = AppResources.Login;
    //btnOneDrive.SignInText = AppResources.SignIn;
    //btnOneDrive.SignOutText = AppResources.SignOut;
    txtInstalling.Text = AppResources.Installing;

    pbInstall.Visibility = Visibility.Collapsed;
    txtInstalling.Visibility = Visibility.Collapsed;
    txtInstalling.BorderBrush = this.LayoutRoot.Background;

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

    piFont.Header = AppResources.Font;
    btnFontFamily.Content = AppResources.Select;
    btnFontSize.Content = AppResources.Select;

    btnInstall.Content = AppResources.Install;
    btnUninstall.Content = AppResources.Uninstall;

    txtBuyukLugat.Text = AppResources.BuyukLugat + " (49331 " + AppResources.Word +
")";
        txtComputer.Text = AppResources.ComputerDictionary + " (3508 " + AppResources.Word + ")";
        txtGerman.Text = AppResources.GermanTurkish + " (17526 " + AppResources.Word + ")";
        txtOxford.Text = AppResources.OxfordDictionary + " (36369 " + AppResources.Word + ")";
        txtWordMeaning.Text = AppResources.WordMeaning + " (10535 " + AppResources.Word + ")";
        txtRisaleNur.Text = AppResources.RisaleNur + " (9478 " + AppResources.Word + ")";
        txtEnglishVol1.Text = AppResources.EnglishTurkishVol1 + " (127157 " + AppResources.Word + ")";
        txtEnglishVol2.Text = AppResources.EnglishTurkishVol2 + " (3699 " + AppResources.Word + ")";


        //cbSync.Content = AppResources.SyncOnOneFile;
        //cbSync.IsEnabled = false;

        SetBackgroundColor();

        using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
        {
            var appSettings = context.AppSettings.First() as AppSettings;

            lblFontFamily.Text = AppResources.FontFamily + " (" + AppResources.Selected + ": " + appSettings.FontFamily + ")";
            lblFontSize.Text = AppResources.FontSize + " (" + AppResources.Selected + ": " + appSettings.FontSize + ")";

            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.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 + ")";
        }
    }
}
```

1.1.2.16.2 GeneralSettingsPage Methods

1.1.2.16.2.1 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();

}
```

1.1.2.16.2.2 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();

}
```

1.1.2.16.2.3 GeneralSettingsPage.ReadFile Method

C#

```
public string ReadFile(string filePath);
```

Description

This is ReadFile, a member of class GeneralSettingsPage.

Body Source

```
public string ReadFile(string filePath)
{
    var ResrouceStream = Application.GetResourceStream(new Uri(filePath,
UriKind.Relative));
    if (ResrouceStream != null)
    {
        Stream myFileStream = ResrouceStream.Stream;
        if (myFileStream.CanRead)
        {
            StreamReader myStreamReader = new StreamReader(myFileStream);

            return myStreamReader.ReadToEnd();
        }
    }
    return "";
}
```

1.1.2.17 KelimeAnlamlari Class

Class Hierarchy

C#

```
[Table]
public class KelimeAnlamlari;
```

File

KelimeAnlamlari.cs (see page 120)

Description

This is class AwesomeDictionary.KelimeAnlamlari.

KelimeAnlamlari Properties

	Name	Description
▤	KelimeAnlamlariId (see page 46)	This is KelimeAnlamlariId, a member of class KelimeAnlamlari.
▤	KelimeAnlamlariMeaning (see page 46)	This is KelimeAnlamlariMeaning, a member of class KelimeAnlamlari.
▤	KelimeAnlamlariName (see page 47)	This is KelimeAnlamlariName, a member of class KelimeAnlamlari.
▤	KelimeAnlamlariNameMeaning (see page 47)	This is KelimeAnlamlariNameMeaning, a member of class KelimeAnlamlari.

1.1.2.17.1 KelimeAnlamlari Properties

1.1.2.17.1.1 KelimeAnlamlari.KelimeAnlamlariId Property

C#

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

Description

This is KelimeAnlamlariId, a member of class KelimeAnlamlari.

1.1.2.17.1.2 KelimeAnlamlari.KelimeAnlamlariMeaning Property

C#

```
[Column]
public string KelimeAnlamlariMeaning;
```

Description

This is KelimeAnlamlariMeaning, a member of class KelimeAnlamlari.

1.1.2.17.1.3 KelimeAnlamlari.KelimeAnlamlariName Property

C#

```
[Column]
public string KelimeAnlamlariName;
```

Description

This is KelimeAnlamlariName, a member of class KelimeAnlamlari.

1.1.2.17.1.4 KelimeAnlamlari.KelimeAnlamlariNameMeaning Property

C#

```
[Column]
public string KelimeAnlamlariNameMeaning;
```

Description

This is KelimeAnlamlariNameMeaning, a member of class KelimeAnlamlari.

1.1.2.18 LanguageSettingsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.LanguageSettingsPage

C#

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

File

LanguageSettingsPage.xaml.cs (see page 121)

Description

This is class AwesomeDictionary.LanguageSettingsPage.

Methods

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

LanguageSettingsPage Methods

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

1.1.2.18.1 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.Sanskrit);
    //lstLanguage.Items.Add(AppResources.Thai);

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

    SetBackgroundColor();
}
```

1.1.2.18.2 LanguageSettingsPage Methods

1.1.2.18.2.1 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);
}
```

1.1.2.18.2.2 LanguageSettingsPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class LanguageSettingsPage.

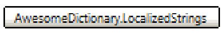
Body Source

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

1.1.2.19 LocalizedStrings Class

Provides access to string resources.

Class Hierarchy



C#

```
public class LocalizedStrings;
```

File

LocalizedStrings.cs (see page 125)

LocalizedStrings Properties

	Name	Description
	LocalizedResources (see page 49)	This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.19.1 LocalizedStrings Properties

1.1.2.19.1.1 LocalizedStrings.LocalizedResources Property

C#

```
public AppResources LocalizedResources;
```

Description

This is LocalizedResources, a member of class LocalizedStrings.

1.1.2.20 MainPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.MainPage

C#

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

File

MainPage.xaml.cs (see page 125)

Description

This is class AwesomeDictionary.MainPage.

Methods

	Name	Description
	MainPage (see page 49)	This is MainPage, a member of class MainPage.

1.1.2.20.1 MainPage.MainPage Constructor

C#

```
public MainPage();
```

Description

This is MainPage, a member of class MainPage.

Body Source

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

    ApplicationBar = new ApplicationBar();

    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);

    SetBackgroundColor();

    piFavourite.Header = AppResources.MyFavourites;
    piRandomWords.Header = AppResources.RandomWords;

}
```

1.1.2.21 NameDetailPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.NameDetailPage

C#

```
public class NameDetailPage : PhoneApplicationPage;
```

File

NameDetailPage.xaml.cs (see page 129)

Description

This is class AwesomeDictionary.NameDetailPage.

Methods

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

NameDetailPage Fields

	Name	Description
◆	flag (see page 52)	This is flag, a member of class NameDetailPage.
◆	isFilled (see page 52)	This is isFilled, a member of class NameDetailPage.
◆	pageName (see page 52)	This is pageName, a member of class NameDetailPage.
◆	ratingValue (see page 52)	This is ratingValue, a member of class NameDetailPage.
◆	wordId (see page 52)	This is wordId, a member of class NameDetailPage.

NameDetailPage Methods

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

1.1.2.21.1 NameDetailPage.NameDetailPage Constructor

C#

```
public NameDetailPage();
```

Description

This is NameDetailPage, a member of class NameDetailPage.

Body Source

```
public NameDetailPage()
{
    InitializeComponent();

    ApplicationBar = new ApplicationBar();

    //ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    //button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
    //button1.Text = "Kaydet";
    //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.SendWithEmail;
    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.Share;
    ApplicationBar.Buttons.Add(button4);
    button4.Click += new EventHandler(ShareNameButton_Click);

    isFilled = false;

    //ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
    //menuItem1.Text = "Sil";
    //ApplicationBar.MenuItems.Add(menuItem1);
    //menuItem1.Click += new EventHandler(DeleteNameMenuItem_Click);

    //List<string> genderList = new List<string>();
    //genderList.Add("Lütfen seçiniz");
    //genderList.Add("Erkek");
    //genderList.Add("Kadin");
    //genderList.Add("Erkek-Kadin");
    //lpGender.ItemsSource = genderList;
    //lpGender.SelectedIndex = 0;

    SetBackgroundColor();
}
```

1.1.2.21.2 NameDetailPage Fields

1.1.2.21.2.1 NameDetailPage.flag Field

C#

```
public bool flag;
```

Description

This is flag, a member of class NameDetailPage.

1.1.2.21.2.2 NameDetailPage.isFilled Field

C#

```
public bool isFilled;
```

Description

This is isFilled, a member of class NameDetailPage.

1.1.2.21.2.3 NameDetailPage.pageName Field

C#

```
public string pageName;
```

Description

This is pageName, a member of class NameDetailPage.

1.1.2.21.2.4 NameDetailPage.ratingValue Field

C#

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

Description

This is ratingValue, a member of class NameDetailPage.

1.1.2.21.2.5 NameDetailPage.wordId Field

C#

```
public int wordId;
```

Description

This is wordId, a member of class NameDetailPage.

1.1.2.21.3 NameDetailPage Methods

1.1.2.21.3.1 NameDetailPage.OnFragmentNavigation Method

C#

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

Description

This is OnFragmentNavigation, a member of class NameDetailPage.

Body Source

```
protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
    // displays "Fragment: Detail"
    //MessageBox.Show("Folder Id: " + e.Fragment);
    base.OnFragmentNavigation(e);
    //string fragmentName = e.Fragment.ToString();
    wordId = Convert.ToInt32(e.Fragment);
    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
    {
        var name = context.AllNames.Where(j => j.AllId.Equals(wordId)).SingleOrDefault() as All;
        allName = name;
        //for (int i = 0; i < lpGender.Items.Count; i++)
        //{
        //    if (lpGender.Items[i].ToString() == name.AllGender)
        //    {
        //        lpGender.SelectedIndex = i;
        //        break;
        //    }
        //}
        lblNameDetail.Text = name.AllName;
        txtMeaning.Text = name.AllMeaning + Environment.NewLine + Environment.NewLine + AppResources.Source + ":" + name.AllSource + "";
        txtMeaning.IsEnabled = false;
        //txtMeaning.Text = name.AllMeaning;

        var favourite = context.Favourites.Where(j => j.FavouriteAllId.Equals(wordId)).SingleOrDefault() as Favourite;
        if (favourite != null)
        {
            ApplicationBarMenuItem menuItem4 = new ApplicationBarMenuItem();
            menuItem4.Text = AppResources.RemoveFromFavourites;
            ApplicationBar.MenuItems.Add(menuItem4);
            menuItem4.Click += new EventHandler(RemoveFavouritesMenuItem_Click);
        }
        else
        {
            ApplicationBarMenuItem menuItem3 = new ApplicationBarMenuItem();
            menuItem3.Text = AppResources.AddToFavourites;
            ApplicationBar.MenuItems.Add(menuItem3);
            menuItem3.Click += new EventHandler(AddFavouritesMenuItem_Click);
        }

        //var myUpdate = context.MyUpdates.Where(j => j.MyUpdateName.Equals(fragmentName)).SingleOrDefault() as MyUpdate; ;
        //if (myUpdate != null)
        //{
        //    ApplicationBarMenuItem menuItem2 = new ApplicationBarMenuItem();
        //    menuItem2.Text = "Sisteme Eklenmesi Için Gönder";
        //    ApplicationBar.MenuItems.Add(menuItem2);
        //    menuItem2.Click += new EventHandler(SaveAndSendMenuItem_Click);

        //}


        isFilled = true;
        //pvName.SelectedIndex = 0;
    }
}
```

1.1.2.21.3.2 NameDetailPage.OnNavigatedFrom Method

C#

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

Description

This is OnNavigatedFrom, a member of class NameDetailPage.

Body Source

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

}
```

1.1.2.21.3.3 NameDetailPage.OnNavigatedTo Method

C#

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

Description

This is OnNavigatedTo, a member of class NameDetailPage.

Body Source

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

    var lastPage = NavigationService.BackStack.FirstOrDefault();
    pageName = lastPage.Source.ToString();
    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
    {
        var appSettings = context.AppSettings.First();

        FontFamily temp = new FontFamily(appSettings.FontFamily);
        double fontsize = double.Parse(appSettings.FontSize);
        txtMeaning.FontFamily = temp;
        txtMeaning.FontSize = fontsize;
    }

    txtMeaning.IsEnabled = false;
    isFilled = false;
    //SetBackgroundColor();
    //while (NavigationService.CanGoBack)
    //NavigationService.RemoveBackEntry();

}
```

1.1.2.22 OxfordEnglishEnglish Class

Class Hierarchy

C#

```
[Table]
public class OxfordEnglishEnglish;
```

File

OxfordEnglishEnglish.cs (see page 139)

Description

This is class AwesomeDictionary.OxfordEnglishEnglish.

OxfordEnglishEnglish Properties

	Name	Description
	OxfordId (see page 55)	This is OxfordId, a member of class OxfordEnglishEnglish.
	OxfordMeaning (see page 55)	This is OxfordMeaning, a member of class OxfordEnglishEnglish.
	OxfordName (see page 55)	This is OxfordName, a member of class OxfordEnglishEnglish.
	OxfordNameMeaning (see page 55)	This is OxfordNameMeaning, a member of class OxfordEnglishEnglish.

1.1.2.22.1 OxfordEnglishEnglish Properties

1.1.2.22.1.1 OxfordEnglishEnglish.OxfordId Property

C#

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

Description

This is OxfordId, a member of class OxfordEnglishEnglish.

1.1.2.22.1.2 OxfordEnglishEnglish.OxfordMeaning Property

C#

```
[Column]
public string OxfordMeaning;
```

Description

This is OxfordMeaning, a member of class OxfordEnglishEnglish.

1.1.2.22.1.3 OxfordEnglishEnglish.OxfordName Property

C#

```
[Column]
public string OxfordName;
```

Description

This is OxfordName, a member of class OxfordEnglishEnglish.

1.1.2.22.1.4 OxfordEnglishEnglish.OxfordNameMeaning Property

C#

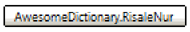
```
[Column]
public string OxfordNameMeaning;
```

Description

This is OxfordNameMeaning, a member of class OxfordEnglishEnglish.

1.1.2.23 RisaleNur Class

Class Hierarchy



C#

```
[Table]
public class RisaleNur;
```

File

RisaleNur.cs (see page 139)

Description

This is class AwesomeDictionary.RisaleNur.

RisaleNur Properties

	Name	Description
	RisaleNurId (see page 56)	This is RisaleNurId, a member of class RisaleNur.
	RisaleNurMeaning (see page 56)	This is RisaleNurMeaning, a member of class RisaleNur.
	RisaleNurName (see page 56)	This is RisaleNurName, a member of class RisaleNur.
	RisaleNurNameMeaning (see page 57)	This is RisaleNurNameMeaning, a member of class RisaleNur.

1.1.2.23.1 RisaleNur Properties

1.1.2.23.1.1 RisaleNur.RisaleNurId Property

C#

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

Description

This is RisaleNurId, a member of class RisaleNur.

1.1.2.23.1.2 RisaleNur.RisaleNurMeaning Property

C#

```
[Column]
public string RisaleNurMeaning;
```

Description

This is RisaleNurMeaning, a member of class RisaleNur.

1.1.2.23.1.3 RisaleNur.RisaleNurName Property

C#

```
[Column]
public string RisaleNurName;
```

Description

This is RisaleNurName, a member of class RisaleNur.

1.1.2.23.1.4 RisaleNur.RisaleNurNameMeaning Property

C#

```
[Column]
public string RisaleNurNameMeaning;
```

Description

This is RisaleNurNameMeaning, a member of class RisaleNur.

1.1.2.24 SearchPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.SearchPage

C#

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

File

SearchPage.xaml.cs (see page 140)

Description

This is class AwesomeDictionary.SearchPage.

Methods

	Name	Description
◆	SearchPage (see page 57)	This is SearchPage, a member of class SearchPage.

SearchPage Methods

	Name	Description
◆	OnNavigatedFrom (see page 58)	This is OnNavigatedFrom, a member of class SearchPage.
◆	OnNavigatedTo (see page 58)	This is OnNavigatedTo, a member of class SearchPage.

1.1.2.24.1 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;
    txtSearchWithMeaning.Text = AppResources.SearchInMeanings;
    lblSearch.Text = AppResources.Search;
    //btnSearch.Content = AppResources.Search;
    //lstSearch.SelectedIndex = -1;
}
```

1.1.2.24.2 SearchPage Methods

1.1.2.24.2.1 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);
}
```

1.1.2.24.2.2 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);
}
```

1.1.2.25 StatisticsPage Class

Class Hierarchy

PhoneApplicationPage
AwesomeDictionary.StatisticsPage

C#

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

File

StatisticsPage.xaml.cs (see page 144)

Description

This is class AwesomeDictionary.StatisticsPage.

Methods

	Name	Description
◆	StatisticsPage (see page 59)	This is StatisticsPage, a member of class StatisticsPage.

StatisticsPage Methods

	Name	Description
◆	OnNavigatedFrom (see page 59)	This is OnNavigatedFrom, a member of class StatisticsPage.
◆	OnNavigatedTo (see page 59)	This is OnNavigatedTo, a member of class StatisticsPage.

1.1.2.25.1 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();
}
```

1.1.2.25.2 StatisticsPage Methods

1.1.2.25.2.1 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();

}
```

1.1.2.25.2.2 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)
{
    base.OnNavigatedTo(e);
    SetStatistic();
}
```

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AboutPage.xaml.cs (see page 60)	This is file AboutPage.xaml.cs.
All.cs (see page 63)	This is file All.cs.
AlmancaTurkce.cs (see page 64)	This is file AlmancaTurkce.cs.
AlphaKeyGroup.cs (see page 64)	This is file AlphaKeyGroup.cs.
App.xaml.cs (see page 66)	This is file App.xaml.cs.
AppResources.Designer.cs (see page 73)	This code was generated by a tool. Runtime Version:4.0.30319.34014 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
AppSettings.cs (see page 87)	This is file AppSettings.cs.
AssemblyInfo.cs (see page 87)	This is file AssemblyInfo.cs.
AwesomeDictionary.csproj (see page 88)	This is file AwesomeDictionary.csproj.
AwesomeDictionary.sln (see page 88)	This is file AwesomeDictionary.sln.
AwesomeDictionaryDataContext.cs (see page 88)	This is file AwesomeDictionaryDataContext.cs.
BackgroundColorSettingsPage.xaml.cs (see page 89)	This is file BackgroundColorSettingsPage.xaml.cs.
BilisimSozlugu.cs (see page 92)	This is file BilisimSozlugu.cs.
BuyukLugat.cs (see page 93)	This is file BuyukLugat.cs.
EnglishTurkishVol1.cs (see page 94)	This is file EnglishTurkishVol1.cs.
EnglishTurkishVol2.cs (see page 94)	This is file EnglishTurkishVol2.cs.
Favourite.cs (see page 95)	This is file Favourite.cs.
FontFamilySettingsPage.xaml.cs (see page 96)	This is file FontFamilySettingsPage.xaml.cs.
FontSizeSettingsPage.xaml.cs (see page 98)	This is file FontSizeSettingsPage.xaml.cs.
GeneralSettingsPage.xaml.cs (see page 101)	This is file GeneralSettingsPage.xaml.cs.
KelimeAnlamlari.cs (see page 120)	This is file KelimeAnlamlari.cs.
LanguageSettingsPage.xaml.cs (see page 121)	This is file LanguageSettingsPage.xaml.cs.
LocalizedStrings.cs (see page 125)	This is file LocalizedStrings.cs.
MainPage.xaml.cs (see page 125)	This is file MainPage.xaml.cs.
NameDetailPage.xaml.cs (see page 129)	This is file NameDetailPage.xaml.cs.
OxfordEnglishEnglish.cs (see page 139)	This is file OxfordEnglishEnglish.cs.
RisaleNur.cs (see page 139)	This is file RisaleNur.cs.
SearchPage.xaml.cs (see page 140)	This is file SearchPage.xaml.cs.
StatisticsPage.xaml.cs (see page 144)	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;
```

```csharp
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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeDictionary
{
    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;
            txtAbout.Text = AppResources.AboutTheAppText;
            txtAbout.IsEnabled = false;
            //var paragraph = new Paragraph();
            //paragraph.Inlines.Add(AppResources.AboutTheAppText);
            //txtAbout.Blocks.Add(paragraph);
        }

        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.SendWithAwesomeDictionaryApp);
            emailComposeTask.Subject = AppResources.AboutAwesomeDictionary;
            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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.2 All.cs

This is file All.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;
using System.ComponentModel;
using Microsoft.Phone.Data.Linq;
// index özelligi asagidaki using sinifi ile etkili bir hale geliyor.
using Microsoft.Phone.Data.Linq.Mapping;

namespace AwesomeDictionary
{
    [Index(Columns = "AllName, AllMeaning, AllNameMeaning, AllNameSource ASC", IsUnique
= false, Name = "indAllNames")]
    [Table]
    public class All
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int AllId { get; set; }

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

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

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

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

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

    }
}
```

Namespaces

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

1.2.3 AlmancaTurkce.cs

This is file AlmancaTurkce.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 AwesomeDictionary
{
    [Table]
    public class AlmancaTurkce
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int AlmancaTurkceId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.4 AlphaKeyGroup.cs

This is file AlphaKeyGroup.cs.

Body Source

```
?using System.Collections.Generic;
using System.Globalization;
using Microsoft.Phone.Globalization;

namespace AwesomeDictionary
{
    public class AlphaKeyGroup<T> : List<T>
    {
        /// <summary>
        /// The delegate that is used to get the key information.
        /// </summary>
        /// <param name="item">An object of type T</param>
        /// <returns>The key value to use for this object</returns>
        public delegate string GetKeyDelegate(T item);

        /// <summary>
        /// The Key of this group.
        /// </summary>
        public string Key { get; private set; }

        /// <summary>
        /// Public constructor.
        /// </summary>
        /// <param name="key">The key for this group.</param>
        public AlphaKeyGroup(string key)
        {
            Key = key;
        }

        /// <summary>
        /// Create a list of AlphaGroup<T> with keys set by a SortedLocaleGrouping.
        /// </summary>
        /// <param name="slg">The </param>
        /// <returns>Theitems source for a LongListSelector</returns>
        private static List<AlphaKeyGroup<T>> CreateGroups(SortedLocaleGrouping slg)
        {
            List<AlphaKeyGroup<T>> list = new List<AlphaKeyGroup<T>>();

            foreach (string key in slg.GroupDisplayNames)
            {
                list.Add(new AlphaKeyGroup<T>(key));
            }

            return list;
        }

        /// <summary>
        /// Create a list of AlphaGroup<T> with keys set by a SortedLocaleGrouping.
        /// </summary>
        /// <param name="items">The items to place in the groups.</param>
        /// <param name="ci">The CultureInfo to group and sort by.</param>
        /// <param name="getKey">A delegate to get the key from an item.</param>
        /// <param name="sort">Will sort the data if true.</param>
        /// <returns>An items source for a LongListSelector</returns>
        public static List<AlphaKeyGroup<T>> CreateGroups(IEnumerable<T> items,
CultureInfo ci, GetKeyDelegate getKey, bool sort)
        {
            SortedLocaleGrouping slg = new SortedLocaleGrouping(ci);
            List<AlphaKeyGroup<T>> list = CreateGroups(slg);

            foreach (T item in items)
            {
                int index = 0;
                if (slg.SupportsPhonetics)
                {
                    //check if your database has yomi string for item
                    //if it does not, then do you want to generate Yomi or ask the user
for this item.
                    //index = slg.GetGroupIndex(getKey(Yomiof(item)));
                }
                else
                {
                    index = slg.GetGroupIndex(getKey(item));
                }
                if (index >= 0 && index < list.Count)
                {
```

1

```
                    list[index].Add(item);
                }
            }

            if (sort)
            {
                foreach (AlphaKeyGroup<T> group in list)
                {
                    group.Sort((c0, c1) => { return ci.CompareInfo.Compare(getKey(c0), getKey(c1)); });
                }
            }

            return list;
        }

    }
}
```

Namespaces

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

1.2.5 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.IO;
using System.Windows;
using System.Windows.Markup;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using System.Collections.Generic;
using Microsoft.Phone.Marketplace;
using AwesomeDictionary.Resources;

namespace AwesomeDictionary
{
    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>
        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;
            }
            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;
            }
        }

        public string ReadFile(string filePath)
        {
            var ResrouceStream = Application.GetResourceStream(new Uri(filePath, UriKind.Relative));
            if (ResrouceStream != null)
            {
                Stream myFileStream = ResrouceStream.Stream;
                if (myFileStream.CanRead)
                {
                    StreamReader myStreamReader = new StreamReader(myFileStream);

                    return myStreamReader.ReadToEnd();
                }
            }
            return "";
        }

        private static void DilAyariOlustur(AwesomeDictionaryDataContext context)
        {
            var appSettings = new AppSettings()
            {
                //AppLangId = 42,
                AppLangName = "EN",
                AppBackgroundColor = "BLA",
                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;
        }

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

                    // burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                    //var male = ReadFile(@"Sources\ErkekAdlari.txt");
                    //var maleLines = male.Split('\n').ToList() as List<string>;
                    //for (int i = 1; i < maleLines.Count; i++)
                    //{
                    //    var information = maleLines[i].Split('_').ToList() as List<string>;
                    //    Male maleName = new Male();
                    //    maleName.MaleName = information[0].ToUpper().TrimEnd().TrimStart();
                    //    maleName.MaleMeaning = information[1].TrimEnd().TrimStart();

                    // context.MaleNames.InsertOnSubmit(maleName);
                    //}

                    //var female = ReadFile(@"Sources\KizAdlari.txt");
                    //var femaleLines = female.Split('\n').ToList() as List<string>;
                    //for (int i = 1; i < femaleLines.Count; i++)
                    //{
                    // var information = femaleLines[i].Split('_').ToList() as List<string>;
                    // Female femaleName = new Female();
                    // femaleName.FemaleName = information[0].ToUpper().TrimEnd().TrimStart();
                    // femaleName.FemaleMeaning = information[1].TrimEnd().TrimStart();

                    // context.FemaleNames.InsertOnSubmit(femaleName);
                    //}

                    //context.SubmitChanges();

                    //var maleNamesTemp = context.MaleNames.ToList() as List<Male>;
                    //var femaleNamesTemp = context.FemaleNames.ToList() as List<Female>;

                    //for (int i = 0; i < maleNamesTemp.Count; i++)
                    //{
                    // for (int k = 0; k < femaleNamesTemp.Count; k++)
                    // {
                    //   if (maleNamesTemp[i].MaleName == femaleNamesTemp[k].FemaleName)
                    //   {
                    //     Unisex unisex = new Unisex();
                    //     unisex.UnisexName = maleNamesTemp[i].MaleName;
                    //     unisex.UnisexMeaning = maleNamesTemp[i].MaleMeaning;
                    //     context.UnisexNames.InsertOnSubmit(unisex);
                    //     var maleNameTemp = context.MaleNames.Where(j => j.MaleId.Equals(maleNamesTemp[i].MaleId)).SingleOrDefault() as Male;
                    //     context.MaleNames.DeleteOnSubmit(maleNameTemp);
                    //     var femaleNameTemp = context.FemaleNames.Where(j => j.FemaleId.Equals(femaleNamesTemp[k].FemaleId)).SingleOrDefault() as Female;
                    //     context.FemaleNames.DeleteOnSubmit(femaleNameTemp);
                    //     break;
                    //   }
                    // }
                    //}

                    //context.SubmitChanges();

                    //var males = context.MaleNames.ToList() as List<Male>;
                    //for (int i = 0; i < males.Count; i++)
                    //{
                    // All allName = new All();
                    // allName.AllName = males[i].MaleName;
                    // allName.AllMeaning = males[i].MaleMeaning;
                    // allName.AllNameMeaning = males[i].MaleName + " " + males[i].MaleMeaning;
                    // allName.AllGender = "Erkek";

                    // context.AllNames.InsertOnSubmit(allName);
                    //}

                    //var females = context.FemaleNames.ToList() as List<Female>;
                    //for (int i = 0; i < females.Count; i++)
                    //{
                    // All allName = new All();
                    // allName.AllName = females[i].FemaleName;
                    // allName.AllMeaning = females[i].FemaleMeaning;
                    // allName.AllNameMeaning = females[i].FemaleName + " " + females[i].FemaleMeaning;
```

```
                    //    allName.AllGender = "Kadin";

                    //    context.AllNames.InsertOnSubmit(allName);
                    //}

                    //var unisexes = context.UnisexNames.ToList() as List<Unisex>;
                    //for (int i = 0; i < unisexes.Count; i++)
                    //{
                    //    All allName = new All();
                    //    allName.AllName = unisexes[i].UnisexName;
                    //    allName.AllMeaning = unisexes[i].UnisexMeaning;
                    //    allName.AllNameMeaning = unisexes[i].UnisexName + " " + unisexes[i].UnisexMeaning;
                    //    allName.AllGender = "Erkek-Kadin";

                    //    context.AllNames.InsertOnSubmit(allName);
                    //}

                    //context.SubmitChanges();
                }
                else
                {
                    using (var context2 = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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.
                }
            }
        }
}
```

Namespaces

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

1.2.6 AppResources.Designer.cs

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

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

Body Source

```
?//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.34014
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AwesomeDictionary.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("AwesomeDictionary.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.EditorBrowsableState.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 Awesome Dictionary.
        /// </summary>
        public static string AboutAwesomeDictionary {
            get {
                return ResourceManager.GetString("AboutAwesomeDictionary", 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 everybody. I am with you with a new app. I investigate a lot of dictionary applications in Windows Phone and i tried to make a simple app which you will like it. If you rate the app and send your thoughts to coderserdar@outlook.com, I will be so appreciated to you. With my best regards. CoderSerdar.
        /// </summary>
        public static string AboutTheAppText {
            get {
                return ResourceManager.GetString("AboutTheAppText", resourceCulture);
            }
        }

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

        /// <summary>
        ///   Looks up a localized string similar to You Should Select At Least One Dictionary.
        /// </summary>
        public static string AtLeastOneDictionary {
            get {
                return ResourceManager.GetString("AtLeastOneDictionary", 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 BackgroundColorChangedSuccessfully {
            get {
                return ResourceManager.GetString("BackgroundColorChangedSuccessfully", 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 BackgroundImageChangedSuccessfully {
            get {
                return ResourceManager.GetString("BackgroundImageChangedSuccessfully", resourceCulture);
            }
        }

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

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

        /// <summary>
        ///   Looks up a localized string similar to Buyuk Lugat (Turkish-&gt;Turkish).
        /// </summary>
        public static string BuyukLugat {
            get {
                return ResourceManager.GetString("BuyukLugat", 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 Computer Dictionary (English-&gt;Turkish).
        /// </summary>
        public static string ComputerDictionary {
            get {
                return ResourceManager.GetString("ComputerDictionary", 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 Dictionaries Have Been Installed Successfully.
        /// </summary>
        public static string DictionariesInstalledSuccessfully {
            get {
                return ResourceManager.GetString("DictionariesInstalledSuccessfully", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Dictionaries Have Been Uninstalled Successfully.
        /// </summary>
        public static string DictionariesUninstalledSuccessfully {
            get {
                return ResourceManager.GetString("DictionariesUninstalledSuccessfully", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Dictionary Install.
        /// </summary>
        public static string DictionaryInstall {
            get {
                return ResourceManager.GetString("DictionaryInstall", 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 English-&gt;Turkish Dictionary Vol. 1.
        /// </summary>
        public static string EnglishTurkishVol1 {
            get {
                return ResourceManager.GetString("EnglishTurkishVol1", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to English-&gt;Turkish Dictionary Vol. 2.
        /// </summary>
        public static string EnglishTurkishVol2 {
            get {
                return ResourceManager.GetString("EnglishTurkishVol2", 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 Are You Sure To Exit The Application?.
        /// </summary>
        public static string ExitAppQuestion {
            get {
                return ResourceManager.GetString("ExitAppQuestion", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Favourite Word Count.
        /// </summary>
        public static string FavouriteNameCount {
            get {
                return ResourceManager.GetString("FavouriteNameCount", 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 FontFamilyChangedSuccessfully {
            get {
                return ResourceManager.GetString("FontFamilyChangedSuccessfully", 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 FontSizeChangedSuccessfully {
            get {
                return ResourceManager.GetString("FontSizeChangedSuccessfully", 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 German-&gt;Turkish Dictionary.
        /// </summary>
        public static string GermanTurkish {
            get {
                return ResourceManager.GetString("GermanTurkish", 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 Install.
        /// </summary>
        public static string Install {
            get {
                return ResourceManager.GetString("Install", resourceCulture);
            }
        }

        /// <summary>
        ///   Looks up a localized string similar to Installing.
        /// </summary>
        public static string Installing {
            get {
                return ResourceManager.GetString("Installing", 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 the application
for changes will be effect..
        /// </summary>
        public static string LanguageWarning {
            get {
                return ResourceManager.GetString("LanguageWarning", resourceCulture);
            }
        }

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

        /// <summary>
        ///   Looks up a localized string similar to My Favourites.
        /// </summary>
        public static string MyFavourites {
            get {
                return ResourceManager.GetString("MyFavourites", 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 Orange.
            /// </summary>
            public static string Orange {
                get {
                    return ResourceManager.GetString("Orange", resourceCulture);
                }
            }
            
            /// <summary>
            ///   Looks up a localized string similar to Oxford (English-&gt;English).
            /// </summary>
            public static string OxfordDictionary {
                get {
                    return ResourceManager.GetString("OxfordDictionary", 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 Random 10 Words.
            /// </summary>
            public static string RandomWords {
                get {
                    return ResourceManager.GetString("RandomWords", 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 Remove From Favourites.
        /// </summary>
        public static string RemoveFromFavourite {
            get {
                return ResourceManager.GetString("RemoveFromFavourite", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to You Will Remove The Word From Favourites. Are You Sure?.
        /// </summary>
        public static string RemoveFromFavouriteQuestion {
            get {
                return ResourceManager.GetString("RemoveFromFavouriteQuestion", resourceCulture);
            }
        }
        
        /// <summary>
        ///   Looks up a localized string similar to Remove From Favourites.
        /// </summary>
        public static string RemoveFromFavourites {
            get {
                return ResourceManager.GetString("RemoveFromFavourites", 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 Risale Nur (Turkish-&gt;Turkish).
        /// </summary>
        public static string RisaleNur {
            get {
                return ResourceManager.GetString("RisaleNur", 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 In Meanings.
        /// </summary>
        public static string SearchInMeanings {
            get {
                return ResourceManager.GetString("SearchInMeanings", 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 Search Criteria Can Not Be Empty.
        /// </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 Dictionary App.
        /// </summary>
        public static string SendWithAwesomeDictionaryApp {
            get {
                return ResourceManager.GetString("SendWithAwesomeDictionaryApp",
resourceCulture);
            }
        }

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

        /// <summary>
        ///   Looks up a localized string similar to Source.
        /// </summary>
        public static string Source {
            get {
                return ResourceManager.GetString("Source", 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 Synchronizing.
        /// </summary>
        public static string Synchronizing {
            get {
                return ResourceManager.GetString("Synchronizing", 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 Total Word Count.
        /// </summary>
        public static string TotalNameCount {
            get {
                return ResourceManager.GetString("TotalNameCount", 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 Uninstall.
        /// </summary>
        public static string Uninstall {
            get {
                return ResourceManager.GetString("Uninstall", resourceCulture);
            }
        }

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

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

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

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

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

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

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

```
873: }
```

Namespaces

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

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

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

        [Column]
        public string AppBackgroundColor { 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; }

        //[Column]
        //public string ActiveDictionary { get; set; }
    }
}
```

Namespaces

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

1.2.8 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 Dictionary")]
[assembly: AssemblyDescription("Best Dictionary App Ever")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("CoderSerdar")]
[assembly: AssemblyProduct("CoderSerdar")]
[assembly: AssemblyCopyright("Copyright ©  2015")]
[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("6e32f667-ec90-4f15-b44d-4bd7611c7ec4")]

// 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("1.0.0.1")]
[assembly: AssemblyFileVersion("1.0.0.1")]
[assembly: NeutralResourcesLanguageAttribute("en-US")]
```

1.2.9 AwesomeDictionary.csproj

This is file AwesomeDictionary.csproj.

1.2.10 AwesomeDictionary.sln

This is file AwesomeDictionary.sln.

1.2.11 AwesomeDictionaryDataContext.cs

This is file AwesomeDictionaryDataContext.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 AwesomeDictionary
{
    public class AwesomeDictionaryDataContext : DataContext
    {
        public const string ConnectionString = @"Data
Source=isostore:/AwesomeDictionary.sdf; Max Database Size=256; Max Buffer Size=4096;";
        public AwesomeDictionaryDataContext(string connectionString)
            : base(connectionString) { }
        public Table<AlmancaTurkce> AlmancaTurkces;
        public Table<RisaleNur> RisaleNurs;
        public Table<BilisimSozlugu> Bilisims;
        public Table<All> AllNames;
        public Table<Favourite> Favourites;
        public Table<EnglishTurkishVol1> EnglishTurkishVol1s;
        public Table<EnglishTurkishVol2> EnglishTurkishVol2s;
        public Table<BuyukLugat> BuyukLugats;
        public Table<OxfordEnglishEnglish> Oxfords;
        public Table<KelimeAnlamlari> Kelimes;
        public Table<AppSettings> AppSettings;
    }
}
```

Namespaces

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

1.2.12 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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeDictionary
{
    public partial class BackgroundColorSettingsPage : PhoneApplicationPage
    {
        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
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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);
        //    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
        //    {
        //        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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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.BackgroundColorChangedSuccessfully);
            }
            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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.13 BilisimSozlugu.cs

This is file BilisimSozlugu.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 AwesomeDictionary
{
    [Table]
    public class BilisimSozlugu
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int BilisimSozluguId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.14 BuyukLugat.cs

This is file BuyukLugat.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 AwesomeDictionary
{
    [Table]
    public class BuyukLugat
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int BuyukLugatId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.15 EnglishTurkishVol1.cs

This is file EnglishTurkishVol1.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 AwesomeDictionary
{
    [Table]
    public class EnglishTurkishVol1
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int EnglishVol1Id { get; set; }

        [Column]
        public string EnglishVol1Name { get; set; }

        [Column]
        public string EnglishVol1Meaning { get; set; }

        [Column]
        public string EnglishVol1NameMeaning { get; set; }

    }
}
```

Namespaces

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

1.2.16 EnglishTurkishVol2.cs

This is file EnglishTurkishVol2.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 AwesomeDictionary
{
    [Table]
    public class EnglishTurkishVol2
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int EnglishVol2Id { get; set; }

        [Column]
        public string EnglishVol2Name { get; set; }

        [Column]
        public string EnglishVol2Meaning { get; set; }

        [Column]
        public string EnglishVol2NameMeaning { get; set; }

    }
}
```

Namespaces

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

1.2.17 Favourite.cs

This is file Favourite.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 AwesomeDictionary
{
    [Table]
    public class Favourite
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int FavouriteId { get; set; }

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

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

Namespaces

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

1.2.18 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 AwesomeDictionary.Resources;

namespace AwesomeDictionary
{
    public partial class FontFamilySettingsPage : PhoneApplicationPage
    {
        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("Albumman 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;

            lblFontFamily.Text = AppResources.SelectFontFamily;
            lblGeneralSettings.Text = AppResources.GeneralSettings;

            SetBackgroundColor();
        }

        private void lstFontFamily_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (lstFontFamily.SelectedIndex != -1)
            {
                using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var item in appSettings)
                    {
                        item.FontFamily = lstFontFamily.SelectedItem.ToString();
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.FontFamilyChangedSuccessfully);
                }
            }
            NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", 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);
            //artistId = int.Parse(e.Fragment);
            //using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            //{
            //    var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single() as Artist;
            //    lblArtistName.Text = artist.ArtistName;
            //    lblFontFamily.Text = AppResources.SelectFontFamily;
            //}

        }

        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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.19 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 AwesomeDictionary.Resources;

namespace AwesomeDictionary
{
    public partial class FontSizeSettingsPage : PhoneApplicationPage
    {
        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;

            lblGeneralSettings.Text = AppResources.GeneralSettings;
            lblFontSize.Text = AppResources.SelectFontSize;

            SetBackgroundColor();
        }

        private void lstFontSize_SelectionChanged(object sender, 
SelectionChangedEventArgs e)
        {
            if (lstFontSize.SelectedIndex != -1)
            {
                using (var context = new 
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                {
                    var appSettings = context.AppSettings;
                    foreach (var item in appSettings)
                    {
                        item.FontSize = lstFontSize.SelectedItem.ToString();
                    }
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.FontSizeChangedSuccessfully);
                }
            }
            NavigationService.Navigate(new Uri("/GeneralSettingsPage.xaml", 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);
            //artistId = int.Parse(e.Fragment);
            //using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            //{
            //    var artist = context.Artists.Where(j => j.ArtistId.Equals(artistId)).Single() as Artist;
            //    lblArtistName.Text = artist.ArtistName;
            //    lblFontSize.Text = AppResources.SelectFontSize;
            //}

        }

        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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.20 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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Phone.Tasks;

namespace AwesomeDictionary
```

```
{
    public partial class GeneralSettingsPage : PhoneApplicationPage
    {

        public GeneralSettingsPage()
        {
            InitializeComponent();
            SetBackgroundColor();

            pvGeneralSettings.Title = AppResources.GeneralSettings;

            piLanguage.Header = AppResources.Language;
            piDictionaryInstall.Header = AppResources.DictionaryInstall;
            //piOtherSettings.Header = AppResources.OtherSettings;
            piBackground.Header = AppResources.Background;

            //lblOneDrive.Text = AppResources.OneDrive;

            btnLanguage.Content = AppResources.Select;
            btnBackgroundColor.Content = AppResources.Select;
            //btnOneDrive.Content = AppResources.Login;
            //btnOneDrive.SignInText = AppResources.SignIn;
            //btnOneDrive.SignOutText = AppResources.SignOut;
            txtInstalling.Text = AppResources.Installing;

            pbInstall.Visibility = Visibility.Collapsed;
            txtInstalling.Visibility = Visibility.Collapsed;
            txtInstalling.BorderBrush = this.LayoutRoot.Background;

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

            piFont.Header = AppResources.Font;
            btnFontFamily.Content = AppResources.Select;
            btnFontSize.Content = AppResources.Select;

            btnInstall.Content = AppResources.Install;
            btnUninstall.Content = AppResources.Uninstall;

            txtBuyukLugat.Text = AppResources.BuyukLugat + " (49331 " +
AppResources.Word + ")";
            txtComputer.Text = AppResources.ComputerDictionary + " (3508 " +
AppResources.Word + ")";
            txtGerman.Text = AppResources.GermanTurkish + " (17526 " +
AppResources.Word + ")";
            txtOxford.Text = AppResources.OxfordDictionary + " (36369 " +
AppResources.Word + ")";
            txtWordMeaning.Text = AppResources.WordMeaning + " (10535 " +
AppResources.Word + ")";
            txtRisaleNur.Text = AppResources.RisaleNur + " (9478 " +
AppResources.Word + ")";
            txtEnglishVol1.Text = AppResources.EnglishTurkishVol1 + " (127157 " +
AppResources.Word + ")";
            txtEnglishVol2.Text = AppResources.EnglishTurkishVol2 + " (3699 " +
AppResources.Word + ")";


            //cbSync.Content = AppResources.SyncOnOneFile;
            //cbSync.IsEnabled = false;

            SetBackgroundColor();

            using (var context = new
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings.First() as AppSettings;

```

```
                      lblFontFamily.Text = AppResources.FontFamily + " (" +
AppResources.Selected + ": " + appSettings.FontFamily + ")";
                      lblFontSize.Text = AppResources.FontSize + " (" +
AppResources.Selected + ": " + appSettings.FontSize + ")";

                      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.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 void SetBackgroundColor()
              {
                  AppSettings appSettings = new AppSettings();
                  using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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 btnLanguage_Click(object sender, RoutedEventArgs e)
              {
                  this.NavigationService.Navigate(new Uri("/LanguageSettingsPage.xaml", UriKind.Relative));
              }

              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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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.BackgroundImageChangedSuccessfully);
                          }
                          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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                  {
                      var appSettings = context.AppSettings;
                      foreach (var appSetting in appSettings)
                      {
                          appSetting.AppBackgroundImage = null;
                      }
                      context.SubmitChanges();
                      MessageBox.Show(AppResources.BackgroundImageRemovedSuccessfully);
                  }
              }

              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;
                  //txtInstalling.Text = AppResources.Synchronizing;

                  //pbSync.Visibility = Visibility.Collapsed;
                  //txtInstalling.Visibility = Visibility.Collapsed;
                  //txtInstalling.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 btnResetSettings_Click(object sender, RoutedEventArgs e)
              {
                  using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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.BackgroundSettingsResetSuccessfully);
                  }
              }

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

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

              // burasi düzenlenecek
              private void btnInstall_Click(object sender, RoutedEventArgs e)
              {
                  try
                  {

                      if (cbBuyukLugat.IsChecked == false && cbComputer.IsChecked == false && cbEnglishVol1.IsChecked == false && cbEnglishVol2.IsChecked == false && cbOxford.IsChecked == false && cbRisaleNur.IsChecked == false && cbWordMeaning.IsChecked == false && cbGerman.IsChecked == false)
                      {
                          MessageBox.Show(AppResources.AtLeastOneDictionary);
                      }
                      else
                      {
                          using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                          {
                              SozlukYukle(context);
                          }
                      }
                  }
                  catch (Exception ex)
                  {
                      MessageBox.Show(AppResources.SystemFault);
                  }
              }

              private void SozlukYukle2(AwesomeDictionaryDataContext context)
              {
                  if (cbBuyukLugat.IsChecked == true)
                  {
                      if (context.BuyukLugats.ToList().Count > 0)
                      {
                          var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.BuyukLugat)).ToList() as List<All>;
                          context.AllNames.DeleteAllOnSubmit(allList);
                          context.BuyukLugats.DeleteAllOnSubmit(context.BuyukLugats.ToList());
                      }
                      //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                      var buyukLugat = ReadFile(@"Sources\BuyukLugatEsas.txt");
                      var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                      for (int i = 1; i < buyukLugatLines.Count; i++)
                      {
                          var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                          BuyukLugat buyukLugatWord = new BuyukLugat();
                          buyukLugatWord.BuyukLugatName = information[0].ToUpper().TrimEnd().TrimStart();
                          buyukLugatWord.BuyukLugatMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                          buyukLugatWord.BuyukLugatNameMeaning = buyukLugatWord.BuyukLugatName + " " + buyukLugatWord.BuyukLugatNameMeaning;
                          context.BuyukLugats.InsertOnSubmit(buyukLugatWord);
                      }
                      context.SubmitChanges();
                      var buyukLugats = context.BuyukLugats.ToList() as List<BuyukLugat>;
                      for (int i = 0; i < buyukLugats.Count; i++)
                      {
                          All allName = new All();
                          allName.AllName = buyukLugats[i].BuyukLugatName;
                          allName.AllMeaning = buyukLugats[i].BuyukLugatMeaning;
                          allName.AllNameMeaning = buyukLugats[i].BuyukLugatName + " " + buyukLugats[i].BuyukLugatMeaning;
                          allName.AllSource = AppResources.BuyukLugat;
                          allName.AllNameSource = allName.AllName + " (" + AppResources.BuyukLugat + ")";
                          context.AllNames.InsertOnSubmit(allName);
                      }
                      context.SubmitChanges();
                  }

                  if (cbGerman.IsChecked == true)
                  {
                      if (context.AlmancaTurkces.ToList().Count > 0)
                      {
                          var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.GermanTurkish)).ToList() as List<All>;
                          context.AllNames.DeleteAllOnSubmit(allList);
                          context.AlmancaTurkces.DeleteAllOnSubmit(context.AlmancaTurkces.ToList());
                      }
                      //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                      var almancaTurkce = ReadFile(@"Sources\AlmancaTurkceSozluk.txt");
                      var almancaTurkceLines = almancaTurkce.Split('\n').ToList() as List<string>;
                      for (int i = 1; i < almancaTurkceLines.Count; i++)
                      {
                          var information = almancaTurkceLines[i].Split('_').ToList() as List<string>;
                          AlmancaTurkce almancaTurkceWord = new AlmancaTurkce();
                          almancaTurkceWord.AlmancaTurkceName = information[0].ToUpper().TrimEnd().TrimStart();
                          almancaTurkceWord.AlmancaTurkceMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                          almancaTurkceWord.AlmancaTurkceNameMeaning = almancaTurkceWord.AlmancaTurkceName + " " + almancaTurkceWord.AlmancaTurkceNameMeaning;
                          context.AlmancaTurkces.InsertOnSubmit(almancaTurkceWord);
                      }
                      context.SubmitChanges();
                      var almancaTurkces = context.AlmancaTurkces.ToList() as List<AlmancaTurkce>;
                      for (int i = 0; i < almancaTurkces.Count; i++)
                      {
                          All allName = new All();
                          allName.AllName = almancaTurkces[i].AlmancaTurkceName;
                          allName.AllMeaning = almancaTurkces[i].AlmancaTurkceMeaning;
                          allName.AllNameMeaning = almancaTurkces[i].AlmancaTurkceName + " " + almancaTurkces[i].AlmancaTurkceMeaning;
                          allName.AllSource = AppResources.GermanTurkish;
                          allName.AllNameSource = allName.AllName + " (" + AppResources.GermanTurkish + ")";
                          context.AllNames.InsertOnSubmit(allName);
                      }
                      context.SubmitChanges();
                  }

                  if (cbComputer.IsChecked == true)
                  {
                      if (context.Bilisims.ToList().Count > 0)
                      {
                          var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.ComputerDictionary)).ToList() as List<All>;
                          context.AllNames.DeleteAllOnSubmit(allList);
                          context.Bilisims.DeleteAllOnSubmit(context.Bilisims.ToList());
                      }
                      //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme
```

```
yapiliyor
                var bilisimSozlugu = ReadFile(@"Sources\BilisimSozlugu.txt");
                var bilisimSozluguLines = bilisimSozlugu.Split('\n').ToList() as List<string>;
                for (int i = 1; i < bilisimSozluguLines.Count; i++)
                {
                    var information = bilisimSozluguLines[i].Split('_').ToList() as List<string>;
                    BilisimSozlugu bilisimSozluguWord = new BilisimSozlugu();
                    bilisimSozluguWord.BilisimSozluguName = information[0].ToUpper().TrimEnd().TrimStart();
                    bilisimSozluguWord.BilisimSozluguMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    bilisimSozluguWord.BilisimSozluguNameMeaning = bilisimSozluguWord.BilisimSozluguName + " " + bilisimSozluguWord.BilisimSozluguNameMeaning;
                    context.Bilisims.InsertOnSubmit(bilisimSozluguWord);
                }
                context.SubmitChanges();
                var bilisimSozlugus = context.Bilisims.ToList() as List<BilisimSozlugu>;
                for (int i = 0; i < bilisimSozlugus.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = bilisimSozlugus[i].BilisimSozluguName;
                    allName.AllMeaning = bilisimSozlugus[i].BilisimSozluguMeaning;
                    allName.AllNameMeaning = bilisimSozlugus[i].BilisimSozluguName + " " + bilisimSozlugus[i].BilisimSozluguMeaning;
                    allName.AllSource = AppResources.ComputerDictionary;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.ComputerDictionary + ")";
                    context.AllNames.InsertOnSubmit(allName);
                }
                context.SubmitChanges();
            }

            if (cbOxford.IsChecked == true)
            {
                if (context.Oxfords.ToList().Count > 0)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.OxfordDictionary)).ToList() as List<All>;
                    context.AllNames.DeleteAllOnSubmit(allList);
                    context.Oxfords.DeleteAllOnSubmit(context.Oxfords.ToList());
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var oxfordEnglishEnglish = ReadFile(@"Sources\OxfordEnglishTurkish.txt");
                var oxfordEnglishEnglishLines = oxfordEnglishEnglish.Split('\n').ToList() as List<string>;
                for (int i = 1; i < oxfordEnglishEnglishLines.Count; i++)
                {
                    var information = oxfordEnglishEnglishLines[i].Split('_').ToList() as List<string>;
                    OxfordEnglishEnglish oxfordEnglishEnglishWord = new OxfordEnglishEnglish();
                    oxfordEnglishEnglishWord.OxfordName = information[0].ToUpper().TrimEnd().TrimStart();
                    oxfordEnglishEnglishWord.OxfordMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    oxfordEnglishEnglishWord.OxfordNameMeaning = oxfordEnglishEnglishWord.OxfordName + " " + oxfordEnglishEnglishWord.OxfordMeaning;
                    context.Oxfords.InsertOnSubmit(oxfordEnglishEnglishWord);
                }
                context.SubmitChanges();
                var oxfordEnglishEnglishs = context.Oxfords.ToList() as List<OxfordEnglishEnglish>;
                for (int i = 0; i < oxfordEnglishEnglishs.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = oxfordEnglishEnglishs[i].OxfordName;
                    allName.AllMeaning = oxfordEnglishEnglishs[i].OxfordMeaning;
                    allName.AllNameMeaning = oxfordEnglishEnglishs[i].OxfordName + " " + oxfordEnglishEnglishs[i].OxfordMeaning;
                    allName.AllSource = AppResources.OxfordDictionary;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.OxfordDictionary + ")";
                    context.AllNames.InsertOnSubmit(allName);
                }
                context.SubmitChanges();
            }

            if (cbRisaleNur.IsChecked == true)
            {
                if (context.RisaleNurs.ToList().Count > 0)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.RisaleNur)).ToList() as List<All>;
                    context.AllNames.DeleteAllOnSubmit(allList);
                    context.RisaleNurs.DeleteAllOnSubmit(context.RisaleNurs.ToList());
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var risaleNur = ReadFile(@"Sources\RisaleNurSozlugu.txt");
                var risaleNurLines = risaleNur.Split('\n').ToList() as List<string>;
                for (int i = 1; i < risaleNurLines.Count; i++)
                {
                    var information = risaleNurLines[i].Split('_').ToList() as List<string>;
                    RisaleNur risaleNurWord = new RisaleNur();
                    risaleNurWord.RisaleNurName = information[0].ToUpper().TrimEnd().TrimStart();
                    risaleNurWord.RisaleNurMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    risaleNurWord.RisaleNurNameMeaning = risaleNurWord.RisaleNurName + " " + risaleNurWord.RisaleNurNameMeaning;
                    context.RisaleNurs.InsertOnSubmit(risaleNurWord);
                }
                context.SubmitChanges();
                var risaleNurs = context.RisaleNurs.ToList() as List<RisaleNur>;
                for (int i = 0; i < risaleNurs.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = risaleNurs[i].RisaleNurName;
                    allName.AllMeaning = risaleNurs[i].RisaleNurMeaning;
                    allName.AllNameMeaning = risaleNurs[i].RisaleNurName + " " + risaleNurs[i].RisaleNurMeaning;
                    allName.AllSource = AppResources.RisaleNur;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.RisaleNur + ")";
                    context.AllNames.InsertOnSubmit(allName);

                }
                context.SubmitChanges();
            }

            if (cbEnglishVol1.IsChecked == true)
            {
                if (context.EnglishTurkishVol1s.ToList().Count > 0)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.EnglishTurkishVol1)).ToList() as List<All>;
                    context.AllNames.DeleteAllOnSubmit(allList);
                    context.EnglishTurkishVol1s.DeleteAllOnSubmit(context.EnglishTurkishVol1s.ToList());
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var englishTurkishVol1 = ReadFile(@"Sources\IngilizceTurkceSozlukv1.txt");
                var englishTurkishVol1Lines = englishTurkishVol1.Split('\n').ToList() as List<string>;
                for (int i = 1; i < englishTurkishVol1Lines.Count; i++)
                {
                    var information = englishTurkishVol1Lines[i].Split('_').ToList() as List<string>;
                    EnglishTurkishVol1 englishTurkishVol1Word = new EnglishTurkishVol1();
                    englishTurkishVol1Word.EnglishVol1Name = information[0].ToUpper().TrimEnd().TrimStart();
                    englishTurkishVol1Word.EnglishVol1Meaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    englishTurkishVol1Word.EnglishVol1NameMeaning = englishTurkishVol1Word.EnglishVol1Name + " " + englishTurkishVol1Word.EnglishVol1NameMeaning;
context.EnglishTurkishVol1s.InsertOnSubmit(englishTurkishVol1Word);
                }
                context.SubmitChanges();
                var englishTurkishVol1s = context.EnglishTurkishVol1s.ToList() as List<EnglishTurkishVol1>;
                for (int i = 0; i < englishTurkishVol1s.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = englishTurkishVol1s[i].EnglishVol1Name;
                    allName.AllMeaning = englishTurkishVol1s[i].EnglishVol1Meaning;
                    allName.AllNameMeaning = englishTurkishVol1s[i].EnglishVol1Name + " " + englishTurkishVol1s[i].EnglishVol1NameMeaning;
                    allName.AllSource = AppResources.EnglishTurkishVol1;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.EnglishTurkishVol1 + ")";
                    context.AllNames.InsertOnSubmit(allName);
                }
                context.SubmitChanges();
            }

            if (cbEnglishVol2.IsChecked == true)
            {
                if (context.EnglishTurkishVol2s.ToList().Count > 0)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.EnglishTurkishVol2)).ToList() as List<All>;
                    context.AllNames.DeleteAllOnSubmit(allList);
context.EnglishTurkishVol2s.DeleteAllOnSubmit(context.EnglishTurkishVol2s.ToList());
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var englishTurkishVol2 = ReadFile(@"Sources\IngilizceTurkceSozlukv2.txt");
                var englishTurkishVol2Lines = englishTurkishVol2.Split('\n').ToList() as List<string>;
                for (int i = 1; i < englishTurkishVol2Lines.Count; i++)
                {
                    var information = englishTurkishVol2Lines[i].Split('_').ToList() as List<string>;
                    EnglishTurkishVol2 englishTurkishVol2Word = new EnglishTurkishVol2();
                    englishTurkishVol2Word.EnglishVol2Name = information[0].ToUpper().TrimEnd().TrimStart();
                    englishTurkishVol2Word.EnglishVol2Meaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    englishTurkishVol2Word.EnglishVol2NameMeaning = englishTurkishVol2Word.EnglishVol2Name + " " + englishTurkishVol2Word.EnglishVol2NameMeaning;
context.EnglishTurkishVol2s.InsertOnSubmit(englishTurkishVol2Word);
```

```
                }
                context.SubmitChanges();
                var englishTurkishVol2s = context.EnglishTurkishVol2s.ToList() as List<EnglishTurkishVol2>;
                for (int i = 0; i < englishTurkishVol2s.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = englishTurkishVol2s[i].EnglishVol2Name;
                    allName.AllMeaning = englishTurkishVol2s[i].EnglishVol2Meaning;
                    allName.AllNameMeaning = englishTurkishVol2s[i].EnglishVol2Name + " " + englishTurkishVol2s[i].EnglishVol2NameMeaning;
                    allName.AllSource = AppResources.EnglishTurkishVol2;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.EnglishTurkishVol2 + ")";
                    context.AllNames.InsertOnSubmit(allName);
                }
                context.SubmitChanges();
            }

            if (cbWordMeaning.IsChecked == true)
            {
                if (context.Kelimes.ToList().Count > 0)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.WordMeaning)).ToList() as List<All>;
                    context.AllNames.DeleteAllOnSubmit(allList);
                    context.Kelimes.DeleteAllOnSubmit(context.Kelimes.ToList());
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var kelimeAnlamlari = ReadFile(@"Sources\KelimeAnlamlari.txt");
                var kelimeAnlamlariLines = kelimeAnlamlari.Split('\n').ToList() as List<string>;
                for (int i = 1; i < kelimeAnlamlariLines.Count; i++)
                {
                    var information = kelimeAnlamlariLines[i].Split('_').ToList() as List<string>;
                    KelimeAnlamlari kelimeAnlamlariWord = new KelimeAnlamlari();
                    kelimeAnlamlariWord.KelimeAnlamlariName = information[0].ToUpper().TrimEnd().TrimStart();
                    kelimeAnlamlariWord.KelimeAnlamlariMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    kelimeAnlamlariWord.KelimeAnlamlariNameMeaning = kelimeAnlamlariWord.KelimeAnlamlariName + " " + kelimeAnlamlariWord.KelimeAnlamlariNameMeaning;
                    context.Kelimes.InsertOnSubmit(kelimeAnlamlariWord);
                }
                context.SubmitChanges();
                var kelimeAnlamlaris = context.Kelimes.ToList() as List<KelimeAnlamlari>;
                for (int i = 0; i < kelimeAnlamlaris.Count; i++)
                {
                    All allName = new All();
                    allName.AllName = kelimeAnlamlaris[i].KelimeAnlamlariName;
                    allName.AllMeaning = kelimeAnlamlaris[i].KelimeAnlamlariMeaning;
                    allName.AllNameMeaning = kelimeAnlamlaris[i].KelimeAnlamlariName + " " + kelimeAnlamlaris[i].KelimeAnlamlariNameMeaning;
                    allName.AllSource = AppResources.WordMeaning;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.WordMeaning + ")";
                    context.AllNames.InsertOnSubmit(allName);
                }
                context.SubmitChanges();
            }

            MessageBox.Show(AppResources.DictionariesInstalledSuccessfully);
        }

        private void SozlukYukle(AwesomeDictionaryDataContext context)
```

```
            {
                //pbInstall.Visibility = Visibility.Visible;

                if (cbBuyukLugat.IsChecked == true)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.BuyukLugat)).ToList() as List<All>;
                    if (allList != null)
                    {
                        context.AllNames.DeleteAllOnSubmit(allList);
                    }
                    //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                    var buyukLugat = ReadFile(@"Sources\BuyukLugatEsas.txt");
                    var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                    string text = AppResources.Installing + " " + AppResources.BuyukLugat;
                    //pbInstall.Value = 0;
                    List<All> allNames = new List<All>();
                    for (int i = 1; i < buyukLugatLines.Count; i++)
                    {
                        //pbInstall.Value = i;
                        var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                        All allName = new All();
                        allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                        allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                        allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                        allName.AllSource = AppResources.BuyukLugat;
                        allName.AllNameSource = allName.AllName + " (" + AppResources.BuyukLugat + ")";
                        allNames.Add(allName);
                        //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                    }
                    context.AllNames.InsertAllOnSubmit(allNames);
                }

                if (cbGerman.IsChecked == true)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.GermanTurkish)).ToList() as List<All>;
                    if (allList != null)
                    {
                        context.AllNames.DeleteAllOnSubmit(allList);
                    }
                    //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                    var buyukLugat = ReadFile(@"Sources\AlmancaTurkceSozluk.txt");
                    var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                    string text = AppResources.Installing + " " + AppResources.GermanTurkish;
                    //pbInstall.Value = 0;
                    List<All> allNames = new List<All>();
                    for (int i = 1; i < buyukLugatLines.Count; i++)
                    {
                        //pbInstall.Value = i;
                        var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                        All allName = new All();
                        allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                        allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                        allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                        allName.AllSource = AppResources.GermanTurkish;
                        allName.AllNameSource = allName.AllName + " (" + AppResources.GermanTurkish + ")";
                        allNames.Add(allName);
                        //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                    }
                    context.AllNames.InsertAllOnSubmit(allNames);

                }

                if (cbComputer.IsChecked == true)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.ComputerDictionary)).ToList() as List<All>;
                    if (allList != null)
                    {
                        context.AllNames.DeleteAllOnSubmit(allList);
                    }
                    //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                    var buyukLugat = ReadFile(@"Sources\BilisimSozlugu.txt");
                    var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                    string text = AppResources.Installing + " " + AppResources.ComputerDictionary;
                    //pbInstall.Value = 0;
                    List<All> allNames = new List<All>();
                    for (int i = 1; i < buyukLugatLines.Count; i++)
                    {
                        //pbInstall.Value = i;
                        var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                        All allName = new All();
                        allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                        allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                        allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                        allName.AllSource = AppResources.ComputerDictionary;
                        allName.AllNameSource = allName.AllName + " (" + AppResources.ComputerDictionary + ")";
                        allNames.Add(allName);
                        //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                    }
                    context.AllNames.InsertAllOnSubmit(allNames);
                }

                if (cbOxford.IsChecked == true)
                {
                    var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.OxfordDictionary)).ToList() as List<All>;
                    if (allList != null)
                    {
                        context.AllNames.DeleteAllOnSubmit(allList);
                    }
                    //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                    var buyukLugat = ReadFile(@"Sources\OxfordEnglishEnglish.txt");
                    var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                    string text = AppResources.Installing + " " + AppResources.OxfordDictionary;
                    //pbInstall.Value = 0;
                    List<All> allNames = new List<All>();
                    for (int i = 1; i < buyukLugatLines.Count; i++)
                    {
                        //pbInstall.Value = i;
                        var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                        All allName = new All();
                        allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                        allName.AllMeaning = information[1].TrimEnd().TrimStart() == null
```

```
? "" : information[1].TrimEnd().TrimStart();
                    allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                    allName.AllSource = AppResources.OxfordDictionary;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.OxfordDictionary + ")";
                    allNames.Add(allName);
                    //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                }
                context.AllNames.InsertAllOnSubmit(allNames);
            }

            if (cbRisaleNur.IsChecked == true)
            {
                var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.RisaleNur)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var buyukLugat = ReadFile(@"Sources\RisaleNurSozlugu.txt");
                var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                string text = AppResources.Installing + " " + AppResources.RisaleNur;
                //pbInstall.Value = 0;
                List<All> allNames = new List<All>();
                for (int i = 1; i < buyukLugatLines.Count; i++)
                {
                    //pbInstall.Value = i;
                    var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                    All allName = new All();
                    allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                    allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                    allName.AllSource = AppResources.RisaleNur;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.RisaleNur + ")";
                    allNames.Add(allName);
                    //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                }
                context.AllNames.InsertAllOnSubmit(allNames);
            }

            if (cbEnglishVol1.IsChecked == true)
            {
                var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.EnglishTurkishVol1)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var buyukLugat = ReadFile(@"Sources\IngilizceTurkceSozlukv1.txt");
                var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                string text = AppResources.Installing + " " + AppResources.EnglishTurkishVol1;
                //pbInstall.Value = 0;
                List<All> allNames = new List<All>();
                for (int i = 1; i < buyukLugatLines.Count; i++)
                {
                    //pbInstall.Value = i;
                    var information = buyukLugatLines[i].Split('_').ToList() as
```

```
List<string>;
                    All allName = new All();
                    allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                    allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                    allName.AllSource = AppResources.EnglishTurkishVol1;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.EnglishTurkishVol1 + ")";
                    allNames.Add(allName);
                    //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                }
                context.AllNames.InsertAllOnSubmit(allNames);
            }

            if (cbEnglishVol2.IsChecked == true)
            {
                var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.EnglishTurkishVol2)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var buyukLugat = ReadFile(@"Sources\IngilizceTurkceSozlukv2.txt");
                var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                string text = AppResources.Installing + " " + AppResources.EnglishTurkishVol2;
                //pbInstall.Value = 0;
                List<All> allNames = new List<All>();
                for (int i = 1; i < buyukLugatLines.Count; i++)
                {
                    //pbInstall.Value = i;
                    var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                    All allName = new All();
                    allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                    allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                    allName.AllSource = AppResources.EnglishTurkishVol2;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.EnglishTurkishVol2 + ")";
                    allNames.Add(allName);
                    //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                }
                context.AllNames.InsertAllOnSubmit(allNames);
            }

            if (cbWordMeaning.IsChecked == true)
            {
                var allList = context.AllNames.Where(j => j.AllSource.Equals(AppResources.WordMeaning)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                //burada metin belgeleri okunuyor ve veritabanindaki tablolara ekleme yapiliyor
                var buyukLugat = ReadFile(@"Sources\KelimeAnlamlari.txt");
                var buyukLugatLines = buyukLugat.Split('\n').ToList() as List<string>;
                string text = AppResources.Installing + " " + AppResources.WordMeaning;
                //pbInstall.Value = 0;
                List<All> allNames = new List<All>();
                for (int i = 1; i < buyukLugatLines.Count; i++)
                {
                    //pbInstall.Value = i;
                    var information = buyukLugatLines[i].Split('_').ToList() as List<string>;
                    All allName = new All();
                    allName.AllName = information[0].ToUpper().TrimEnd().TrimStart();
                    allName.AllMeaning = information[1].TrimEnd().TrimStart() == null ? "" : information[1].TrimEnd().TrimStart();
                    allName.AllNameMeaning = allName.AllName + " " + allName.AllNameMeaning;
                    allName.AllSource = AppResources.WordMeaning;
                    allName.AllNameSource = allName.AllName + " (" + AppResources.WordMeaning + ")";
                    allNames.Add(allName);
                    //txtInstalling.Text = text + " (" + i + "/" + buyukLugatLines.Count + ")";
                }
                context.AllNames.InsertAllOnSubmit(allNames);
            }
            context.SubmitChanges();
            //pbInstall.Visibility = Visibility.Collapsed;

            MessageBox.Show(AppResources.DictionariesInstalledSuccessfully);
        }

        public string ReadFile(string filePath)
        {
            var ResrouceStream = Application.GetResourceStream(new Uri(filePath, UriKind.Relative));
            if (ResrouceStream != null)
            {
                Stream myFileStream = ResrouceStream.Stream;
                if (myFileStream.CanRead)
                {
                    StreamReader myStreamReader = new StreamReader(myFileStream);

                    return myStreamReader.ReadToEnd();
                }
            }
            return "";
        }

        private void btnUninstall_Click(object sender, RoutedEventArgs e)
        {
            try
            {

                if (cbBuyukLugat.IsChecked == false && cbComputer.IsChecked == false && cbEnglishVol1.IsChecked == false && cbEnglishVol2.IsChecked == false && cbOxford.IsChecked == false && cbRisaleNur.IsChecked == false && cbWordMeaning.IsChecked == false && cbGerman.IsChecked == false)
                {
                    MessageBox.Show(AppResources.AtLeastOneDictionary);
                }
                else
                {
                    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                    {
                        SozlukKaldir(context);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(AppResources.SystemFault);
            }
```

1

```
        }

        private void SozlukKaldir(AwesomeDictionaryDataContext context)
        {
            //pbInstall.Visibility = Visibility.Visible;

            if (cbBuyukLugat.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.BuyukLugat)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }

            if (cbGerman.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.GermanTurkish)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }

            if (cbComputer.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.ComputerDictionary)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            if (cbOxford.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.OxfordDictionary)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            if (cbRisaleNur.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.RisaleNur)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            if (cbEnglishVol1.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.EnglishTurkishVol1)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            if (cbEnglishVol2.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.EnglishTurkishVol2)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            if (cbWordMeaning.IsChecked == true)
            {
                var allList = context.AllNames.Where(j =>
j.AllSource.Equals(AppResources.WordMeaning)).ToList() as List<All>;
                if (allList != null)
                {
                    context.AllNames.DeleteAllOnSubmit(allList);
                }
                context.SubmitChanges();
            }
            //context.SubmitChanges();

            MessageBox.Show(AppResources.DictionariesUninstalledSuccessfully);
        }
    }
}
```

Namespaces

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

1.2.21 KelimeAnlamlari.cs

This is file KelimeAnlamlari.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 AwesomeDictionary
{
    [Table]
    public class KelimeAnlamlari
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int KelimeAnlamlariId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.22 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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeDictionary
{
    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.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 AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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 = "sa";
                lang = "SA";
            }
            else if (index == 11)
            {
                    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
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.23 LocalizedStrings.cs

This is file LocalizedStrings.cs.

Body Source

```
?using AwesomeDictionary.Resources;

namespace AwesomeDictionary
{
    /// <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
AwesomeDictionary (see page 1)	This is namespace AwesomeDictionary.

1.2.24 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.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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

```

```
namespace AwesomeDictionary
{
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            ApplicationBar = new ApplicationBar();

            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);

            SetBackgroundColor();

            piFavourite.Header = AppResources.MyFavourites;
            piRandomWords.Header = AppResources.RandomWords;

        }

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

        private void pNames_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString);
            if (pNames.SelectedIndex == 0)
            {
                List<Favourite> favourites = context.Favourites.OrderBy(j => j.FavouriteName).ToList() as List<Favourite>;
                List<All> allNames = new List<All>();
                for (int i = 0; i < favourites.Count; i++)
                {
                    var allName = context.AllNames.Where(j => j.AllId.Equals(favourites[i].FavouriteAllId)).SingleOrDefault() as All;
                    allNames.Add(allName);
                }
                List<AlphaKeyGroup<All>> DataSource = AlphaKeyGroup<All>.CreateGroups(allNames,
                   System.Threading.Thread.CurrentThread.CurrentUICulture,
                   (All a) => { return a.AllName; }, true);
                llsFavourites.ItemsSource = DataSource;
            }

            else if (pNames.SelectedIndex == 1)
            {
                List<All> allNames = new List<All>();
                if (context.AllNames.ToList().Count > 0)
                {
                    List<int> sayiListesi = new List<int>();
                    Random random = new Random();
                    int tableCount = context.AllNames.ToList().Count;
                    for (int i = 0; i < 10; i++)
                    {
                        int sayi = random.Next(i, tableCount);
                        sayiListesi.Add(sayi);
                    }
                    for (int k = 0; k < sayiListesi.Count(); k++)
                    {
                        //int sayi = random.Next(0, 500000);
                        var allName = context.AllNames.Where(j => j.AllId.Equals(sayiListesi[k])).SingleOrDefault() as All;
                        //var allName = context.AllNames.Where(j => j.AllId.Equals(sayi)).SingleOrDefault() as All;
                        if (allName != null)
                        {
                            allNames.Add(allName);
                        }
                        //if(allNames.Count == 10)
                        //{
                        //    break;
                        //}
                    }
                }
                List<AlphaKeyGroup<All>> DataSource = AlphaKeyGroup<All>.CreateGroups(allNames,
                   System.Threading.Thread.CurrentThread.CurrentUICulture,
                   (All a) => { return a.AllName; }, true);
                llsRandomWords.ItemsSource = DataSource;
            }
        }

        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));
        }

        private void AddNameButton_Click(object sender, EventArgs e)
        {
            NavigationService.Navigate(new Uri("/NameDetailPage.xaml", UriKind.Relative));
            //PhoneApplicationPage_Loaded(this, new RoutedEventArgs());
        }


        private void llsFavourites_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            //var favourite = llsFavourites.SelectedItem as Favourite;
            var favourite = llsFavourites.SelectedItem as All;
            NavigationService.Navigate(new Uri("/NameDetailPage.xaml#" + favourite.AllId, UriKind.Relative));
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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 llsRandomWords_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            var randomWord = llsRandomWords.SelectedItem as All;
            NavigationService.Navigate(new Uri("/NameDetailPage.xaml#" +
randomWord.AllId, UriKind.Relative));
        }

    }
}
```

Namespaces

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

1.2.25 NameDetailPage.xaml.cs

This is file NameDetailPage.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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeDictionary
{
    public partial class NameDetailPage : PhoneApplicationPage
    {
        public string pageName;
        public int wordId;
        double InputHeight = 0.0;
        public bool flag;
        public bool isFilled;
        public double ratingValue = 0;
        string gender;
        All allName = new All();

        public NameDetailPage()
        {
            InitializeComponent();

            ApplicationBar = new ApplicationBar();

            //ApplicationBarIconButton button1 = new ApplicationBarIconButton();
            //button1.IconUri = new Uri("/Assets/Save.png", UriKind.Relative);
            //button1.Text = "Kaydet";
            //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.SendWithEmail;
            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.Share;
            ApplicationBar.Buttons.Add(button4);
            button4.Click += new EventHandler(ShareNameButton_Click);

            isFilled = false;

            //ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
            //menuItem1.Text = "Sil";
            //ApplicationBar.MenuItems.Add(menuItem1);
            //menuItem1.Click += new EventHandler(DeleteNameMenuItem_Click);

            //List<string> genderList = new List<string>();
            //genderList.Add("Lütfen seçiniz");
            //genderList.Add("Erkek");
            //genderList.Add("Kadin");
            //genderList.Add("Erkek-Kadin");
            //lpGender.ItemsSource = genderList;
            //lpGender.SelectedIndex = 0;

            SetBackgroundColor();
        }

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

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

            var lastPage = NavigationService.BackStack.FirstOrDefault();
            pageName = lastPage.Source.ToString();
            using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            {
                var appSettings = context.AppSettings.First();

                FontFamily temp = new FontFamily(appSettings.FontFamily);
                double fontsize = double.Parse(appSettings.FontSize);
                txtMeaning.FontFamily = temp;
                txtMeaning.FontSize = fontsize;
            }

            txtMeaning.IsEnabled = false;
            isFilled = false;
            //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);
            //string fragmentName = e.Fragment.ToString();
            wordId = Convert.ToInt32(e.Fragment);
            using (var context = new
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            {
                var name = context.AllNames.Where(j =>
j.AllId.Equals(wordId)).SingleOrDefault() as All;
                allName = name;
                //for (int i = 0; i < lpGender.Items.Count; i++)
                //{
                //    if (lpGender.Items[i].ToString() == name.AllGender)
                //    {
                //        lpGender.SelectedIndex = i;
                //        break;
                //    }
                //}
                lblNameDetail.Text = name.AllName;
                txtMeaning.Text = name.AllMeaning + Environment.NewLine +
Environment.NewLine + AppResources.Source + ":" + name.AllSource + "";
                txtMeaning.IsEnabled = false;
                //txtMeaning.Text = name.AllMeaning;

                var favourite = context.Favourites.Where(j =>
j.FavouriteAllId.Equals(wordId)).SingleOrDefault() as Favourite;
                if (favourite != null)
                {
                    ApplicationBarMenuItem menuItem4 = new ApplicationBarMenuItem();
                    menuItem4.Text = AppResources.RemoveFromFavourites;
                    ApplicationBar.MenuItems.Add(menuItem4);
                    menuItem4.Click += new
EventHandler(RemoveFavouritesMenuItem_Click);
                }
                else
                {
                    ApplicationBarMenuItem menuItem3 = new ApplicationBarMenuItem();
                    menuItem3.Text = AppResources.AddToFavourites;
                    ApplicationBar.MenuItems.Add(menuItem3);
                    menuItem3.Click += new EventHandler(AddFavouritesMenuItem_Click);
                }

                //var myUpdate = context.MyUpdates.Where(j =>
j.MyUpdateName.Equals(fragmentName)).SingleOrDefault() as MyUpdate; ;
                //if (myUpdate != null)
                //{
                //    ApplicationBarMenuItem menuItem2 = new ApplicationBarMenuItem();
                //    menuItem2.Text = "Sisteme Eklenmesi Için Gönder";
                //    ApplicationBar.MenuItems.Add(menuItem2);
                //    menuItem2.Click += new EventHandler(SaveAndSendMenuItem_Click);

                //}


                isFilled = true;
                //pvName.SelectedIndex = 0;
            }
        }

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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 txtName_KeyDown(object sender, KeyEventArgs e)
        {
            //if (e.Key == Key.Enter)
            //{
            //    pvName.SelectedIndex = 2;
            //    txtMeaning.Focus();
            //}
        }

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

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

                InputHeight = CurrentInputHeight;
            });
        }

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

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

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

        private void txtMeaning_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                svMeaning.ScrollToVerticalOffset(txtMeaning.ActualHeight);
            }
        }

        private void svMeaning_GotFocus(object sender, RoutedEventArgs e)
        {
            this.svMeaning.ScrollToVerticalOffset(this.txtMeaning.ActualHeight);
            this.svMeaning.UpdateLayout();
        }

        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 ShareNameButton_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();

            EmailComposeTask emailComposeTask = new EmailComposeTask();

            emailComposeTask.Subject = AppResources.WordAndMeaning + " (" + lblNameDetail.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.Word + ": " + lblNameDetail.Text);
            sb.AppendLine(AppResources.Meaning + ": " + txtMeaning.Text);
            //sb.AppendLine("Cinsiyeti: " + lpGender.Items[lpGender.SelectedIndex]);
            sb.AppendLine();
            sb.AppendLine();
            sb.AppendLine(AppResources.SendWithAwesomeDictionaryApp);
            return sb.ToString();
        }

        private void AddFavouritesMenuItem_Click(object sender, EventArgs e)
        {
            using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            {
                var favourites = context.Favourites.Where(j => j.FavouriteName.Equals(lblNameDetail.Text)).ToList() as List<Favourite>;
                if (favourites.Count > 0)
                {
                    MessageBox.Show(AppResources.WordAlreadyFavourite);
                }
                else
                {
                    Favourite favourite = new Favourite();
                    favourite.FavouriteName = lblNameDetail.Text;
                    favourite.FavouriteAllId = wordId;
                    context.Favourites.InsertOnSubmit(favourite);
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.WordAddedFavouriteSuccessfully);
                }
            }
        }

        private void RemoveFavouritesMenuItem_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show(AppResources.RemoveFromFavouriteQuestion,
                   AppResources.RemoveFromFavourite, MessageBoxButton.OKCancel)
                   != MessageBoxResult.OK)
            {
            }
            else
            {
                using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                {
                    var favourite = context.Favourites.Where(j => j.FavouriteName.Equals(lblNameDetail.Text)).ToList() as List<Favourite>;
                    context.Favourites.DeleteAllOnSubmit(favourite);
                    context.SubmitChanges();
                    MessageBox.Show(AppResources.WordRemovedFavouriteSuccessfully);
                }
            }
        }

        private void SaveAndSendMenuItem_Click(object sender, EventArgs e)
        {
            //EmailComposeTask emailComposeTask = new EmailComposeTask();

            //emailComposeTask.Subject = "Isim ve Anlami (" + txtName.Text + ")";
            //emailComposeTask.Body = CreateSendMaterial();
            //emailComposeTask.To = "coderserdar@outlook.com";
            //emailComposeTask.Cc = "";
            //emailComposeTask.Bcc = "";

            //emailComposeTask.Show();
        }

        private void DeleteNameMenuItem_Click(object sender, EventArgs e)
        {
            //if (isFilled == true)
            //{
            //    if (MessageBox.Show("Ismi Silmek Istediginize Emin Misiniz?",
            //           "Ismi Sil", MessageBoxButton.OKCancel)
            //           != MessageBoxResult.OK)
            //    {
            //    }
            //    else
            //    {
            //        using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            //        {

            //            switch (allName.AllGender)
            //            {
            //                case "Erkek":
            //                    var maleName = context.MaleNames.Where(j => j.MaleName.Equals(allName.AllName)).SingleOrDefault() as Male;
            //                    context.MaleNames.DeleteOnSubmit(maleName);
            //                    break;
            //                case "Kadin":
            //                    var femaleName = context.FemaleNames.Where(j => j.FemaleName.Equals(allName.AllName)).SingleOrDefault() as Female;
            //                    context.FemaleNames.DeleteOnSubmit(femaleName);
            //                    break;
            //                case "Erkek-Kadin":
            //                    var unisexName = context.UnisexNames.Where(j => j.UnisexName.Equals(allName.AllName)).SingleOrDefault() as Unisex;
            //                      context.UnisexNames.DeleteOnSubmit(unisexName);
            //                      break;
            //                  default:
            //                      break;
            //              }

            //              var allNames = context.AllNames.Where(j => j.AllName.Equals(allName.AllName)).SingleOrDefault() as All;
            //              context.AllNames.DeleteOnSubmit(allNames);

            //              var myUpdates = context.MyUpdates.Where(j => j.MyUpdateName.Equals(allName.AllName)).SingleOrDefault() as MyUpdate;
            //              if (myUpdates != null)
            //              {
            //                  context.MyUpdates.DeleteOnSubmit(myUpdates);
            //              }

            //              var favourites = context.Favourites.Where(j => j.FavouriteName.Equals(allName.AllName)).SingleOrDefault() as Favourite;
            //              if (favourites != null)
            //              {
            //                  context.Favourites.DeleteOnSubmit(favourites);
            //              }

            //              context.SubmitChanges();
            //              MessageBox.Show("Isim, Sistemden Silindi");
            //          }
            //      }
            //}
            //this.NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
        }

        private void SaveButton_Click(object sender, EventArgs e)
        {
            //using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            //{
            //      if (isFilled == false)
            //      {
            //          if (lpGender.SelectedIndex < 1 || txtName.Text.TrimEnd().TrimStart().Length < 1 || txtMeaning.Text.TrimEnd().TrimStart().Length < 1)
            //          {
            //              MessageBox.Show("Bütün Alanlari Dolduraniz Gerekmektedir.");
            //          }
            //          else
            //          {
            //              var allName = context.AllNames.Where(j => j.AllName.ToLower().Equals(txtName.Text.TrimEnd().TrimStart().ToLower())).SingleOrDefault() as All;
            //              if (allName != null)
            //              {
            //                  MessageBox.Show("Bu Isim Sistemde Mevcut");
            //              }
            //              else
            //              {

            //                  switch (lpGender.SelectedIndex)
            //                  {
            //                      case 1:
            //                          Male maleName = new Male();
            //                          maleName.MaleName = txtName.Text.ToUpper();
            //                          maleName.MaleMeaning = txtMeaning.Text;
            //                          context.MaleNames.InsertOnSubmit(maleName);
            //                          break;
            //                      case 2:
            //                          Female femaleName = new Female();
            //                          femaleName.FemaleName = txtName.Text.ToUpper();
            //             femaleName.FemaleMeaning = txtMeaning.Text;
            //             context.FemaleNames.InsertOnSubmit(femaleName);
            //             break;
            //           case 3:
            //             Unisex unisexName = new Unisex();
            //             unisexName.UnisexName = txtName.Text.ToUpper();
            //             unisexName.UnisexMeaning = txtMeaning.Text;
            //             context.UnisexNames.InsertOnSubmit(unisexName);
            //             break;
            //           default:
            //             break;
            //         }

            //         All allNameTemp = new All();
            //         allNameTemp.AllName = txtName.Text.ToUpper();
            //         allNameTemp.AllMeaning = txtMeaning.Text;
            //         allNameTemp.AllNameMeaning = txtName.Text + " " + txtMeaning.Text;
            //         allNameTemp.AllGender = lpGender.Items[lpGender.SelectedIndex].ToString();
            //         context.AllNames.InsertOnSubmit(allNameTemp);

            //         MyUpdate myUpdate = new MyUpdate();
            //         myUpdate.MyUpdateName = txtName.Text.ToUpper();
            //         context.MyUpdates.InsertOnSubmit(myUpdate);

            //         context.SubmitChanges();
            //         MessageBox.Show("Isim, Sisteme Basarili Bir Sekilde Eklendi");

            //       }
            //     }
            //   }
            //   else
            //   {
            //     if (allName.AllGender == lpGender.Items[lpGender.SelectedIndex].ToString() && allName.AllMeaning == txtMeaning.Text.TrimEnd().TrimStart())
            //     {
            //       MessageBox.Show("Herhangi bir degisiklik yapmadiniz.");
            //     }
            //     else
            //     {
            //       // önce eski kayit siliniyor
            //       switch (allName.AllGender)
            //       {
            //         case "Erkek":
            //           var maleName = context.MaleNames.Where(j => j.MaleName.Equals(allName.AllName)).SingleOrDefault() as Male;
            //           context.MaleNames.DeleteOnSubmit(maleName);
            //           break;
            //         case "Kadin":
            //           var femaleName = context.FemaleNames.Where(j => j.FemaleName.Equals(allName.AllName)).SingleOrDefault() as Female;
            //           context.FemaleNames.DeleteOnSubmit(femaleName);
            //           break;
            //         case "Erkek-Kadin":
            //           var unisexName = context.UnisexNames.Where(j => j.UnisexName.Equals(allName.AllName)).SingleOrDefault() as Unisex;
            //           context.UnisexNames.DeleteOnSubmit(unisexName);
            //           break;
            //         default:
            //           break;
            //       }

            //       // sonrasinda yeni kayit ekleniyor
            //       switch (lpGender.SelectedIndex)
            //       {
            //         case 1:
```

```
            //                      Male maleName = new Male();
            //                      maleName.MaleName = txtName.Text.ToUpper();
            //                      maleName.MaleMeaning = txtMeaning.Text;
            //                      context.MaleNames.InsertOnSubmit(maleName);
            //                      break;
            //                  case 2:
            //                      Female femaleName = new Female();
            //                      femaleName.FemaleName = txtName.Text.ToUpper();
            //                      femaleName.FemaleMeaning = txtMeaning.Text;
            //                      context.FemaleNames.InsertOnSubmit(femaleName);
            //                      break;
            //                  case 3:
            //                      Unisex unisexName = new Unisex();
            //                      unisexName.UnisexName = txtName.Text.ToUpper();
            //                      unisexName.UnisexMeaning = txtMeaning.Text;
            //                      context.UnisexNames.InsertOnSubmit(unisexName);
            //                      break;
            //                  default:
            //                      break;
            //              }

            //              var all = context.AllNames.Where(j => j.AllName.Equals(allName.AllName)).Select(j => j);
            //              foreach (var item in all)
            //              {
            //                  item.AllGender = lpGender.Items[lpGender.SelectedIndex].ToString();
            //                  item.AllMeaning = txtMeaning.Text.TrimEnd().TrimStart();
            //                  item.AllNameMeaning = item.AllName + " " + item.AllMeaning;
            //              }
            //              var myUpdateTemp = context.MyUpdates.Where(j => j.MyUpdateName.Equals(allName.AllName)).SingleOrDefault() as MyUpdate;
            //              if (myUpdateTemp == null)
            //              {
            //                  MyUpdate myUpdate = new MyUpdate();
            //                  myUpdate.MyUpdateName = allName.AllName;
            //                  context.MyUpdates.InsertOnSubmit(myUpdate);
            //              }

            //              context.SubmitChanges();

            //              MessageBox.Show("Isim Bilgisi Basarili Bir Sekilde Güncellendi");
            //          }
            //      }
            //      this.NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
            //}
        }

        private void lpGender_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            //if (lpGender.SelectedIndex != 0)
            //{
            //    gender = lpGender.Items[lpGender.SelectedIndex].ToString();
            //    pvName.SelectedIndex = 1;
            //    txtName.Focus();
            //}
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {

        }

    }
}
```

Namespaces

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

1.2.26 OxfordEnglishEnglish.cs

This is file OxfordEnglishEnglish.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 AwesomeDictionary
{
    [Table]
    public class OxfordEnglishEnglish
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int OxfordId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.27 RisaleNur.cs

This is file RisaleNur.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 AwesomeDictionary
{
    [Table]
    public class RisaleNur
    {
        [Column(IsPrimaryKey = true,
            IsDbGenerated = true,
            DbType = "INT NOT NULL Identity",
            CanBeNull = false)]
        public int RisaleNurId { get; set; }

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

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

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

    }
}
```

Namespaces

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

1.2.28 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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

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

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

        private void SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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)
        {
            lstSearch.Items.Clear();
            var nameList = new List<All>();
            if (txtSearch.Text.TrimStart().TrimEnd().Length < 1)
            {
```

1

```
                MessageBox.Show(AppResources.SearchTrimFault);
            }
            else
            {
                using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                {
                    if (cbSearchWithMeaning.IsChecked == true)
                    {
                        nameList = context.AllNames.Where(j => j.AllNameMeaning.ToLower().Contains(txtSearch.Text.TrimEnd().TrimStart().ToLower())).ToList() as List<All>;
                    }
                    else
                    {
                        nameList = context.AllNames.Where(j => j.AllName.ToLower().Contains(txtSearch.Text.TrimEnd().TrimStart().ToLower())).ToList() as List<All>;
                    }
                    //var noteList = context.Notes.ToList() as List<Note>;

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

                    //List<All> allNames = nameList.OrderBy(j => j.AllName).ToList() as List<All>;
                    //List<AlphaKeyGroup<All>> DataSource = AlphaKeyGroup<All>.CreateGroups(allNames,
                    //    System.Threading.Thread.CurrentThread.CurrentUICulture,
                    //    (All a) => { return a.AllName; }, true);
                    //llsAllNames.ItemsSource = DataSource;

                    var nameList2 = nameList.OrderBy(j => j.AllName).ToList() as List<All>;
                    //lstSearch.ItemsSource = nameList;
                    for (int i = 0; i < nameList2.Count; i++)
                    {
                        lstSearch.Items.Add(nameList2[i] as All);
                    }
                    //lstSearch.ItemTemplate.
                    //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate" + ")";
                    lstSearch.DisplayMemberPath = "AllNameSource";
                    MessageBox.Show(AppResources.SearchCompleted);
                }
            }
        }

        private void lstSearch_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
                if (lstSearch.SelectedIndex != -1)
                {
                    All selectedName = lstSearch.SelectedItem as All;
                    NavigationService.Navigate(new Uri("/NameDetailPage.xaml#" + selectedName.AllId, UriKind.Relative));
                    lstSearch.SelectedIndex = -1;
                }

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

1

```
            }
        }

        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)
            {
                lstSearch.Items.Clear();
                var nameList = new List<All>();
                if (txtSearch.Text.TrimStart().TrimEnd().Length < 1)
                {
                    MessageBox.Show(AppResources.SearchTrimFault);
                }
                else
                {
                    using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
                    {
                        if (cbSearchWithMeaning.IsChecked == true)
                        {
                            nameList = context.AllNames.Where(j => j.AllNameMeaning.ToLower().Contains(txtSearch.Text.TrimEnd().TrimStart().ToLower())).ToList() as List<All>;
                        }
                        else
                        {
                            nameList = context.AllNames.Where(j => j.AllName.ToLower().Contains(txtSearch.Text.TrimEnd().TrimStart().ToLower())).ToList() as List<All>;
                        }
                        //var noteList = context.Notes.ToList() as List<Note>;

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

                        //List<All> allNames = nameList.OrderBy(j => j.AllName).ToList() as List<All>;
                        //List<AlphaKeyGroup<All>> DataSource = AlphaKeyGroup<All>.CreateGroups(allNames,
                        //    System.Threading.Thread.CurrentThread.CurrentUICulture,
                        //    (All a) => { return a.AllName; }, true);
                        //llsAllNames.ItemsSource = DataSource;

                        var nameList2 = nameList.OrderBy(j => j.AllName).ToList() as List<All>;
                        //lstSearch.ItemsSource = nameList;
                        for (int i = 0; i < nameList2.Count; i++)
                        {
                            lstSearch.Items.Add(nameList2[i] as All);
                        }
                        //lstSearch.ItemTemplate.
                        //lstSearch.DisplayMemberPath = "NoteName" + " (" + "CreationDate" + ")";
                        lstSearch.DisplayMemberPath = "AllNameSource";
                        MessageBox.Show(AppResources.SearchCompleted);
                    }
                }
            }
        }

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

Namespaces

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

1.2.29 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 AwesomeDictionary.Resources;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AwesomeDictionary
{
    public partial class StatisticsPage : PhoneApplicationPage
    {
        public StatisticsPage()
        {
            InitializeComponent();
            lblStatistics.Text = AppResources.Statistics;
            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 SetBackgroundColor()
        {
            AppSettings appSettings = new AppSettings();
            using (var context = new AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.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);
            SetStatistic();
        }

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

        }

        private void SetStatistic()
        {
            StringBuilder sb = new StringBuilder();
            int allCount, favouriteCount;


            using (var context = new
AwesomeDictionaryDataContext(AwesomeDictionaryDataContext.ConnectionString))
            {
                favouriteCount = context.Favourites.ToList().Count;
                allCount = context.AllNames.ToList().Count;
            }

            sb.AppendLine(AppResources.TotalNameCount + ": " + allCount);
            sb.AppendLine(AppResources.FavouriteNameCount + ": " + favouriteCount);
            //sb.AppendLine(AppResources.MostListenArtist + ": " + artistName);
            //sb.AppendLine(AppResources.MostListenLabel + ": " + labelName);
            //sb.AppendLine(AppResources.BestAlbum + ": " + bestAlbum);
            //sb.AppendLine(AppResources.WorstAlbum + ": " + worstAlbum);

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

Namespaces

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

Index