namespace BooksDatabase.ViewModel

{

public class MainViewModel : ViewModelBase

{

string text;

public string Text

{

get

{

return text;

}

set

{

text = value;

RaisePropertyChanged("text");

}

}

/// <summary>

/// Initializes a new instance of the MainViewModel class.

/// </summary>

public MainViewModel()

{

BooksEntities dbcontext = new BooksEntities();

//Get a list of all the titles and the authors who wrote them.

//Sort the result by title.

var allTitles =

from author in dbcontext.Authors

from book in author.Titles

orderby book.Title1

select new { author.FirstName, author.LastName, book.Title1 };

Text += "Titles and Authors:";

// display titles and author

foreach (var element in allTitles)

{

Text += "\r\n\t" + element.Title1 + "\t" + element.FirstName + "\t" + element.LastName;

} // end foreach

//Get a list of all the titles and the authors who wrote them.

//Sort the result by title.

//For each title sort the authors alphabetically by last name, then first name.

var allTitlesLF =

from author in dbcontext.Authors

from book in author.Titles

orderby book.Title1, author.LastName, author.FirstName

select new { author.FirstName, author.LastName, book.Title1 };

Text += "\r\n\r\nAuthors and titles with authors sorted for each title:";

// display titles and author

foreach (var element in allTitlesLF)

{

Text += "\r\n\t" + element.Title1 + "\t" + element.LastName + "\t" + element.FirstName;

} // end foreach

//Get a list of all the authors grouped by title, sorted by title

//for a given title sort the author names alphabetically by last name first then first name.

var authorsByTitle =

from author in dbcontext.Authors

from book in author.Titles

group new { author.FirstName, author.LastName } by book.Title1 into bookGroup

orderby bookGroup.Key

select bookGroup;

Text += "\r\n\r\nTitles grouped by author:";

// display titles written by each author, grouped by titles

foreach (var title in authorsByTitle)

{

// display titles's name

Text += "\r\n\t" + title.Key + ":";

// display authors

foreach (var author in title.OrderBy(name => name.LastName + name.FirstName))

{

Text += "\r\n\t\t" + author.FirstName + " " + author.LastName;

} // end inner foreach

} // end outer foreach

this.RaisePropertyChanged(() => this.Text);

}

}

}

namespace CustomerMaintenanceMVVM.ViewModel

{

public class MainViewModel : ViewModelBase

{

.

.

.

/// <summary>

/// deletes a customer

/// </summary>

public void DelMethod()

{

MessageBoxResult result = MessageBox.Show("Delete " + selectedCustomer.Name + "?", "Confirm Delete", MessageBoxButton.YesNo, MessageBoxImage.Question);

if(result == MessageBoxResult.Yes)

{

try

{

MMABooksEntity.MMABooks.Customers.Remove(selectedCustomer);

MMABooksEntity.MMABooks.SaveChanges();

IDText = "";

Empty();

}

// concurrency error handling.

catch (DbUpdateConcurrencyException ex)

{

ex.Entries.Single().Reload();

if (MMABooksEntity.MMABooks.Entry(selectedCustomer).State == EntityState.Detached)

{

MessageBox.Show("Another User has deleted" + "that customer", "Concurrency Error");

IDText = "";

Empty();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, ex.GetType().ToString());

}

}

}

}

}

namespace CustomerMaintenanceMVVM.ViewModel

{

public class AddUpdateViewModel : ViewModelBase

{

.

.

.

/// <summary>

/// saves customer to database

/// </summary>

public void SaveMethod()

{

//Add

.

.

.

//Mod

else

{

PutCustomerData(customer);

try

{

MMABooksEntity.MMABooks.SaveChanges();

}

// concurrency error handling.

catch (DbUpdateConcurrencyException ex)

{

ex.Entries.Single().Reload();

if (MMABooksEntity.MMABooks.Entry(customer).State == EntityState.Unchanged)

{

MessageBox.Show("Another user has updated that customer", "Concurrency Error");

}

else if (MMABooksEntity.MMABooks.Entry(customer).State == EntityState.Detached)

{

MessageBox.Show("Another user has deleted that customer", "Concurrency Error");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, ex.GetType().ToString());

}

}

Messenger.Default.Send<NotificationMessage>(new NotificationMessage(this, "Close"));

Messenger.Default.Send<Customer>(customer);

}

}

}