To-Do List

By Team Rubellite

Team members (in alphabetical order):

1. *Ангел Димитров (Angel Dimitrov)* - Vazzzz
2. *Митко Николов (Mitko Nikolov)* – velmond
3. *Никола Тодоров (Nikola Todorov)* – nmtodorov
4. *Николай Кичев (Nikolay Kichev)* – nkichev
5. *Радослав Стоянов (Radoslav Stoyanov)* – rodi1i
6. *Цветан Борисов (Tsvetan Borisov)* – tsvetan.borissov

Work organization:

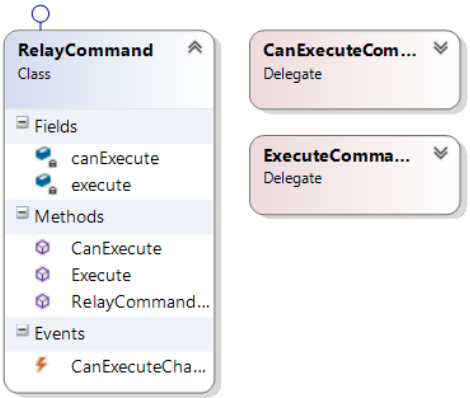
The work was organized through several meetings in Telerik academy. The basic communication was in a group created in Skype, which has more than 3000 messages (counted last on 8.12.2013).

Project explanation:

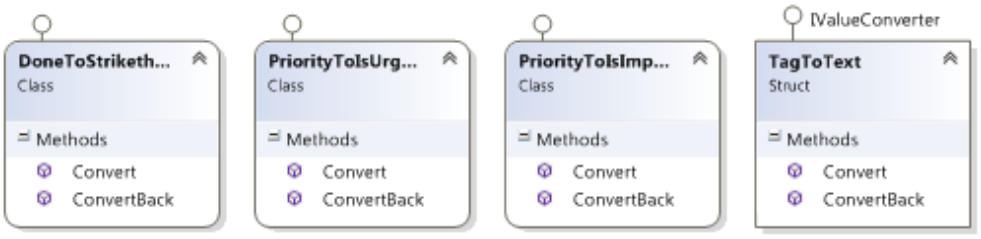
The program is in essence a To-Do list in which the user can save his short term tasks or long term goals, add information about upcoming meetings, people's birthdays and plain notes.

The program implements the MVVM design patter (Model-View-ViewModel). All the classes are separated into the following namespaces:

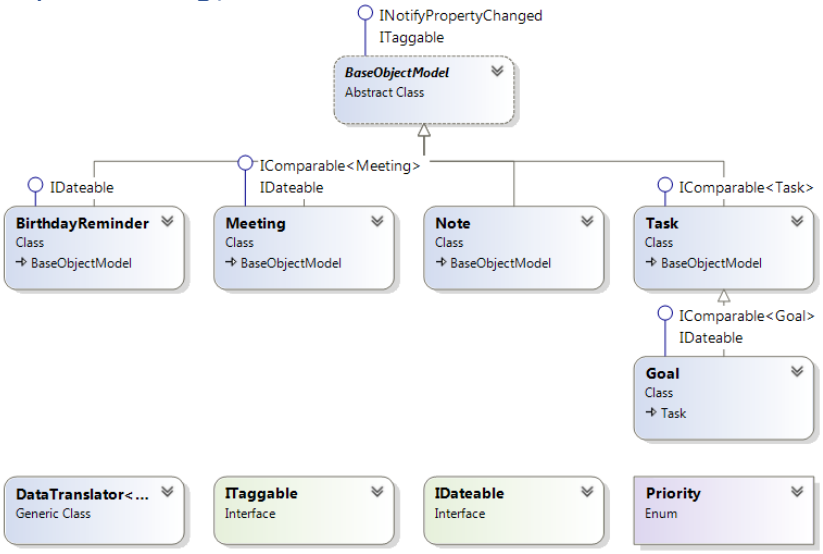
- ToDoList.Commands – for all the classes dedicated to executing commands;



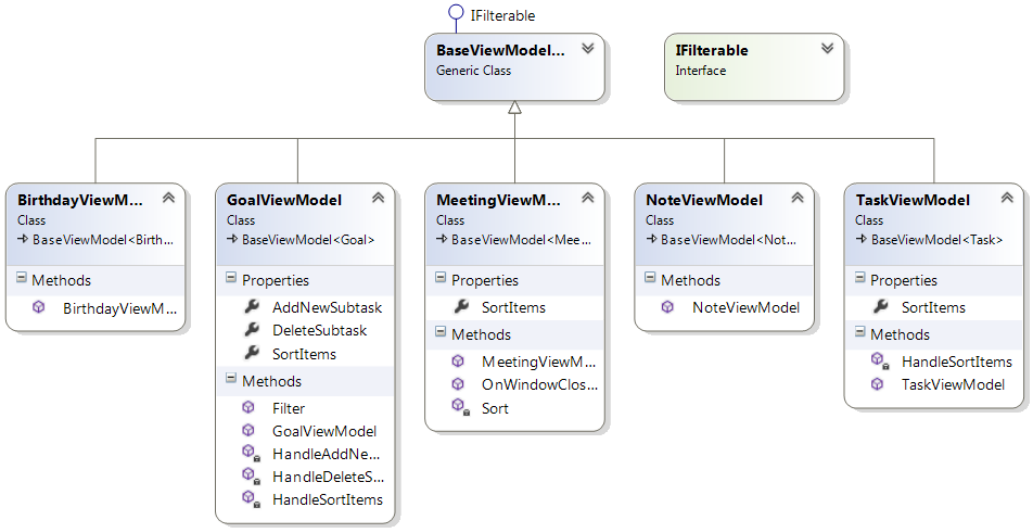
- ToDoList.Converters – for all the conversions of data from one type to another;



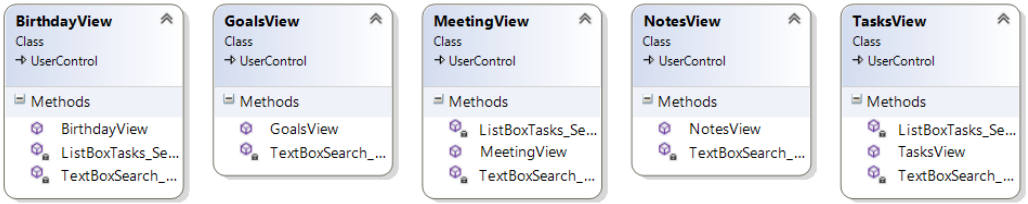
- ToDoList.Models – for all the types of to-do entities that the program uses (tasks, notes, etc.) and the interfaces that they are implementing;



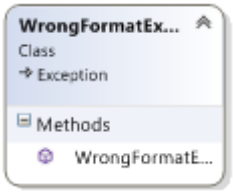
- ToDoList.ViewModels – for the view models for each of the types of to-dos and the interfaces that they are implementing;



- ToDoList.Views – for all the XAML files for each model view;



- WrongFormatException – custom exception class made for the project. Used when there's an input, which is not the preferred format (ToDoList.Exceptions);



SVN repository:

To-Do List (team Rubellite)

Additional information:

There are three interfaces specifically written for the project – ITaggable, IDateable and IFilterable. From existing interfaces in the .NET framework INotifyPropertyChanged, IValueConverter, ICommand and IComparable<T> are used.

ITaggable makes sure the class implementing it has a property 'Tags' which is a collection of descriptive strings. It is used in all types of to-dos.

IDateable makes sure an object has a DateTime property 'EventDate' which is specific for each type of to-do that implements this interface. For example for ToDoList.Models.BirthdayReminder it is the birth date, for ToDoList.Models.Meeting it marks the date of the meeting and for the ToDoList.Models.Goal it marks the deadline by which the goal must be achieved.

IFilterable makes sure that the class implementing it will allow filtering through a collection of objects which is used as a search function.

There is one abstract class which is the base for each of the 'models'. It is named BaseObjectModel and ensures that all classes deriving from it have a title, description, a property showing if it done or not and a collection of tags. The BaseObjectModel has an event OnPropertyChanged which notifies if a property is ever changed.

There is an enumeration for the priority types which can be "Urgent", "Important" or "None".

There are five types of to-dos that the program uses:

- Tasks - simple task without deadline and optional priority.
- Meetings - it keeps all the information about a meeting.
- Goals - long term task that can contain subtasks and has a deadline
- Birthday reminders - a reminder task that keeps birthday information.
- Notes - simple task containing just a title and a description

Each of these five has a 'model', 'viewModel' and 'view' class. The view is done through XAML.

All information about the collections of each of these types of to-dos is kept in *.xml files.

We have used generic classes for the DataTranslator and the BaseViewModel.

LINQ is used for the specific sorting of each type of to-do.

Full class diagram: