Lakana v2 WPF Framework

Overview

[1. Overview 3](#_Toc355366228)

[2. Lakana WPF Navigation 4](#_Toc355366229)

[2.1 Why and when should I need it? 4](#_Toc355366230)

[2.2 Features 4](#_Toc355366231)

[3. Lakana WPF Forms 6](#_Toc355366232)

[3.1 Why and when should I need it? 6](#_Toc355366233)

[3.2 Features 6](#_Toc355366234)

# Overview

**Lakana** is a set of lightweight frameworks which main goal is to tackle some tricky problems that Windows Presentation Foundation (WPF) developers can encounter very often, especially when they make use of the Model View View-Model (MVVM) design pattern.

It is composed of two independent yet compatible components:

* **Lakana WPF Navigation**: that is an easy to use and lightweight navigation framework.
* **Lakana WPF Forms**: a lightweight framework that eases the creation of editable data forms.

As a side note, a “Lakana” is a rudimentary hand-made canoe that is used by Malagasy fishermen.

# Lakana WPF Navigation

**Lakana WPF Navigation** is a lightweight yet powerful navigation Framework for Windows Presentation Foundation applications, it is part of Lakana Framework.

It offers powerful navigation capabilities (similar to MDI navigation style) that can be very helpful especially if you design a multi-task application and make use of MVVM.

## Why and when should I need it?

Users use applications by navigating from one screen to another to achieve a certain task. To help him as efficiently as possible, the application should offer a coherent and clear navigation style that is not always obvious to implement.

Lakana is a lightweight framework that can serve as a navigation engine for your application. It will offer you common features so that you do not have to implement them from scratch.

Below is the list of all features that the framework offers.

## Features

* MDI-like navigation style
* Display non-blocking modal windows
* Customizable messages boxes
* Graceful application shutdown management
* Support for both MVVM and non-MVVM applications.
* Custom animation during screen transitions

MDI-like navigation style

Lakana implements a non-linear style of navigation that is you can navigate between views in almost any direction you want, unlike linear navigation as implemented by internet browser for instance where you only stack and unstack views. This kind of navigation suits well for multi-task applications as users can begin one task, suspend it and do another task then be back on it again later.

Display non-blocking modal windows

Lakana offers API that allows you to display your views as a non-blocking modal view on top of another one. Usually, modal views block your entire application until you close it but Lakana allows you to display a modal view and still you can work with other part of your application.

Customizable message boxes

Lakana implements standard and customizable/themable message boxes.

Graceful application shutdown

Lakana manages for you the tricky task of gracefully shutdown the application.

Sometimes for a lot of reasons, you just want to prevent the application from closing when requested so. This will be the case if some tasks are still running and you want to be able to terminate them gracefully before closing the application. Lakana offers an elegant manner to handle that kind of problem.

Support for both MVVM and non-MVVM projects

You can use the framework with existing or new project (may it use MVVM or not). Because Lakana is simple and enough decoupled from other application concerns, you can use it with both MVVM and non-MVVM applications.

Custom animation for screen transitions

You may want to animate screen transitions. Lakana offers some animations out of the box, and you can also easily implement your own if you want fancy 2D or 3D animation during your screen transitions.

# Lakana WPF Forms

**Lakana WPF Forms** is a lightweight framework that aims to offer you very useful features when implementing editable data forms using Windows Presentation Foundation and the Model View View-Model (MVVM) design pattern.

## Why and when should I need it?

The WPF/MVVM combination has become the mainstream nowadays.

Despite this fact, developing applications that uses forms to edit data (that is the case for many line of business applications for example), is still tedious. Managing things like field validation, validation rule definition on fields or dynamic localization of texts, can become a nightmare especially if you want more advanced features such as asynchronous validation or the ability to define more complex validation rules.

Lakana WPF Forms fills this gap by offering the basic and common features that you can leverage in your application.

## Features

* Rich view model properties.
* UI localization made easier.
* Asynchronous validation on fields.
* Three validation engines for fields: Data annotations, Fluent Validation API or Custom.

Rich view model properties

Lakana relies on special rich properties abstracted by the IViewModelProperty interface, in order to enforce features like property value change detection, error notifications and so forth.

View model properties are kind of rich properties in sense that they contain more than just the property value. In fact, each View model property will wrap each property of your model to make them more “UI friendly”.

UI Localization made easier

UI metadata (represented by IUIMetadata interface) are associated with View model properties. They are used to put things like field descriptions, labels or icons. IUIMetadata facilitates the dynamic localization of the UI.

Asynchronous validation on fields

To validate a property and display error notifications, WPF developers are used to choose either implementing IDataErrorInfo interface or attaching ValidationRule object on their bindings.

These two approaches are synchronous, meaning your validation code should execute very quickly to avoid freezing the UI. For cases where the validation process can take longer (for instance if you have to ask to some web service for existence or not of the current value), an asynchronous approach is preferable.

Lakana natively supports both synchronous and asynchronous validation of fields.

Validation Engines

The built-in validation engines allow to define validation rules and to validate field’s value. They also support both synchronous and asynchronous validation of the form.

In WPF, you have two choices as far as form validation is concerned: per-field validation or batch validation.

Lakana implements per-field validation, meaning the validation process take place every time you change the value on a field.

* Data annotations

Data annotations are part of the .NET framework, and they are used to enforce validation rules on object properties.

They are powerful but unfortunately, were not designed natively to work with WPF. However Lakana again fills this gap. All you have to do is to define data annotations attribute on your view model properties and that’s all, the framework will manage the validation for you.

* Fluent Validation API

Fluent APIs have gained in popularity in recent years. To allow you to gain in clarity and expressiveness for your form validation rules declaration, Lakana offers a built-in Fluent API. The fluent syntax will improve significantly the readability of your code. Furthermore, you will be able to write rules that involve many properties or asynchronous operations.

* Custom validation

If any of the previous validation engines do not fit your needs, you can just implement your own. You can inherit from ValidationEngineBase class. You still can benefit the ability to make a synchronous or asynchronous validation.