Writing task for Ric Generator

Abstract: This document explains how to upgrade a task from the old to the new UI, it also show some guidelines and advices everyone should follow when writing tasks for the Ric Generator application.

Contents

[Task upgrade from old to new Ui 2](#_Toc388965028)

[Namespaces 2](#_Toc388965029)

[Config 2](#_Toc388965030)

[Task 3](#_Toc388965031)

[Mandatory 3](#_Toc388965032)

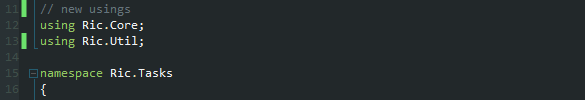
[Optional but recommended 4](#_Toc388965033)

[Database 4](#_Toc388965034)

# Task upgrade from old to new Ui

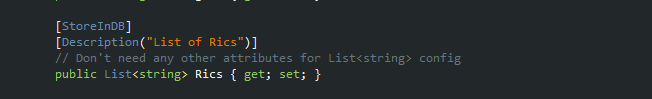
## Namespaces

* Now tasks namespace is “Ric.Tasks”. When more tasks are working, we can divide by market like the old ric generator and have namespace like “Ric.Tasks.[MarketName]”. But now just one project and namespace for all tasks to simplify the upgrade and testing.
* In the using section you may have to rename some of them, RicDb -> Ric.Db, Reuters.ProcessQuality.ContentAuto.Lib -> Ric.Util, etc …

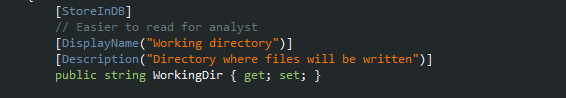


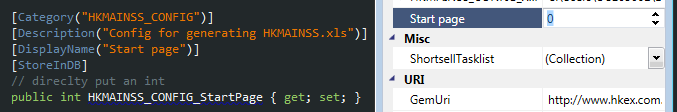
## Config

* For string list, don’t need the very long attribute anymore



* Should use more [DisplayName(“”)]



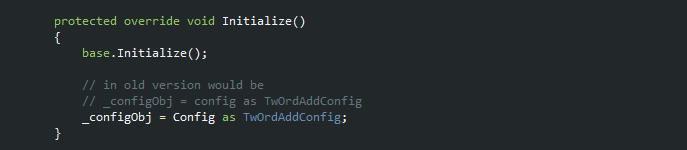
* Now can have more types than just string or list<string> (int for example) 

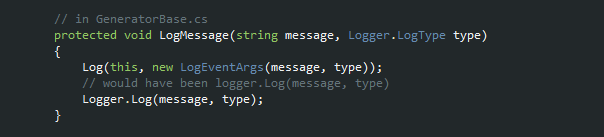
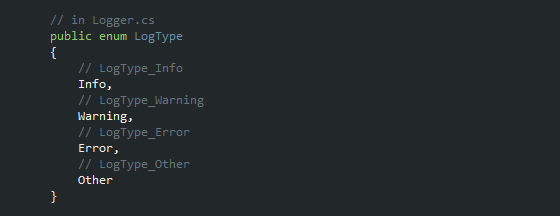
## Task

### Mandatory

Those changes are mandatory, it won’t compile if you don’t do the changes.

* **C**onfig instead of **c**onfig. To respect naming standards.



* **L**ogger instead of **l**ogger. To respect naming standards.
* 
* LogType changes: Logger.LogType.LogType\_Error -> Logger.LogType.Error . for all types the “LogType\_” is deleted to avoid useless repetition.
* 

### Optional but recommended

Those changes are optional and if you don’t do them the task should work fine, but could be useful for the user to see what is happening live for long running tasks.

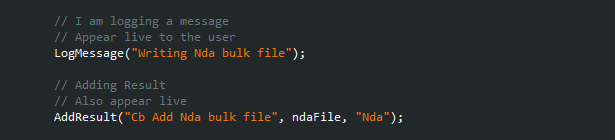
* Replace the logger.Log by LogMessage. Logmessage sent an event to the UI to show the message live, it also calls logger.Log so you still write log in a file, same as before.

LogMessage(string message) -> use LogType.Info

LogMessage(string message, Logger.LogType type) uses the type you wrote

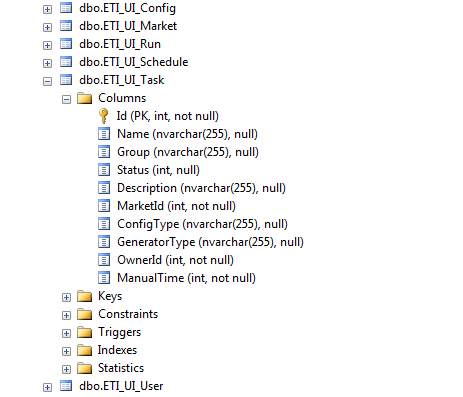
* Replace TaskResult.Add() by AddResult(), it will also send an event to the UI to show user in real time that there is a new file. And will call TaskResult.Add () this way nothing changes compare to before.

AddResult(string fileName, string filePath, string fileType)



# Database

The Table where tasks are is “ETI\_UI\_Task”.



If you need to add a Market, it’s in ETI\_UI\_Market, then look at the Id and refer to it when adding a new task in the Task table. You can also add new market by adding a row in Market table, and refer the new Id in Task table.

