Markdown Monster

download latest chocolatey v2.8.5 downloads 122k

Markdown Monster is an easy to use and extensible **Markdown Editor**, **Viewer** and **Weblog Publisher** for Windows. Our goal is to provide the best Markdown specific editor for Windows and make it as easy as possible to create Markdown documents. We provide a core editor and previewer, and a number of non-intrusive helpers to help embed content like images, links, tables, code and more into your documents with minimal effort.

- See what's new
- License

> ### Show your Support
> If you like what you see here, please consider **starring this repo** (click the ★ in the top right corner of this page). If you have a favorite feature in Markdown Monster, it'd be awesome if you could tweet about it and mention @markdownmonstr. Please help us spread the word.

Installation

Download Installer

You can download Markdown Monster using the self-contained installer:

Download Markdown Monster

Chocolatey

You can you use Chocolatey to install from the Windows Command Prompt *(maintained by us - always up to date)*

```
c:\> choco install markdownmonster
```

WinGet

You can also use the built-in Windows WinGet tool *(not maintained by us)*:

```
c:\> winget install markdown-monster
```

Scoop

You can also use Scoop to install and update. You need to use the `extras` bucket. *(not maintained by us)*

```
c:\> scoop install markdown-monster
```

License

Markdown Monster is a licensed product and while we provide a fully functional, free download, **for continued use a reasonably priced license is required**.

For more detailed info on licensing please visit licensing information here:

Markdown Monster Licensing

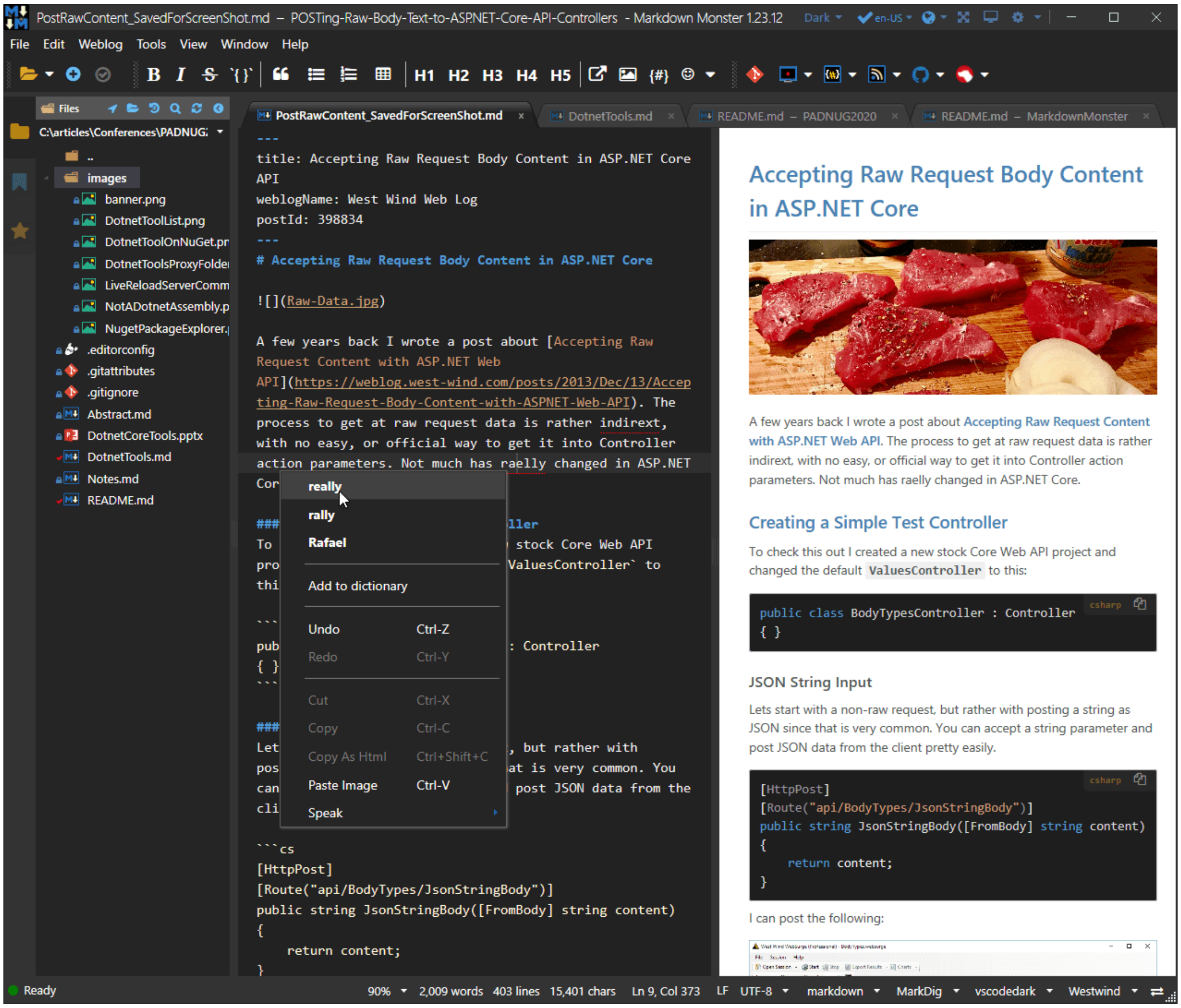
Source code no longer available here

Due to rampant license abuse and misuse of the source code by a few bad actors, this repository no longer holds any of Markdown Monster's source code. The source code has moved to a private repository and access is **available by request**.

This repository serves as the GitHub information page for downloads and features, as well as for feedback on bug reports and enhancement requests in the **Issues section**. For more general questions, you an also use our **support message board** in the Markdown Monster section.

Overview

Here's what Markdown Monster looks like using the default **Dark Theme**:



```
---
title: Accepting Raw Request Body Content in ASP.NET Core
API
weblogName: West Wind Web Log
postId: 398834
---
# Accepting Raw Request Body Content in ASP.NET Core

![](Raw-Data.jpg)

A few years back I wrote a post about [Accepting Raw
Request Content with ASP.NET Web
API](https://weblog.west-wind.com/posts/2013/Dec/13/Accep
ting-Raw-Request-Body-Content-with-ASPNET-Web-API). The
process to get at raw request data is rather indirext,
with no easy, or official way to get into Controller
action parameters. Not much has raelly changed in ASP.NET
Cor

###                                        ller
To                                       stock Core Web API
pro                                      ValuesController` to
thi

```
pub   : Controller
{ }

###
Let   , but rather with
pos   at is very common. You
can   post JSON data from the
cli

```cs
[HttpPost]
[Route("api/BodyTypes/JsonStringBody")]
public string JsonStringBody([FromBody] string content)
{
    return content;
}
```

Accepting Raw Request Body Content in ASP.NET Core

A few years back I wrote a post about Accepting Raw Request Content with ASP.NET Web API. The process to get at raw request data is rather indirext, with no easy, or official way to get into Controller action parameters. Not much has raelly changed in ASP.NET Core.

Creating a Simple Test Controller

To check this out I created a new stock Core Web API project and changed the default `ValuesController` to this:

```csharp
public class BodyTypesController : Controller
{ }
```

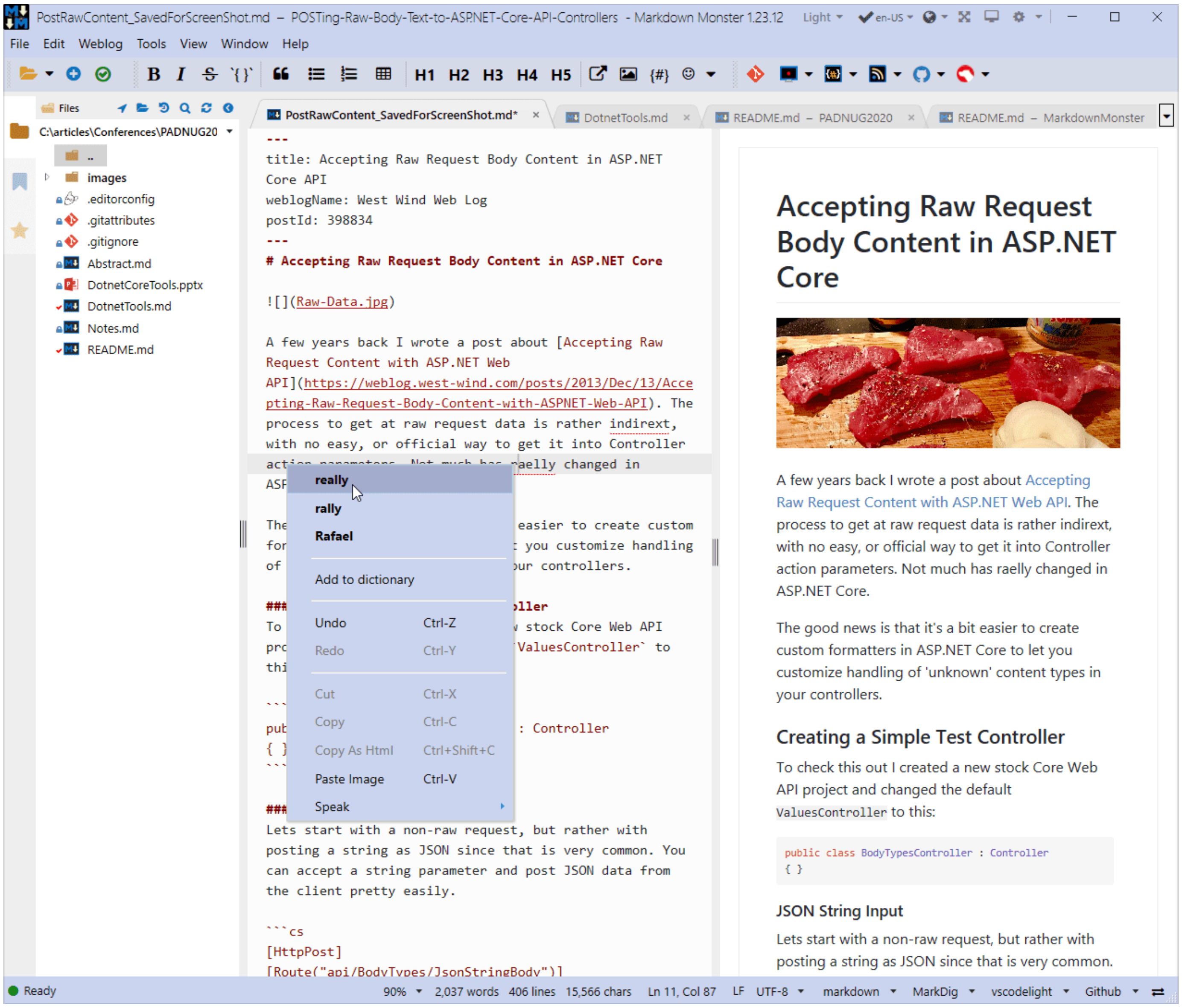
JSON String Input

Lets start with a non-raw request, but rather with posting a string as JSON since that is very common. You can accept a string parameter and post JSON data from the client pretty easily.

```csharp
[HttpPost]
[Route("api/BodyTypes/JsonStringBody")]
public string JsonStringBody([FromBody] string content)
{
    return content;
}
```

I can post the following:

and here is the **Light Theme**:



```
---
title: Accepting Raw Request Body Content in ASP.NET
Core API
weblogName: West Wind Web Log
postId: 398834
---
# Accepting Raw Request Body Content in ASP.NET Core

![](Raw-Data.jpg)

A few years back I wrote a post about [Accepting Raw
Request Content with ASP.NET Web
API](https://weblog.west-wind.com/posts/2013/Dec/13/Acce
pting-Raw-Request-Body-Content-with-ASPNET-Web-API). The
process to get at raw request data is rather indirext,
with no easy, or official way to get it into Controller
action parameters. Not much has raelly changed in
ASP

The                                  easier to create custom
for                                  you customize handling
of                                   our controllers.

###                                  oller
To                                   stock Core Web API
pro                                  ValuesController` to
thi

```
pub   : Controller
{ }

###
Lets start with a non-raw request, but rather with
posting a string as JSON since that is very common. You
can accept a string parameter and post JSON data from
the client pretty easily.

```cs
[HttpPost]
[Route("api/BodyTypes/JsonStringBody")]
```

Accepting Raw Request Body Content in ASP.NET Core

A few years back I wrote a post about Accepting Raw Request Content with ASP.NET Web API. The process to get at raw request data is rather indirext, with no easy, or official way to get it into Controller action parameters. Not much has raelly changed in ASP.NET Core.

The good news is that it's a bit easier to create custom formatters in ASP.NET Core to let you customize handling of 'unknown' content types in your controllers.

Creating a Simple Test Controller

To check this out I created a new stock Core Web API project and changed the default `ValuesController` to this:

```
public class BodyTypesController : Controller
{ }
```

JSON String Input

Lets start with a non-raw request, but rather with posting a string as JSON since that is very common.

Note that you can individually customize the theming for the editor, the preview and even the code snippet display in the previewer.

Productive

Markdown Monster has a lot of features, but we work hard to keep those features out of your way when you don't need them. If you just want to write, MM provides a minimal editing experience that provides pleasant white space around your content and an optional distraction free mode. But if you want to be more productive use shortcuts and helpers to enhance your editing experience, we help you with many useful tools to enhance your editing experience with thoughtful productivity features. Most commands have keyboard shortcuts and can be accomplished with one or two keystrokes.

Easily embed things like images, links, tables, code snippets and more with a couple of keystrokes and smart link fix ups. Because images are such a big part of content creation, our image embedding support is especially rich and supports pasting images from the clipboard, embedding with the image file/url dialog, drag and drop from explorer or a browser, and even a built-in image capture utility.

Customizable

Markdown Monster is highly customizable with themes for the main shell, editor and preview displays. The editor and preview themes can be easily customized using HTML and CSS based templates so you can make Markdown Monster your own. There are also many options to customize the editor's fonts, sizes and many display options.

You can also run in distraction free mode, that removes toolbars, menus, preview, sidebar and other distractions to let you focus on your code. Or use Presentation mode to focus on reading content without the editor view.

Extensible

Markdown Monster is extensible with .NET code. You can use our Snippets or Commander addins to create custom text expansions, or automate common tasks using .NET Code snippets. A full addin extensibility model allows deep integration with most of Markdown Monster's features and UI to create sophisticated extensions using a straightforward Addin model. There is an Addin Manager that lets you easily install, update and manage available Addins.

Weblog Publishing

If you have a blog chances are you can use Markdown Monster to create your content and publish it to your blogging service. Markdown Monster can publish to many common Weblog engines including WordPress, Medium and any service that use MetaWeblog API. MM automatically handles converting your Markdown content to HTML, fixing up links and publishing and re-publishing your Markdown content cleanly to blogging services.

The **New Weblog Post** feature automatically sets up a post folder in a Weblog Posts directory hierarchy to let you organize posts in a shared DropBox or OneDrive (if available) folder so you can easily work on posts from multiple machines.

Markdown Monster's integrated Git features also work great for any Git based service like Jekyll, Ghost, Hugo or Wyam to write your posts locally and immediately push them to your remote Git repository to publish. Git integration is built into MM to commit your changes either individually or in batch or you can open your favorite Git Client to publish your changes to your hosted content blog.

Links

- Markdown Monster Site
- What's New Change Log
- Documentation
- Bug Reports & Feature Requests
- Support Forum for Questions & Discussions
- License
- Follow @MarkdownMonstr on Twitter

Addins

- Markdown Monster Addin Registry
- Create Addins with .NET

> ### Please report any Issues you run into!

> If you run into a problem with Markdown Monster, **please** let us know by filing an issue or feature request here on GitHub. We want to know what doesn't work and get it fixed. **Help us make Markdown Monster better**! We know your time is valuable, but we really appreciate any feedback.

Features

Markdown Monster provides many useful features:

Markdown Editor

- Syntax highlighted Markdown editing
- Live and synced HTML preview
- Gentle, optional toolbar support for Markdown newbies
- Inline spell checking
- Line and Word counts
- Synced Document Outline
- Distraction free mode
- Markdown folding
- Split view

Previewer

- Scroll synced preview window
- Optional external previewer for multi-screen
- Preview in Web Browser
- Presentation mode support
- Distraction-free mode support
- Document Navigation from embedded Markdown Links

Image Features

- Paste images from Clipboard
- Smartly select and embed images from disk or URL
- Drag images from Folder Browser
- Drag images from Explorer
- Edit images in your image editor of choice
- Built-in screen capture
- Automatic image compression on pasted images

Editing Features

- Easy link embedding from clipboard or disk

- Embed code snippets and see highlighted syntax coloring
- Two-way table editor for interactively creating and editing tables
- Text Snippet Expansion with C# Code via **Snippets Addin**
- Embed Emojii
- Smart, unobtrusive toolbar and shortcut key helpers
- Snippet expansion from text templates
- Document Outline to navigate large documents
- Many Editor customization options

Output and Selections

- Save rendered output to self-contained HTML or HTML Fragment
- Save rendered output to PDF
- Copy Markdown selection as HTML
- Paste HTML text as Markdown
- Open rendered output in your favorite Web browser
- Print rendered output to the printer or PDF driver
- Generate and embed document Table of Contents

Theme Support

- Dark and Light application themes
- Customizable Editor Themes
- Customizable Preview Themes
- Customizable output syntax coloring themes
- Each type of theme can be individually applied
- Use HTML and CSS to customize Preview and Editor Themes

File Operations

- Editor remembers open documents by default (optional)
- Auto-Save and Auto-Backup support
- Many file operations on each file
    - Shell Viewer
    - Open With...
    - Edit in appropriate editors
    - View/Edit Images in configured apps
    - Compress images
    - Commit to Git
    - Open on Github (if Github repo)
- Save files with encryption

- Drag and drop documents from Explorer and Folder Browser
- Open a Terminal, Explorer or Git Client

Organization and File Access

- Integrated File and Folder browser
- Quick File Search
- Find in Files (search files and content)
- Favorites Sidebar - save, organize and search
- Group files into Projects
- Drag and Drop files everywhere

Git Integration

- Show Git Status in Folder Browser
- Commit and push Dialog
- Commit and push active file, folder browser file
- Commit and push all pending changes
- Compare changes in configured Git Diff client
- Undo Changes
- Add Ignored Files
- Clone Repository
- Open in Git Client

Weblog Publishing

- Create or edit Weblog posts using Markdown
- Publish your Markdown directly to your blog
- Re-publish posts at any time
- Post data stored as YAML metadata in Markdown
- Send custom meta data with posts
- Supports MetaWebLog, Wordpress and Medium (limited)
- Supports document based blogs (Jekyll, Hugo, Wyam, Ghost etc.)
- Download and edit existing posts
- Very fast publish and download process
- Support for multiple blogs
- Dropbox and OneDrive shared post storage

Non Markdown Features

- HTML file editing with live preview
- Many other file formats can also be edited:
  JSON, XML, CSS, JavaScript, Typescript, FoxPro, CSharp and more

- Optional shared configuration on Cloud drives
- High DPI Monitor Aware

Command Line features

- Use `mm` or `markdown` to launch Markdown Monster
- Markdown Monster path added to user path
- `mm readme.md` - open single file
- `mm readme.md changelog.md` - open multiple files
- `mm .` - open folder browser in current folder
- `mm reset` - reset all Markdown Monster settings
- `mm uninstall` - remove all non-local system settings

Extensibility

- Automate Markdown Monster with C# using the Commander Addin
- Create Addins with .NET code
- Visual Studio Project Template available
- Simple interface, easy to implement
- Access UI, menu and active documents
- Access document and application lifecycle events
- Add Custom Markdown Parsers
- Replace the Preview Rendering Engine
- Add Tabs to left and right sidebar panels
- Some published addins available:
    - Console: A pinned Terminal Window
    - Commander: C# based Script Automation
    - Gist: Open from and Save As Gists, and Paste Code as Gist
    - Save Image to Azure Blob Storage

Why another Markdown Editor?

Markdown is everywhere these days, and it's becoming a favorite format for many developers, writers and documentation experts to create lots of different kinds of content in this format. Markdown is used in a lot of different places:

- Source Code documentation files (like this one)
- Weblog posts
- Product documentation
- Message Board message entry
- Application text entry for formatted text

Personally I use Markdown for my Weblog, my message board, of course on GitHub and in a number of applications that have free form text fields that allow for formatted text - for example in our Webstore product descriptions are in Markdown.

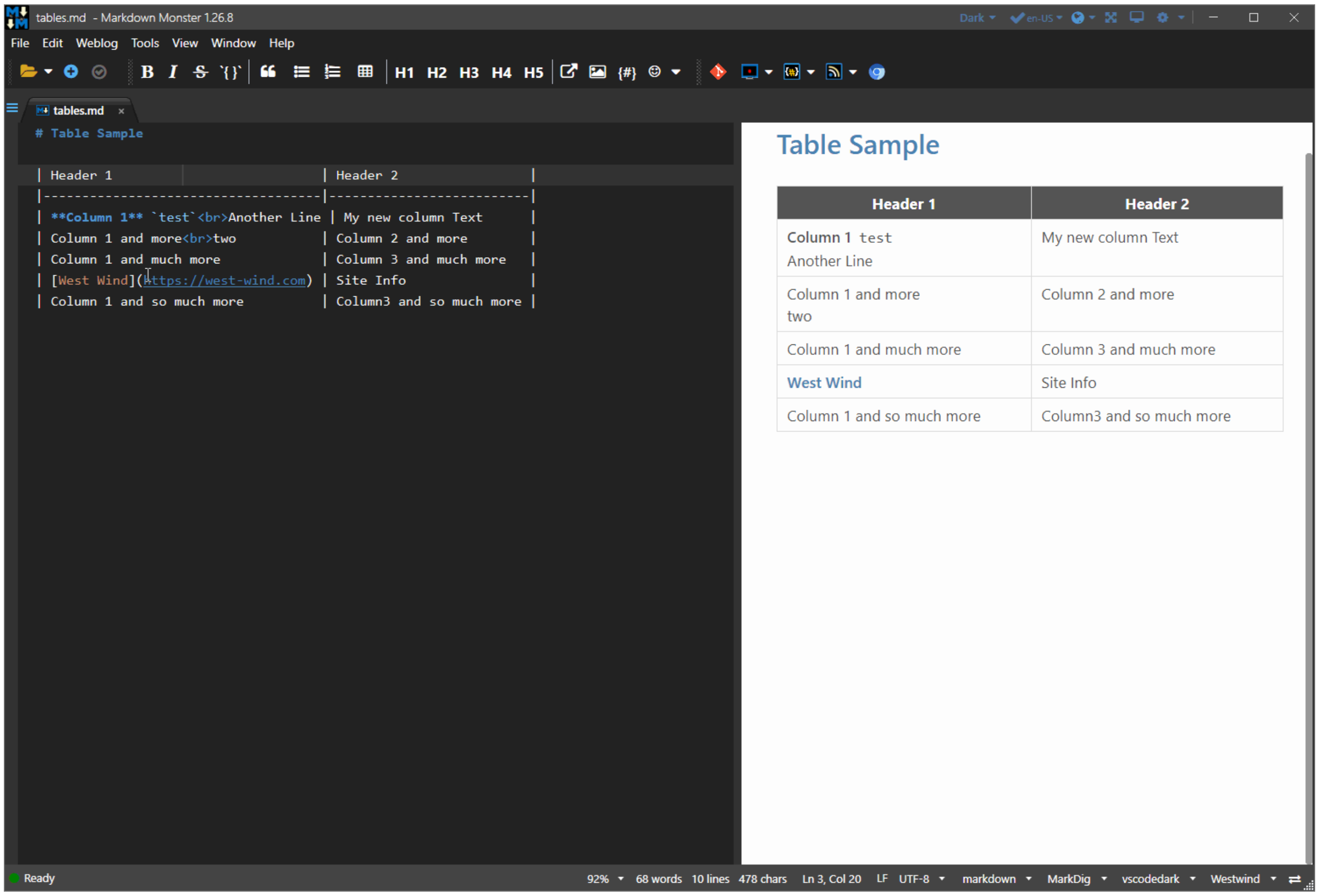
Having a dedicated Markdown Editor that gets out of your way, yet provides a few helpful features **and lets you add custom features** that make your content creation sessions more productive is important. **Check out this post** on why it makes sense to use a dedicated Markdown Editor rather than a generic text editor for Markdown document creation. The ability to easily publish your Markdown to any MetaWebLog or Wordpress API endpoint is also useful as it allows you to easily publish to blogs or any application that supports for either for these formats.

Markdown Monster wants to eat your Markdown!

Markdown Monster is a Markdown editor and Viewer for Windows that lets you create edit or simply preview Markdown text. It provides basic editing functionality with a few nice usability features for quickly embedding images, links, code, tables, screen shots and other markup. You get a responsive text editor that's got you covered with Markdown syntax highlighting, a collapsible live preview, so you can see what your output looks like, inline spellchecking and a handful of optimized menu options that help you mark up your text and embed and link content into your Markdown document. Additionally utility features let you quickly jump to the command line or an Explorer window, commit a document to Git, or even edit images in your favorite image editor.

Table Editor

Markdown Monster includes a powerful two-way table editor that lets you interactively edit content in tables. You can use the rich table editor to enter your Table content, and re-open Markdown table code in the editor for later editing and re-formatting.



```
# Table Sample

| Header 1                         | Header 2                |
|----------------------------------|-------------------------|
| **Column 1** `test`<br>Another Line | My new column Text   |
| Column 1 and more<br>two         | Column 2 and more       |
| Column 1 and much more           | Column 3 and much more  |
| [West Wind](https://west-wind.com) | Site Info             |
| Column 1 and so much more        | Column3 and so much more |
```

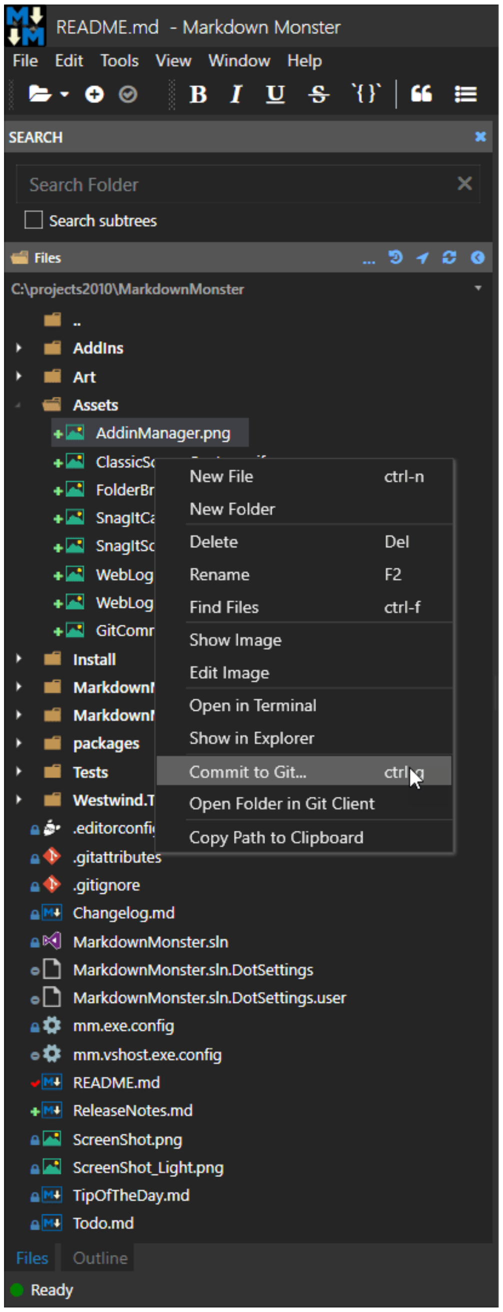
Table Sample

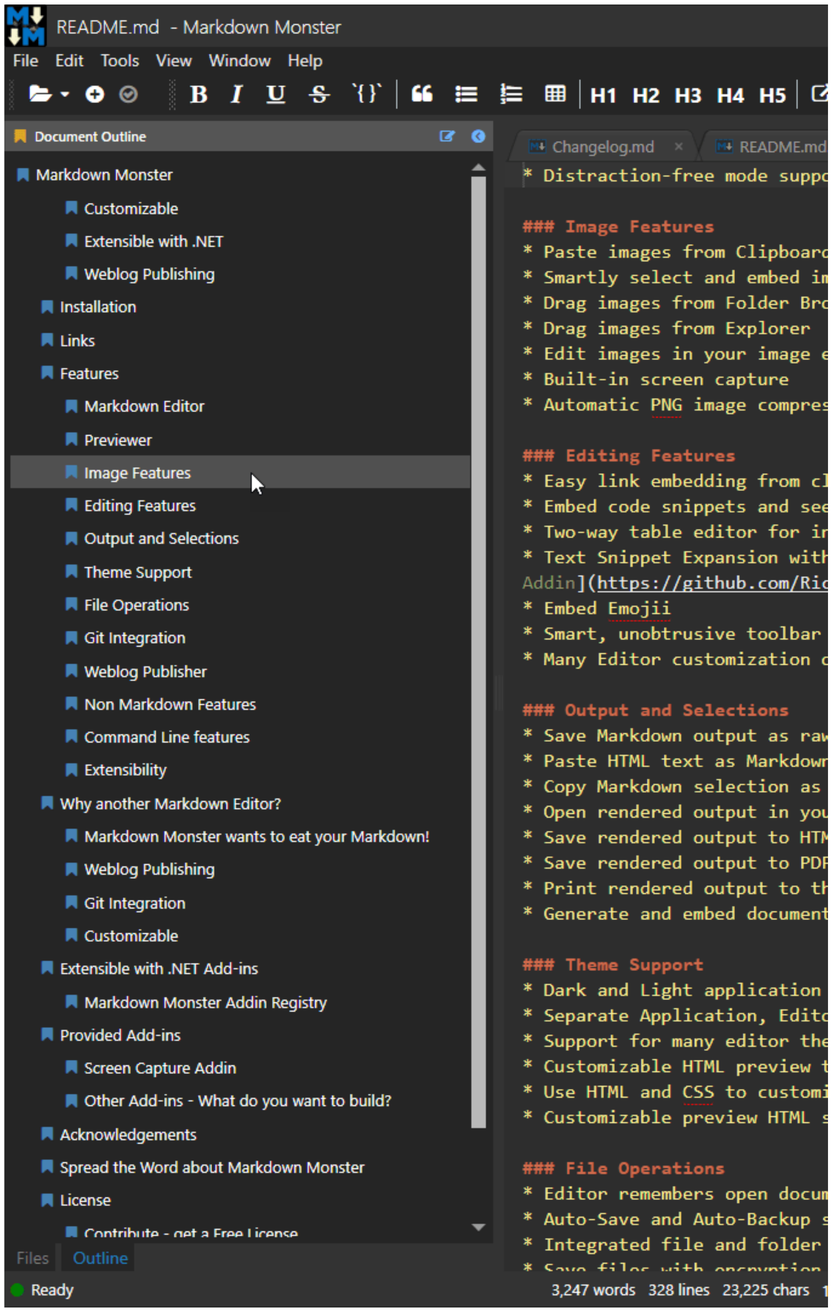
Header 1	Header 2
Column 1 `test` Another Line	My new column Text
Column 1 and more two	Column 2 and more
Column 1 and much more	Column 3 and much more
West Wind	Site Info
Column 1 and so much more	Column3 and so much more

Folder Browser

Markdown editing often requires managing related content like images, or multiple files for links etc. and Markdown Monster provides a folder browser to see and navigate files while in the editor. The browser has built-in Git status support so you can immediately see what documents have changed in your folder tree and you can easily review and undo changes. The browser lets you navigate folders, create, delete and move files and there are many options to view, open and edit files.

The folder browser also supports searching, moving and renaming of files as you would expect of a file browser.





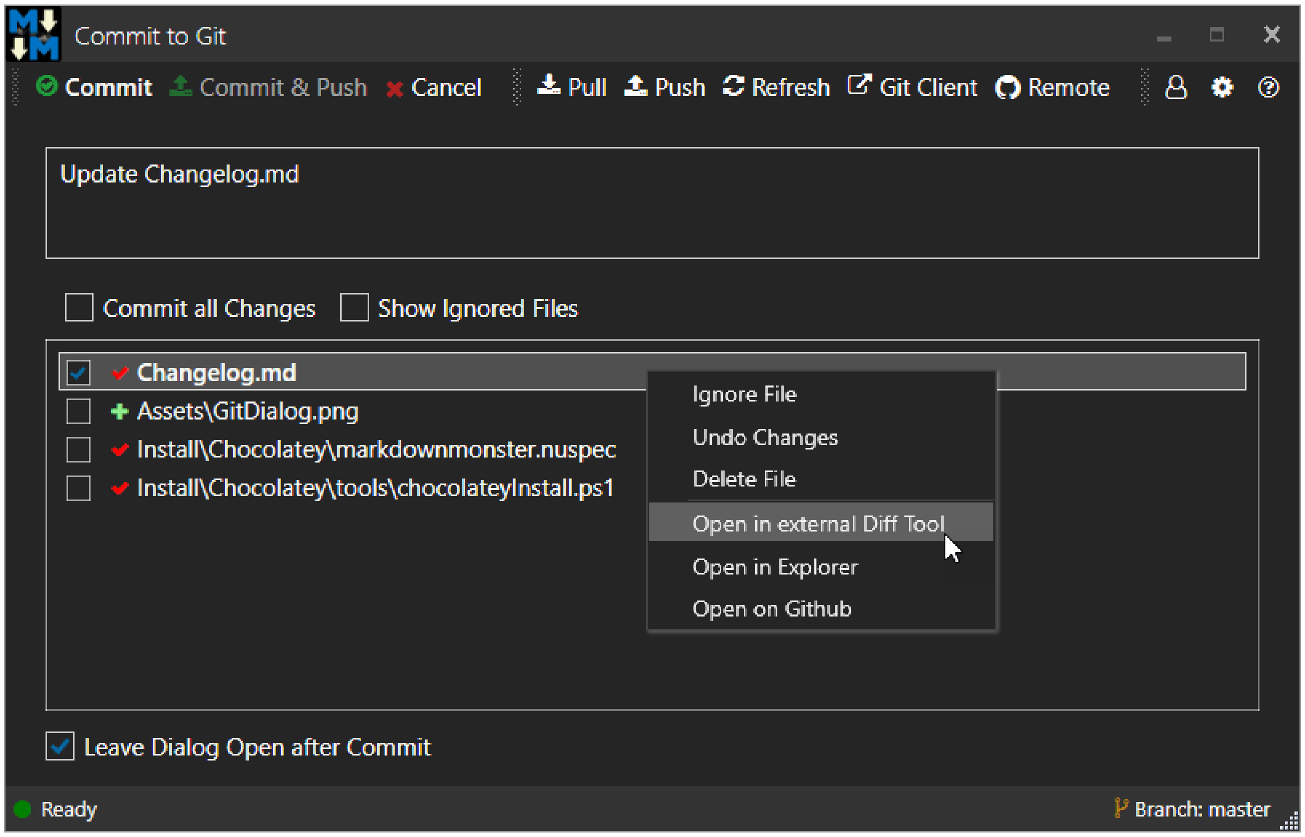
```
* Distraction-free mode suppo

### Image Features
* Paste images from Clipboar
* Smartly select and embed im
* Drag images from Folder Bro
* Drag images from Explorer
* Edit images in your image e
* Built-in screen capture
* Automatic PNG image compres

### Editing Features
* Easy link embedding from cl
* Embed code snippets and see
* Two-way table editor for in
* Text Snippet Expansion with
Addin](https://github.com/Ric
* Embed Emojii
* Smart, unobtrusive toolbar
* Many Editor customization o

### Output and Selections
* Save Markdown output as raw
* Paste HTML text as Markdow
* Copy Markdown selection as
* Open rendered output in you
* Save rendered output to HTM
* Save rendered output to PDF
* Print rendered output to th
* Generate and embed document

### Theme Support
* Dark and Light application
* Separate Application, Edito
* Support for many editor the
* Customizable HTML preview t
* Use HTML and CSS to customi
* Customizable preview HTML s

### File Operations
* Editor remembers open docum
* Auto-Save and Auto-Backup s
* Integrated file and folder
```

Document Outline

When working with long documents it's important to have an easy way to navigate the document's structure quickly, and the Document Outline makes it easy to see the document's structure at a glance and jump to any section quickly. The outline also lets you pick up document Ids for quickly embedding same-document links.

Git Integration

These days working with Markdown often means working with Git repositories and Markdown Monster integrates common Git tasks that you need to perform during editing. You can easily review, commit and push changes, as well as cloning and creating of new repositories.



Update Changelog.md

Commit all Changes   Show Ignored Files

- [✓] Changelog.md
- Assets\GitDialog.png
- Install\Chocolatey\markdownmonster.nuspec
- Install\Chocolatey\tools\chocolateyInstall.ps1

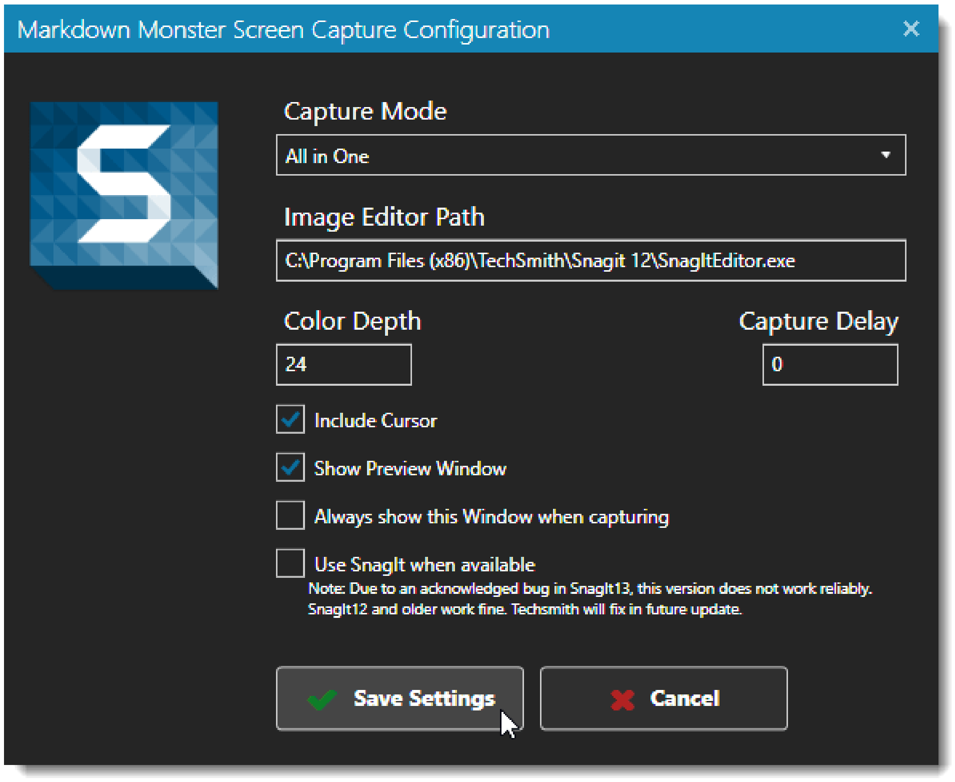
[✓] Leave Dialog Open after Commit

Screen Captures

The Screen Capture addin supports two separate capture modes:

- Techsmith's popular and versatile **SnagIt** Screen Capture Tool
- Built-in Screen Capture Addin

To capture, simply click the capture button (camera icon) and the main app minimizes and either SnagIt or the integrated capture tool pops up to let you select the object to capture. With SnagIt you can use most of SnagIt's native capture modes, for the built-in tool you can select Windows and Controls on the screen. Captures are previewed and have options for editing, and when finished the captured image is embedded and linked directly into the Markdown content.


Markdown Monster Screen Capture Configuration

Capture Mode: All in One

Image Editor Path: C:\Program Files (x86)\TechSmith\Snagit 12\SnagItEditor.exe

Color Depth: 24

Capture Delay: 0

[✓] Include Cursor

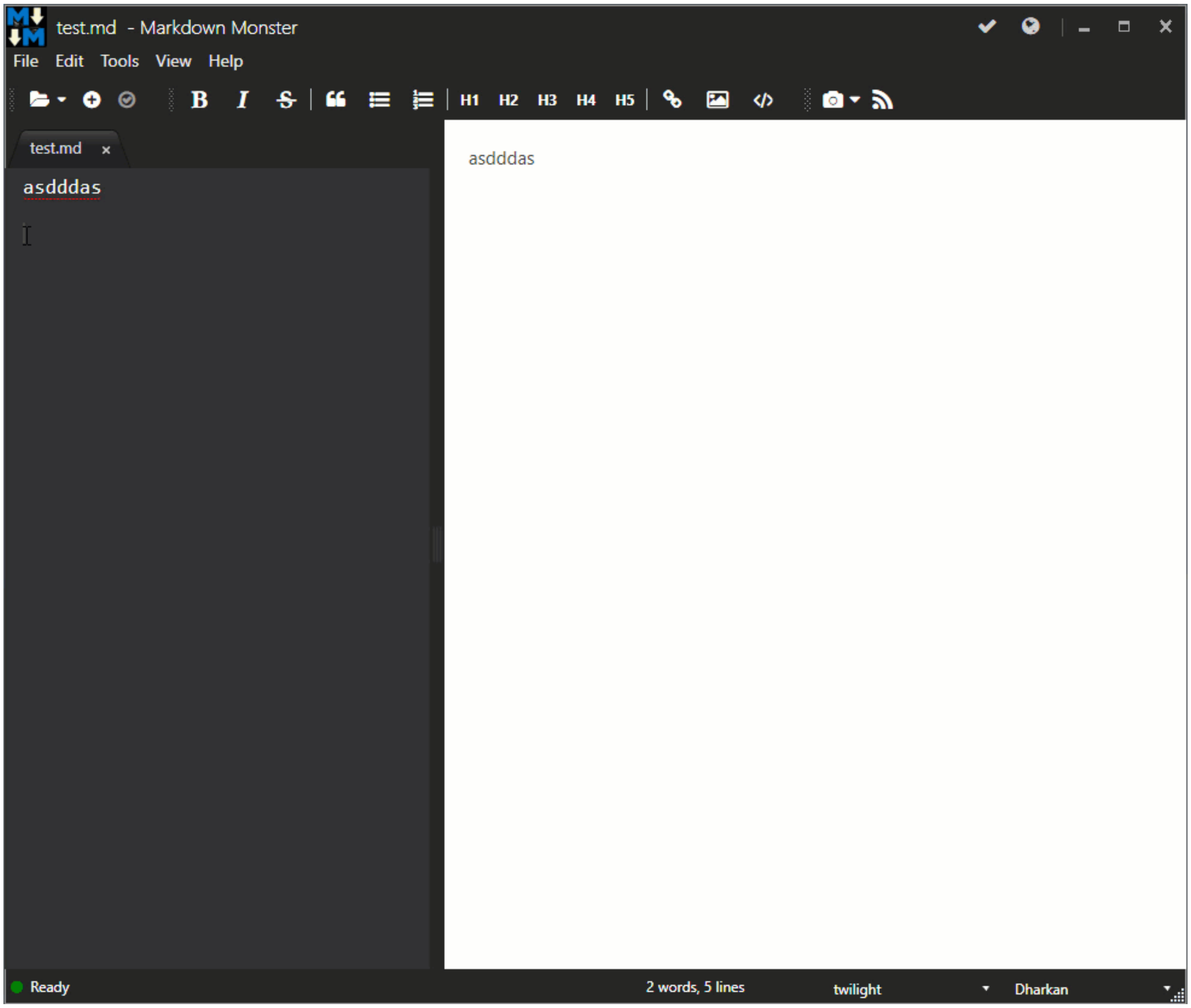
[✓] Show Preview Window

[ ] Always show this Window when capturing

[ ] Use SnagIt when available

Note: Due to an acknowledged bug in SnagIt13, this version does not work reliably. SnagIt12 and older work fine. Techsmith will fix in future update.

Here's the **SnagIt Screen Capture** in action:



asdddas

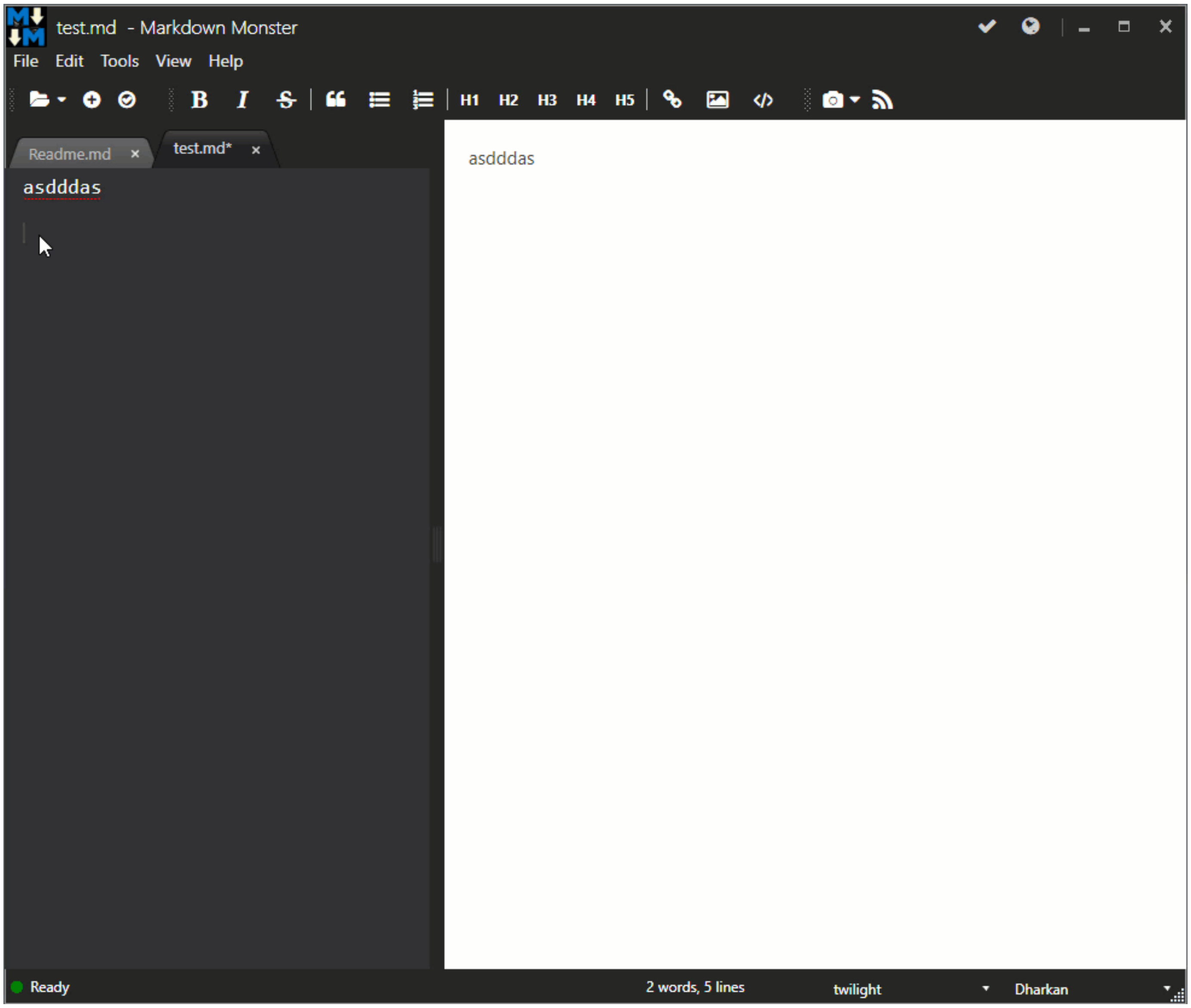
asdddas

2 words, 5 lines

twilight

Dharkan

And here is the self-contained, built-in Screen Capture Module:

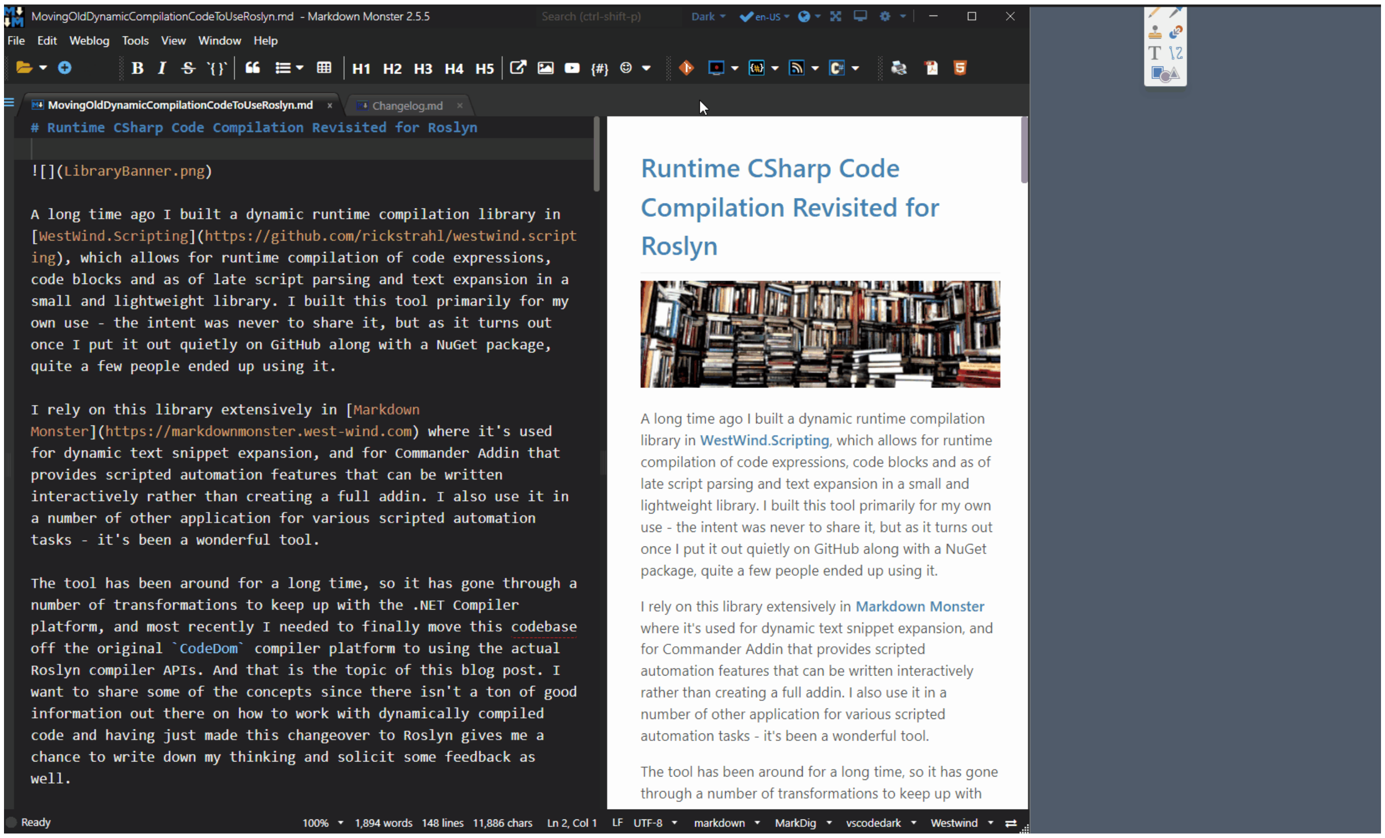


asdddas

asdddas

Template Expansions

The built-in Snippets addin allows you to create templates - including dynamic templates using C# code - to expand text either on a typed expansion keyword, via interactive selection or via a pre-defined shortcut key.


MovingOldDynamicCompilationCodeToUseRoslyn.md - Markdown Monster 2.5.5
Search (ctrl-shift-p)
Dark
en-US
File Edit Weblog Tools View Window Help
H1 H2 H3 H4 H5
MovingOldDynamicCompilationCodeToUseRoslyn.md
Changelog.md
Ready
100% 1,894 words 148 lines 11,886 chars Ln 2, Col 1 LF UTF-8 markdown MarkDig vscodedark Westwind

```
# Runtime CSharp Code Compilation Revisited for Roslyn

![](LibraryBanner.png)

A long time ago I built a dynamic runtime compilation library in
[Westwind.Scripting](https://github.com/rickstrahl/westwind.script
ing), which allows for runtime compilation of code expressions,
code blocks and as of late script parsing and text expansion in a
small and lightweight library. I built this tool primarily for my
own use - the intent was never to share it, but as it turns out
once I put it out quietly on GitHub along with a NuGet package,
quite a few people ended up using it.

I rely on this library extensively in [Markdown
Monster](https://markdownmonster.west-wind.com) where it's used
for dynamic text snippet expansion, and for Commander Addin that
provides scripted automation features that can be written
interactively rather than creating a full addin. I also use it in
a number of other application for various scripted automation
tasks - it's been a wonderful tool.

The tool has been around for a long time, so it has gone through a
number of transformations to keep up with the .NET Compiler
platform, and most recently I needed to finally move this codebase
off the original `CodeDom` compiler platform to using the actual
Roslyn compiler APIs. And that is the topic of this blog post. I
want to share some of the concepts since there isn't a ton of good
information out there on how to work with dynamically compiled
code and having just made this changeover to Roslyn gives me a
chance to write down my thinking and solicit some feedback as
well.
```

Runtime CSharp Code Compilation Revisited for Roslyn

A long time ago I built a dynamic runtime compilation library in WestWind.Scripting, which allows for runtime compilation of code expressions, code blocks and as of late script parsing and text expansion in a small and lightweight library. I built this tool primarily for my own use - the intent was never to share it, but as it turns out once I put it out quietly on GitHub along with a NuGet package, quite a few people ended up using it.

I rely on this library extensively in Markdown Monster where it's used for dynamic text snippet expansion, and for Commander Addin that provides scripted automation features that can be written interactively rather than creating a full addin. I also use it in a number of other application for various scripted automation tasks - it's been a wonderful tool.

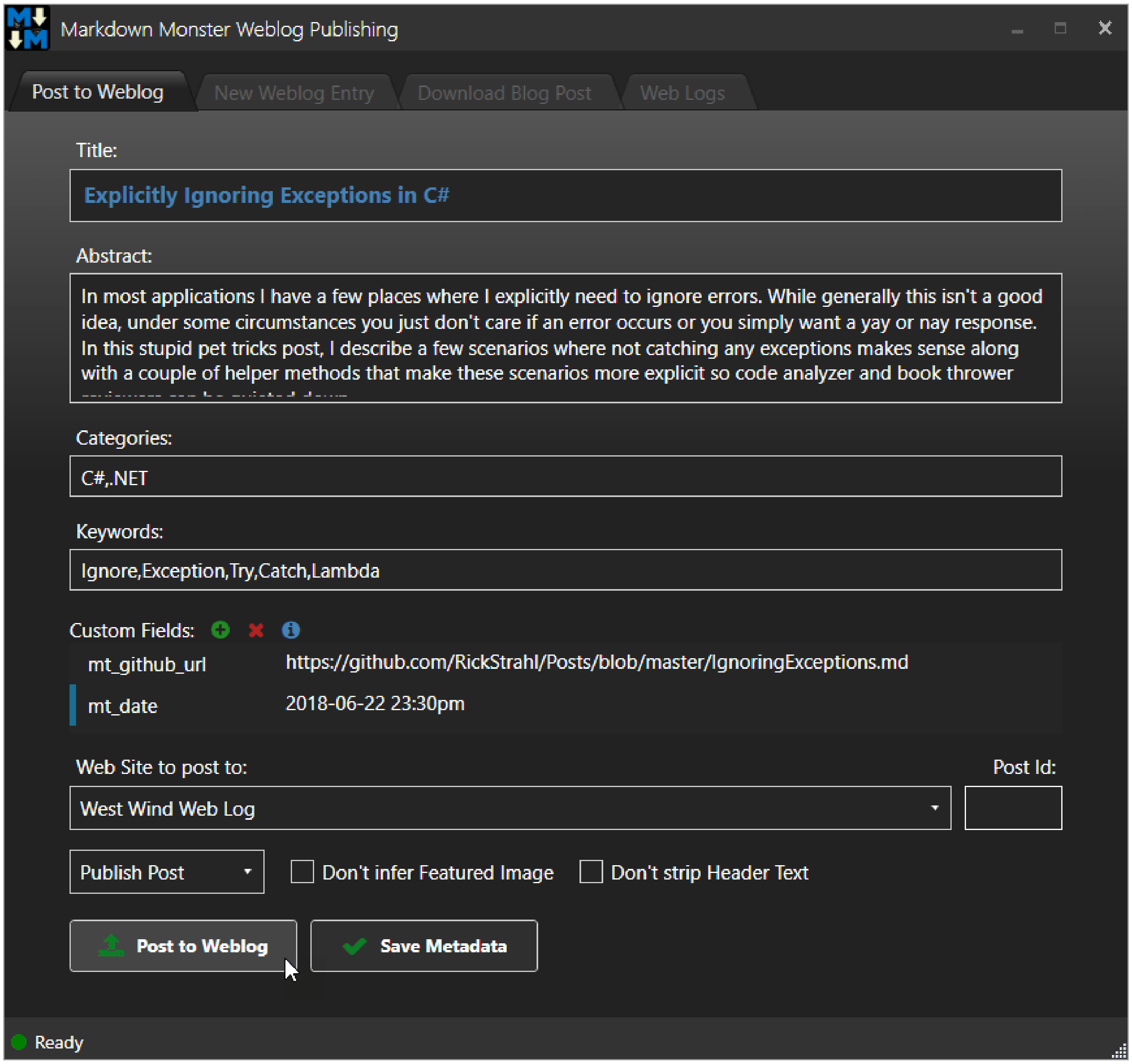
The tool has been around for a long time, so it has gone through a number of transformations to keep up with

Weblog Publishing

A common use case for Markdown is to create rich blog posts with embedded links and content and Markdown Monster makes it easy to pull together content from various sources. You can easily embed images by pasting from the the clipboard, or by linking images from URLs or files or using the built-in Screen Capture addin or SnagIt support.

Writing long blog posts and articles, and coordinating lots of related content is one of the main reasons I built this tool in the first place as is publishing and updating content easily to various types of blogs and content generators.

You can take any Markdown and turn it into a blog post by using the Weblog publishing feature. If you use WordPress, a MetaWeblog API Blog, Medium or Jekyll, you can click the Weblog button on the toolbar and set up your blog (MetaWebLog, WordPress or Medium), and then specify the Weblog specifics like title, abstract, tags and Web Site to publish to. Posts can be easily published and also re-published with the click of a button. You can also download existing blog posts from your blog and edit them as Markdown and then republish them.



Title:

Explicitly Ignoring Exceptions in C#

Abstract:

In most applications I have a few places where I explicitly need to ignore errors. While generally this isn't a good idea, under some circumstances you just don't care if an error occurs or you simply want a yay or nay response. In this stupid pet tricks post, I describe a few scenarios where not catching any exceptions makes sense along with a couple of helper methods that make these scenarios more explicit so code analyzer and book thrower

Categories:

C#,.NET

Keywords:

Ignore,Exception,Try,Catch,Lambda

Custom Fields:

mt_github_url	https://github.com/RickStrahl/Posts/blob/master/IgnoringExceptions.md
mt_date	2018-06-22 23:30pm

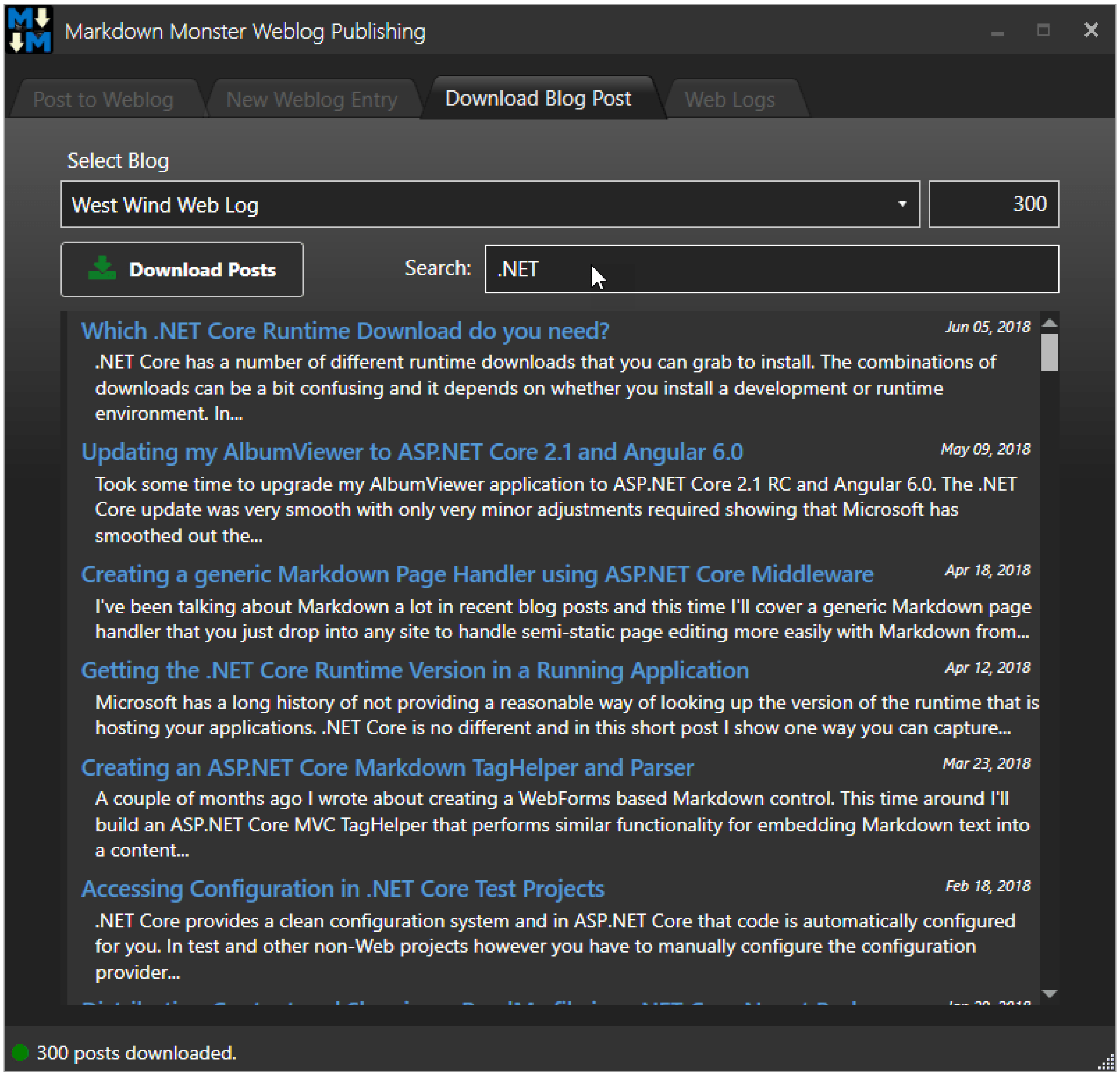
Web Site to post to: West Wind Web Log

Post Id:

Publish Post

Don't infer Featured Image

Don't strip Header Text



West Wind Web Log | 300

Search: .NET

Which .NET Core Runtime Download do you need? — Jun 05, 2018
.NET Core has a number of different runtime downloads that you can grab to install. The combinations of downloads can be a bit confusing and it depends on whether you install a development or runtime environment. In...

Updating my AlbumViewer to ASP.NET Core 2.1 and Angular 6.0 — May 09, 2018
Took some time to upgrade my AlbumViewer application to ASP.NET Core 2.1 RC and Angular 6.0. The .NET Core update was very smooth with only very minor adjustments required showing that Microsoft has smoothed out the...

Creating a generic Markdown Page Handler using ASP.NET Core Middleware — Apr 18, 2018
I've been talking about Markdown a lot in recent blog posts and this time I'll cover a generic Markdown page handler that you just drop into any site to handle semi-static page editing more easily with Markdown from...

Getting the .NET Core Runtime Version in a Running Application — Apr 12, 2018
Microsoft has a long history of not providing a reasonable way of looking up the version of the runtime that is hosting your applications. .NET Core is no different and in this short post I show one way you can capture...

Creating an ASP.NET Core Markdown TagHelper and Parser — Mar 23, 2018
A couple of months ago I wrote about creating a WebForms based Markdown control. This time around I'll build an ASP.NET Core MVC TagHelper that performs similar functionality for embedding Markdown text into a content...

Accessing Configuration in .NET Core Test Projects — Feb 18, 2018
.NET Core provides a clean configuration system and in ASP.NET Core that code is automatically configured for you. In test and other non-Web projects however you have to manually configure the configuration provider...

300 posts downloaded.

Customizable

Most editing and UI features in Markdown Monster are optional and can be turned on and off. Want to work distraction free and see no preview or spell checking hints? You can turn them off. Want to store configuration data in a shared cloud folder? You can do that too.

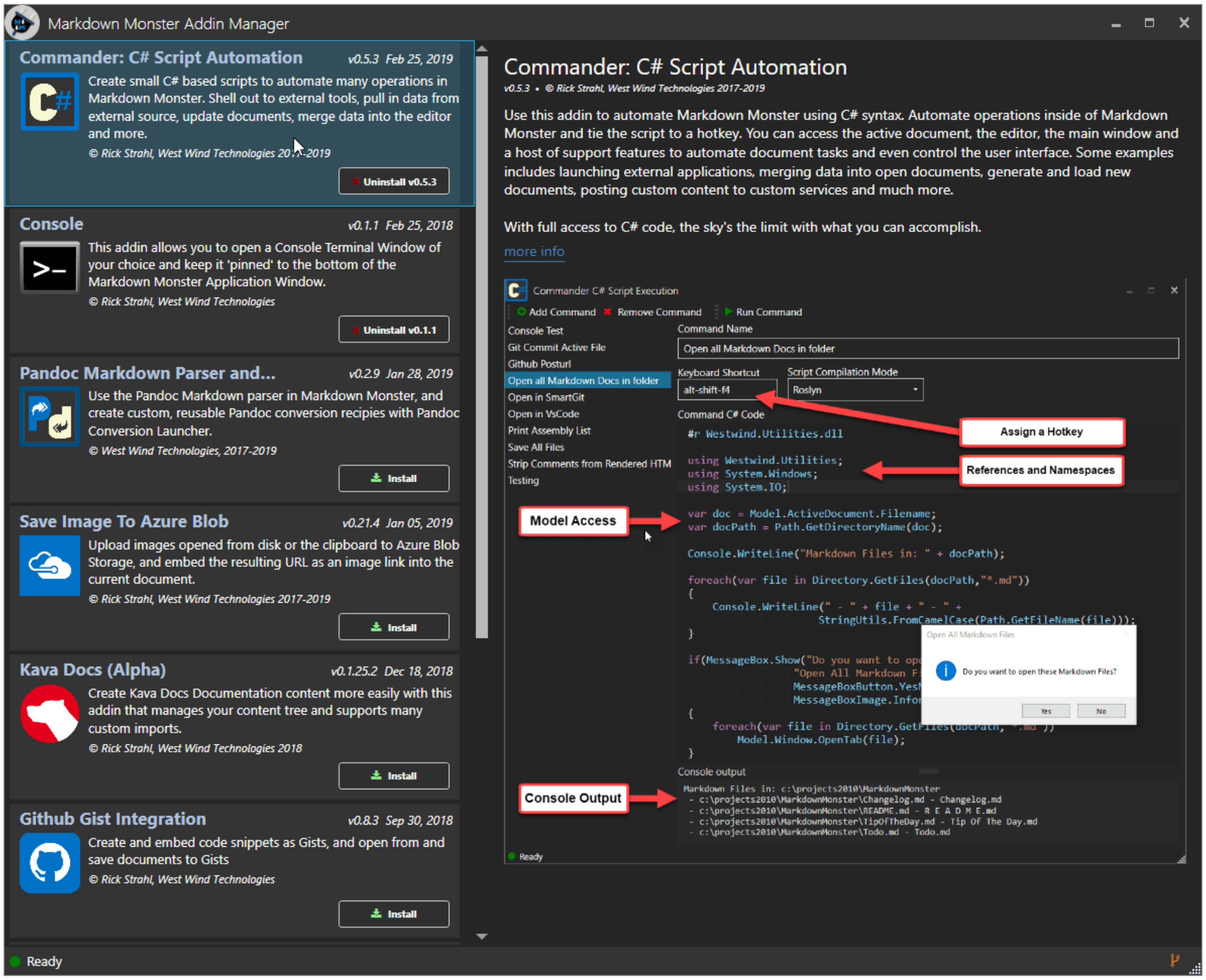
Want a different editor theme than the dark default or a preview theme that matches your blog or branding? You can easily switch to one of the many built-in editor themes. For previews you can use either one of several built-in themes or add your own with a simple, plain HTML/CSS template. You can even create themes that link to your own online styles.

The editor and previewer are HTML and JavaScript based, so you can also apply any custom styling and even hook up custom JavaScript code if you want to get fancy beyond the basic configuration. The preview themes are easy to modify as they are simply HTML and CSS templates.

Extensible with .NET Add-ins

One of the **key feature** and the main reason I built Markdown Monster is that it is **extensible**, so that you and I can plug additional functionality into it without bloating the main product.

You can find available public Addins you can install in the **Markdown Monster Addin Manager** from the **Tools** menu:



Commander: C# Script Automation v0.5.3 Feb 25, 2019
Create small C# based scripts to automate many operations in Markdown Monster. Shell out to external tools, pull in data from external source, update documents, merge data into the editor and more.
© Rick Strahl, West Wind Technologies 2017-2019

Console v0.1.1 Feb 25, 2018
This addin allows you to open a Console Terminal Window of your choice and keep it 'pinned' to the bottom of the Markdown Monster Application Window.
© Rick Strahl, West Wind Technologies

Pandoc Markdown Parser and... v0.2.9 Jan 28, 2019
Use the Pandoc Markdown parser in Markdown Monster, and create custom, reusable Pandoc conversion recipies with Pandoc Conversion Launcher.
© West Wind Technologies, 2017-2019

Save Image To Azure Blob v0.21.4 Jan 05, 2019
Upload images opened from disk or the clipboard to Azure Blob Storage, and embed the resulting URL as an image link into the current document.
© Rick Strahl, West Wind Technologies 2017-2019

Kava Docs (Alpha) v0.1.25.2 Dec 18, 2018
Create Kava Docs Documentation content more easily with this addin that manages your content tree and supports many custom imports.
© Rick Strahl, West Wind Technologies 2018

Github Gist Integration v0.8.3 Sep 30, 2018
Create and embed code snippets as Gists, and open from and save documents to Gists
© Rick Strahl, West Wind Technologies

Commander: C# Script Automation
v0.5.3 • © Rick Strahl, West Wind Technologies 2017-2019

Use this addin to automate Markdown Monster using C# syntax. Automate operations inside of Markdown Monster and tie the script to a hotkey. You can access the active document, the editor, the main window and a host of support features to automate document tasks and even control the user interface. Some examples includes launching external applications, merging data into open documents, generate and load new documents, posting custom content to custom services and much more.

With full access to C# code, the sky's the limit with what you can accomplish.

more info

Console Test
Git Commit Active File
Github Posturl
Open all Markdown Docs in folder
Open in SmartGit
Open in VsCode
Print Assembly List
Save All Files
Strip Comments from Rendered HTM
Testing

Command Name: Open all Markdown Docs in folder
Keyboard Shortcut: alt-shift-f4
Script Compilation Mode: Roslyn
Command C# Code

```
#r Westwind.Utilities.dll

using Westwind.Utilities;
using System.Windows;
using System.IO;

var doc = Model.ActiveDocument.Filename;
var docPath = Path.GetDirectoryName(doc);

Console.WriteLine("Markdown Files in: " + docPath);

foreach(var file in Directory.GetFiles(docPath,"*.md"))
{
    Console.WriteLine(" - " + file + " - " +
                      StringUtils.FromCamelCase(Path.GetFileName(file)));
}

if(MessageBox.Show("Do you want to ope
                   "Open All Markdown F
                   MessageBoxButton.YesN
                   MessageBoxImage.Infor
{
    foreach(var file in Directory.GetFiles(docPath, "*.md"))
        Model.Window.OpenTab(file);
}
```

Console output

```
Markdown Files in: c:\projects2010\MarkdownMonster
 - c:\projects2010\MarkdownMonster\Changelog.md - Changelog.md
 - c:\projects2010\MarkdownMonster\README.md - R E A D M E.md
 - c:\projects2010\MarkdownMonster\TipOfTheDay.md - Tip Of The Day.md
 - c:\projects2010\MarkdownMonster\Todo.md - Todo.md
```

There is a variety of functionality available in addins. Here are a few examples:

Right now the registry is pretty sparse, but here are a few Addins you can check out:

- **Console**
  Lets you pin an always-active Terminal Console window to the bottom of Markdown Monster.
- **Commander C# Scripting**
  A C# based scripting addin that lets you automate tasks using script code. For simple tasks this is quicker and easier than creating a full addin.
- **Save Image to Azure Blob Storage**
  Lets you save images as Azure Blob storage items and embeds a link to the an uploaded resoure into your Markdown.
- **Gist**

This addin allows you to **Open from Gist** and **Save as Gist** as well as letting you create code snippets and embed them as Gists into your Markdown content.

- **Pandoc Markdown Parser**
  This addin provides a PanDoc Markdown processor that can be used instead of the default MarkDig parser. This addin also provides a host of document conversion options to convert your Markdown to PDF, DOC, EPub and a few other formats using an interactive dialog.

.NET Based Extensibility

Markdown Monster Addins have access to an add-in model that lets you manipulate and automate any open documents and the editor, lets you load new documents, launch external processes, add menu options and other UI features, open a new sidebar, generally interact with the entire UI and attach to life cycle events to get notifications of various application events like documents opening and closing, documents being saved and the application shutting down, etc...

Complexity of Addins can vary greatly from very simple automation tools like the Console Addin that simply pins a Terminal window to Markdown Monster, or something as complex as the KavaDocs addin that manages an entire documentation application with many custom windows and sidebars that are integrated into Markdown Monster.

Addins have access to most features of Markdown Monster and they are fairly easy to create. We as well as several third parties have created a number of useful addins using the powerful addin model and I encourage you to browse addins to see what you can do.

Creating Addins

One of the key features of Markdown Monster is that you can also create your own addins using the .NET based Markdown Monster Addin model. It's very easy to create new addins and we provide a **Visual Studio Project Template Extension** to facilitate the process of getting started with creating an Addin. All of our plug-ins are also available on GitHub, so you can easily check out how other addins were created.

You can find documentation for creating Addins here:

- **Creating a Markdown Monster Addin**
- **Markdown Monster Addin Visual Studio Project Template**
- **Accessing and Manipulating the Active Editor**
- **Bringing up UI from your Addin**

Markdown Monster Addin Registry

You can create addins for your own use, simply by copying them into the `%appdata%\Addins` folder, or if you created an Addin that you think might be useful for others you can publish on the **Markdown Monster Addin Registry**. The registry holds public Addins that show in the Addin Manager inside of Markdown Monster:

You can find out more on how to publish your Addins in this GitHub repository:

- **Markdown Monster Addin Registry**

Other Add-ins - What do you want to build?

I can think of a few add-in ideas - a quick way to commit to Git and Push would be useful for documentation solutions, or Git based blogs, so you can easily persist changes to a GitHub repository. Embedding all sorts of content like reference links, AdSense links, Amazon product links, a new post template engine, etc., etc.

Or maybe you have custom applications that use Markdown text and provide an API that allows you to post the Markdown (or HTML) to the server. It's easy to build a custom add-in that lets you take either the Markdown text or rendered HTML and push it to a custom REST interface in your custom application.

Acknowledgements

This application heavily leans several third party libraries without which this tool would not have been possible. Many thanks for the producers of these libraries that are used extensively in Markdown Monster:

- **Ace Editor**
  Ace Editor is a power HTML based editor platform that makes it easy to plug syntax highlighted software style editing possible in a browser. Markdown Monster uses Ace Editor for the main Markdown editing experience inside of a Web browser control that interacts with the WPF application.
- **MarkDig Markdown Parser**
  This extensible Markdown parser library is used for the rendering Markdown to HTML in Markdown Monster. The library is fast and supports a number of useful extensions like GitHub Flavored Markdown, table support, auto-linking and various add-on protocols. The feature set is extensible via a plug-in pipeline.
- **MahApps.Metro**
  This library provides the Metro style window and theming support of the top level application shell. It's an easy to use library that makes it a snap to build nice looking WPF applications.
- **Dragablz**
  This library provides the tab control support for the editor allowing for nicely styled tab reordering and overflow. The library also supports tab tear off tabs and layout docking although this feature is not used in Markdown Monster.
- **nHunspell Spell Checking**
  Spell checking is handled via the hunspell library and the .NET wrapper in nhunspell. This library checks for mispellings and provides lookups for misspelled words. Word parsing is done in JavaScript and the spell checking is done in .NET by piping word lists to .NET to check which is drastically faster than doing the spell checking in the browser using JavaScript.

Additional shout outs to these libraries used:

- **HtmlAgility Pack**
- **LibGit2Sharp**
- **YamlDotnet**
- **Windows API Codepack**
- **FontAwesome.WPF**
- **ReverseMarkdown**

Spread the Word about Markdown Monster

If you like Markdown Monster please pass it on to help spread the word. Let your friends know, mention it to others who ask about Markdown and help us grow this community to encourage building the best Markdown Editor around.

Here are a few things you can do to help spread the word:

- **Follow us on Twitter**: @MarkdownMonstr
- **Tweet about Markdown Monster** and mention @MarkdownMonstr
- **Star this repo** by clicking on the Star icon in the header
- **Install from Chocolatey** with the Markdown Monster Package
- **Write an Addin**: Create a Markdown Monster Addin
- **Write a blog post** and mention how you use Markdown Monster
- **Link to the** Markdown Monster Web Site to help us spread the Google Foo.

The support from the community so far with feedback, bug reports and ideas for new features has been awesome, and I look forward for that to continue with a growing community of active users and contributors.

License

Markdown Monster comes in several license modes: Evaluation, Single User, Multiple User and Site License.

Markdown Monster is **Source Open** with source code available on GitHub, but it is a licensed product that requires a paid-for license for continued use. The software is licensed as © Rick Strahl, West Wind Technologies, 2015-2019.

A fully functional, free evaluation version is available for evaluation use, but continued use requires purchase of a license.

Licenses can be purchased from:

https://store.west-wind.com/product/markdown_monster

Evaluation License

The Evaluation version has all the features and functionality of the registered version, except that it shows occasional freeware notices. Tampering with or removing of the notices is not allowed with the evaluation license.

You can use the evaluation version with the notices enabled, but if you use Markdown Monster regularly or for commercial use, please register and support further development and maintenance.

Purchased License

For continued or commercial use of Markdown Monster a paid-for license is required. The paid-for license removes the freeware notices.

Each licensed user must have a separate license, but a single user may use multiple copies of Markdown Monster on multiple machines.

The multi-user licenses work the same as a single user license applied to the number of users specified on the license purchased. An organizational site license is available to allow any number of users running unlimited numbers of Markdown Monster instances within a single organization.

Any purchased license is valid for the duration of the major release that it was purchased for (ie. 1.00-1.99) and minor version updates within that major version are always free. Upgrade pricing is available for major version upgrades, usually at half of full price, and it's our policy to allow for free upgrades to the next major version within a year of purchase.

Source Code

Markdown Monster is **Source Open** and source code is available on GitHub at **https://github.com/RickStrahl/MarkdownMonster**, but the licensing outlined above is applies regardless. We allow modification of source code for internal use of Markdown Monster in your organization or for submitting pull requests to the Markdown Monster main repository. Under no circumstances are you allowed to re-package and re-distribute any part of Markdown Monster outside of your organization.

Help us out - Get a free License

We encourage pull requests for feature suggestions or bug fixes to be submitted back to the Markdown Monster repository. Any contributors that provide meaningful enhancements, help with identifying and or fixing of bugs or by actively promoting Markdown Monster can qualify for a free license (at our discretion). Additionally Microsoft MVPs and Insiders and Microsoft Employees can **apply for a free license**.

WARRANTY DISCLAIMER: NO WARRANTY!

YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE LICENSED APPLICATION IS AT YOUR SOLE RISK AND THAT THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH YOU. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE LICENSE APPLICATION AND ANY SERVICES PERFORMED OR PROVIDED BY THE LICENSED APPLICATION ("SERVICES") ARE PROVIDED "AS IS" AND "AS AVAILABLE," WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, AND APPLICATION PROVIDER HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH RESPECT TO THE LICENSED APPLICATION AND ANY SERVICES, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. APPLICATION PROVIDER DOES NOT WARRANT AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE LICENSED APPLICATION, THAT THE FUNCTIONS CONTAINED IN, OR SERVICES PERFORMED OR PROVIDED BY, THE LICENSED APPLICATION WILL MEET YOUR REQUIREMENTS, THAT THE OPERATION OF THE LICENSED APPLICATION OR SERVICES WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT EFFECTS IN THE LICENSED APPLICATION OR SERVICES WILL BE CORRECTED. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY APPLICATION PROVIDER OR ITS AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SHOULD THE LICENSED APPLICATION OR SERVICES PROVE DEFECTIVE, YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT SHALL THE AUTHOR, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THIS PROGRAM AND DOCUMENTATION, BE LIABLE FOR ANY COMMERCIAL, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM INCLUDING, BUT NOT LIMITED TO, LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR LOSSES SUSTAINED BY THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS, EVEN IF YOU OR OTHER PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
