DockPanel Suite Documentation

Release 2.10

Ryan Rastedt, Lex Li, and others

January 25, 2016

Attention: DockPanel Suite 2.10 Beta 2 is now available on NuGet! Please read the Getting Started instructions for installing the latest version from NuGet.

CHAPTER 1

Topics

1.1 Getting Started

1.1.1 Installing DockPanel Suite On Windows

By Lex Li

This page shows you how to install DockPanel Suite to your project on Windows.

In this article:

Install DockPanel Suite via NuGet

The easiest way to get started building applications with DockPanel Suite is to install via NuGet in the latest version of Visual Studio 2015 (including the free Community edition).

1. Install Visual Studio 2015.

   Be sure to specify that you include the Windows and Web Development.

2. Install latest NuGet Package Manager.

   This will install the latest NuGet tooling.

3. Open/create an empty Windows Forms project.

4. Install DockPanel Suite NuGet packages following NuGet conventions.

   The latest packages can be found at,

   - Main Library with VS2005 Theme.
   - VS2003 Theme.
   - VS2012 Light Theme.
   - VS2013 Blue Theme.
   - VS2005 Theme for Multiple UI Threads.

Note that the 2.10 packages are pre-release, and the ''VS2005MultithreadingTheme'' is not recommended for general usage.

5. Create the `DockPanel` control in code and insert to the main form

```
public MainForm()
{
    InitializeComponent();

    this.dockPanel = new WeifenLuo.WinFormsUI.Docking.DockPanel();
    this.dockPanel.Dock = System.Windows.Forms.DockStyle.Fill;
    this.Controls.Add(this.dockPanel);
}
```

6. Create other panels by creating a new `Form` or new `UserControl` in Visual Studio

```
public class NewForm : Form
{

}
```

Change the base type to `WeifenLuo.WinFormsUI.Docking.DockContent`

```
public class NewDockContent : WeifenLuo.WinFormsUI.Docking.DockContent
{

}
```

7. Show the custom `DockContent` in `DockPanel` as a document

```
public void ShowDockContent()
{
    var dockContent = new NewDockContent();
    dockContent.Show(this.dockPanel, DockState.Document);
}
```

Install DockPanel Suite via source code

DockPanel Suite source code can be directly used in your project.

1. Download the source code from GitHub, or clone the repo directly.
2. Open/create a empty Windows Forms project in a solution.
3. Add WinFormsUI.csproj in `WinFormsUI` directory to your solution.
4. (optional) Add other theme projects such as ThemeVS2003.csproj to your solution.
5. Compile the solution and DockPanel Suite controls are automatically added to Toolbox panel.
6. Open main form of the empty project, and drag the `DockPanel` control from Toolbox on to it.

   This will let Visual Studio generate the necessary code.

7. Create other panels by creating new `Form` or new `UserControl` in Visual Studio

```
public class NewForm : Form
{

}
```

Change the base type to `WeifenLuo.WinFormsUI.Docking.DockContent`

```
public class NewDockContent : WeifenLuo.WinFormsUI.Docking.DockContent
{

}
```

8. Show the custom `DockContent` in `DockPanel` as a document

```
public void ShowDockContent()
{
    var dockContent = new NewDockContent();
    dockContent.Show(this.dockPanel, DockState.Document);
}
```

1.1.2 Adding DockPanel Suite to Toolbox in Visual Studio

By Lex Li

This page shows you how to install DockPanel Suite to your project on Windows.

In this article:

Recommended Steps

1. Download the NuGet package and extract the assemblies.

Note: The assemblies can also be built from source code.

---

2. Open Visual Studio, and navigate to its Toolbox panel.
3. Expand a tab such as General, or create a new tab by right clicking and choosing "Add Tab" menu item.
4. Right click in the tab area and choose "Choose Items..." menu item.
5. Under the ".NET Framework Components" "Choose Toolbox Items" dialog, click the "Browse..." button.

6. In the "Open" file dialog, navigate to the folder that contains the assemblies, and choose all related files (WeifenLuo.WinFormsUI.Docking.dll, ThemeVS2003.dll, ThemeVS2012Light.dll, and ThemeVS2013Blue.dll).

7. Click the "Open" button to exit the dialog.
8. In ".NET Framework Components" tab, click "Namespace" column header to reorder the list.
9. Make sure that "DockPanel", "VS2003Theme", "VS2005Theme", "VS2012LightTheme", and "VS2013BlueTheme" are checked.
10. Click the "OK" button to exit the dialog.

1.1.3 Project History

By Lex Li

Microsoft first introduced the docking panel layout in Visual Studio .NET (2002), and soon it became popular in application design. Many commercial .NET component vendors started to provide docking libraries initially, but

there was no good free and open source alternative, until WeiFen Luo released DockPanel Suite (DPS for short) on SourceForge.net in 2006.

http://sourceforge.net/projects/dockpanelsuite

WeiFen Luo's Efforts and Early Years

Its 1.0 release was available on Feb 13, 2006, one day before the Valentine's day [1]. From the SVN repository we could no longer find the commits earlier than Mar 2, 2007. Therefore, we don't know exactly when WeiFen decided to implement this docking library and the day he started. This release has been downloaded more than 57,000 times on SF.net alone (binaries + source package), which is a huge success.

After that, WeiFen published several new releases. Release 2.0 RC was available on Mar 02, 2007. Release 2.0 followed on Apr 02, while release 2.2 was available on Nov 04, 2007 (more than 68,000 downloads) [2]

Danilo Corallo wrote an article titled "A Visual Studio 2005-like Interface"[3] on CodeProject.com initially on Jun 06, 2006 to introduce this library for broader audience. The article has been reviewed for more than 586,000 times with an average rate of 4.89. This article was last updated on 22 Jan, 2007, so it only targets DPS 1.0. However, DPS 2.0 and above do contains breaking changes, so the sample of this article does not work with newer DPS releases.

SharpDevelop [4], the open source C#/VB.NET IDE, has chosen DPS as its docking library for years (till SD 4.0 migrates to WPF and uses AvalonDock [5] instead of DPS). It is interesting that from SD code base, DPS source files appeared as early as Jan 04, 2005 [6].

On Aug 16, 2009, WeiFen wrote in a discussion thread, that he would like to move this project to CodePlex.com [7]. However, this move was never carried out. But in this thread WeiFen linked one of his important blog posts on DPS [8], which documented his ideas on why WPF based docking library is better. WeiFen's interest has been moved to WPF side product called WPF Docking [9].

Extended Maintenance by Steve Overton and Others

Steve Overton and other contributors stepped up and started to maintain this library in 2009 [10]. He managed to release 2.3 on May 08, 2009 (more than 68,000 downloads) [11] [12]. 2.4 Followed on Oct 30, 2010 with a few new patches [13]. For the first time, DPS is released with binaries/source code/release notes. This release has been downloaded over 8000 times. Soon release 2.5.0 (with RC1 flag) was available on Nov 25 the same year with more patches included [14]. This is the last stable release that can be found on SF.net with accumulated downloads of 61,000.

GitHub Fork and The DockPanel Suite Organization

Frustrated DPS users started to discuss about the future of this project [15], and soon some agreed to create a fork on GitHub [16].

[1] http://sourceforge.net/projects/dockpanelsuite/files/DockPanel%20Suite/1.0.0.0/
[2] http://sourceforge.net/projects/dockpanelsuite/files/DockPanel%20Suite/2.2/
[3] http://www.codeproject.com/Articles/14336/A-Visual-Studio-2005-like-Interface
[4] http://www.icsharpcode.net/OpenSource/SD/Default.aspx
[5] http://avalondock.codeplex.com/
[6] https://github.com/icsharpcode/SharpDevelop/tree/c4336b038c23fa37ee19bdd7d27bfa29b575a4a4/src/Libraries/DockPanel_Src
[7] http://sourceforge.net/projects/dockpanelsuite/forums/forum/402316/topic/3368441
[8] http://www.devzest.com/blog/post/WPF-vs-Windows-Forms-From-Control-Authoring-Perspective.aspx
[9] http://www.devzest.com/WpfDocking.aspx?Show=Overview
[10] http://sourceforge.net/projects/dockpanelsuite/forums/forum/402316/topic/3879095
[11] http://sourceforge.net/projects/dockpanelsuite/files/DockPanel%20Suite/2.3.1/
[12] https://sourceforge.net/news/?group_id=110642
[13] http://sourceforge.net/projects/dockpanelsuite/files/DockPanel%20Suite/2.4.0/
[14] http://sourceforge.net/projects/dockpanelsuite/files/DockPanel%20Suite/2.5.0%20RC1/
[15] http://sourceforge.net/projects/dockpanelsuite/forums/forum/402316/topic/5080422
[16] http://sourceforge.net/projects/dockpanelsuite/forums/forum/402316/topic/5271451

The new repository was created using svn2git [17] by Lex Li and now is hosted on GitHub under dockpanelsuite organization,

https://github.com/dockpanelsuite/dockpanelsuite

Ryan Rastedt purchased the dockpanelsuite.com domain name and designed a new home page [18] for this project.

This organization evaluated and merged many community patches. A few releases, such as 2.6, 2.7, 2.8, and 2.9 have been published in the past few years.

1.2 Tutorials

1.2.1 Basics

By Lex Li

This page shows you the basics about DockPanel Suite.

In this article:

Show Contents as Documents

It is quite common to have a list of dock contents shown as a list of documents.

To show a single document, the following code can be used

```
doc1.Show(dockPanel, DockState.Document);
```

To show a list of documents in order, the following code can be used,

```
doc1.Show(dockPanel, DockState.Document);
doc2.Show(doc1.Pane, null);
doc3.Show(doc1.Pane, null);
doc4.Show(doc1.Pane, null);

// reorder doc1 and doc2
doc1.Show(doc1.Pane, null);
doc2.Show(doc1.Pane, null);
```

Note: Note that the `DockContent.Pane` property is used so that documents are placed in the same dock pane.

Show Multiple Contents in The Same Dock State

By default if two or more dock contents are shown in the same dock state will share the same dock pane and become tabs in that pane.

[17] https://github.com/nirvdrum/svn2git
[18] http://dockpanelsuite.com

```
dmcDownloadMonitor.Show(dockPanel, DockState.DockBottom);
amcActivateModsMonitor.Show(dockPanel, DockState.DockBottom);
```

However, sometimes the contents should be placed into different panes and shown side by side. This requires some changes in code,

```
dmcDownloadMonitor.Show(dockPanel, DockState.DockBottom);
amcActivateModsMonitor.Show(dmcDownloadMonitor.Pane, DockAlignment.Right, 0.5);
```

How to Lock Layout and Prevent Dock Panels From Moving

Lock Entire Layout

To lock the entire layout, the following snippet can be used to disable all end user docking at `DockPanel` level,

```
dockPanel.AllowEndUserDocking = false;
```

Lock A Single Panels

To lock only a single panel, the following snippet can be used to disable all end user docking at `DockContent` level,

```
dockContent.DockHandler.AllowEndUserDocking = false;
```

Avoid A Few Dock States

To prevent a dock panel from entering some dock states, set `DockContent.DockAreas` to only the areas that are desired.

1.2.2 Customizing DockContent

By Ryan Rastedt, Lex Li

This page shows you how to customize `DockContent`.

In this article:

Bring a DockContent to Front

Simply call `DockContent.Activate`. This is usually used to show a tab as active tab in code.

Prevent The DockContent From Being Closed

To prevent a `DockContent` from every being closed, utilize the `CloseButton` property. You can also utilize the `CloseButtonVisible` property to hide the close button when docked in the `DockPanel` control

```
public CustomContent : DockContent
{
    public CustomContent()
    {
        InitializeContent();

        // Prevent this content from being closed
        CloseButton = false;

        // Hide the close button so the user isn't even tempted
        CloseButtonVisible = false;
    }
}
```

You can also use the `OnFormClosing` method to prevent the user from closing under certain conditions that cannot be determined until runtime (for example, showing a dialog box asking the user if they are really sure they wish to close the tab)

```
public CustomContent : DockContent
{
    protected override void OnFormClosing(FormClosingEventArgs e)
    {
        bool cancel = /* add your closing validation here */;
        e.Cancel = cancel;
        base.OnFormClosing(e);
    }
}
```

Controlling Default Height or Width When Set to Auto-Hide

The property `DockContent.AutoHidePortion` controls the auto-hide behaviors.

A value greater than 1 indicates a pixel size, while a value less than 1 indicates a percentage size (as a percentage of the `DockPanel`).

Dismissing Visible Auto-Hide Panel

Usually by double clicking the the tab of an auto-hide panel it will be dismissed. Below shows how to do it in code,

```
this.dockPanel.ActiveAutoHideContent = null;
```

Switching Among Documents

It is possible to add keyboard shortcuts to enable switching among document tabs, not yet part of the default code base,

1. Derive from `DockContent`.
2. Override `ProcessCmdKey`.
3. Find the next `DockContent` in `DockPanel.Documents`.

4. Call its `Activate` method.

Remove a DockContent From DockPanel

To release a `DockContent` from `DockPanel`, simply set `DockContent.DockHandler.DockPanel` to `null`.

1.2.3 Customizing DockWindow

By Lex Li

This page shows you how to customize `DockWindow`.

In this article:

Change DockWindow Z-Order

There are four `DockWindow` instances created in `DockPanel` control, where later new `DockContent` instances can be inserted to. By changing their Z-order, the layout style can be controlled.

```
this.dockPanel.UpdateDockWindowZOrder(DockStyle.Right, true);
```

1.2.4 Customizing FloatWindow

By Ryan Rastedt, Lex Li

This page shows you how to customize `FloatWindow`.

In this article:

Maximizable FloatWindow

Out of the box, a window that is undocked to a floating state lacks a maximize button. Utilizing the `Extender` functionality of `DockPanel` it's easy to customize the `FloatWindow` class!

The first step is to create a custom class that extends `FloatWindow`. By default, `FloatWindow` has a `FormBorderStyle` of `SizableToolWindow` which will only provide a close button. To expose the maximize and minimize button, set `FormBorderStyle` of the custom window to `Sizable`

```
public class CustomFloatWindow : FloatWindow
{
    public CustomFloatWindow(DockPanel dockPanel, DockPane pane)
        : base (dockPanel, pane)
    {
        FormBorderStyle = FormBorderStyle.Sizable;
    }

    public CustomFloatWindow(DockPanel dockPanel, DockPane pane, Rectangle bounds)
        : base (dockPanel, pane, bounds)
    {
        FormBorderStyle = FormBorderStyle.Sizable;
    }
}
```

Next, create a factory class to create the `CustomFloatWindow`. This is done by implementing the `IFloatWindowFactory` interface

```
public class CustomFloatWindowFactory : DockPanelExtender.IFloatWindowFactory
{
    public FloatWindow CreateFloatWindow(DockPanel dockPanel, DockPane pane, Rectangle bounds)
    {
        return new CustomFloatWindow(dockPanel, pane, bounds);
    }

    public FloatWindow CreateFloatWindow(DockPanel dockPanel, DockPane pane)
    {
        return new CustomFloatWindow(dockPanel, pane);
    }
}
```

Lastly, attach the new factory to the `DockPanel` control,

```
this.dockPanel.Extender.FloatWindowFactory = new CustomFloatWindowFactory();
```

Alt+Tab Support

To enable Alt+Tab between your undocked forms and your main form add this to your `CustomFloatWindow` constructors

```
ShowInTaskbar = true;
Owner = null;
```

Overriding Double Clicking Behavior

Note: This requires DockPanel Suite version 2.8 and above.

By default, when a float window's title bar is double clicked it is redocked into the `DockPanel`. This behavior can be disabled (and allow the Windows default behavior of maximizing/restoring the window) by setting the following property in your `CustomFloatWindow` constructor

```
DoubleClickTitleBarToDock = false;
```

Positioning for Multiple Monitors

When a dock content is set to float, the created `FloatWindow` might be at a secondary monitor (depending on WinForms underlying positioning).

To force the `FloatWindow` to appear on a desired monitor, a custom `FloatWindow` can be created. Then override its `SetBoundsCore` method to check the monitors based on the information exposed by the `Screen` class.

Top-Most FloatWindow

To make the `FloatWindow` top-most, simply create a custom `FloatWindow` class and set its `TopMost` property to true.

1.2.5 Customizing Persistence

By Lex Li

This page shows you how to customize persistence.

In this article:

Layout File Format

`DockPanel.Persistor.SaveAsXml` is the method who saves current layout to an XML file.

It is not recommended to manually edit the generated layout file, as its syntax is complicated.

To achieve customization, it is recommended to override the persistent string.

Overriding Persistent String

As the sample project shows, by default when current layout is saved to disk only class name is persisted. As a result, in delegate `IDockContent DeserializeDockContent(string persistString)` only class name is used to reconstruct the layout.

The current design gives you flexibility to control what other information about each dock items to be persisted, as the method `DockContent.GetPersistString()` can be overridden by derived classes. By properly overriding it, extra data (such as custom data about the internal components of a dock item) can be appended to the string. Thus, when the layout is reconstructed, the extended persist string can be analyzed and extra data can be extracted and used.

1.3 Themes

1.3.1 Existing Themes

By Lex Li

This page shows you what are the existing themes and how to use them.

In this article

Visual Studio 2003 Theme

This is a theme that simulates Visual Studio 2003.

To apply this theme, the following code can be used

```
var theme = new VS2003Theme();
this.dockPanel.Theme = theme;
```

Note: for 2.10 and above, a reference to `ThemeVS2003.dll` (from NuGet package DockPanel-Suite.ThemeVS2003) is needed.

Visual Studio 2005 Theme

This is the default theme of DockPanel Suite. When you create a new instance of `DockPanel` class, the dock panel and its contents use this theme automatically.

Visual Studio 2012 Light Theme

This is a theme that simulates Visual Studio 2012 Light.

Note: It is introduced in 2.9 release.

To apply this theme, the following code can be used:

```
.. code-block:: csharp
```

var theme = new VS2012LightTheme(); this.dockPanel.Theme = theme;

Note: for 2.10 and above, a reference to `ThemeVS2012Light.dll` (from NuGet package DockPanel-Suite.ThemeVS2012Light) is needed.

Visual Studio 2013 Blue Theme

This is a new theme that simulates Visual Studio 2013 Blue.

Note: It is introduced in 2.10 release.

To apply this theme, the following code can be used:

```
.. code-block:: csharp
```

var theme = new VS2013BlueTheme(); this.dockPanel.Theme = theme;

Note: for 2.10 and above, a reference to `ThemeVS2013Blue.dll` (from NuGet package DockPanelSuite.ThemeVS2013Blue) is needed.

Visual Studio 2005 Theme for Multiple UI Threads

This is derived from the default theme of DockPanel Suite. It is released for applications that use multiple UI threads only, so not recommended for general usage.

Switching among Themes

The sample project demonstrates how to switch among themes,

https://github.com/dockpanelsuite/dockpanelsuite/blob/master/DockSample/MainForm.cs#L159

1.3.2 Creating A New Theme

By Lex Li

This page shows you how to create a new theme.

In this article

New Themes in Incubation

Lex Li has a few blog posts on theming when he attempted to integrate the VS 2012 Light theme to the code base, which might get you started.

The following themes are currently under development (mainly be community members),

- VS 2013 Blue (in vs2013blue branch for incubation, see #316 for progress)
- VS 2012 Dark (in various forks by different authors, see #317 for details)

1.4 Contribute

Note: We reuse the ASP.NET docs style guide.

1.4.1 GitHub Guide

By Lex Li

This page shows you how to use DockPanel Suite repo on GitHub to collaborate.

In this article:

Submitting a Patch

An issue might be opened to discuss with the maintainers before creating the patch. This helps the maintainers track your progress, and provide guidance if needed.

1. Learn about GitHub via http://help.github.com/.
2. Create your own fork from the main repo at https://github.com/dockpanelsuite/dockpanelsuite.
3. Create a new branch with a meaningful name.
4. Make the changes on this new branch in your fork, and test it fully.
5. Create a pull request back (link the new branch in your fork to the master branch of main repo).
6. Document all your changes in details as part of the pull request, which is critical to better communicate with maintainers.

Patches will be reviewed and merged as early as possible.

Note: If the patch is large, it might take several weeks to review and merge.

Note: To see what makes a good pull request, please follow *Suggestions on Creating Pull Requests* section.

Reviewing a Patch

Maintainers should respond to new pull requests as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Comments have been left at **. Please revise your patch like this,"
- "Reviewed. ** will be merged, while ** will not. The reasons are,"

which will keep all discussions public to reveal all necessary technical information for future reference.

Reporting a Bug

Note: If you already have a patch for the bug, please follow *Submitting a Patch* section.

Note: If you are not sure whether it is a bug, please follow *Asking a Question* section.

1. Learn about GitHub via http://help.github.com/.
2. Review existing issues at https://github.com/dockpanelsuite/dockpanelsuite/issues that are marked as bug or enhancement to avoid duplicate.
3. Please also review all open bugs on SF.net tracker before reporting. Existing bugs recorded on SF.net may be gradually fixed in this fork, so you don't need to create duplicate items on GitHub.
4. Create a new issue on https://github.com/dockpanelsuite/dockpanelsuite/issues and provide all information in details.

You are welcome to provide step by step instructions, as that can help other reproduce and investigate the issue. If you are willing to share a sample project, please use a service such as DropBox or OneDrive.

Reviewing a Bug

Maintainers should respond to bug reports as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Could not reproduce it. Please provide more information to assist investigation such as **,"
- "Reviewed. ** is a bug that can be reproduced. Will perform further investigation on how to resolve it. This may take a long time."

which will keep all discussions public to reveal all necessary technical information for future reference.

Asking a Question

Asking a Question on StackOverflow (Recommended)

1. Log into StackOverflow.
2. Before starting asking a new question, please review all questions under tag `dockpanel-suite` in case yours has already been answered.
3. Click Ask Question to create a new question.
4. Add tag `dockpanel-suite` to this question, and also include all information in details.

Then you can wait till users reply to your question.

Asking a Question on GitHub

1. Learn about GitHub via http://help.github.com/.
2. Before creating the issue, please review all existing issues especially our FAQ in case the issue has already been reported and resolved.
3. Create a new issue on https://github.com/dockpanelsuite/dockpanelsuite/issues and provide all information in details.

Answering a Question

Maintainers might join StackOverflow and monitor discussions under `dockpanel-suite` tag.

Maintainers should respond to questions on GitHub as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Could not reproduce it. Please provide more information to assist investigation such as **,"
- "Reviewed. ** is a bug that can be reproduced. Will perform further investigation on how to resolve it. This may take a long time."

which will keep all discussions public to reveal all necessary technical information for future reference.

Tag such an issue with question tag.

Close such issues once a meaningful answer is given.

Mark an issue as `faq candidate` if it should be considered as an FAQ.

Overview of Issue Tags

Maintainers should use the tags as early as possible so as to help each other to easily track the progress. The decoration tags are most useful for items which are not yet assigned to milestones.

Tags for Item Categories

The following are used to assign an item to a specific category,

- bug This item was reported as a bug of this product. The reporter expects a fix.
- enhancement This item was reported as an enhancement request. The reporter expects a certain feature to be enhanced or a new feature to be implemented.
- task This item was reported as a task. The reporter expects a maintainer to perform a piece of work (usually not development).
- idea This item was reported as a new idea. The reporter expects some discussion on a feature request. Once discussed, this item might be upgraded to an enhancement.
- question This item was reported as a question. The reporter expects some discussion on a problem met about this product. Once discussed, this item might be upgraded to a bug, an enhancement, or an idea.
- tech debt This item was reported as bad smells detected in the code base. The reporter expects changes in the code base to remove the bad smells.

- pull request This item was used to handle a pull request.

Tags for Decoration

The following are used to decorate an item so as to make it easy to see its status and required actions,

- dependency bug This only applies to bug items. It means the bug was caused by a bug of one of the dependencies (such as bugs of .NET Framework/Mono bugs, or bugs of the operating systems).
- not an issue This means after discussion, there is nothing to be done further (usually for false positives).
- wontfix This means the item (usually bugs) won't be fixed due to a strong justification. An agreement must be achieved among the maintainers.
- duplicate This means the item is exactly the same as another existing item. The maintainers should explicitly point out which item will be the focus and mark all the rest as duplicate.
- tentative This means based on the provided information it is not likely to move on. The reporter should provide more information and drive the discussion.
- soon to close This means there is little left to do on the item. The maintainers are going to close the item after a few more days (usually applied to tentative and cannot reproduce items).
- cannot reproduce This means the maintainers failed to reproduce the symptoms described in a bug report. The reporter should provide more information (process dumps, sample projects, screen shots, video clips and so on) and drive the investigation.
- in progress This means the item has been actively investigated by the maintainers.
- up for grabs This means community contribution is welcome.

Suggestions on Creating Pull Requests

All pull requests are appreciated (even if some we cannot merge). The following can make the pull requests simpler for reviewers, so hope you can follow them.

- If possible, send multiple pull requests for individual tasks and avoid a pull request for multiple tasks. Properly isolating changes to meaningful batches makes it quicker to analyze and assert the changes.
- Fork and create a new branch with a meaningful name first before making the changes.
- Squash all commits on this new branch to only one or two before sending the pull request.
- Wait for comments from the reviewers. It usually takes weeks as the reviewers might not be able to finish quickly. Don't make further changes at this stage to avoid changes of this pull request.
- Revise the code based on feedbacks, and then make a second commit with necessary changes and push to the branch in your fork, where GitHub automatically appends it to the pull request for further review.

Then the reviewers will decide whether to accept or reject the pull request based on code quality.

One important notice is that some pull requests might not be accepted, but they are still valuable to the community,

- It contains a nice-to-have feature (such as options to enable/disable part of a theme, or a visual element) for some users but not all.
- It introduces a feature (such as new visual elements) that goes beyond Visual Studio look and feel.

Such pull requests are of great value of course. But since the primary goal of DPS is to simulate Visual Studio look and feel, and the code base is already huge to maintain, we try to avoid bringing in non-core features.

Suggestions on Reviewing Pull Requests

Please leave a message that you are going to review a pull request. That should let the submitter know it's been reviewed.

Leave all comments at a time, so that the submitter can revise them altogether to form a new commit.

Decide carefully whether to accept or reject a pull request. Leave explanation for future reference.

CHAPTER 2

Related Resources

- genindex
- modindex
- search

CHAPTER 3

Contribute

The documentation on this site is the handiwork of our many contributors.

We accept pull requests! But you're more likely to have yours accepted if you follow these guidelines:

1. Read the Contributing Guide.
2. Follow the ASP.NET 5 Docs Style Guide.