

Contents

參與者指南概觀和以瀏覽器為基礎的工作

在本機進行設定與工作

1. 註冊 GitHub
2. 安裝 Git 和 Markdown 工具
3. 設定本機 Git 存放庫

Git 和 GitHub 基礎

完整工作流程

有關撰寫的基本資訊

Markdown 參考

風格和語氣快速入門

樣式指南

將程式碼加入文章

設定文字格式

連結

中繼資料

Docs 編寫套件

概觀

功能

開發語言完成

影像壓縮

中繼資料瀏覽器

中繼資料更新

重新格式化 Markdown 資料表

智慧引號取代

排序重新導向

排序選取項目

Jupyter Notebook

文件集特定指引

.NET 文件

[如何參與](#)

[程式碼分析文件](#)

[標籤、專案與里程碑藍圖](#)

[風格慣例](#)

[語態和語調指南](#)

[提取要求檢閱流程](#)

[檔中的 .NET 命名](#)

[PowerShell 文件](#)

[Hacktoberfest](#)

[其他資源](#)

Microsoft Docs 參與者指南的概觀

2021/6/23 •

歡迎使用 docs.microsoft.com (Docs) 參與者指南！

我們有幾個開放原始碼的 Microsoft 文件集，由 GitHub 代管。並非所有文件集都是完全開放原始碼的，但許多都有公開的存放庫，可讓您透過提取要求來建議變更。這種開放原始碼的方法可簡化並改善產品工程師、內容小組與客戶之間的通訊，而且還有其他優點：

- 開放原始碼存放庫「公開規畫」以取得意見反應，了解最迫切需要的文件有哪些。
- 開放原始碼存放庫「公開檢閱」，以在我們初次發行時發佈最實用的內容。
- 開放原始碼存放庫「公開更新」，讓持續改善內容變得更容易。

docs.microsoft.com 上的使用者體驗會直接整合 [GitHub](#) (英文) 工作流程，使其變得更加容易。請從[編輯您正在檢閱的文件](#)開始。或者，透過[檢閱新主題](#)或[建立品質問題](#)來提供協助。

IMPORTANT

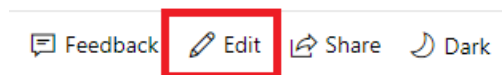
所有發佈至 docs.microsoft.com 的存放庫皆採用 [Microsoft 開放原始碼管理辦法](#) (英文) 或 [.NET Foundation 管理辦法](#) (英文)。如需詳細資訊，請參閱[管理辦法常見問題集](#) (英文)。若有任何問題或意見，也可以連絡 opencode@microsoft.com 或 conduct@dotnetfoundation.org。

您在公用存放庫中針對文件和程式碼範例所提交的次要修正或釐清，將受到 [docs.microsoft.com 使用條款](#) 的約束。如果您不是 Microsoft 的員工，在做出全新或大規模的變更時，系統會在提取要求中產生意見，要求您提交線上版的貢獻授權合約 (CLA)。您必須先完成填寫線上表單，我們才會檢閱或接受您的提取要求。

快速編輯現有文件

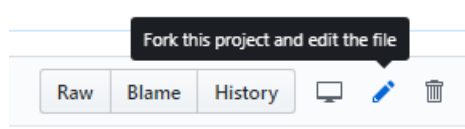
快速編輯可簡化回報並修正文件中小錯誤和遺漏的流程。即便我們非常努力，發佈的文件中仍不免存在少許文法和拼字錯誤。儘管您可以建立問題來回報錯誤，建立提取要求 (PR) 來解決問題會更加快速且輕鬆 (當該選項可用時)。

1. 某些文件頁面允許您直接在瀏覽器中編輯內容。如果是這樣，您將會看到如下 [編輯] 按鈕。按一下 [編輯] (或當地語系化的同等文字) 按鈕，即前往 GitHub 上的原始程式檔。若沒有看到 [編輯] 按鈕，表示文件頁面無法變更。



如果未出現 [編輯] 按鈕，表示該內容並未開放進行公開參與。

2. 選取鉛筆圖示以編輯文章。如果鉛筆圖示顯示為灰色，表示您必須登入您的 GitHub 帳戶或建立新帳戶。



3. 在 web 編輯器中進行變更。按一下 [預覽變更] 索引標籤來檢查變更的格式。
4. 完成變更後，請向下捲動到頁面底部。為變更的內容輸入標題與說明，然後按一下 [建議檔案變更]，如下圖所示：

Propose changes

Update load-tests.md

Fix typos and format text

Propose changes

Cancel

5. 現在您已提出變更建議，您必須要求存放庫的擁有者將您的變更「提取」到其存放庫中。使用「提取要求」即可完成此動作。當您選取 [建議檔案變更] 時，會顯示類似下列的新頁面：

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base fork: MicrosoftDocs/MAX-CPUB-Test

base: master

head fork: dstrome3/MAX-CPUB-Test

compare: patch-3

✓ Able to merge. These branches can be automatically merged.

Create pull request

Discuss and review the changes in this comparison with others.

選取 [建立提取要求]，輸入提取要求的標題，並選擇性地輸入描述，然後選取 [建立提取要求]。如果您不熟悉 GitHub，請參閱 [關於提取要求](#) 以取得詳細資訊。

6. 大功告成！內容小組成員會在核准時檢查併合並您的 PR。您可能會收到要求變更的意見反應。

GitHub 編輯 UI 會回應您對存放庫的權限。上述影像適用於不具有目標存放庫寫入權限的參與者。GitHub 會自動在您的帳戶中建立目標存放庫的分叉。如果您有目標存放庫的寫入權限，GitHub 會在目標存放庫中建立一個新的分支。分支名稱的格式為 <GitHubId>-patch-n，並使用您的 GitHub 識別碼，以及針對修補分支的數值識別碼。

我們會使用提取要求來進行所有變更，即使是針對有寫入權限的參與者也是如此。大部分的存放庫都會保護預設分支，因此必須將更新提交為提取要求。

瀏覽器內編輯體驗，最適合用於小規模或非常態的變更。如果您做出較大規模的貢獻，或是使用進階 Git 功能（例如分支管理或進階的合併衝突解決），則需要[派生存放庫並在本機處理](#)。

NOTE

大部分當地語系化的檔不提供透過 GitHub 編輯或提供意見反應的功能。若要針對當地語系化的內容提供意見反應，請使用 [aka.ms/DocSiteLoFeedback](#) 提供的電子郵件範本。Azure 檔有一個例外狀況。Azure 檔有公開當地語系化的 GitHub 存放庫，適用於 6 種語言的社區貢獻：簡體中文、法文、德文、日文、韓文和西班牙文。針對所有其他語言和內容集，請使用 [電子郵件範本](#)（以任何語言接收您的輸入）來回報翻譯問題或提供技術意見反應。

檢閱未處理的 PR

您可以透過查看目前未處理的 PR，來在新主題發佈之前閱讀它們。檢閱需遵循 [GitHub 流程](#) (英文) 的程序。您可以查看公開存放庫中的建議更新或新文章。請檢閱它們並新增您的意見。查看我們的文件存放庫，並針對您感興趣的領域查看未處理的提取要求 (PR)。任何針對建議更新的社群意見反應，都能為整個社群提供協助。

建立品質問題

我們的文件都是持續進行中的工作。好的問題有助於我們將工作重點放在社群的最高優先事項上。您可以提供的細節越多，該問題就越有幫助。請告訴我們您想要的資訊。請告訴我們您所使用的搜尋字詞。如果您無法開

始使用，請告訴我們您想要如何開始探索不熟悉的技術。

許多 Microsoft 文件頁面底部都有 [意見反應] 區段，在其中按一下即可留下 [產品意見反應] 或 [內容意見反應] 以追蹤該文章的特定問題。

問題能促進所需項目的相關討論。內容小組將回應這些問題，提供我們可以新增內容的想法，並徵求您的意見。當我們建立草稿時，我們會要求您[檢閱 PR](#)。

深入參與

其他主題可協助您開始有效率地為 Microsoft Docs 做出貢獻。這些主題會說明如何使用 GitHub 存放庫、Markdown 工具，以及 Microsoft Docs 平台所使用的延伸模組。

GitHub 帳戶設定

2021/5/13 •

設定您的 GitHub 帳戶

若要對 Docs 技術內容做出貢獻，您必須設定您自己的 GitHub 帳戶。好消息是您通常只需要執行這些步驟一次。

1. 建立 GitHub 帳戶並設定設定檔

如果您還沒有 GitHub 帳戶，請[建立一個](#)。在您的 GitHub 設定檔中識別所有關係。針對 docs.microsoft.com 的參與會計入 [MVP 獎勵](#) 考量。識別可協助我們建置一個您所有活動的完整設定檔。

NOTE

參與開放原始碼專案的 Microsoft 員工，在 GitHub 設定檔中一律將其本身識別為員工。社群參與者應該確定其設定檔不會錯誤地暗示僱傭關係。

下一步

- 繼續閱讀[工具安裝](#)文章以安裝 Git Bash (Markdown 編輯器) 等工具。

安裝內容撰寫工具

2021/6/6 •

本文說明以互動方式安裝 Git 用戶端工具和 Visual Studio Code 的步驟。

- 安裝 [Git](#)
- 安裝 [Visual Studio Code](#)
- 安裝 [Docs 編寫套件](#) (英文)

IMPORTANT

若只是要對文章進行很少量的變更，不需要完成此文章中的步驟，而可繼續直接進入 [快速變更工作流程](#)。

建議將進行重大變更的參與者完成這些步驟，這可讓您使用 [主要/長期執行的變更工作流程](#)。即使您在主要存放庫中具有寫入權限，仍然 [強烈建議您](#) (且此指南假設您會) 對存放庫進行派生和複製，這樣您會具有在分支中儲存所建議變更的讀取/寫入權限。

安裝 Git 用戶端工具

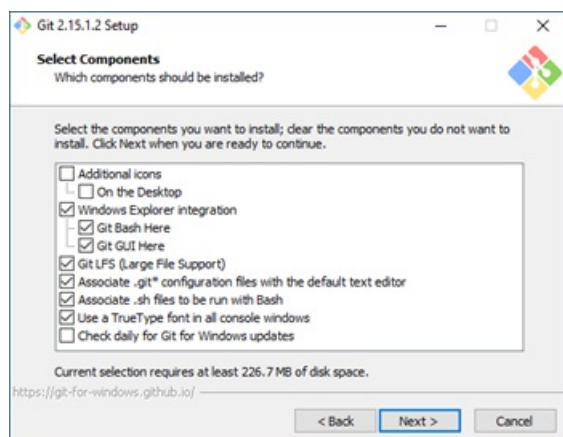
安裝包含了 Git 版本控制系統和 Git Bash，您可以使用此命令列應用程式來與本機 Git 存放庫互動。

安裝適用於 Windows 的 Git 用戶端工具

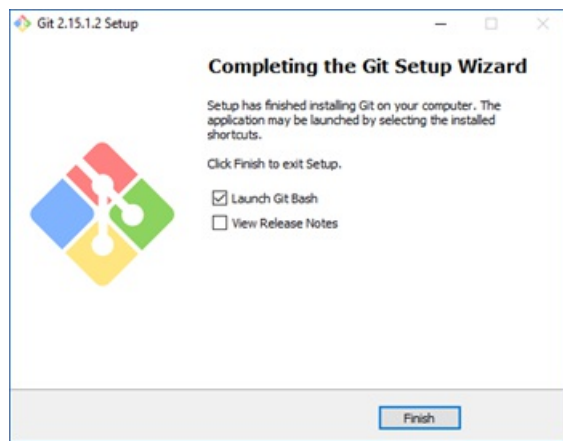
NOTE

- If you prefer a graphical user interface (GUI) over a command-line interface (CLI), [Visual Studio Code](#), [Software Freedom Conservancy's available GUI Clients page](#), and [GitHub's GitHub Desktop](#) offer that functionality.

1. Download [Git for Windows](#).
2. Run the downloaded executable (.EXE) file and follow the prompts to install. Select **Next** at each prompt to accept all default settings.



3. Select **Finish** to complete the installation.



After installing the Git for Windows, you'll have to configure your Git name and your email address, before installing Visual Studio Code.

WARNING

Git on Windows: Enable long path names. By default, Git for Windows disables support for long filepaths, which prevents any file with a destination path longer than 255 characters from being cloned. Enable long filepaths to avoid this issue by running the following command **as an administrator**:

```
git config --global core.longpaths true
```

安裝適用於 Mac 和 Linux 的 Git 用戶端工具

- 適用於 Mac 的 Git 是隨著 Xcode 命令列工具提供。只要從命令列執行 `git`。系統將會提示您視需要安裝該命令列工具。您也可以從 Software Freedom Conservancy 下載[適用於 Mac 的 Git](#)。
- [適用於 Linux 與 Unix 的 Git](#)

請遵循您所選擇用戶端的指示，以進行安裝與設定。

在下一篇文章中，您將[設定本機 Git 存放庫](#)。

您可以從這裡取得其他的 git 資源：[git 術語](#) | [git 基本概念](#) | [學習 git 和 GitHub](#)

安裝 Visual Studio Code

[Visual Studio Code](#) 也稱為 VS Code，是一種輕量型編輯器，適用於 Windows、Linux 和 Mac。它包含 Git 整合，並支援擴充功能。

TIP

若要啟動 VS Code 並開啟目前的資料夾，請在命令列或 bash 殼層中執行 `code .` 命令。如果目前資料夾為本機 Git 存放庫的一部分，則 GitHub 整合會自動在 Visual Studio Code 中顯示。

安裝適用於 Windows、mac 和 Linux 的 Visual Studio Code

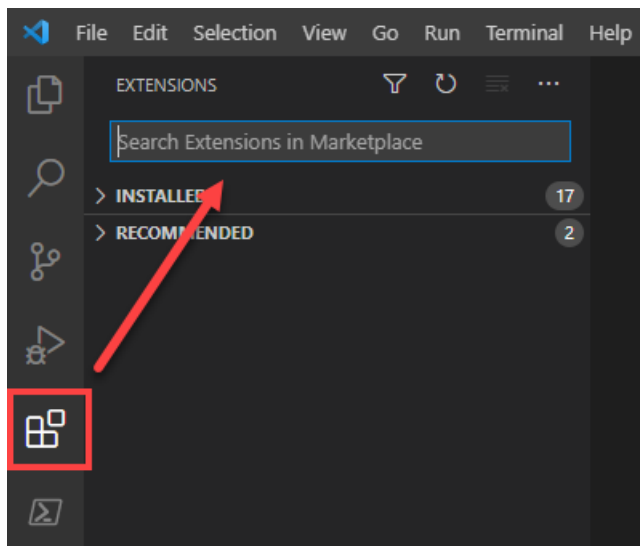
- [Windows](#)。VS Code 首頁應該會正確地偵測到您的作業系統。
- [Mac](#)
- [Linux](#)

安裝 Visual Studio Code 延伸模組

To install the extensions:

1. Start Visual Studio Code.

2. Select the square **Extensions** icon on the left nav. The **Extensions: Marketplace** pane appears.
3. In the **Search Extensions in the Marketplace** search box, type the name of an extension you want to find.



4. In the results that appear, locate the extension you want and select **Install**.

安裝 Docs 編寫套件 (英文)

IMPORTANT

The Docs Authoring Pack for Visual Studio Code includes basic Markdown authoring assistance, page previews, support for Markdown templates, markdownlint, and Code Spell Checker. These features ease and streamline the contributions process. As such, we consider the Docs Authoring Pack a **required** extension for contributors.

To install the Docs Authoring Pack, choose **Install** from the [Docs Authoring Pack page](#) in the VS Code Marketplace.

To use the Docs Authoring Pack functionality, press Alt+M in Visual Studio Code. To configure a toolbar to show the functions available, edit the Visual Studio Code settings (Control+comma), and add user setting

```
"markdown.showToolbar": true .
```

For more information, see [Docs Authoring Pack for Visual Studio Code](#).

了解 Markdown 編輯器

Markdown 是用來撰寫內容的輕量標記語言。[Visual Studio Code](#) 是 Microsoft 用來編輯 Markdown 的偏好工具。[Atom](#) 是另一個用來編輯 Markdown 的熱門工具。如 (OPS) 自訂 Markdown 延伸模組所支援的 Markdown 基本概念和功能, 請 [參閱 Markdown 參考](#) 文章。

下一步

現在您已準備好 [設定本機 Git 存放庫](#)。

在本機針對文件設定 Git 存放庫

2021/5/13 •

此文章說明在本機電腦上設定 Git 存放庫以針對 Microsoft 文件進行貢獻的步驟。參與者可能會使用本機複製的存放庫來加入新的文章、在現有的文章上進行主要編輯，或變更圖檔。

您會執行這些一次性安裝活動來開始參與：

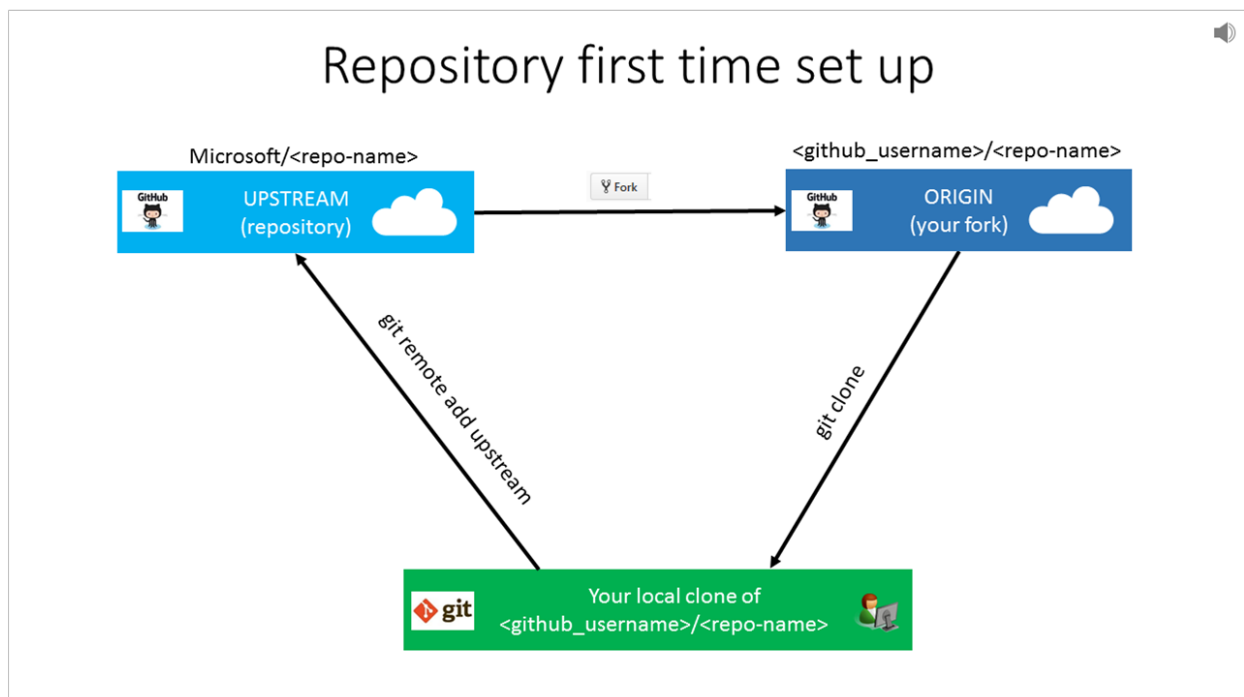
- 決定適當的存放庫
- 將存放庫派生至您的 GitHub 帳戶
- 選擇複製檔案的本機資料夾
- 將存放庫複製到本機電腦
- 設定上游遠端值

IMPORTANT

如果只是對文章進行次要變更，則不需要完成本文中的步驟。您可以直接跳到[快速變更工作流程](#)。

概觀

若要參與 Microsoft 的文件網站，您可以複製對應的文件存放庫，以便在本機製作和編輯 Markdown 檔案。Microsoft 需要您將適當的存放庫派生至您本身的 GitHub 帳戶，這樣您就有儲存建議變更的讀取/寫入權限。然後，您就可以使用提取要求，將變更合併到唯獨中央共用存放庫中。

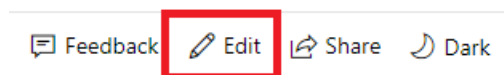


如果您還不熟悉 GitHub，請觀賞以下影片以了解分叉和複製程序的概念性概觀：

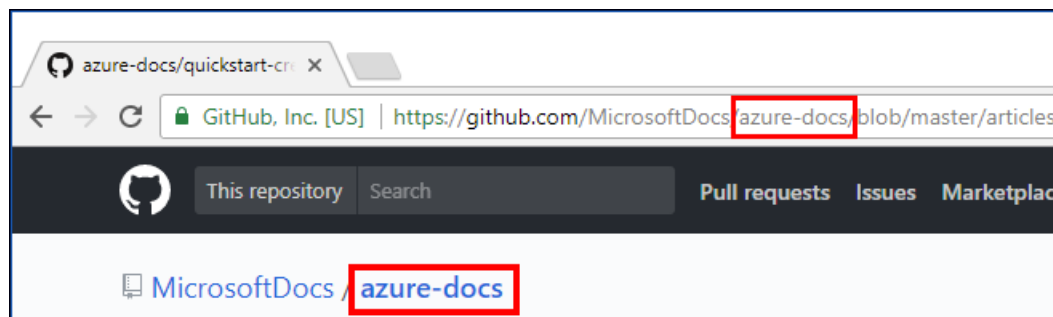
決定存放庫

裝載於 docs.microsoft.com 的文件位於 github.com 上數個不同的存放庫。

1. 如果您不確定要使用哪個存放庫，請使用您的網頁瀏覽器瀏覽 docs.microsoft.com 上的文章。選取文章右上角的 [編輯] 連結 (鉛筆圖示)。



2. 該連結會帶您前往適當存放庫中所對應 Markdown 檔案的 github.com 位置。請記下檢視存放庫名稱的 URL。



例如，這些受歡迎的存放庫可供公開參與使用：

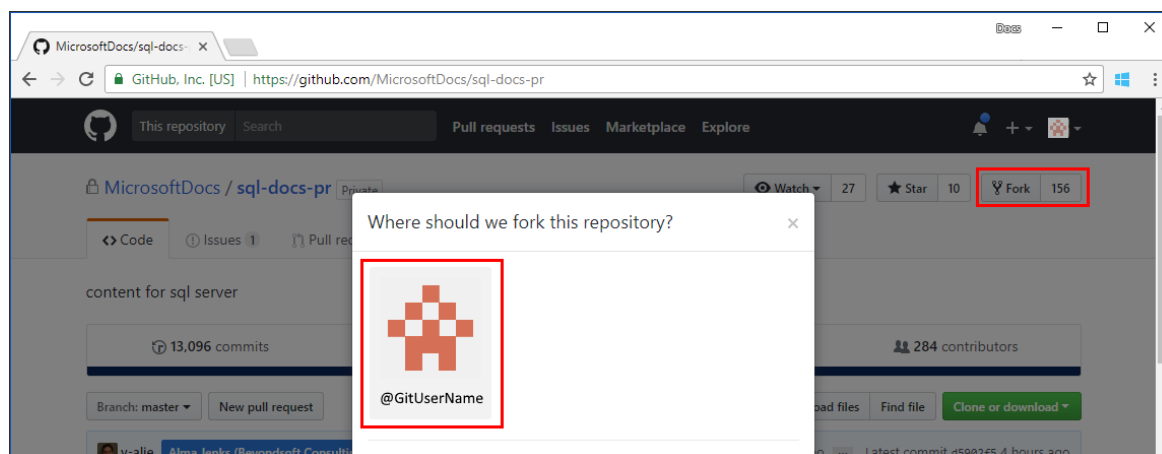
- Azure 文件 <https://github.com/MicrosoftDocs/azure-docs>
- SQL Server 文件 <https://github.com/MicrosoftDocs/sql-docs>
- Visual Studio 文件 <https://github.com/MicrosoftDocs/visualstudio-docs>
- .NET 文件 <https://github.com/dotnet/docs>
- Azure .Net SDK 文件 <https://github.com/azure/azure-docs-sdk-dotnet>
- ConfigMgr 文件 <https://github.com/MicrosoftDocs/SCCMdocs>

派生存放庫

透過 GitHub 網站使用適當的存放庫，建立該存放庫到您自己 GitHub 帳戶的分支。

因為所有主要文件存放庫都提供唯讀存取權，所以需要個人分支。若要進行變更，您必須從主要存放庫的分支提交 [提取要求](#)。為了加快此程序，您首先需要有自己的存放庫複本，而您在其中具有寫入權限。GitHub 分叉 即符合此用途。

1. 移至主要存放庫的 GitHub 頁面，然後按一下右上角的 [Fork] (分叉) 按鈕。



2. 出現提示時，請選取您的 GitHub 帳戶磚作為應建立分支的目的。這個提示會在您的 GitHub 帳戶中建立該存放庫的複本 (稱為分支)。

撰擇本機資料夾

請建立一個本機資料夾，以便在本機保留存放庫的複本。某些存放庫可能很大；例如，對於 azure-docs，最高可

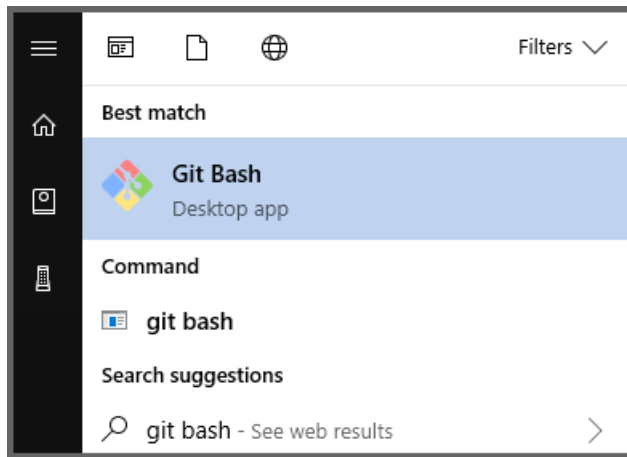
達 5 GB。請選擇具有可用磁碟空間的位置。

1. 選擇應該容易記住和鍵入的名稱。例如，考慮使用根資料夾 `C:\docs\`，或者在您的使用者設定檔目錄 `~/Documents/docs/` 中建立一個資料夾

IMPORTANT

請避免選擇以巢狀方式內嵌於另一個 Git 存放庫資料夾位置的本機資料夾路徑。雖然接受以彼此相隣的方式儲存 Git 複製的資料夾，但以巢狀方式將 Git 資料夾內嵌於另一個資料夾會導致檔案追蹤發生錯誤。

2. 啟動 Git Bash



Git Bash 的預設啟動位置通常是 Windows 作業系統上的主目錄 (~) 或 `/c/users/<Windows-user-account>/`。

若要判定目前的目錄，請在 \$ 提示下鍵入 `pwd`。

3. 將目錄切換 (cd) 到您為了在本機裝載存放庫而建立的資料夾。請注意，Git Bash 會針對資料夾路徑使用 Linux 正斜線慣例，而不是反斜線。

例如，`cd /c/docs/` 或 `cd ~/Documents/docs/`

建立本機複製品

使用 Git Bash，準備執行 `clone` 命令，以將存放庫的複本 (您的分支) 下拉至裝置的目前目錄中。

使用 Git 認證管理員進行驗證

如果您已安裝最新版本的 Git for Windows 並已接受預設安裝，Git 認證管理員預設便會啟用。Git 認證管理員能使驗證變得更加輕鬆，因為您不需要在重新建立與 GitHub 的驗證連線/遠端時，重新叫用自己的個人存取權杖。

1. 執行 `clone` 命令，並提供存放庫名稱。複製作業會將分支存放庫下載 (複製) 到本機電腦上。

TIP

從 GitHub UI 中的 [Clone or download] (複製或下載) 按鈕，可以取得複製命令的分叉 GitHub URL：

Clone or download ▾

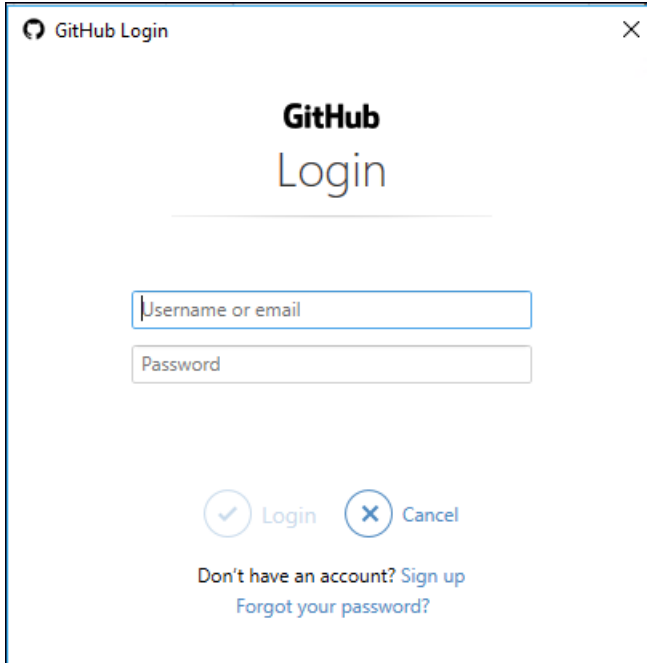
請務必於複製程序期間指定「您的分叉」的路徑，而非用於建立分叉的主要存放庫。否則，您無法參與變更。個人 GitHub 使用者帳戶可用來參考分支，例如：`github.com/<github-username>/<repo>`。

```
git clone https://github.com/<github-username>/<repo>.git
```

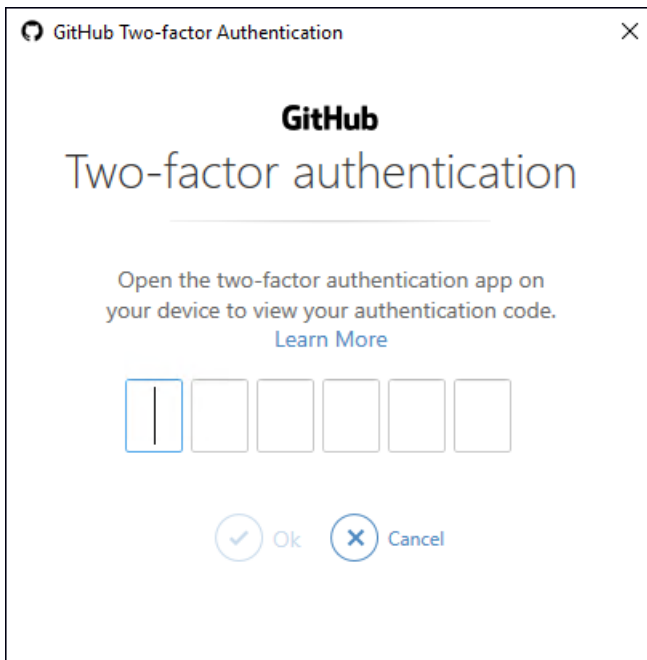
您的 clone 命令應該類似此範例：

```
git clone https://github.com/smithj/azure-docs.git
```

2. 出現提示時，請輸入您的 GitHub 認證。

A screenshot of the GitHub Login dialog box. It has a title bar with the GitHub logo and the text "GitHub Login". The main content area displays the GitHub logo and the word "Login" in a large font. Below this, there are two input fields: "Username or email" and "Password". At the bottom, there are two buttons: "Login" (with a checkmark icon) and "Cancel" (with an 'X' icon). Below the buttons, there are two links: "Don't have an account? Sign up" and "Forgot your password?".

3. 出現提示時，請輸入您的雙因素驗證碼。

A screenshot of the GitHub Two-factor Authentication dialog box. It has a title bar with the GitHub logo and the text "GitHub Two-factor Authentication". The main content area displays the GitHub logo and the text "Two-factor authentication" in a large font. Below this, there is a message: "Open the two-factor authentication app on your device to view your authentication code." followed by a link "Learn More". Below the message, there are six input boxes for the authentication code. At the bottom, there are two buttons: "Ok" (with a checkmark icon) and "Cancel" (with an 'X' icon).

NOTE

您的認證將會被儲存，並用於驗證日後的 GitHub 要求。您在每部電腦上只需執行此驗證一次。

4. clone 命令即會執行，並將分支中的存放庫檔案複本下載到本機磁碟。將會在目前的資料夾中建立新的資料夾。根據存放庫大小而定，這可能需要幾分鐘的時間。您可以在此作業完成後，瀏覽資料夾以查看結

構。

設定遠端上游

複製存放庫之後，請設定與主要存放庫 (名為**上游**) 的唯讀遠端連線。您將使用此上游 URL，讓本機存放庫與其他人所進行的最新變更保持同步。**git remote** 命令可用來設定設定值。您將會使用 **fetch** 命令從上游存放庫重新整理分支資訊。

1. 如果您使用 Git 認證管理員，請使用下列命令。取代 **<repo>** 和 **<organization>** 預留位置。

```
cd <repo>
git remote add upstream https://github.com/<organization>/<repo>.git
git fetch upstream
```

2. 檢視設定的值，並確認 URL 都正確。請確定**原始** URL 指向您的個人分支。請確定**上游** URL 指向主要存放庫，例如 MicrosoftDocs 或 Azure。

```
git remote -v
```

隨即顯示遠端輸出範例。名為 MyGitAccount 的虛構 Git 帳戶設有用來存取存放庫 azure-docs 的個人存取權杖：

```
origin https://github.com/MyGitAccount/azure-docs.git (fetch)
origin https://github.com/MyGitAccount/azure-docs.git(push)
upstream https://github.com/MicrosoftDocs/azure-docs.git (fetch)
upstream https://github.com/MicrosoftDocs/azure-docs.git (push)
```

3. 如果犯了一個錯誤，您可以移除遠端值。若要移除上游值，請執行 `git remote remove upstream`。

後續步驟

- 若要深入了解新增和更新內容，請繼續移至 [GitHub 參與工作流程](#) 頁面。

適用於 Docs 的 Git 和 GitHub 基本資訊

2021/5/13 •

概觀

身為 Docs 內容的貢獻者，您將與多種工具和程序互動。您會同時與其他貢獻者在相同專案上共同作業，而且甚至可能同時處理相同內容。這一切都是透過 Git 和 GitHub 軟體來達成。

Git 是開放原始碼版本控制系統。它透過「存放庫」中之檔案的「分散式版本控制」來促成此類型的專案共同作業。本質上來說，Git 可整合由多個參與者在不同時間，針對指定的存放庫所完成的工作流。

GitHub 是 Git 存放庫的 web 架構主機服務，例如用來儲存 docs.microsoft.com 內容的服務。針對任何專案，GitHub 都裝載主要存放庫，供貢獻者針對其自己的工作建立複本。

Git

若您熟悉集中式版本控制系統 (例如 Team Foundation Server、SharePoint 或 Visual SourceSafe)，您將注意到 Git 擁有可支援其分散式模型的唯一貢獻工作流程與術語。例如，它並沒有通常會與簽出/簽入作業一起出現的檔案鎖定功能。事實上，Git 所關心的是更精細層級中的變更，因此會進行逐位元組檔案比較。

Git 也使用分層結構來儲存及管理專案內容：

- **存放庫**：亦稱為 *repo*，這是最高的儲存單位。存放庫包含一或多個分支。
- **分支**：包含組成專案內容集合之檔案與資料夾的儲存單位。分支會分隔工作串流 (通常稱為版本)。貢獻一律是在特定分支的範圍所進行。所有存放庫都包含預設分支 (通常名為 "main")，以及一或多個目標要合併回預設分支的分支。預設分支可作為專案的最新版本和「單一真實來源」。它是存放庫中所有其他分支的建立來源。

參與者會與 Git 互動，以在本機和 GitHub 層級對存放庫進行更新和操作：

- 在本機透過 Git Bash 主控台這類工具，而這類工具支援用來管理本機存放庫以及與 GitHub 存放庫通訊的 Git 命令。
- 透過 www.github.com，它會整合 Git 來管理流回主要儲存機制的投稿對帳。

GitHub

NOTE

雖然 Docs 指導方針是以使用 GitHub 為基礎，某些小組會使用 Visual Studio Team Service 來裝載 Git 存放庫。Visual Studio Team Explorer 用戶端提供用來與 Team Services 存放庫互動的 GUI，做為透過命令列使用 Git 命令之外的替代方法。此外，下列許多指導方針都是透過多年來在 GitHub 中裝載 Azure 服務內容的經驗發展而來的最佳做法。它們某些 Docs 存放庫中可能為必要做法。

所有的工作流程都是在 GitHub 層級開始與結束，這也是任何 Doc 專案主要存放庫的儲存位置。參與者為自己的使用目的而建立的複本會跨多部電腦散佈。這些複本最終會協調回專案的主要 GitHub 存放庫。

目錄組織方式

如先前所述，專案的預設分支是專案的最新版本內容。預設分支中的內容 (以及從它建立的分支) 會與相對應檔頁面上的文章組織鬆散地對齊。系統會使用子目錄來區隔內容 (例如服務)、媒體內容 (例如影像檔案)，以及可讓內容重複使用的 "include" 檔案。

您通常可以在存放庫的根目錄找到主要 `articles` 目錄。articles 目錄包含一組子目錄。子目錄中的文章格式為 Markdown 檔案，其副檔名為 `.md`。某些支援多個服務的存放庫會使用一般的 `/articles` 子目錄 (例如 [Azure-Docs](#) 存放庫)。其他存放庫可能會使用服務特定的名稱 (例如使用 `/IntuneDocs` 的 [IntuneDocs](#) 存放庫)。

在此目錄的根內，您可以找到有關整體服務或產品的一般文章。您通常也可以接著找到另一系列的子目錄，這些子目錄會符合功能/服務或常見案例。例如，Azure "virtual machine" (虛擬機器) 文章是位於 `/virtual-machines` 子目錄，而 Intune "understand and explore" (了解與探索) 文章則位於 `/understand-explore` 子目錄。

Media 子目錄

每個 article 目錄都包含對應之媒體檔案的 `/media` 子目錄。媒體檔案包含具有影像參考之文章所使用的影像。

Includes 子目錄

只要有兩篇或多篇文章共用的可重複使用內容，系統就會將它放置在主要 `articles` 目錄的 `/includes` 子目錄中。在使用 include 檔案的 Markdown 檔案中，系統會將對應的 "include" Markdown 擴充功能放置在需要參考 include 檔案的位置。

如需其他指引，請參閱 [Markdown 參考: 包含](#)。

Markdown 檔範本

為了方便起見，每個存放庫的根目錄通常包含一個名為 `template.md` 的 Markdown 檔案。若需要建立新文章以提交到存放庫，您可以將此範本檔案做為「開始檔案」使用。此檔案包含：

- 檔案頂端的 **中繼資料標頭** 其中包含兩個具有 3 個破折號的行。它包含用於與文章相關之追蹤資訊的多種標記。文章中繼資料可啟用特定功能，例如作者姓名標示、參與者姓名標示、階層連結與文章描述。它也包括 SEO 最佳化與報告程序，Microsoft 使用它們來評估內容的成效。因此，中繼資料非常重要！
- 描述各種元資料標記和值的 **中繼資料區段**。如果您不確定在中繼資料區段中應使用的值，則可以留白，或在前面加上雜湊標記 (#) 使其成為註解，讓存放庫的提取要求檢閱者來檢閱/完成它們。
- 格式化文章元素的各種 **Markdown 使用範例**。
- 一般的 **Markdown 擴充功能使用指示**，可用於各種類型的警示。
- 使用 `iframe` 內嵌視訊的範例。
- 一般 `docs.microsoft.com` **擴充功能使用指示**，您可以將其用於特殊控制項，例如按鈕與選取器。

提取要求

「提取要求」為讓參與者建議要套用至預設分支之一系列變更的便利方式。變更 (亦稱為「認可」) 會儲存在參與者的分支中，讓 GitHub 可先模擬將它們「合併」至預設分支後所會帶來的影響。提取要求也可作為提供參與者來自建置/驗證流程、提取要求檢閱者之意見反應的機制，以在變更合併至預設分支之前解決可能發生的問題。

有兩種方式可透過提取要求來做出貢獻，這取決於您要建議之變更的大小。我們稍後會在本指南的 [GitHub 工作流程](#) 小節中詳細說明這一部分。

適用於主要或長期變更的 GitHub 參與工作流程

2021/10/9 •

IMPORTANT

所有發佈至 docs.microsoft.com 的存放庫皆採用 [Microsoft 開放原始碼管理辦法 \(英文\)](#) 或 [.NET Foundation 管理辦法 \(英文\)](#)。如需詳細資訊，請參閱[管理辦法常見問題集 \(英文\)](#)。若有任何問題或意見，也可以連絡 opencode@microsoft.com 或 conduct@dotnetfoundation.org。

您在公用存放庫中針對文件和程式碼範例所提交的次要修正或釐清，將受到 [docs.microsoft.com 使用條款](#) 的約束。如果您不是 Microsoft 的員工，在做出全新或重要的變更時，系統將會在提取要求中產生意見，要求您提交線上版的貢獻授權合約 (CLA)。您必須先完成線上表單，您的提取要求才會被合併。

概觀

此工作流程適用於需要對某個存放庫進行主要變更，或會成為該存放庫頻繁參與者的參與者。頻繁的參與者通常會有進行中 (長期) 的變更，這些變更在提取要求完成簽核並合併之前，都需要經歷多個建置/驗證/暫存循環或是花費許多天才能完成。

這些參與類型的範例包括：

- **進行大量貢獻。**例如，您的投稿 (新增/變更/刪除) 可能涉及多篇文章且需要在單一提取要求中以一個工作單位進行認可/測試。
- **建立與發佈新文章** 通常需要較強固的本機編輯器。
- **新增新的影像或更新影像** 一般需要同時建立新的 `media` 子目錄、影像檔、更新文章中的影像連結，以及使用本機編輯器預覽 Markdown 檔案，以測試影像轉譯。
- **發佈數日前的文章，請先予以更新。**在這些情況下，您通常需要定期整合預設分支中發生的其他變更。此項整合會透過 Git Bash 和本機編輯很容易執行。如果透過 GitHub UI 執行此作業並等待認可變更，您也要承擔遺失編輯的風險。
- **開啟提取要求之後，持續更新相同的文章** (除非您覺得透過 GitHub UI 執行此作業很自在)。使用 GitHub UI 有可能針對同一檔案建立多個未處理的提取要求，從而互相衝突。

術語

在您開始之前，讓我們先檢閱一些用於此工作流程的 Git/GitHub 詞彙及代號。您暫時還不需要了解它們。您只需要知道自己將會學習它們，並可以在需要確認定義時回來參考本節。

“	“
分叉 (Fork)	通常是作為名詞使用，用來描述主要 GitHub 存放庫的複本。實際上，分叉只是另一個存放庫。但它的特殊之處在於 GitHub 會維護與主要/父存放庫之間的關係。它有時會作為動詞使用，例如「您必須先針對存放庫進行分叉」。
遠端	針對遠端存放庫的具名關係，例如「原始」或「上游」遠端。Git 會將此稱為遠端，因為它是用來參考裝載於另一部電腦上的存放庫。在此工作流程中，「遠端」一律會是 GitHub 存放庫。
原始	指派給本機存放庫和複製該存放庫之原始存放庫之間關係的名稱。在此工作流程中，「原始」代表針對您分叉的關係。它有時候會作為原始存放庫本身的代號，例如「記得將您的變更推送到原始」。

“	“
---	---

上游	上游和原始遠端相同，是針對另一個存放庫的具名關係。在此工作流程中，「上游」代表您的本機存放庫及主要存放庫（也就是從之建立分叉的存放庫）之間的關係。它有時候會作為上游存放庫本身的代號，例如「記得從上游提取變更」。
----	---

工作流程

IMPORTANT

如果您尚未完成[設定](#)一節中的步驟，請務必完成它們。本節會引導您設定 GitHub 帳戶、安裝 Git Bash 和 Markdown 編輯器、建立分叉，以及設定本機存放庫。如果您不熟悉 Git 和 GitHub 概念（例如存放庫或分支等），請先檢閱[Git 和 GitHub 基本概念](#)。

在此工作流程中，變更流程會是重複性的循環。變更會從裝置的本機存放庫開始，回流至 GitHub 分叉，接著傳送至主要 GitHub 存放庫，然後隨著您納入來自其他參與者的變更再次返回本機。

使用 GitHub 流程

回想一下[git 和 GitHub 基礎](#)，git 存放庫包含預設分支，以及任何其他尚未整合到預設分支中的工作進行中分支。每當您要導入一系列邏輯相關的變更時，最佳做法為建立「工作分支」來透過工作流程管理這些變更。我們在此將其稱為工作分支，因為它是用來逐一查看/縮小變更的工作區，直到可以整合回預設分支為止。

將相關聯的變更隔離限制於特定分支中，可讓您獨立地控制及導入變更，並將其目標設定為發佈週期中的特定發行時間。在現實中，根據所進行工作的不同，您的存放庫中很容易就會出現數個工作分支。同時處理多個分支（每個分支都代表不同的專案）的情況，其實相當常見。

TIP

在預設分支中進行變更並不是理想的作法。Imagine 您使用預設分支來引進一組計時功能版本的變更。您完成變更，並正在等待發行這些變更。然後，在過渡期間，您會有緊急的要求來修正某個問題，因此您會對預設分支中的檔案進行變更，然後發佈變更。在此範例中，您會在無意中使該修正「連同」先前正在為特定發行日期保留的變更一起發佈。

現在，讓我們在本機存放庫中建立新的工作分支，以擷取您所建議的變更。如果您已設定 Git Bash（請參閱[安裝內容撰寫工具](#)），則您可以建立新的分支，並使用下列命令，從您已複製的存放庫內將該分支「簽出」：

```
git checkout -b "branchname"
```

每個 Git 用戶端都不同，因此請參閱適您慣用用戶端的說明。在[GitHub 流程](#)上的 GitHub 指南中有程序的概觀。

進行變更

您已經有 Microsoft 存放庫的複本（「複製的存放庫」），且已經建立分支，現在您可以隨意使用任何文字或 Markdown 編輯器（如[安裝內容撰寫工具](#)頁面上所述），進行您認為對社群有所幫助的任何變更。您可以在本機儲存變更，在完成變更之前都無需將其提交至 Microsoft。

儲存變更至您的存放庫

將變更傳送給作者之前，您必須先將其儲存至 Github 存放庫。同樣地，雖然所有工具都不相同，但如果您使用 Git Bash 命令列，只需要幾個簡單的步驟就可以完成這項作業。

首先，在存放庫中，您需要 **暫存** 所有變更，以準備進行下一個認可。您可以執行下列命令來完成此動作：

```
git add --all
```

接下來，您需要將已儲存的變更認可至本機存放庫。您可以執行下列命令以在 Git Bash 中完成此動作：

```
git commit -m "Short Description of Changes Made"
```

最後，由於您是在本機電腦上建立此分支，因此需要為 GitHub.com 帳戶中的分支同步更新資訊。如果您使用 Git Bash，則可執行下列命令來完成此動作：

```
git push --set-upstream origin <branchname>
```

大功告成！您的程式碼現在已經在 GitHub 存放庫中，並可供建立提取要求。

TIP

雖然您的變更會在推送時顯示於您個人 GitHub 帳戶中，但您並不需要立即提交提取要求。如果您想要暫停並稍後再回來進行其他調整也沒關係！

需要修正您已經提交的東西嗎？沒問題！只要在相同分支中進行變更，然後再次認可並推送即可 (不需要在相同分支的後續推送上設定上游伺服器)。

進行您的下一個變更

還需要進行更多其他項目的變更嗎？切換回預設分支，從上游存放庫中提取以確定您的分支是最新的，並查看新的分支。在 Git Bash 中執行下列命令：

```
git checkout main
git pull upstream main
git checkout -b "branchname"
```

NOTE

上述命令會假設您正在使用的存放庫 `main` 做為其預設分支。如果第一個命令失敗，可能是預設分支未重新命名。`main` 將前兩個命令中的取代 `master` 為，以確認這一點。

您現在已回到新的分支，而且您已準備好成為專家參與者。

提取要求會處理

上一節引導您完成提交建議變更的程序，方法是將它們繫結在新的提取要求 (PR) 中，此 PR 會新增到目的地存放庫的 PR 佇列。提取要求會要求提取您工作分支的變更，合併到其他分支，以啟用 GitHub 的共同作業模型。在大部分情況下，其他分支是主要存放庫中的預設分支。

驗證

您的提取要求可能要先通過一或多項 PR 驗證程序，才能合併到其目的地分支。驗證程序因建議變更範圍及目的地存放庫規則而異。提交提取要求之後，您可以期待出現下列一或多種情況：

- **合併功能**：基準 GitHub 合併功能測試會先發生，以確認分支中的建議變更是否與目的分支不衝突。如果提取要求指出此測試失敗，您即必須先協調造成合併衝突的內容，處理程序才能繼續。
- **CLA**：如果您要向公共存放庫發表貢獻，但不是 Microsoft 員工，視建議變更的範圍而定，第一次向該存放庫提交提取要求時，系統可能會要求您先完成簡短的貢獻授權合約 (CLA)。完成 CLA 步驟之後，就會處理您的提取要求。
- **標記**：標籤會自動套用至您的提取要求，當提取要求通過驗證工作流程時指示其狀態。例如，新的提取要求會自動接收 "do-not-merge" 標籤，指出提取要求尚未完成驗證、檢閱及登出步驟。
- **驗證和建置**：自動檢查會驗證變更是否通過驗證測試。驗證測試可能會發出警告或錯誤，要求您先對提取要求中的一或多個檔案進行變更，再執行合併。驗證測試結果會新增為提取要求中的註解供您檢閱，也可能以電子郵件傳送給您。
- **預備位置**：受變更影響的文章頁面會自動部署到預備環境，於驗證及建置成功時檢閱。PR 註解會顯示預覽 URL。
- **自動合併**：如果提取要求通過驗證測試及特定條件，可能會自動合併。如果是這種情況，您不需要採取任何進一步的動作。

檢閱與登出

完成所有 PR 處理程序之後，您應該檢閱結果 (PR 註解、預覽 URL 等等) 以決定是否需要對其檔案進行額外變更，再登出進行合併。如果 PR 檢閱者已檢閱過您的提取要求，如有未處理的問題/疑問需要在合併前解決，他們也可以透過註解提供意見反應。

註解自動化可以將適當的標籤指派給提取要求，讓讀取層級的使用者 (在存放庫中沒有寫入權限的使用者) 執行寫入層級的動作。如果您是在已實作註解自動化的存放庫中工作，請使用下列表格中的雜湊註解來指派標籤、變更標籤或關閉提取要求。每當您所撰寫的文章收到變更的建議時，Microsoft 員工也會透過電子郵件收到檢閱及簽核公用存放庫提取要求的通知。

#####	##	#####
#sign-off	<p>當某篇文章的作者在意見資料流中輸入 <code>#sign-off</code> 留言時，會指派 ready-to-merge 標籤。此標籤可讓存放庫中的檢閱者知道提取要求已準備好進行檢閱/合併。</p> <p>若有未列於作者清單中的參與者嘗試使用 <code>#sign-off</code> 留言來簽核公開存放庫提取要求，系統會在提取要求中寫入訊息，表示只有作者可以指派該標籤。</p>	公用和私人
#hold-off	<p>當作者改變心意或犯下錯誤時，可以在提取要求意見中輸入 <code>#hold-off</code>，移除 ready-to-merge 標籤。在私人存放庫中，這會指派 do-not-merge 標籤。</p>	公用和私人
#please-close	<p>若作者決定不要合併變更，可以在意見資料流中輸入 <code>#please-close</code>，關閉提取要求。</p>	公用

當提取要求沒有問題且已登出後，您的變更即會合併回父分支並關閉提取要求。

發佈

請記住，提取要求必須先由 PR 檢閱者合併，下次排程的發佈回合才能納入變更。通常是依提交順序檢閱/合併提取要求。如果您的提取要求需要針對特定的發佈回合進行合併，您就需要提前與 PR 檢閱者合作，以確保先合併再發佈。

投稿經核准與合併後，就交由 docs.microsoft.com 發佈程序挑選。發佈時間會因您投稿的存放庫管理小組而異。

經由下列路徑發佈的文章通常約在太平洋時間週一至週五上午 10:30 和下午 3:30 部署：

- <https://docs.microsoft.com/azure/>
- <https://docs.microsoft.com/aspnet/>
- <https://docs.microsoft.com/dotnet/>
- <https://docs.microsoft.com/enterprise-mobility-security>

文章發佈上線最多約需 45 分鐘。文章發佈後，您可在正確的 URL 驗證變更：

`https://docs.microsoft.com/<path-to-your-article-without-the-md-extension>`。

後續步驟

就這麼簡單！您已對 docs.microsoft.com 內容做出貢獻！

- 若要深入了解 Markdown 和 Markdown 延伸模組語法等主題，請繼續閱讀 [Markdown 參考](#)一文。

檔 Markdown 參考

2021/9/24 •

本文提供針對 docs.microsoft.com (檔撰寫 Markdown 的字母順序參考)。

Markdown 是採用純文字格式語法的輕量型標記語言。檔支援透過**Markdig**剖析引擎進行**CommonMark**相容的 Markdown。檔也支援自訂的 Markdown 延伸模組，可在檔網站上提供更豐富的内容。

您可以使用任何文字編輯器來撰寫 Markdown，但我們建議使用檔**撰寫套件 Visual Studio Code**。檔撰寫套件提供編輯工具和預覽功能，可讓您在轉譯檔時查看文章的外觀。

警示 (附註、提示、重要、注意、警告)

警示是用來建立區塊引號的 Markdown 延伸模組，其色彩和圖示會指出内容的重要性。

避免附註、秘訣和重要方塊。讀者傾向於略過它們。最好將該資訊直接放入文章文字中。

如果您需要使用警示，請將其限制為每一篇文章一或兩次。文章中的多個附註絕對不能彼此相鄰。

支援的警示類型如下：

```
> [!NOTE]
> Information the user should notice even if skimming.

> [!TIP]
> Optional information to help a user be more successful.

> [!IMPORTANT]
> Essential information required for user success.

> [!CAUTION]
> Negative potential consequences of an action.

> [!WARNING]
> Dangerous certain consequences of an action.
```

這些警示在檔上看起來像這樣：

NOTE

Information the user should notice even if skimming.

TIP

Optional information to help a user be more successful.

IMPORTANT

Essential information required for user success.

Caution

Negative potential consequences of an action.

WARNING

動作可能會導致危險的結果。

角括號

如果您在檔案的文字中使用角括弧 (例如, 表示預留位置), 您必須手動編碼角括弧。否則 Markdown 會將它們視為 HTML 標籤。

例如, 將編碼 `<script name>` 為 `<script name>` 或 `\<script name>`。

角括弧不需要在格式化為內嵌程式碼或程式碼區塊中的文字中進行轉義。

縮寫符號和雙引號

如果您將內容從 Word 複製到 Markdown 編輯器中, 文字可能會包含「智慧」(彎曲) 縮寫符號或雙引號。這些必須編碼或變更為基本縮寫符號或雙引號。否則, 您會在發行檔案時得到如下的內容: 最根本 € ™ s

以下是這些標點符號的「智慧」版本編碼:

- 左 (開頭) 雙引號: `“`
- 右 (結尾) 雙引號: `”`
- 右 (結尾) 單引號或縮寫符號: `’`
- 左 (開頭) 單引號 (很少使用): `‘`

引文

區塊引述使用 `>` 字元建立:

```
> This is a blockquote. It is usually rendered indented and with a different background color.
```

前述範例會如下呈現:

這是區塊引述。它通常會以縮排方式轉譯並具有不同的背景色彩。

粗體與斜體文字

若要将文字格式化為 **粗體**, 請以兩個星號括住:

```
This text is **bold**.
```

若要将文字格式設定為 *斜體*, 請將它括在單一星號:

```
This text is *italic*.
```

若要将文字格式設定為 ***粗體和斜體***, 請以三個星號括住:

```
This text is both ***bold and italic***.
```

如需何時使用粗體和斜體文字的指引, 請參閱 [文字格式設定指導方針](#)。

程式碼片段

檔 Markdown 支援將程式碼片段內嵌在句子中，並在句子之間以個別的 "隔離" 區塊來放置。如需詳細資訊，請參閱 [如何將程式碼新增至檔](#)。

資料行

Markdown **延伸** 模組可讓檔作者新增以資料行為基礎的內容版面配置，此配置比基本 Markdown 資料表更具彈性且功能強大，而且僅適用於真正的表格式資料。您最多可以加入四個數據行，並使用選擇性 `span` 屬性來合併兩個或多個資料行。

資料行的語法如下所示：

```
:::row::  
  :::column span="":  
    Content...  
  :::column-end::  
  :::column span="":  
    More content...  
  :::column-end::  
:::row-end::
```

資料行應該只包含基本 Markdown，包括影像。不應包含標題、資料表、索引標籤和其他複雜結構。資料列不能有資料行以外的任何內容。

例如，下列 Markdown 會建立一個跨越兩個數據行寬度的資料行，而一個標準 (沒有) 資料行 `span`：

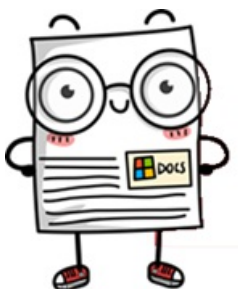
```
:::row::  
  :::column span="2":  
    **This is a 2-span column with lots of text.**  
  
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vestibulum mollis nunc  
    ornare commodo. Nullam ac metus imperdiet, rutrum justo vel, vulputate leo. Donec  
    rutrum non eros eget consectetur.  
  :::column-end::  
  :::column span="":  
    **This is a single-span column with an image in it.**  
  
    ![Doc.U.Ment](media/markdown-reference/document.png)  
  :::column-end::  
:::row-end::
```

這會轉譯為：

這是具有大量文字的 2 span 資料行。

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vestibulum mollis nunc ornare commodo. Nullam ac metus imperdiet, rutrum justo vel, vulputate leo. Donec rutrum non eros eget consectetur。

這是單一範圍的資料行，其中有影像。



註解

如果您必須將文章的章節批註為批註，檔可支援 HTML 批註：

```
<!-- Here's my comment -->
```

WARNING

請勿在 HTML 批註中放入私用或機密資訊。檔會將 HTML 批註傳送至公開的已發佈 HTML。雖然讀者的眼睛看不到 HTML 批註，但它們會在下方的 HTML 中公開。

標題

Docs 支援六種層級的 Markdown 標題：

```
# This is a first level heading (H1)

## This is a second level heading (H2)

...

##### This is a sixth level heading (H6)
```

- 最後一個 `#` 和標題文字之間必須有一個空格。
- 每個 Markdown 檔都只能有一個 H1 標題。
- H1 標題必須是檔案中 YML 中繼資料區塊後的第一個內容。
- H2 標題會自動出現在已發佈檔案的右手邊導覽功能表中。較低層級的標題不會出現，因此請策略性地使用 H2 來協助讀者流覽您的內容。
- HTML 標題（例如 `<h1>`）不是建議的做法，而且在某些情況下會造成組建警告。
- 您可以透過 [書籤連結](#)，連結至檔案中的個別標題。

HTML

雖然 Markdown 支援內嵌 HTML，但不建議您在發行至檔時使用 HTML，但有限的值清單會造成組建錯誤或警告。

影像

以下是預設支援的影像檔案類型：

- .jpg
- .png

標準概念影像 (預設 Markdown)

內嵌映射的基本 Markdown 語法為：

```
![<alt text>](<folderPath>)

Example:
![alt text for image](../images/Introduction.png)
```

其中，`<alt text>` 是影像的簡短描述，而 `<folder path>` 是影像的相對路徑。適用於視障者的螢幕助讀程式需要使用替代文字。如果有無法轉譯影像的網站錯誤，它也很有用。

除非您在前面加上反斜線 ()，否則不會正確轉譯替代文字中的底線 _。不過，請勿複製檔案名以做為替代文字。例如，而不是：

```
![ADextension_2FA_Configure_Step4](./media/bogusfilename/ADextension_2FA_Configure_Step4.PNG)
```

撰寫如下：

```
![Active Directory extension for two-factor authentication, step 4: Configure]
(./media/bogusfilename/ADextension_2FA_Configure_Step4.PNG)
```

標準概念影像 (檔 Markdown)

檔自訂 `:::image:::` 延伸模組支援標準影像、複雜影像和圖示。

針對標準映射，較舊的 Markdown 語法仍可運作，但建議使用新的擴充功能，因為它支援更強大的功能，例如指定與父主題不同的當地語系化範圍。未來將提供其他先進的功能，例如從共用映射庫中選取，而不是指定本機映射。新的語法如下所示：

```
:::image type="content" source="<folderPath>" alt-text="<alt text>":::
```

如果 `type="content"` (預設)，則 `source` 和 `alt-text` 都是必要的。

具有完整描述的複雜影像

您也可以使用此延伸模組來新增具有詳細描述的影像，該影像會由螢幕讀取器讀取，但無法在已發行的頁面上以視覺化方式呈現。詳細描述是複雜影像 (例如圖形) 的協助工具需求。語法如下所示：

```
:::image type="complex" source="<folderPath>" alt-text="<alt text>":::
<long description here>
:::image-end:::
```

如果 `type="complex"`、`source`、`alt-text`、long 描述，而且 `:::image-end:::` 標記全部都是必要的。

指定 loc 範圍

有時候影像的當地語系化範圍與包含它的文章或模組的當地語系化範圍不同。這可能會造成不好的全球體驗：例如，如果產品的螢幕擷取畫面意外當地語系化成語言，則無法使用產品。若要避免這種情況，您可以 `loc-scope` 在類型和的影像中指定選擇性屬性 `content` `complex`。

圖示

影像延伸模組支援圖示，這些圖示是裝飾性影像，不應該有替代文字。圖示的語法為：

```
:::image type="icon" source="<folderPath>":::
```

如果 `type="icon"` 為，則只 `source` 應指定。

包含的 Markdown 檔案

在多篇文章中需要重複 markdown 檔案的地方，您可以使用 include 檔案。[包含] 功能會指示檔在組建時以 include 檔的內容取代參考。您可以透過下列方式使用 include：

- 內嵌：重複使用內嵌于句子內的一般文字片段。
- 區塊：您重複使用整個 Markdown 檔案作為區塊 (巢狀嵌入文章中的小節)。

內嵌或區塊 include 檔案是 Markdown (md) 檔。它可以包含任何有效的 Markdown。Include 檔案通常位於存

放庫根目錄中的通用 *include* 子目錄中。當文章發行之後，包含的檔案會直接整合到其中。

包含語法

區塊包含在自己的行上：

```
[!INCLUDE [<title>](<filepath>)]
```

內嵌包含在一行內：

```
Text before [!INCLUDE [<title>](<filepath>)] and after.
```

其中 `<title>` 是檔案名，而是檔案的 `<filepath>` 相對路徑。`INCLUDE` 必須是大寫，而且前面必須有一個空格 `<title>`。

以下是 Include 檔案的需求與考量：

- 針對大量內容 (例如一兩個段落、共用的程序或共用的節) 使用區塊包含。請勿將它們用在小於一個句子的內容。
- include 不會在您的文章的 GitHub 轉譯視圖中轉譯，因為它們依賴檔延伸模組。它們將會在發行集之後才轉譯。
- 確定 include 檔案中的所有文字都是以完整的句子或片語所撰寫，而不相依赖于參考包含的文章中的前幾個文字或之後的文字。忽略此指導方針會在本文中建立產生無法翻譯字串。
- 請勿將 include 檔案內嵌在其他 include 檔案中。
- 將媒體檔案放在包含子目錄的特定媒體檔案夾中 (例如，`<repo>/includes/media` 資料夾)。媒體目錄的根目錄中不應包含任何影像。如果 include 沒有影像，則不需要對應的 媒體 目錄。
- 對於一般文章，請勿在不同包含檔案之間共用媒體。請針對每個包含檔案和文章，使用具有唯一名稱的個別檔案。將媒體檔案儲存在與該包含檔案相關聯的 [media] 資料夾。
- 請勿將包含檔案做為文章的唯一內容。包含檔案是設計成用來補充文章其他部分中的內容。

縮排

在 Markdown 中，行第一個字元之前的空格決定相對於前幾行的行對齊。縮排尤其會影響編號和項目符號清單，以階層式或大綱格式轉譯多個層級的嵌套。

若要縮排文字以對齊先前的段落或編號或項目符號清單中的專案，請使用空格。

下列兩個範例顯示縮排段落如何根據與其他段落的關聯性呈現。

```
1. This is a numbered list example (one space after the period before the letter T).
   This sentence is indented three spaces.
   This code block is indented three spaces.

- This is a bulleted list example (one space after the bullet before the letter T).
  This sentence is indented two spaces.
  > [!TIP]
  > This tip is indented two spaces.
- This is a second-level bullet (indented two spaces, with one space after the bullet before the letter T).
  This sentence is indented four spaces.
  > This quote block is indented four spaces.
```

上述範例會轉譯為：

1. 這是編號清單範例 (在字母 T) 之前的一個空格之後。

這個句子是縮排的三個空格。

This code block is indented three spaces.

- 這是項目符號清單範例 (在字母 T) 之前的專案符號後面一個空格。

此句子會縮排兩個空格。

TIP

這是縮排兩個空格的提示。

- 這是第二層的專案符號 (縮排兩個空格, 在字母 T) 之前的專案符號後面有一個空格。

這個句子是縮排的四個空格。

這個引號區塊是縮排的四個空格。

連結

如需連結語法的詳細資訊, 請參閱 [使用檔中的連結](#)。

清單 (編號、分項、檢查清單)

編號清單

若要建立編號清單, 您可以全部使用1個。在發行時, 數位會以遞增順序轉譯為連續清單。為了提高來源可讀性, 您可以手動遞增清單。

請勿在清單中使用字母, 包括嵌套清單。發行至檔時, 不會正確轉譯。使用數位的嵌套清單會在發行時轉譯成小寫字母。例如:

```
1. This is
1. a parent numbered list
    1. and this is
    1. a nested numbered list
1. (fin)
```

這會轉譯為:

1. This is
2. a parent numbered list
 - a. and this is
 - b. a nested numbered list
3. (fin)

項目符號清單

若要建立項目符號清單, `-` 請 `*` 在每一行開頭使用或後面接著空格:

```
- This is
- a parent bulleted list
  - and this is
  - a nested bulleted list
- All done!
```

這會轉譯為:

- This is
- a parent bulleted list
 - and this is
 - a nested bulleted list
- All done!

無論您使用哪種語法，`-` 或在 `*` 一篇文章中一致使用它。

檢查清單

您可以透過自訂 Markdown 延伸模組，在檔上使用檢查清單：

```
> [!div class="checklist"]
> * List item 1
> * List item 2
> * List item 3
```

此範例會在檔中呈現，如下所示：

- 清單項目 1
- List item 2
- 清單項目 3

您可在文章的開頭或結尾使用檢查清單，來摘要「您將學到什麼」或「您已學到的內容」。請不要在整篇文章中新增隨機檢查清單。

下一步動作

您可以使用自訂延伸模組，將下一步動作按鈕新增至檔頁面。

語法如下：

```
> [!div class="nextstepaction"]
> [button text](link to topic)
```

例如：

```
> [!div class="nextstepaction"]
> [Learn about adding code to articles](code-in-docs.md)
```

這會轉譯為：

[瞭解如何將程式碼新增至文章](#)

您可以在下一步動作中使用任何支援的連結，包括其他網頁的 Markdown 連結。在大部分情況下，下一個動作連結會是相同 docset 中另一個檔案的相對連結。

非當地語系化的字串

您可以使用自訂 `no-loc` Markdown 延伸模組來識別您想要當地語系化程式忽略的內容字串。

所有呼叫的字串都會區分大小寫；也就是，字串必須完全符合，才能針對當地語系化加以忽略。

若要將個別字串標示為不可當地語系化，請使用下列語法：

```
:::no-loc text="String":::
```

例如，在下列範例中，當地語系化程式期間只會忽略的單一實例 `Document`：

```
# Heading 1 of the Document
```

Markdown content within the `:::no-loc text="Document":::`. The are multiple instances of `Document`, `document`, and `documents`.

NOTE

用 `\` 來對特殊字元進行換用：

```
Lorem :::no-loc text="Find a \"Quotation\""::: Ipsum.
```

您也可以使用 YAML 標頭中的中繼資料，將目前 Markdown 檔案內字串的所有實例標記為不可當地語系化：

```
author: cillroy
no-loc: [Global, Strings, to be, Ignored]
```

NOTE

非 `loc` 中繼資料在 *docfx* 中不支援做為全域中繼資料。當地語系化管線不會讀取 *docfx*，因此必須將非 `loc` 中繼資料加入至每個個別的原始檔。

在下列範例中，在中繼資料 `title` 和 Markdown 標頭中，`Document` 會在當地語系化程式期間忽略單字。

在中繼資料 `description` 和 Markdown 的主要內容中 `document`，文字會當地語系化，因為它不是以大寫字母開頭 `D`。

```
---
title: Title of the Document
author: author-name
description: Description for the document
no-loc: [Title, Document]
---
# Heading 1 of the Document

Markdown content within the document.
```

選取器

選取器是使用者介面元素，可讓使用者在同一篇文章的多個類別之間切換。在某些檔集中使用它們來解決跨技術或平臺的各項差異。選取器通常最適用於開發人員的行動平臺內容。

因為相同的選取器 Markdown 會放在使用選取器的每個發行項檔案中，所以建議您將文章的選擇器放在 `include` 檔案中。然後，您可以在所有使用相同選取器的發行項檔案中參考該 `include` 檔案。

選取器有兩種類型：單一選取器和多重選取器。

單一選取器

```
> [!div class="op_single_selector"]
> - [Universal Windows](../articles/notification-hubs-windows-store-dotnet-get-started/)
> - [Windows Phone](../articles/notification-hubs-windows-phone-get-started/)
> - [iOS](../articles/notification-hubs-ios-get-started/)
> - [Android](../articles/notification-hubs-android-get-started/)
> - [Kindle](../articles/notification-hubs-kindle-get-started/)
> - [Baidu](../articles/notification-hubs-baidu-get-started/)
> - [Xamarin.iOS](../articles/partner-xamarin-notification-hubs-ios-get-started/)
> - [Xamarin.Android](../articles/partner-xamarin-notification-hubs-android-get-started/)
```

... 轉譯結果如下：

多重選取器

```
> [!div class="op_multi_selector" title1="Platform" title2="Backend"]
> - [(iOS | .NET)](/mobile-services-dotnet-backend-ios-get-started-push.md)
> - [(iOS | JavaScript)](/mobile-services-javascript-backend-ios-get-started-push.md)
> - [(Windows universal C# | .NET)](/mobile-services-dotnet-backend-windows-universal-dotnet-get-started-push.md)
> - [(Windows universal C# | Javascript)](/mobile-services-javascript-backend-windows-universal-dotnet-get-started-push.md)
> - [(Windows Phone | .NET)](/mobile-services-dotnet-backend-windows-phone-get-started-push.md)
> - [(Windows Phone | Javascript)](/mobile-services-javascript-backend-windows-phone-get-started-push.md)
> - [(Android | .NET)](/mobile-services-dotnet-backend-android-get-started-push.md)
> - [(Android | Javascript)](/mobile-services-javascript-backend-android-get-started-push.md)
> - [(Xamarin iOS | Javascript)](/partner-xamarin-mobile-services-ios-get-started-push.md)
> - [(Xamarin Android | Javascript)](/partner-xamarin-mobile-services-android-get-started-push.md)
```

... 轉譯結果如下：

注標和上標

您應該只在需要時使用注標或上標，以取得技術精確度，例如在撰寫數學公式時。請勿將它們用於非標準樣式，例如註腳。

針對注標和上標，使用 HTML：

```
Hello <sub>This is subscript!</sub>
```

這會轉譯為：

您好，這是注標！

```
Goodbye <sup>This is superscript!</sup>
```

這會轉譯為：

再見！

資料表

在 Markdown 中建立表格的最簡單做法是使用直立線符號及線條。若要建立含標題的標準表格，請沿著第一個線段與虛線：

```
|This is |a simple |table header|
|-----|-----|-----|
|table |data |here |
|it doesn't|actually |have to line up nicely!|
```

這會轉譯為：

THIS IS	A SIMPLE	TABLE HEADER
資料表	data	這裡
it doesn't	actually	have to line up nicely!

您可以使用冒號對齊資料行：

```
| Fun | With | Tables |
| :----- | -----: |:-----:|
| left-aligned column | right-aligned column | centered column |
| $100 | $100 | $100 |
| $10 | $10 | $10 |
| $1 | $1 | $1 |
```

轉譯如下：

⌵	⌵	⌴
left-aligned column	right-aligned column	centered column
美金 100 元	美金 100 元	美金 100 元
\$10	\$10	\$10
\$1	\$1	\$1

TIP

適用於 VS Code 的 Docs 編寫延伸模組可讓您輕鬆新增基本 Markdown 表格！

您也可以使用[線上表格產生器](#)。

任何表格儲存格中的單字內換行

Markdown 資料表中的長字可能會讓資料表展開至正確的導覽，並變成無法讀取。若要解決此問題，您可以在必要時，允許檔轉譯自動在單字內插入分行符號。您只要使用自訂類別 `[!div class="mx-tdBreakAll"]` 加在表格前後即可。

以下是具有三列的表格 Markdown 範例，會以類別名稱 `mx-tdBreakAll` 的 `div` 加在表格前後。


```
> [!div class="mx-tdBreakAll"]
> |Name|Syntax|Mandatory for silent installation?|Description|
> |-----|-----|-----|-----|
> |Quiet|/quiet|Yes|Runs the installer, displaying no UI and no prompts.|
> |NoRestart|/norestart|No|Suppresses any attempts to restart. By default, the UI will prompt before
restart.|
> |Help|/help|No|Provides help and quick reference. Displays the correct use of the setup command, including
a list of all options and behaviors.|
```

轉譯結果如下：

NAME	⌘	????????? ?	⌘
Quiet	/quiet	是	執行安裝程式，而不顯示 UI 和提示。
NoRestart	/norestart	否	抑制任何重新啟動嘗試。根據預設，UI 會在重新啟動前出現提示。
[說明]	/help	否	提供說明和快速參考。顯示安裝命令的正確用法，包括所有選項和行為的清單。

第二個數據行資料表單元格的單字內換行

您可能只想要在資料表的第二個數據行中，將分行符號自動插入單字內。若要將中斷限制在第二個數據行，請使用包裝函式語法來套用類別， `mx-tdCol2BreakAll` `div` 如先前所示。

資料矩陣資料表

資料矩陣資料表有標頭和加權的第一個資料行，在左上方建立具有空白資料格的矩陣。檔具有適用於資料矩陣資料表的自訂 Markdown：

```
| |Header 1 |Header 2|
|-----|-----|-----|
|**First column A**|Cell 1A |Cell 2A |
|**First column B**|Cell 1B |Cell 2B |
```

第一個資料行中的每個專案都必須將樣式設為粗體 (`**bold**`) ; 否則，這些資料表將無法供螢幕讀取器存取，也不適用於檔。

HTML 表格

檔不建議使用 HTML 表格。它們在來源中不是人類可讀取的，這是 Markdown 的主要原則。

Docs 風格和語氣快速入門

2021/6/23 •

此快速入門是有關撰寫技術內容以發佈在 docs.microsoft.com 的簡短指南。無論您要建立新文件或更新現有文件，這些指導方針均適用。

最佳做法：

- 即使您需要複製並貼入 Microsoft Word 來進行檢查，也請檢查您文章中的拼字與文法。
- 使用輕鬆友善的語氣，如同您正在和另一個人交談一般。
- 使用簡單的句子。容易閱讀的句子表示讀者可以快速使用您分享的指導方針。

使用 Microsoft 語氣原則

當我們撰寫 docs.microsoft.com 的技術內容時，我們追求遵循這些原則。雖然有時可能做不到，但我們必須持續努力！

- **聚焦於目的上：**當客戶參閱我們的文件時，他們的內心一定具有某個特定的目的。在您開始撰寫之前，請明確判斷出目標客戶的類型，以及客戶要嘗試執行的工作內容。接著，請撰寫文章以協助此類特定客戶執行該項特定工作。
- **使用日常用字：**嘗試使用自然語言，您的客戶使用的語言。不用太正式，但是必須維持科技風格的正確性。提供說明新概念的範例。
- **以一致的風格撰寫：**避免使用贅字。使用肯定句，並避免使用贅字或大量修飾詞。保持句子簡短明瞭。讓您的文章成為焦點。如果工作有限制條件，請將它置於句子或段落的開頭。此外，請盡量將附註的數目降到最低。如果使用螢幕擷取畫面可以節省字數，請使用它。
- **讓文章易於概覽：**將最重要的事項放在最前面。使用小節將較長程序分割為更容易管理的步驟群組。具有 12 個以上步驟的（程式可能太長。）在新增更清楚的情況時使用螢幕擷取畫面。
- **表現同理心：**在文章中使用具支持性的正面語氣，並使免責聲明的數目降到最低。誠實地說明容易使客戶感到挫折的區域。確保文章著重在對客戶而言重要的內容，不要只是提供技術性說明。

考量當地語系化和機器翻譯

我們的技術文章會翻譯為數種語言，且某些文章會針對特定市場或地理位置進行修改。讀者也可能會使用網路上的機器翻譯來閱讀技術文章。因此在撰寫時，請務必將下列指導方針牢記在心：

- **確定文章沒有文法、拼字或標點符號錯誤：**這是我們應該做到的基本門檻。有些 Markdown 編輯器（例如 MarkdownPad 2.0）有基本的拼字檢查工具，但最佳作法是將文章的（已轉譯的 HTML）內容貼入 Word 來檢查，因為 Word 擁有更健全的拼字和文法檢查工具。
- **儘可能縮短句子：**複合句或是一連串的子句會使翻譯變得困難。如果可以，請在不會使句子過於冗長或讀起來很奇怪的前提下，將句子分成多段。我們也不希望文章是以不自然的語言撰寫。
- **使用簡單且一致的句子結構：**一致性會使翻譯更加容易。避免括號和旁白，並盡可能讓主詞靠近句子的開頭。查看一些新發行的文章。若文章有友善、易讀的風格，請以它為範本。
- **用字和大小寫維持一致：**再次強調，一致性很重要。如果某個字不是句子的開頭或專有名詞，請勿將它大寫。
- **使用「小字」：**我們會認為英文中的某些字詞是無關緊要的（例如 "a"、"the"、"that" 和 "is"），因為我們可以就內容來了解整體的意思。但這類字詞對於機器翻譯而言非常重要。因此請務必包括這類「小字」。

其他應注意的風格和語氣問題

- 不要使用註解或旁白來分割步驟。

- 針對包括程式碼片段的步驟，可將關於該步驟的其他資訊放入程式碼做為註解。這樣可減少讀者必須閱讀的文字量。重要資訊會複製到程式碼專案中，以在讀者於稍後參考它時，能提醒讀者該程式碼的作用。
- 針對所有標題使用句子大小寫 (僅限英文)。
- 使用 "sign-in" 而不是 "log-in"。
- 如需詳細方針，請參閱 [Microsoft 書寫樣式指南](#)。

當地語系化的文件

- 如果您正參與將文件當地語系化的工作，請參閱 [Microsoft 語言入口網站](#)。
- 如需當地語系化指導方針、技術性出版物之語言風格及使用方式的相關資訊，以及市場特定資料格式的相關資訊，請下載您所使用之語言的[樣式指南](#)。
- 如需已當地語系化的 Microsoft 詞彙，請搜尋[產品特定的已核准詞彙](#)，或下載您所使用之語言的 [Microsoft 詞彙集](#)。
- 若要深入了解當地語系化，請參閱 [Microsoft 書寫樣式指南](#)中的〈全球通訊〉。

NOTE

大部分當地語系化的檔不提供透過 GitHub 編輯或提供意見反應的功能。若要針對當地語系化的內容提供意見反應，請使用 [aka.ms/DocSiteLocFeedback](#)提供的電子郵件範本。Azure 檔有一個例外狀況。Azure 檔有公開當地語系化的 GitHub 存放庫，適用於6種語言的社區貢獻：簡體中文、法文、德文、日文、韓文和西班牙文。針對所有其他語言和內容集，請使用 [電子郵件範本](#) (以任何語言接收您的輸入) 來回報翻譯問題或提供技術意見反應。

如何在文件中包含程式碼

2021/7/16 •

有數種方法可以將程式碼包含在於 docs.microsoft.com 上發行的文章之中：

- 位於文字內的個別元素 (單字)。

以下是 `code` 樣式的範例。

當涉及文字裡附近程式碼區塊中的具名參數和變數時，請使用程式碼格式。程式碼格式也可以用於屬性、方法、類別、語言關鍵字。如需詳細資訊，請參閱本文稍後的[程式碼項目](#)。

- 位於文章 Markdown 檔案中的程式碼區塊。

```
```csharp
public static void Log(string message)
{
 _logger.LogInformation(message);
}
```
```

當藉由參考程式碼檔案來顯示程式碼的做法不可行時，請使用內嵌程式碼區塊。如需詳細資訊，請參閱本文稍後的[程式碼區塊](#)。

- 參考本機存放庫中程式碼檔案的程式碼區塊。

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" range="2-24,26":::
```

如需詳細資訊，請參閱本文稍後的[存放庫內的程式碼片段參考](#)。

- 參考位於另一個存放庫中程式碼檔案的程式碼區塊。

```
:::code language="csharp" source="~/samples-durable-functions/samples/csx/shared/Location.csx" highlight="2,5":::
```

如需詳細資訊，請參閱本文稍後的[存放庫外的程式碼片段參考](#)。

- 讓使用者在瀏覽器中執行程式碼的程式碼區塊。

```
:::code source="PowerShell.ps1" interactive="cloudshell-powershell":::
```

如需詳細資訊，請參閱本文稍後的[互動式程式碼片段](#)。

除了解釋上述包含程式碼方式的 Markdown 之外，文章也會提供一些[針對所有程式碼區塊的一般指引](#)。

程式碼元素

「程式碼元素」為程式設計語言關鍵字、類別名稱、屬性名稱等等。要界定什麼是程式碼，本身並不是一件簡單明瞭的事。例如，NuGet 套件名稱應該要視為程式碼進行處理。若有疑慮，請參閱[文字格式設定指導方針](#)。

內嵌程式碼樣式

若要將程式碼元素包含在文章文字中，請以倒引號 (```) 環繞它以指出程式碼樣式。內嵌程式碼樣式不應使用三個倒引號格式。

| MARKDOWN | HTML |
|---|---|
| 根據預設，Entity Framework 會將名為 <code>`Id`</code> 或 <code>`ClassnameID`</code> 的屬性解譯為主索引鍵。 | 根據預設，Entity Framework 會將名為 <code>Id</code> 或 <code>ClassnameID</code> 的屬性解譯為主索引鍵。 |

對文章進行當地語系化 (翻譯成其他語言) 之後，樣式被設定為程式碼的文字將不會被翻譯。如果您想要在不使用程式碼樣式的情況下防止當地語系化，請參閱 [不可當地語系化的字串](#)。

粗體樣式

某些較舊的樣式指南會針對內嵌程式碼指定粗體。在程式碼樣式非常突兀並會影響可讀性時，便可以使用粗體。例如，如果 Markdown 資料表中多為程式碼元素，滿滿的程式碼樣式看起來會很擁擠。如果您選擇使用粗體樣式，請使用 [不可當地語系化的字串語法](#)，確保不會將程式碼當地語系化。

連結

在某些內容中，指向參考文件的連結可能比程式碼格式更有用。如果您使用連結，請勿將程式碼格式套用至連結文字。將連結的樣式設為程式碼，可能會無法明確表示該文字實際上為連結。

如果您使用連結，並於相同內容的稍後又參考相同的元素，請在之後的例項使用程式碼格式 (而不是連結)。

預留位置

如果您想要讓使用者以自己的值取代顯示的程式碼區段，請使用預留位置文字 (以角括弧或大括弧標示)。例如：
`az group delete -n <ResourceGroupName>`。說明在替代成實際值時，必須移除角括弧或大括弧。

程式碼區塊

在文件中包含程式碼的語法，會視程式碼所在的位置而有所不同：

- [位於文章 Markdown 檔案中](#)
- [位於相同存放庫中的程式碼檔案中](#)
- [位於不同存放庫中的程式碼檔案中](#)

下列指導方針適用於全部三種類型的程式碼區塊：

- [自動化程式碼驗證](#)。
- [反白顯示關鍵程式碼行](#)。
- [避免水平捲軸](#)。

螢幕擷取畫面

上一節所列的所有方法都會產生可使用的程式碼區塊：

- 您可以複製並貼上這些區塊。
- 這些區塊均由搜尋引擎編制索引。
- 螢幕助讀程式可以存取這些區塊。

這些只是不建議使用 IDE 螢幕擷取畫面在文章中包含程式碼的其中幾個原因。只有當您要顯示與 IDE 本身相關的內容時 (例如 IntelliSense)，才使用 IDE 螢幕擷取畫面。單只為了秀出顏色標示和反白顯示，請勿使用螢幕擷取畫面。

程式碼驗證

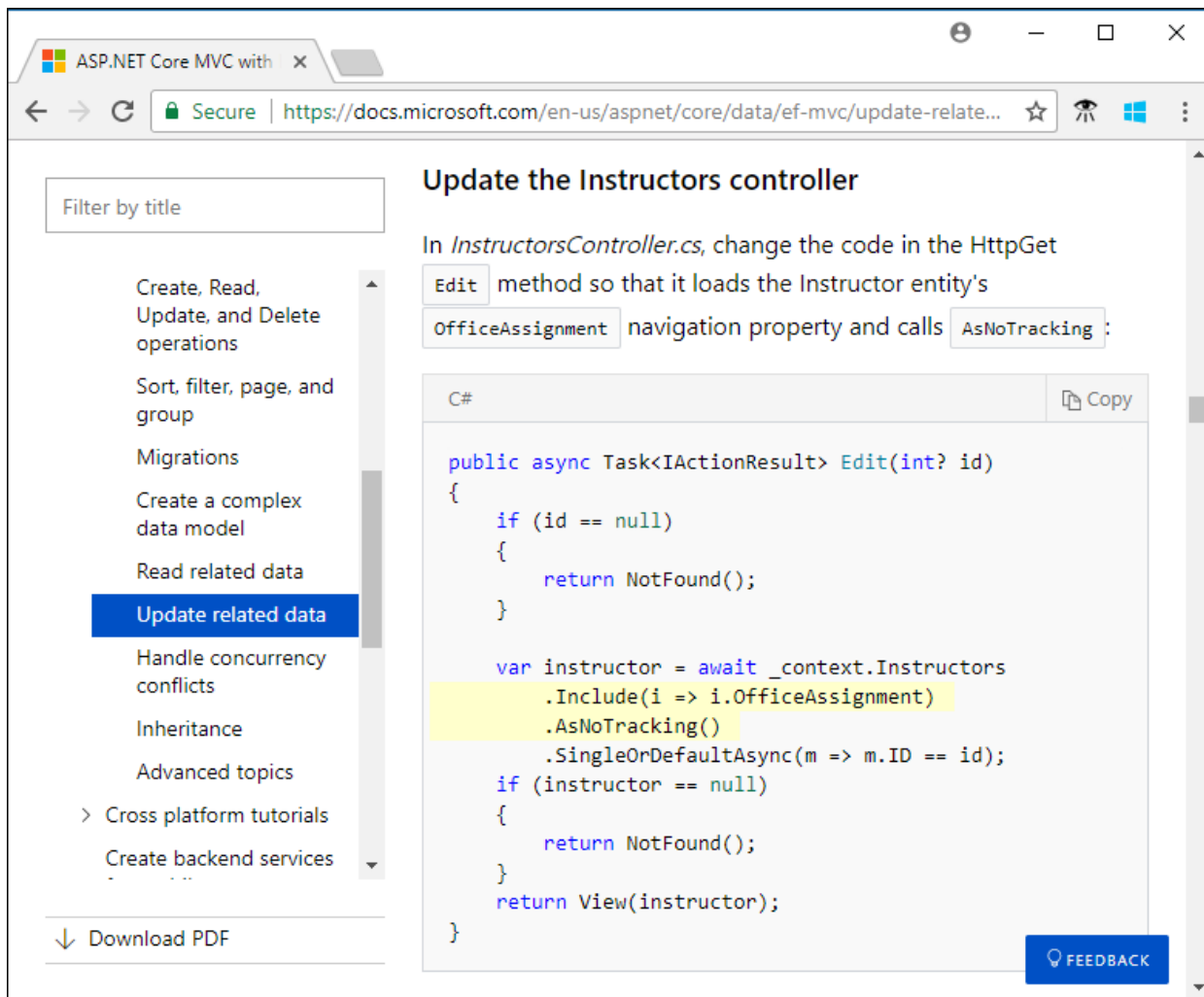
某些存放庫會實作能自動編譯所有範例程式碼以檢查錯誤的處理程序。.NET 存放庫就做得好。如需詳細資訊，請參閱 .NET 存放庫中的 [參與貢獻](#) (英文)。

如果您要包含另一個存放庫中的程式碼區塊，對於這些程式碼的維護策略請與其擁有者合作，讓您包含的程式碼

不會因為程式碼所使用的程式庫有了新版本而中斷或過期。

反白顯示

程式碼片段通常會包含比必要程式碼還要多的程式碼，以提供背景內容。在此情況下，若能將程式碼片段中的關鍵程式碼反白顯示，便能協助提升可讀性，如此範例所示：



當您將程式碼包含在文章 Markdown 檔案中時，將無法反白顯示它。此功能僅適用於參考程式碼檔案的程式碼片段。

水平捲軸

請將較長的行分段，以避免水平捲軸。程式碼片段上的捲軸會使程式碼較難閱讀。水平捲軸對較長程式碼區塊的影響更為明顯，因為讀者可能會無法同時看見捲軸和想要閱讀的程式碼行。

想讓程式碼區塊的水平捲軸盡可能最小，其中一個好的做法是將超過 85 個字元的程式碼行分段。但請記住，捲軸的存在與否不是可讀性的唯一準則。如果在 85 個字元前中斷一行會影響可讀性或複製貼上的便利性，超過 85 個也無妨。

內嵌程式碼區塊

只有當藉由參考程式碼檔案來顯示程式碼的做法不可行時，才使用內嵌程式碼區塊。相較於完整專案中的程式碼檔案，內嵌程式碼通常比較不容易測試和保持最新狀態。且內嵌程式碼可能會省略可協助開發人員瞭解及使用程式碼的內容。這些考量主要適用於程式設計語言。內嵌程式碼區塊也可以用於輸出和輸入 (例如 JSON)、查詢語言 (例如 SQL)、指令碼語言 (例如 PowerShell)。

有兩種方式可以將文章檔案中的某個文字區塊指定為程式碼區塊：將它 *隔離* 在三個倒引號 (```) 之內，或是對它進行縮排。建議使用隔離的方式，因為它能讓您指定語言。請避免使用縮排，因為太容易出錯，而且當其他作者需要編輯您的文章時，可能很難瞭解您的意圖。

語言指標是放置在開頭的三個倒引號之後，如下列範例所示：

Markdown：

```
```json
{
 "aggregator": {
 "batchSize": 1000,
 "flushTimeout": "00:00:30"
 }
}
```
```

已轉譯：

```
{
  "aggregator": {
    "batchSize": 1000,
    "flushTimeout": "00:00:30"
  }
}
```

對於可作為語言指標使用的值，如需相關資訊，請參閱[語言名稱和別名](#) (英文)。

如果您放在三個倒引號 (```) 後的語言或環境字組不被支援，該字組會出現在轉譯頁面的程式碼區塊標題列中。盡可能在內嵌程式碼區塊中使用語言或環境指標。

NOTE

如果您從 Word 檔案複製並貼上程式碼，請確定它在程式碼中不是有效的「大括弧」。如果有，請將它們變更成一般引號 (' 和 ")。或者，依賴 Docs 編寫套件的[智慧引號取代功能](#)。

存放庫內的程式碼片段參考

想要在文件中包含程式設計語言的程式碼片段，建議透過參考程式碼檔案的方式。這個方法讓您能夠反白顯示一至數行的程式碼，並讓 GitHub 上提供的程式碼片段範圍更廣，以利開發人員使用。您可以手動方式使用三個冒號格式 (:::)，或在 Visual Studio Code 中借助 docs.microsoft.com 編寫套件，來包含程式碼。

1. 在 Visual Studio Code 中，按一下 Alt + M 或 選項 + M，然後選取 [程式碼片段]。
2. 選取 [程式碼片段] 後，系統會提示您選取 [全文檢索搜尋]、[Scoped Search] (限定範圍搜尋) 或 [Cross-Repository Reference] (跨存放庫參考)。若要在本機搜尋，選取 [全文檢索搜尋]。
3. 輸入搜尋字詞以尋找檔案。找到檔案之後，請選取檔案。
4. 接下來，選取一個選項以決定應在程式碼片段中包含哪些程式碼。這些選項包括：[識別碼]、[範圍] 和 [無]。
5. 根據您在步驟 4 中的選擇，視需要提供值。

顯示完整程式碼檔案：

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs":::
```

透過指定行號來顯示部分程式碼檔案：

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" range="2-24,26":::
```

透過指定程式碼片段名稱來顯示部分程式碼檔案：


```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" id="snippet_Create":::
```

下列各節將說明這些範例：

- [使用程式碼檔案的相對路徑](#)
- [僅包含選取的行號](#)
- [參考具名的程式碼片段](#)
- [反白顯示選取的行](#)

如需詳細資訊，請參閱本文稍後的[程式碼片段語法參考](#)。

程式碼檔案的路徑

範例：

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" range="2-24,26":::
```

此範例是來自 ASP.NET 文件存放庫中的 [aspnetcore/data/ef-mvc/crud.md](#) (英文) 文章檔案。程式碼檔案的參考方式，是透過針對相同存放庫中 [aspnetcore/data/ef-mvc/intro/samples/cu/Controllers/StudentsController.cs](#) 的相對路徑。

選取的行號

範例：

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" range="2-24,26":::
```

這個範例只顯示 *StudentController.cs* 程式碼檔案中的第 2-24 行和第 26 行。

相較於硬式編碼的行號，請盡量使用具名程式碼片段，原因會於下一節中說明。

具名的程式碼片段

範例：

```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" id="snippet_Create":::
```

名稱只能使用字母和底線。

此範例會顯示程式碼檔案的 `snippet_Create` 區段。這個範例的程式碼檔案會在 C# 的程式碼註解中放上程式碼片段標記：

```
// code excluded from the snippet
// <snippet_Create>
// code included in the snippet
// </snippet_Create>
// code excluded from the snippet
```

請盡可能參考具名的區段，而非指定行號。行號參考較容易出錯，因為程式碼檔案終究會變更，進而使行號產生變更。而您不一定會收到這些變更的通知。您的文章最後會開始顯示錯誤的程式碼行，且您將不會意識到這項變更。

反白顯示選取的行

範例：


```
:::code language="csharp" source="intro/samples/cu/Controllers/StudentsController.cs" range="2-24,26"
highlight="2,5":::
```

此範例會反白顯示行 2 和 5，這是從顯示的程式碼片段開頭開始計算。要反白顯示的行號並不是從程式碼檔案的開頭開始計算。換句話說，系統會醒目提示程式碼檔案的第 3 行和第 6 行。

存放庫外的程式碼片段參考

若您想要參考的程式碼檔案位於另一個存放庫，請將該程式碼存放庫設定為「相依存放庫」。這麼做時，便需要為它指定名稱。接著，基於程式碼參考的目的，該名稱會被作為資料夾名稱使用。

例如，文件存放庫為 *Azure/azure-docs*，且程式碼存放庫為 *Azure/azure-functions-durable-extension*。

在 *azure-docs* 的根資料夾中，新增 *.openpublishing.publish.config.json* 中的下列區段：

```
{
  "path_to_root": "samples-durable-functions",
  "url": "https://github.com/Azure/azure-functions-durable-extension",
  "branch": "master",
  "branch_mapping": {}
},
```

現在，當您以 *azure-docs* 中資料夾的形式參考 *sample-durable-functions* 時，實際上是在參考 *azure-functions-durable-extension* 存放庫中的根資料夾。

您可以手動方式使用三個冒號格式 (:::)，或在 Visual Studio Code 中借助 docs.microsoft.com 編寫套件，來包含程式碼。

1. 在 Visual Studio Code 中，按一下 **Alt + M** 或 **選項 + M**，然後選取 [程式碼片段]。
2. 選取 [程式碼片段] 後，系統會提示您選取 [全文檢索搜尋]、[Scoped Search] (限定範圍搜尋) 或 [Cross-Repository Reference] (跨存放庫參考)。若要搜尋不同的存放庫，請選取 [Cross-Repository Reference] (跨存放庫參考)。
3. *.openpublishing.publish.config.json* 會提供您存放庫選項。選取存放庫。
4. 輸入搜尋字詞以尋找檔案。找到檔案之後，請選取檔案。
5. 接下來，選取一個選項以決定應在程式碼片段中包含哪些程式碼。這些選項包括：[識別碼]、[範圍] 和 [無]。
6. 根據您在步驟 5 中的選擇提供值。

您的程式碼片段參考看起來如下：

```
:::code language="csharp" source="~/samples-durable-functions/samples/csx/shared/Location.csx"
highlight="2,5":::
```

在 *azure-functions-durable-extension* 存放庫中，該程式碼檔案是位於 *samples/csx/shared* 資料夾中。如[先前所述](#)，反白顯示行的行號是相對於程式碼片段開頭，而不是檔案的開頭。

NOTE

您指派給相依存放庫的名稱是相對於主要存放庫的根目錄，但波狀符號 (~) 則是指文件集的根目錄。文件集根目錄是由 *.openpublishing.publish.config.json* 中的 `build_source_folder` 決定。在上述範例中，程式碼片段的路徑適用於 *azure-docs* 檔存放庫，因為 `build_source_folder` 是指存放庫根目錄 (.)。如果 `build_source_folder` 是 `articles`，路徑的開頭會是 `~/../samples-durable-functions`，而不是 `~/samples-durable-functions`。

Jupyter 筆記本中的程式碼片段

您可以參考 Jupyter 筆記本中的資料格作為程式碼片段。為了參考資料格：

1. 針對您想要參考的資料格，將資料格中繼資料加入至筆記本。
2. 設定存放庫的存取權。
3. 在 markdown 檔案中使用 Jupyter 筆記本程式碼片段語法。

將中繼資料新增至筆記本

1. 藉由在 Jupyter 筆記本中新增資料格中繼資料來命名儲存格。
 - 在 Jupyter 中，您可以先啟用資料格工具列：**View > 資料格工具列 > 編輯元 資料**，以 [編輯資料格中繼資料](#)。
 - 啟用儲存格工具列之後，請選取您想要命名的資料格上的 [編輯中繼資料]。
 - 或者，您也可以直接在筆記本的 JSON 結構中編輯中繼資料。
2. 在資料格中繼資料中，加入 "name" 屬性：

```
"metadata": {"name": "<name>"},
```

例如：

```
"metadata": {"name": "workspace"},
```

TIP

您可以新增任何其他您想要的中繼資料，以協助您追蹤資料格的使用位置。例如：

```
"metadata": {  
  "name": "workspace",  
  "msdoc": "how-to-track-experiments.md"  
},
```

設定存放庫存取

如果您想要參考的筆記本檔案位於不同的存放庫，請將程式碼存放庫設定為 [相依存放庫](#)。

Jupyter 筆記本程式碼片段語法參考

一旦您的筆記本包含必要的中繼資料，請在您的 markdown 檔案中參考它。使用 `<cell-name-value>` 您新增至筆記本的，並將 `<path>` 設定為相依存放庫。

```
[!notebook-<language>[] (<path>/<notebook-name.ipynb?name=<cell-name-value>)]
```

例如：

```
[!notebook-python[] (~/MachineLearningNotebooks/train-on-local.ipynb?name=workspace)]
```

IMPORTANT

此語法是區塊 Markdown 延伸。必須用於本身的行。

使用任何 [支援](#) 的 `<language>` 識別碼語言。

互動式程式碼片段

內嵌互動式程式碼區塊

針對下列語言，可以使程式碼片段能在瀏覽器視窗中執行：

- Azure Cloud Shell
- Azure PowerShell Cloud Shell
- C# REPL

在啟用互動模式的情況下，所呈現的程式碼方塊會顯示 [試試看] 或 [執行] 按鈕。例如：

```
```azurepowershell-interactive
New-AzResourceGroup -Name myResourceGroup -Location westeurope
```
```

會轉譯為：

```
New-AzResourceGroup -Name myResourceGroup -Location westeurope
```

若要針對特定程式碼區塊開啟此功能，請使用特殊的語言識別項。可用的選項包括：

- `azurepowershell-interactive` - 啟用 Azure PowerShell Cloud Shell，如上述範例所示
- `azurecli-interactive` - 啟用 Azure Cloud Shell
- `csharp-interactive` - 啟用 C# REPL

針對 Azure Cloud Shell 和 PowerShell Cloud Shell，使用者可以僅針對其 Azure 帳戶執行命令。

以參考方式包含程式碼片段

您可以針對依參考所包含的程式碼片段啟用互動模式。以下為範例：

```
:::code source="PowerShell.ps1" interactive="cloudshell-powershell":::
```

```
:::code source="Bash.sh" interactive="cloudshell-bash":::
```

若要針對特定程式碼區塊開啟此功能，請使用 `interactive` 屬性。可用的屬性值如下：

- `cloudshell-powershell` - 啟用 Azure PowerShell Cloud Shell，如上述範例所示
- `cloudshell-bash` - 啟用 Azure Cloud Shell
- `try-dotnet` - 啟用 .NET Interactive
- `try-dotnet-class` - 啟用具有類別 Scaffolding 的 .NET Interactive
- `try-dotnet-method` - 啟用具有方法 Scaffolding 的 .NET Interactive

針對 Azure Cloud Shell 和 PowerShell Cloud Shell，使用者可以僅針對其 Azure 帳戶執行命令。

針對 .NET Interactive 體驗，您程式碼區塊的內容取決於您選擇三個 Scaffolding 體驗的其中一個：

- **無 scaffolding** (`try-dotnet`)：程式碼區塊應該代表完整程式文字。例如，由 `dotnet new console` 產生的 `Program.cs` 檔案將會有效。這些最適合用來顯示整個小程式，包括所需的 `using` 指示詞。目前不支援頂層陳述式。
- **方法 Scaffolding** (`try-dotnet-method`)：程式碼區塊應該代表主控台應用程式中 `Main` 方法的內容。您可以假設 `using` 指示詞由 `dotnet new console` 範本新增。此設定最適合用於示範單一功能的簡短程式碼片段。
- **類別 Scaffolding** (`try-dotnet-class`)：程式碼區塊應該代表具有作為程式進入點之 `Main` 方法的類別。這些可用來顯示類別成員的互動方式。

程式碼片段語法參考

語法：

```
:::code language="<language>" source="<path>" <attribute>="<attribute-value>":::
```

IMPORTANT

此語法是區塊 Markdown 延伸。必須用於本身的行。

- `<language>` (選擇性)
 - 程式碼片段的語言。如需詳細資訊，請參閱本文稍後的<支援的語言>一節。
- `<path>` (強制)
 - 檔案系統中的相對路徑，表示要參考的程式碼片段檔案。
- `<attribute>` 和 `<attribute-value>` (選擇性)

同時使用以指定從檔案擷取程式碼的方式以及如何顯示程式碼：

- `range`：1,3-5 行的範圍。此範例包含第 1、3、4 及 5 行。
- `id`：snippet_Create 需要從程式碼檔案插入的程式碼片段識別碼。這個值與範圍不能並存。
- `highlight`：2-4,6 需要在產生的程式碼片段中醒目提示的範圍及/或行數。編號是相對於顯示的行(如範圍或識別碼所指定)，而不是檔案。
- `interactive`：cloudshell-powershell、cloudshell-bash、try-dotnet、try-dotnet-class、try-dotnet-method 字串值決定啟用的互動功能類型。
- 如需程式碼片段原始程式檔中依語言分類的標籤名稱表示法詳細資料，請參閱 [DocFX 指南](#)。

支援的語言

[Docs 編寫套件](#) 有一項功能，可提供陳述式完成和驗證程式碼隔離區塊的可用語言識別碼。

隔離的程式碼區塊

| 語言 | 識別碼 |
|---------------|------------------|
| .NET Core CLI | dotnetcli |
| 1C | 1c |
| ABNF | abnf |
| 存取記錄 | accesslog |
| Ada | ada |
| XML 組譯工具 | armasm, arm |
| AVR 組譯工具 | avrasm |
| ActionScript | actionscript, as |

| 「 | 「「「 |
|------------------------|-----------------------------|
| Alan | alan , i |
| AngelScript | angelscript , asc |
| ANTLR | antlr |
| Apache | apache , apacheconf |
| AppleScript | applescript , osascript |
| 大型電玩 | arcade |
| AsciiDoc | asciidoc , adoc |
| AspectJ | aspectj |
| ASPX | aspx |
| ASP.NET (C#) | aspx-csharp |
| ASP.NET (VB) | aspx-vb |
| AutoHotkey | autohotkey |
| Autolt | autoit |
| Awk | awk , mawk , nawk , gawk |
| Axapta | axapta |
| AzCopy | azcopy |
| Azure CLI | azurecli |
| Azure CLI (互動式) | azurecli-interactive |
| Azure Powershell | azurepowershell |
| Azure Powershell (互動式) | azurepowershell-interactive |
| Bash | bash , sh , zsh |
| 基本 | basic |
| BNF | bnf |
| C | c |

| 「 | 」 |
|---------------------|--|
| C# | csharp, cs |
| C# (互動式) | csharp-interactive |
| C++ | cpp, c, cc, h, c++, h++, hpp |
| C++/CX | cppcx |
| C++/WinRT | cppwinrt |
| C/AL | cal |
| Cache Object Script | cos, cls |
| CMake | cmake, cmake.in |
| Coq | coq |
| CSP | csp |
| CSS | css |
| Cap'n Proto | capnproto, capnp |
| Clojure | clojure, clj |
| CoffeeScript | coffeescript, coffee, cson, iced |
| Crmsb | crmsb, crm, pcmk |
| Crystal | crystal, cr |
| Cypher (Neo4j) | cypher |
| D | d |
| DAX Power BI | dax |
| DNS Zone 檔案 | dns, zone, bind |
| DOS | dos, bat, cmd |
| Dart | dart |
| Delphi | delphi, dpr, dfm, pas, pascal, freepascal, lazarus, lpr, lfm |
| Diff | diff, patch |

| 〃 | 〃〃〃 |
|--------------|--|
| Django | <code>django</code> , <code>jinja</code> |
| Dockerfile | <code>dockerfile</code> , <code>docker</code> |
| dsconfig | <code>dsconfig</code> |
| DTS (裝置樹狀目錄) | <code>mts</code> |
| Dust | <code>dust</code> , <code>dst</code> |
| Dylan | <code>dylan</code> |
| EBNF | <code>ebnf</code> |
| Elixir | <code>elixir</code> |
| Elm | <code>elm</code> |
| Erlang | <code>erlang</code> , <code>erl</code> |
| Excel | <code>excel</code> , <code>xls</code> , <code>xlsx</code> |
| Extempore | <code>extempore</code> , <code>xtlang</code> , <code>xm</code> |
| F# | <code>fsharp</code> , <code>fs</code> |
| FIX | <code>fix</code> |
| Fortran | <code>fortran</code> , <code>f90</code> , <code>f95</code> |
| G-Code | <code>gcode</code> , <code>nc</code> |
| Gams | <code>gams</code> , <code>gms</code> |
| GAUSS | <code>gauss</code> , <code>gss</code> |
| GDScript | <code>godot</code> , <code>gdscrip</code> |
| Gherkin | <code>gherkin</code> |
| GN for Ninja | <code>gn</code> , <code>gni</code> |
| Go | <code>go</code> , <code>golang</code> |
| Golo | <code>golo</code> , <code>gololang</code> |
| Gradle | <code>gradle</code> |

| 「 | 「「「 |
|-----------------|---|
| Groovy | groovy |
| HTML | html , xhtml |
| HTTP | http , https |
| Haml | haml |
| 把手 | handlebars , hbs , html.hbs , html.handlebars |
| Haskell | haskell , hs |
| Haxe | haxe , hx |
| Hy | hy , hylang |
| Ini | ini |
| Inform7 | inform7 , i7 |
| IRPF90 | irpf90 |
| JSON | json |
| Java | java , jsp |
| JavaScript | javascript , js , jsx |
| Kotlin | kotlin , kt |
| Kusto | kusto |
| Leaf | leaf |
| Lasso | lasso , ls , lassoscript |
| 小於 | less |
| LDIF | ldif |
| Lisp | lisp |
| LiveCode Server | livecodeserver |
| LiveScript | livescript , ls |
| Lua | lua |

| 「 | 「「「「 |
|-------------------------|--|
| Makefile | <code>makefile</code> , <code>mk</code> , <code>mak</code> |
| Markdown | <code>markdown</code> , <code>md</code> , <code>mkdown</code> , <code>mkd</code> |
| Mathematica | <code>mathematica</code> , <code>mma</code> , <code>wl</code> |
| Matlab | <code>matlab</code> |
| Maxima | <code>maxima</code> |
| Maya 內嵌語言 | <code>mel</code> |
| Mercury | <code>mercury</code> |
| mIRC 指令碼語言 | <code>mirc</code> , <code>mrc</code> |
| Mizar | <code>mizar</code> |
| 受控物件格式 | <code>mof</code> |
| Mojolicious | <code>mojolicious</code> |
| Monkey | <code>monkey</code> |
| Moonscript | <code>moonscript</code> , <code>moon</code> |
| MS Graph (互動式) | <code>msgraph-interactive</code> |
| N1QL | <code>n1ql</code> |
| NSIS | <code>nsis</code> |
| Nginx | <code>nginx</code> , <code>nginxconf</code> |
| Nimrod | <code>nimrod</code> , <code>nim</code> |
| Nix | <code>nix</code> |
| OCaml | <code>ocaml</code> , <code>ml</code> |
| Objective C | <code>objectivec</code> , <code>mm</code> , <code>objc</code> , <code>obj-c</code> |
| OpenGL Shading Language | <code>glsl</code> |
| OpenSCAD | <code>openscad</code> , <code>scad</code> |
| Oracle 規則語言 | <code>ruleslanguage</code> |

| 「 | 「「「 |
|-----------------------|--|
| Oxygene | <code>oxygene</code> |
| PF | <code>pf</code> , <code>pf.conf</code> |
| PHP | <code>php</code> , <code>php3</code> , <code>php4</code> , <code>php5</code> , <code>php6</code> |
| Parser3 | <code>parser3</code> |
| Perl | <code>perl</code> , <code>p1</code> , <code>pm</code> |
| 純文字無反白 | <code>plaintext</code> |
| Pony | <code>pony</code> |
| PostgreSQL & PL/pgSQL | <code>pgsql</code> , <code>postgres</code> , <code>postgresql</code> |
| PowerShell | <code>powershell</code> , <code>ps</code> |
| PowerShell (互動式) | <code>powershell-interactive</code> |
| Processing | <code>processing</code> |
| Prolog | <code>prolog</code> |
| 屬性 | <code>properties</code> |
| 通訊協定緩衝區 | <code>protobuf</code> |
| Puppet | <code>puppet</code> , <code>pp</code> |
| Python | <code>python</code> , <code>py</code> , <code>gyp</code> |
| Python 分析工具結果 | <code>profile</code> |
| Q# | <code>qsharp</code> |
| Q | <code>k</code> , <code>kdb</code> |
| QML | <code>qml</code> |
| R | <code>r</code> |
| Razor CSHTML | <code>cshtml</code> , <code>razor</code> , <code>razor-cshtml</code> |
| ReasonML | <code>reasonml</code> , <code>re</code> |
| RenderMan RIB | <code>rib</code> |

| 「 | ''' |
|-------------------|---|
| RenderMan RSL | rs1 |
| Roboconf | graph , instances |
| Robot Framework | robot , rf |
| RPM 規格檔案 | rpm-specfile , rpm , spec , rpm-spec , specfile |
| Ruby | ruby , rb , gemspec , podspec , thor , irb |
| Rust | rust , rs |
| SAS | SAS , sas |
| SCSS | scss |
| SQL | sql |
| STEP Part 21 | p21 , step , stp |
| Scala | scala |
| 配置 | scheme |
| Scilab | scilab , sci |
| Shape Expressions | shexc |
| 殼層 | shell , console |
| Smali | smali |
| Smalltalk | smalltalk , st |
| Solidity | solidity , sol |
| Stan | stan |
| Stata | stata |
| 結構化文字 | iecst , scl , stl , structured-text |
| 手寫筆 | stylus , styl |
| SubUnit | subunit |
| Supercollider | supercollider , sc |

| 語言 | 檔案副檔名 |
|-------------------------------|--|
| Swift | <code>swift</code> |
| Tcl | <code>tcl</code> , <code>tk</code> |
| Terraform (HCL) | <code>terraform</code> , <code>tf</code> , <code>hcl</code> |
| 測試任何通訊協定 | <code>tap</code> |
| TeX | <code>tex</code> |
| Thrift | <code>thrift</code> |
| TOML | <code>toml</code> |
| TP | <code>tp</code> |
| Twig | <code>twig</code> , <code>craftcms</code> |
| TypeScript | <code>typescript</code> , <code>ts</code> |
| VB.NET | <code>vbnet</code> , <code>vb</code> |
| VBScript | <code>vbscript</code> , <code>vbs</code> |
| VHDL | <code>vhdl</code> |
| Vala | <code>vala</code> |
| Verilog | <code>verilog</code> , <code>v</code> |
| Vim Script | <code>vim</code> |
| Visual Basic | <code>vb</code> |
| Visual Basic for Applications | <code>vba</code> |
| X++ | <code>xpp</code> |
| x86 Assembly | <code>x86asm</code> |
| XL | <code>xl</code> , <code>tao</code> |
| XQuery | <code>xquery</code> , <code>xpath</code> , <code>xq</code> |
| XAML | <code>xaml</code> |
| XML | <code>xml</code> , <code>xhtml</code> , <code>rss</code> , <code>atom</code> , <code>xjb</code> , <code>xsd</code> , <code>xsl</code> , <code>plist</code> |

| “ | ” |
|--------|--|
| YAML | <code>yml</code> , <code>yaml</code> |
| Zephir | <code>zephir</code> , <code>zep</code> |

TIP

Docs 編寫套件[開發語言完成功能](#)在有多個別名可用時, 會使用第一個有效的別名。

後續步驟

如需程式碼以外內容類型的文字格式設定資訊, 請參閱[文字格式設定指導方針](#)。

文字格式設定指導方針

2021/7/8 •

對文字項目一致且適當地使用粗體、斜體和程式碼樣式可提升可讀性並避免誤解。

粗體

使用粗體來表示使用者介面元素，例如功能表選取項目、對話方塊名稱、輸入欄位名稱。

範例

- **正確**：在 **方案總管** 中，以滑鼠右鍵按一下專案節點，然後選取 **[新增] > [新項目]**。
- **不正確**：在方案總管中，以滑鼠右鍵按一下專案節點，然後選取 [新增] > [新項目]。
- **不正確**：在 **方案總管** 中，以滑鼠右鍵按一下專案節點，然後選取 **[新增] > [新項目]**。

斜體

使用斜體來表示：

- 介紹新的字詞，包含其定義或說明。
- 檔案名稱、資料夾名稱、路徑。
- 使用者輸入。

範例

- **正確**：在 App Service 中，應用程式會在 *App Service 方案* 中執行。App Service 方案定義一組計算資源供 Web 應用程式執行。
- **不正確**：在 App Service 中，應用程式會在 "App Service 方案" 中執行。App Service 方案定義一組計算資源供 Web 應用程式執行。
- **正確**：將 *HttpTriggerCSharp.cs* 中的程式碼取代為下列程式碼。
- **不正確**：將 `HttpTriggerCSharp.cs` 中的程式碼取代為下列程式碼。
- **正確**：名稱 請輸入 *ContosoUniversity*，然後選取 **新增**。
- **不正確**：名稱 請輸入 "ContosoUniversity"，然後選取 **新增**。

程式碼樣式

使用程式碼樣式來表示：

- 程式碼項目，例如方法名稱、屬性名稱和語言關鍵字。
- SQL 命令
- NuGet 套件名稱
- 命令列的命令*
- 資料庫資料表和資料行名稱
- 不應當地語系化的資源名稱，例如虛擬機器名稱
- 不想讓人按的 URL

為什麼？較舊的樣式指南會為許多這類文字元素指定粗體。然而，大部分的文章都已當地語系化，而程式碼樣式能告訴譯者保留該部分的文字不要翻譯。

可將程式碼樣式「內嵌」（以 ` 括住），或將跨越數行的程式碼區塊「隔離」（以 ``` 括住）。請將較長的程式碼片段和路徑放在括住的程式碼區塊中。

* 在命令列命令中，如果所有平臺都支援，請在檔案路徑中使用正斜線。當僅支援反斜線時，請使用反斜線來說明在 Windows 上執行的命令。例如，正斜線可在所有平臺上的 .NET CLI 上運作，因此您可以使用

`dotnet build foldername/filename.csproj` 而不是 `dotnet build foldername\filename.csproj`。

內嵌樣式的範例

- **正確**：根據預設，Entity Framework 會將名為 `Id` 或 `ClassnameID` 的屬性解譯為主索引鍵。
- **不正確**：根據預設，Entity Framework 會將名為 `Id` 或 `ClassnameID` 的屬性解譯為主索引鍵。
- **正確**：`Microsoft.EntityFrameworkCore` 套件提供 EF Core 的執行階段支援。
- **不正確**：`Microsoft.EntityFrameworkCore` 套件提供 EF Core 的執行階段支援。

括住的程式碼區塊範例

- **正確**：只變更 `IQueryable` 的陳述式不會將任何命令傳送至資料庫，如下列程式碼：

```
```csharp
var students = context.Students.Where(s => s.LastName == "Davolio")
```
```

- **不正確**：只變更 `IQueryable` 的陳述式不會將任何命令傳送至資料庫，例如 `var students = context.Students.Where(s => s.LastName == "Davolio")`。

- **正確**：例如，若要在 `C:\Scripts` 目錄中執行 `Get-ServiceLog.ps1` 指令碼，請輸入：

```
```powershell
C:\Scripts\Get-ServiceLog.ps1
```
```

- **不正確**：例如，若要在 `C:\Scripts` 目錄中執行 `Get-ServiceLog.ps1` 指令碼，請輸入：`"C:\Scripts\Get-ServiceLog.ps1"`。
- **所有** 隔離的程式碼區塊都必須有已核准的語言標記。如需支援的語言標記清單，請參閱[如何在 Docs 中包含程式碼](#)。

標題和連結文字

請勿將斜體或粗體等內嵌樣式套用至標題或超連結文字。

為什麼？

大眾會根據標準超連結文字確認文字項目為可按的連結。例如，將連結的樣式設定為斜體，可能會混淆文字為連結的事實。標題有其專屬樣式，混用其他樣式並不美觀。

範例

- **正確**：

```
The *function.json* file is generated by the NuGet package
[Microsoft.NET.Sdk.Functions](http://www.nuget.org/packages/Microsoft.NET.Sdk.Functions).
```

- **不正確**：

```
The *function.json* file is generated by the NuGet package
[*Microsoft.NET.Sdk.Functions*](http://www.nuget.org/packages/Microsoft.NET.Sdk.Functions).
```

- **正確**：

```
### The Microsoft.NET.Sdk.Functions package
```

- 不正確：

```
### The *Microsoft.NET.Sdk.Functions* package
```

鍵盤和鍵盤快速鍵

用到按鍵或按鍵組合時，請遵循下列慣例：

- 將按鍵名稱的第一個字母大寫。
- 不要套用斜體、粗體或程式碼等內嵌樣式。
- 使用 "+" 來聯結同時選取的按鍵。

範例

- 正確：選取 Alt + Ctrl + S。
- 不正確：選取 ALT+CTRL+S。
- 不正確：點擊 ALT+CTRL+S。

如需詳細資訊，請參閱[描述與 UI 的互動](#)。

例外狀況

樣式指導方針並不是僵化的規則。在會危害可讀性的情況下，可採行不同的做法。例如，如果 HTML 資料表中多為程式碼項目，滿滿的程式碼樣式看起來會很擁擠。您可以在該內容中選擇粗體樣式。

如果您在通常會呼叫程式碼之處選擇使用替代的文字樣式，請確定該文字可以在當地語系化版本的文章中翻譯。畢竟，程式碼是唯一可以自動防止翻譯的樣式。如果您想要在不使用程式碼樣式的情況下防止當地語系化，請參閱 [不可當地語系化的字串](#)。

在文件中使用連結

2021/5/13 •

本文描述如何在 docs.microsoft.com 上裝載的頁面中使用超連結。使用一些不同的慣例，可以輕易地將連結新增至 Markdown。連結可將使用者指向相同頁面、其他相鄰頁面，或外部網站與 URL 中的內容。

docs.microsoft.com 網站後端使用「開放式發行服務」(OPS)，可支援透過 [Markdig](#) 剖析引擎且符合 [CommonMark](#) 規範的 Markdown。這個 Markdown 變體大多與 [GitHub 變體 Markdown \(GFM\)](#) 相容，因為大多數文件都儲存在 GitHub 中並可在該處編輯。額外的功能可透過 Markdown 延伸模組新增。

IMPORTANT

只要目標有支援 (絕大部分都會支援) 安全協定，所有連結都必須使用安全協定 ([https](#) 相對於 [http](#))。

連結文字

您包含在連結文字中的字詞應該淺顯易懂。換句話說，它們應該是簡單的英文單字，或您要連結之網頁的標題。

IMPORTANT

請勿使用「按一下這裡」。這對搜尋引擎最佳化而言並不是一個好的選擇，且沒有適當地描述目標。

正確：

- For more information, see the [contributor guide index](<https://github.com/Azure/azure-content/blob/master/contributor-guide/contributor-guide-index.md>).
- For more details, see the [SET TRANSACTION ISOLATION LEVEL](<https://docs.microsoft.com/sql/t-sql/statements/set-transaction-isolation-level-transact-sql>) reference.

不正確：

- For more details, see <https://msdn.microsoft.com/library/ms173763.aspx>.
- For more information, click [here](<https://github.com/Azure/azure-content/blob/master/contributor-guide/contributor-guide-index.md>).

文章之間的連結

發佈系統支援兩種超連結類型：URL 和檔案連結。

URL 連結可以是相對於 docs.microsoft.com 根目錄的 URL 路徑，或包含完整 URL 語法的絕對 URL (例如 <https://github.com/MicrosoftDocs/PowerShell-Docs>)。

- 連結到目前 *docset* 的外部內容，或在 docset 內自動產生的參照與概念性文章之間連結時，請使用 URL 連結。
- 建立相對連結的最簡單方式，就是從瀏覽器複製 URL，然後將 <https://docs.microsoft.com/en-us> 從貼到 Markdown 的值中移除。
 - 請勿在 Microsoft 屬性的 URL 中包含地區設定 (例如，請移除 URL 中的 "/en-us")。

檔案連結是用來將 docset 內的某篇文章連結到另一篇文章。

- 所有檔案路徑都會使用正斜線 ([/](#)) 字元，而不是反斜線字元。

- 將文章連結到相同目錄中的另一篇文章：

```
[link text](article-name.md)
```

- 將文章連結到當前目錄的父目錄中文章：

```
[link text](../article-name.md)
```

- 將文章連結到當前目錄的子目錄中文章：

```
[link text](directory/article-name.md)
```

- 將文章連結到當前目錄的父目錄中，其他子目錄中文章：

```
[link text](../directory/article-name.md)
```

NOTE

上述範例均未在連結中使用 `~/`。若要連結到從存放庫根開始的絕對路徑，請使用 `/` 來啟動連結。瀏覽 GitHub 上的原始碼存放庫時，置入 `~/` 會產生無效的連結。在路徑的開頭使用 `/` 即可正確解決。

docs.microsoft.com 上的連結結構

在 docs.microsoft.com 上發佈的內容具有下列 URL 結構：

```
https://docs.microsoft.com/<locale>/<product-service>/[<feature-service>]/[<subfolder>]/<topic>[?view=<view-name>]
```

範例：

```
https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview
https://docs.microsoft.com/en-us/powershell/azure/overview?view=azurermps-5.1.1
```

- `<locale>` - 識別文章的語言 (例如: en-us 或 de-de)
- `<product-service>` - 產品或服務的名稱 (例如: powershell、dotnet 或 azure)
- `[<feature-service>]` - (選用) 產品的功能或子服務名稱 (例如: csharp 或 load-balancer)
- `[<subfolder>]` - (選用) 功能內的子資料夾名稱
- `<topic>` - 主題的文章檔案名稱 (例如: load-balancer-overview 或 overview)
- `[?view=\<view-name>]` - (選用) 版本選取器針對具有多個可用版本的內容所使用檢視名稱 (例如: azps-3.5.0)

TIP

在大部分情況下，相同 *docset* 中的文章會有相同 `<product-service>` URL 片段。例如：

- 相同的 docset
 - `https://docs.microsoft.com/dotnet/core/get-started`
 - `https://docs.microsoft.com/dotnet/framework/install`
- 不同的 docset
 - `https://docs.microsoft.com/dotnet/core/get-started`
 - `https://docs.microsoft.com/visualstudio/whats-new`

書籤連結

若要讓書籤連結到「目前」檔案中的標題，請使用井字符號，後面接著小寫標題文字。移除標題中的標點符號，並

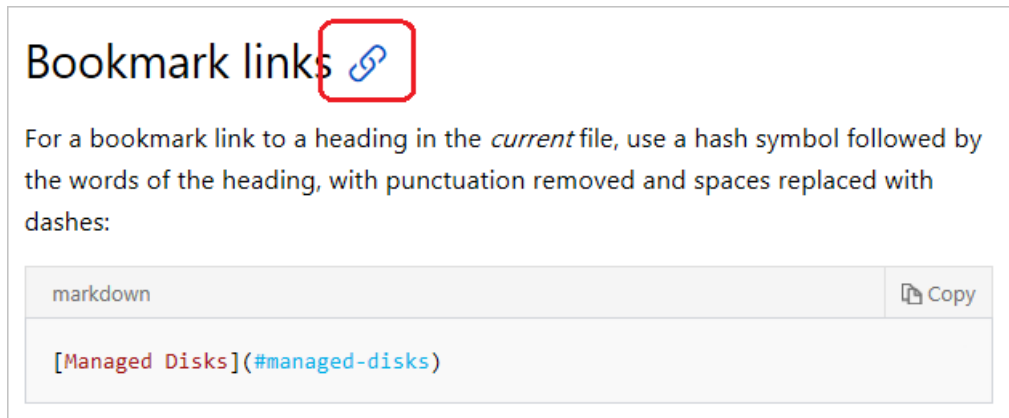
以破折號取代空格：

```
[Managed Disks](#managed-disks)
```

若要連結到另一篇文章中的書籤標題，請使用檔案相對或網站相對連結加上井字符號，後面接著標題文字。移除標題中的標點符號，並以破折號取代空格：

```
[Managed Disks](../../linux/overview.md#managed-disks)
```

您也可以從 URL 複製書籤連結。若要尋找 URL，請將滑鼠暫留在 docs.microsoft.com 的標題列上。您應該會看到出現一個連結圖示：



按一下連結圖示，然後從 URL 複製書籤錨點文字 (也就是井字符號後面的部分)。

NOTE

Docs 延伸模組也有協助建立連結的工具。

明確錨點連結

除非在中樞和登陸頁面中，否則沒有必要也不建議新增使用 `<a>` HTML 標籤的明確錨點連結。相反地，請使用自動產生的書籤，如書籤連結中所述。針對中樞和登陸頁面，請如下所示宣告錨點：

```
## <a id="anchortext" />Header text
```

或

```
## <a name="anchortext" />Header text
```

且下列會連結到錨點：

```
To go to a section on the same page:  
[text](#anchortext)
```

```
To go to a section on another page.  
[text](filename.md#anchortext)
```

NOTE

錨點文字必須一律為小寫，且不能包含空格。

XRef (交互參照) 連結

XRef 連結是連結到 API 的建議方式，因為這些連結會在建置時進行驗證。若要連結到目前 docset 或其他 docset 中自動產生的 API 參照頁面，請搭配類型或成員的唯一識別碼 (UID) 來使用 XRef 連結。

TIP

您可使用適用於 VS Code 的 Docs Markdown 延伸模組 (Docs Authoring Pack 的一部分)，以插入 .NET API 瀏覽器中呈現的 .NET XRef 連結。

在 .NET API 瀏覽器 或 Windows UWP 搜尋方塊中鍵入完整名稱或其中一部分，以檢查所要連結的 API 是否在 docs.microsoft.com 上。如果沒有看到任何顯示的結果，則該類型目前還不在 docs.microsoft.com 上。

您可以使用下列其中一個語法：

- 自動連結：

```
<xref:UID>  
<xref:UID?displayProperty=nameWithType>
```

根據預設，連結文字只會顯示成員或型別名稱。選用的 `displayProperty=nameWithType` 查詢參數會產生完整連結文字；亦即，`namespace.type` 表示類型，而 `type.member` 表示類型成員 (包括列舉類型成員)。

- Markdown 式連結：

```
[link text](xref:UID)
```

當想要自訂顯示的連結文字時，請針對 XRef 使用 Markdown 式連結。

範例：

- `<xref:System.String>` 會顯示為 [String](#)
- `<xref:System.String?displayProperty=nameWithType>` 會顯示為 [System.String](#)
- `[String 類別](xref:System.String)` 會顯示為 [String 類別](#)。

在類別中，`displayProperty=fullName` 查詢參數的運作方式與 `displayProperty=nameWithType` 相同。亦即，連結文字會變成 `namespace.classname`。不過，若是成員，則連結文字會顯示為 `namespace.classname.membername`，這可能不適當。

NOTE

UID 會區分大小寫。例如，`<xref:System.Object>` 會正確解析，但 `<xref:system.object>` 則不會。

XRef 建置警告和累加建置

累加建置只會建置已變更或受變更影響的檔案。如果看到有關 XREF 連結無效的建置警告，但此連結實際上有有效，這可能是因為建置是累加的。造成警告的檔案並未變更，因此不會建置，且會重新顯示過去的警告。當檔案變更時，或如果觸發完整建置 (可在 ops.microsoft.com 上啟動完整建置)，警告就會消失。這是累加建置的缺點，因為 DocFX 無法偵測到 XREF 服務內的資料更新。

判斷 UID

UID 通常是完整類別或成員名稱。至少有兩種方式可判斷 UID：

- 以滑鼠右鍵按一下類型或成員的 Docs 頁面，選取 [檢視原始檔]，然後複製 ms.assetid 的 content 值：

```

87 <meta name="page_type" content="dotnet" />
88 <meta name="page_kind" content="class" />
89 <meta name="description" content="Represents text as a sequence of UTF-16 code units. " />
90 <meta name="toc_rel" content="_splitted/System/toc.json" />
91 <meta name="source_url" content="" />
92 <meta name="ms.assetid" content="System.String" />
93 <meta name="pdf_url_template" content="https://docs.microsoft.com/pdfstore/en-us/V5.dotnet-api-docs/{branchName}/{pdfName}" />
94 <meta name="search.mshattr.devlang" content="csharp vb cpp" />

```

- 將部分或完整的類型名稱附加到 URL，以使用[自動完成網站](#)。例如，在瀏覽器的網址列中輸入 `https://xref.docs.microsoft.com/autocomplete?text=Writeline`，會顯示其名稱中包含 **Writeline** 的所有類型和方法，以及其 UID。

驗證 UID

若要測試是否有正確的 UID，請以 UID 取代下列 URL 中的 `System.String`，然後將其貼到瀏覽器的網址列中：

```
https://xref.docs.microsoft.com/query?uid=System.String
```

TIP

URL 中的 UID 會區分大小寫，且如果正在檢查方法多載 UID，請不要在參數類型之間加入空格。

如果看到類似如下的內容，表示具有正確的 UID：

```

[{"uid":"System.String","name":"String","fullName":"System.String","href":"https://docs.microsoft.com/dotnet/api/system.string","tags":",/dotnet,netframework-4.5.1,netframework-4.5.2,netframework-4.5,...xamarinmac-3.0,public,", "vendor":null,"hash":null,"commentId":"T:System.String","nameWithType":"System.String"}, {"uid":"System.String","name":"String","fullName":"System.String","href":"https://docs.microsoft.com/dotnet/api/system.string","tags":",/dotnet,netframework-4.5.1,netframework-4.5.2,netframework-4.5,netframework-4.6,netframework-4.6.1,...xamarinmac-3.0,public,", "vendor":null,"hash":null,"commentId":"T:System.String","nameWithType":"System.String"}]

```

如果只看到頁面上顯示 `[]`，則表示具有錯誤的 UID。

URL 的百分號編碼

UID 中的特殊字元必須以 HTML 編碼，如下所示：

| ⌞ | HTML ⌞ |
|----------------|------------------|
| <code>`</code> | <code>%60</code> |
| <code>#</code> | <code>%23</code> |
| <code>*</code> | <code>%2A</code> |

請參閱[百分號編碼](#)的完整清單。

編碼範例：

- `System.Threading.Tasks.Task`1` 會編碼為 `System.Threading.Tasks.Task%601` (請參閱[泛型型別](#)一節)
- `System.Exception.#ctor` 會編碼為 `System.Exception.%23ctor`
- `System.Object.Equals*` 會編碼為 `System.Object.Equals%2A`

泛型型別

泛型型別是指 `System.Collections.Generic.List<T>` 之類的類型。如果在 [.NET API 瀏覽器](#) 中瀏覽至此類型並查看其 URL，則會看到 `<T>` 在 URL 中寫成 `-1`，這實際上代表 ``1`：

```
https://docs.microsoft.com/dotnet/api/system.collections.generic.list-1
```

若要連結到泛型型別 (例如 `List<T>`), 請將 ` 倒單引號字元編碼為 `%60`, 如下列範例所示:

```
<xref:System.Collections.Generic.List%601>
```

方法

若要連結到方法, 可在方法名稱後面加上星號 (`*`) 來連結到一般方法, 或連結到特定多載。例如, 當想要連結到不含特定參數類型的 `<xref:System.Object.Equals%2A?displayProperty=nameWithType>` 方法時, 請使用一般頁面。星號字元會編碼為 `%2A`。例如:

```
<xref:System.Object.Equals%2A?displayProperty=nameWithType>
```

 會連結到 [Object.Equals](#)

若要連結到特定多載, 請在方法名稱後面加上括弧, 並包含每個參數的完整類型名稱。請勿在類型名稱之間加入空白字元, 否則連結將無法使用。例如:

```
<xref:System.Object.Equals(System.Object,System.Object)?displayProperty=nameWithType>
```

 會連結到 [Object.Equals\(Object, Object\)](#)

從包含檔案的連結

因為包含檔案位於另一個目錄中, 所以您必須使用較長的相對路徑。若要從包含檔案連結到文章, 請使用此格式:

```
[link text](../articles/folder/article-name.md)
```

TIP

適用於 Visual Studio Code 的 [Docs Authoring Pack](#) 延伸模組有助於正確插入相對連結和書籤, 而不需與冗長的路徑奮戰。

選取器中的連結

選取器是在文件文章中顯示為下拉式清單的導覽元件。當讀者在下拉式清單中選取一個值時, 瀏覽器會開啟選取的文章。一般而言, 選取器清單會包含密切相關的文章連結, 例如多種程式設計語言的相同主題, 或密切相關的系列文章。

如果您擁有內嵌在包含檔案中的選取器, 則可以使用下列連結結構:

```
> [AZURE.SELECTOR-LIST (Dropdown1 | Dropdown2 )]  
- [(Text1 | Example1 )](../articles/folder/article-name1.md)  
- [(Text1 | Example2 )](../articles/folder/article-name2.md)  
- [(Text2 | Example3 )](../articles/folder/article-name3.md)  
- [(Text2 | Example4 )](../articles/folder/article-name4.md)
```

參考風格連結

您可以使用參考風格連結, 讓來源內容更容易閱讀。參考風格連結會將內嵌連結語法取代為簡化語法, 允許您將較長的 URL 移動到文章的結尾。以下是 [Daring Fireball](#) 的範例:

內嵌文字:

```
I get 10 times more traffic from [Google][1] than from [Yahoo][2] or [MSN][3].
```

文章結尾處的連結參考：

```
<!--Reference links in article-->
[1]: http://google.com/
[2]: http://search.yahoo.com/
[3]: http://search.msn.com/
```

請確定您包含冒號之後的空格 (連結之前)。當您連結到其他技術文章時，如果忘記包含空格，已發佈的文章中的連結將會中斷。

連結到不屬於技術文件集的頁面

若要連結到 Microsoft 網站上的另一個頁面 (例如定價頁面、SLA 頁面或任何不屬於文件文章的頁面)，請使用絕對 URL，但省略地區設定。此目的為本連結可於 GitHub 和轉譯的網站上運作：

```
[link text](https://azure.microsoft.com/pricing/details/virtual-machines/)
```

連結到協力廠商網站

最佳使用者體驗會減少將使用者傳送到其他網站的情況。因此，以此資訊作為我們有時需要之任何協力廠商網站連結的基礎：

- **責任**：當所要共用的資訊是協力廠商資訊時，連結到協力廠商內容。例如，告知大家如何使用 Android 開發人員工具，並不是 Microsoft 的立場，那是 Google 的資訊。如果有需要，我們可以解釋如何**搭配** Azure 使用 Android 開發人員工具，但 Google 應該說明如何使用其工具。
- **PM 簽核**：要求 Microsoft 簽核協力廠商內容。透過與之連結，表示我們對其信任以及我們在人們遵循指示時的義務。
- **時效性檢閱**：確定協力廠商資訊仍是最新、正確且相關，且連結並未變更。
- **異地**：讓使用者了解他們將前往另一個網站。如果上下文不清楚，請加入一個限定詞組。例如：「必要條件包括 Android 開發人員工具，其可在 Android Studio 網站上下載」。
- **後續步驟**：舉例來說，可在〈後續步驟〉一節中新增一個 MVP 部落格連結。同樣地，請確認使用者了解他們將會離開網站。
- **法律**：在所有 microsoft.com 頁面的**使用條款**頁尾中，**第三方網站的連結**規定了相關法律事宜。

檔的中繼資料

2021/7/7 •

在 Microsoft, 我們會在檔上使用中繼資料來進行報告、透過搜尋探索內容, 以及推動網站體驗的各個層面。中繼資料可套用於 YAML front) 的文章 (, 或在存放庫的 *docfx.js* 檔案中全域套用。

如果您正在編輯現有的文章, 您可能不需要變更任何中繼資料。但是, 如果您要加入新的發行項, 則必須在 YAML front 中包含某些需要的中繼資料值。

必要的中繼資料

下表顯示必要的中繼資料索引鍵。如果您省略其中任何一項, 您可能會在組建期間收到驗證錯誤。

| II | I | III ? |
|--|---|---|
| <code>author</code> | 作者的 GitHub 帳戶識別碼。 | 藉由 GitHub 識別碼識別作者, 以防內容有問題或發生問題。在某些情況下, GitHub 自動化可能會通知作者有關該檔案的活動。 |
| <code>description</code> | 內容摘要。75-300 個字元。 | 用於網站搜尋。有時會用在搜尋引擎的 [結果] 頁面上, 以改善 SEO。 |
| <code>ms.author</code> | 作者的 Microsoft 別名, 不含 "@microsoft.com"。如果您不是 Microsoft 員工, 請在此欄位中尋找適合使用的 Microsoft 員工。 | 識別文章的擁有者。擁有者會負責有關文章內容的決策, 以及文章的報告和 BI 相關決策。 |
| <code>ms.date</code> | 格式為 MM/DD/YYYY 的日期。 | 顯示在 [已發佈] 頁面上, 以指出最後一次編輯發行項的時間, 或保持最新。輸入的日期沒有時間, 而且會在 UTC 時區中轉譯為 0:00 和。顯示給使用者的日期會轉換成其時區。 |
| <code>ms.service</code> 或 <code>ms.prod</code> | 服務或產品識別碼。使用其中一種, 絕不能同時使用。此值通常會在 <i>docfx.json</i> 檔案進行全域設定。 | 用於問題分級和報告。

一般而言, 用於 <code>ms.service</code> 雲端應用程式, 並用於內部 <code>ms.prod</code> 部署伺服器 and 應用程式。 |
| <code>ms.topic</code> | 通常是下列其中一個值:

<code>article</code> , <code>conceptual</code> ,
<code>contributor-guide</code> ,
<code>interactive-tutorial</code> , <code>overview</code> ,
<code>quickstart</code> , <code>reference</code> , <code>sample</code> ,
<code>tutorial</code> . | 識別報表用途的內容類型。 |
| <code>title</code> | 頁面標題。 | 這是顯示在 [瀏覽器] 索引標籤上的頁面標題。這是最重要的 SEO 中繼資料。 |

選擇性中繼資料

除了必要的中繼資料之外，還有許多您可以指定的選擇性中繼資料索引鍵。下表顯示一些選用的中繼資料索引鍵。

| II | I | III? |
|----------------------------|--|---|
| <code>ms.custom</code> | <div>██████████</div> <p>通常用來追蹤遙測工具中的特定檔或內容集。它是單一字串值，而是由使用中的工具進行剖析。範例：</p> <div><code>ms.custom: "experiment1, content_reporting, all_uwp_docs, CI_Id=101022"</code></div> <div>██████: ████████125████。</div> | <code>ms.custom</code> 是一種自訂欄位，可供寫入器用來追蹤特殊專案或內容子集。 |
| <code>ms.reviewer</code> | 評論內容之人員的 Microsoft 別名。 | |
| <code>ms.subservice</code> | 內容所屬的更細微服務。僅在 <code>ms.subservice</code> 您也使用 <code>ms.service</code> | <code>ms.subservice</code> 本身不是有效的中繼資料。作者必須將它與父值產生關聯 <code>ms.service</code> 。這個屬性可讓您進一步向下切入指定的報告 <code>ms.service</code> 。 |
| <code>ms.technology</code> | 內容所屬的技術。只有 <code>ms.technology</code> 當您同時使用時，才使用 <code>ms.prod</code> 。 | <code>ms.technology</code> 本身不是有效的中繼資料。作者必須將它與父值產生關聯 <code>ms.prod</code> 。這個屬性可讓您進一步向下切入指定的報告 <code>ms.prod</code> 。 |
| <code>ROBOTS</code> | <code>NOINDEX</code> ， <code>UNFOLLOW</code> | 在您的中繼資料區段中使用機器人，以防止組建和發行程式在搜尋頁面上顯示內容。當您想要使用 <code>ROBOTS</code> (和 [是] 時，雖然不) 其他元資料標記，但全部都是大寫的：
-新增 <code>ROBOTS: NOINDEX</code> 至中繼資料區段。
- <code>NOINDEX</code> 導致資產不會顯示在搜尋結果中。
- <code>NOFOLLOW</code> 僅在封存整個內容集時使用。 |
| <code>no-loc</code> | 文章中不應 (當地語系化) 轉譯的單字清單。 | 使用此中繼資料可避免「overlocalization」。 |

另請參閱

- [中繼資料瀏覽器](#)

適用於 VS Code 的 Docs 編寫套件

2021/9/24 •

Docs 編寫套件是 VS Code 延伸模組的集合，有助於 docs.microsoft.com 的 Markdown 編寫。此套件可在 [VS Code Marketplace](#) 中取得，且包含下列延伸模組：

- **檔 Markdown**：提供 docs.microsoft.com (檔) 內容的 Markdown 撰寫協助，包括基本 Markdown 支援，以及檔中自訂 Markdown 語法的支援，例如警示、程式碼片段和不可當地語系化的文字。現在也包含一些基本的 YAML 撰寫協助，例如插入 TOC 專案。
- **markdownlint**：David Anson 的熱門 Markdown linter，協助確保您的 Markdown 有效。
- **Code Spell Checker** (程式碼拼字檢查程式)：由 Street Side Software 開發的完整離線拼字檢查程式。
- **Docs Preview** (文件預覽)：使用 docs.microsoft.com CSS 來取得更精確的 Markdown 預覽，包括自訂 Markdown。
- **檔文章範本**：允許使用者 scaffold 學習模組，並將 Markdown 的基本架構內容套用至新檔案。
- **檔 YAML**：提供檔 YAML 架構驗證和自動完成。
- **檔影像**：提供資料夾和個別檔案的影像壓縮和調整大小，以協助 docs.microsoft.com 的作者。

必要條件與假設

若要使用檔 Markdown 擴充功能插入相對連結、影像和其他內嵌內容，您必須將 VS Code 工作區的範圍設為已複製之開放式發佈系統 (OPS) 存放庫的根目錄。例如，如果您已將檔存放庫複製到 `C:\git\SomeDocsRepo\`，請在 [VS Code: > 開啟資料夾] 功能表或 `code C:\git\SomeDocsRepo\` 從命令列中開啟該資料夾或子資料夾。

延伸模組支援的某些語法 (例如警示和程式碼片段) 是 OPS 的自訂 Markdown。除非透過 OPS 發佈，否則自訂 Markdown 將不會正確呈現。

如何使用 Docs Markdown 延伸模組

若要存取 [檔 Markdown] 功能表，請輸入 ALT + M。您可以按一下或使用向上箭號和向下箭號來選取您要的命令。或者，您可以輸入以開始篩選，然後在功能表中反白顯示您想要的函式時按 ENTER 鍵。

請參閱 [檔 Markdown 讀我檔案](#) 以取得最新的命令清單。

如何產生主要重新導向檔案

檔 Markdown 延伸模組包含一個腳本，可根據個別檔案中的中繼資料，產生或更新存放庫的主要重新導向檔案 `redirect_url`。此腳本會檢查存放庫中的每個 Markdown 檔 `redirect_url`，並將重新導向中繼資料新增至存放庫的主要重新導向檔案 (`.openpublishing.publish.config.json`)，並將重新導向的檔案移至存放庫以外的資料夾。執行指令碼：

1. 選取 F1 以開啟 VS Code 的命令選擇區。
2. 開始鍵入 "Doc: 產生 ..."
3. 選取命令 `Docs: Generate master redirection file`。
4. 當腳本執行完成時，重新導向的結果會顯示在 [VS Code 輸出] 窗格中，而移除的 Markdown 檔案將會新增至預設路徑底下的 [檔 Authoring\redirects] 資料夾中。
5. 檢閱結果。如預期般，請提交提取要求以更新存放庫。

如何指派鍵盤快速鍵

1. 輸入 ctrl + K，然後按 ctrl + S 以開啟 [鍵盤快速鍵] 清單。

2. 搜尋 `formatBold` 您要建立自訂按鍵系結的命令(例如)。
3. 按一下當您將滑鼠移至該行上時, 顯示在命令名稱附近的加號。
4. 看到新的輸入方塊之後, 請鍵入您要連結至該特定命令的鍵盤快速鍵。例如, 若要使用粗體的一般快速鍵, 請輸入 `Ctrl + B`。
5. 最好將子句插入金鑰系結 `when` 中, 如此一來, 就不會在 Markdown 以外的檔案中使用。若要執行此作業, 請開啟 `keybindings.json`, 並在命令名稱下面插入下行 (務必在各行之間新增逗號):

```
"when": "editorTextFocus && editorLangId == 'markdown'"
```

您完成的自訂按鍵系結在 `keybindings.json` 中看起來應該像這樣:

```
[
  {
    "key": "ctrl+b",
    "command": "formatBold",
    "when": "editorTextFocus && editorLangId == 'markdown'"
  }
]
```

TIP

將您的金鑰系結放置於此檔案以覆寫預設值

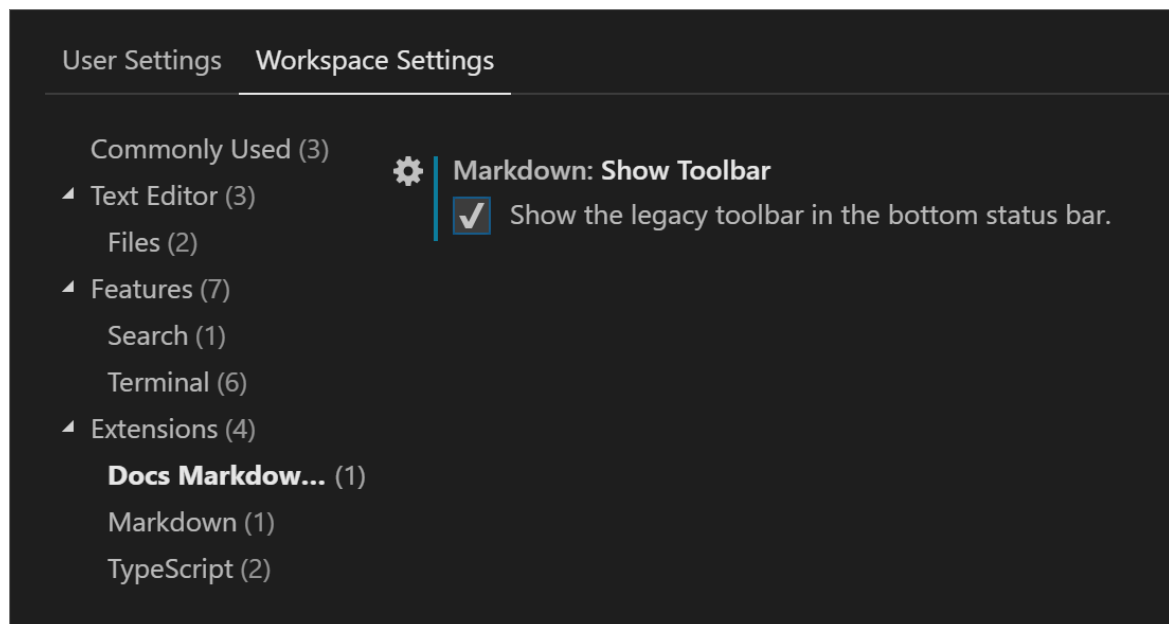
6. 儲存 `keybindings.json`。

如需金鑰系結的詳細資訊, 請參閱[VS Code](#)檔。

如何顯示舊版 "Gauntlet" 工具列

"Gauntlet" 延伸模組程式碼的先前使用者將會注意到, 安裝 Docs Markdown 延伸模組後, 編寫工具列不再顯示於 VS Code 視窗的底端。這是因為工具列佔用了 VS Code 狀態列上的大型空間, 但未遵循延伸模組 UX 的最佳做法, 所以在新的延伸模組中已淘汰。不過, 您可以更新 VS Code 的 `settings.json` 檔案, 以便選擇性地顯示工具列, 如下所示:

1. 在 VS Code 中, 移至 [檔案喜好設定] > > 設定 或選取 `Ctrl + ,`。
2. 選取 [使用者設定] 來變更所有 VS Code 工作區或 工作區 的設定, 設定只針對目前的工作區變更這些設定。
3. 選取延伸模組檔 > Markdown 延伸 模組設定], 然後選取 底部狀態列中的 [顯示舊版工具列]。



完成選取之後，VS Code 會更新 設定的 *json* 檔案。系統會提示您重載視窗，讓變更生效。

新增至擴充功能的較新命令將無法從工具列使用。

如何使用文件範本

Docs Article Templates (文件文章範本) 延伸模組可讓 VS Code 中的作者從中央存放庫提取 Markdown 範本並將它套用到檔案。範本可以協助確保文章中包含必要中繼資料，以及遵循內容標準等等。範本是在 GitHub 存放庫中以 Markdown 檔案方式管理。

在 VS Code 中套用範本

1. 確定已安裝並啟用檔文章範本延伸模組。
2. 如果您未安裝檔 Markdown 延伸模組，請按一下 F1 開啟 [命令選擇區]，開始輸入「範本」進行篩選，然後按一下 `Docs: Template`。如果您已安裝檔 Markdown，您可以使用 [命令選擇區]，或按一下 [Alt + M] 以顯示檔 Markdown QuickPick 功能表，然後 `Template` 從清單中選取。
3. 從出現的清單中選取想要的範本。

將您的 GitHub ID 和/或 Microsoft 別名新增到您的 VS Code 設定

Templates (範本) 延伸模組支援三種動態中繼資料欄位：作者、ms.author 與 ms.date。這表示若範本建立者在 Markdown 範本的中繼資料標頭中使用這些欄位，當您套用此範本時，系統將會在您的檔案中自動填入，如下所示：

| 欄位 | 說明 |
|------------------------|--|
| <code>author</code> | 您的 GitHub 別名 (如果在 VS Code 設定檔中指定的話)。 |
| <code>ms.author</code> | 您的 GitHub 識別碼 (若已在您的 VS Code 設定檔中指定)。如果您不是 Microsoft 員工，請不要指定。 |
| <code>ms.date</code> | 檔支援格式的目前日期 <code>MM/DD/YYYY</code> 。如果您後續更新檔案，日期不會自動更新，您必須以手動方式更新該檔案。這個欄位是用來表示「發行項有效期限」。 |

若要設定作者和/或 ms. 作者

1. 在 VS Code 中，移至 [檔案喜好設定] > > 設定 或選取 Ctrl +、。
2. 選取 [使用者 設定] 來變更所有 VS Code 工作區的設定，或選取 [工作區] 設定，只變更目前工作區的設定。
3. 在左側的預設設定窗格中，尋找檔 文章範本延伸模組 設定，按一下所需設定旁的鉛筆圖示，然後按一下設定

中的 [取代]。

4. [使用者 設定] 窗格將會並排開啟，而且底部會有新專案。
5. 視需要新增您的 GitHub 識別碼或 Microsoft 電子郵件別名，然後儲存檔案。
6. 您可能需要關閉並重新啟動 VS Code，變更才會生效。
7. 現在，當您套用使用動態欄位的範本時，您的 GitHub 識別碼和/或 Microsoft 別名將會在中繼資料標頭中自動填入。

若要在 VS Code 中提供新範本

1. 將您的範本草稿為 Markdown 檔。
2. 提交提取要求至 [MicrosoftDocs/內容範本](#) 存放庫的 [範本] 資料夾。

如果您的範本符合 docs.microsoft.com 樣式指導方針，docs.microsoft.com 小組將會審查您的範本併合並 PR。合併後，範本將可供檔文章範本延伸模組的所有使用者使用。

示範數項功能

以下是簡短的视频，示範檔製作套件的下列功能：

- **YAML 檔案**
 - 支援「檔：程式庫中的檔案連結」
- **Markdown 檔案**
 - 更新 "ms.date" 中繼資料值內容功能表選項
 - 程式碼隔離語言識別項的程式碼自動完成支援
 - 無法辨識的程式碼隔離語言識別項警告/自動校正支援
 - 排序選取專案遞增 (A 到 Z)
 - 以遞減方式排序選取範圍 (Z 到)

下一步

探索檔製作套件中提供的各種功能，Visual Studio Code 延伸模組。

- [開發語言完成](#)
- [影像壓縮](#)
- [中繼資料更新](#)
- [重新格式化資料表](#)
- [智慧引號取代](#)
- [排序重新導向](#)
- [排序選取項目](#)

開發語言完成

2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

參與者需要協助, 來判斷 Markdown 檔案中跟在三個倒引號 (程式碼柵欄的開頭) 之後的有效語言識別碼 (dev lang)。可惜的是, 開發語言的組建階段驗證並不存在。結果是相同概念文件集中的同一語言卻有不同的表示法。

以 C# 為例。參與者已使用 `c#`、`C#`、`cs`、`csharp` 和其他字眼做為該語言的開發語言表示法。這些表示法中哪一個正確?

「開發語言完成」功能會顯示已知的開發語言清單來消除混淆。從 IntelliSense 中選取開發語言名稱時:

- 程式碼隔離已結尾。
- 插入點位於程式碼隔離中。

喜好設定

無法停用此功能。可用的設定如下:

- [顯示常用的開發語言](#)
- [顯示所有已知的開發語言](#)

顯示常用的開發語言

只有有效開發語言的子集會用於單一文件集中。若要提升使用者體驗:

1. 在 Visual Studio Code 中, 開啟文件集根目錄。
2. 選取 [檔案] > [喜好設定] > [設定], 然後依 *Docs Markdown Extension* 篩選。
3. 按一下 [在 settings.json 中編輯] 連結 (位於 [Markdown: 文件集語言] 區段中)。
4. 在 settings.json 檔案中加入以下 `markdown.docsetLanguages` 屬性:

```
{
  "markdown.docsetLanguages": [

  ]
}
```

5. 將插入點放在屬性的空白陣列中, 然後啟動 IntelliSense (透過 Ctrl + Space)。已知的開發語言名稱清單隨即出現。
6. 將開發語言名稱新增至陣列, 直到您滿意清單為止。例如, 下列清單會在使用者輸入三個倒引號之後, 對使用者顯示四個開發語言名稱:

```
{
  "markdown.docsetLanguages": [
    ".NET Core CLI",
    "C#",
    "Markdown",
    "YAML"
  ]
}
```

7. 將您的變更儲存到 settings.json 檔案。

WARNING

空的 `markdown.docsetLanguages` 陣列會使所有已知的開發語言都顯示出來。

顯示所有已知的開發語言

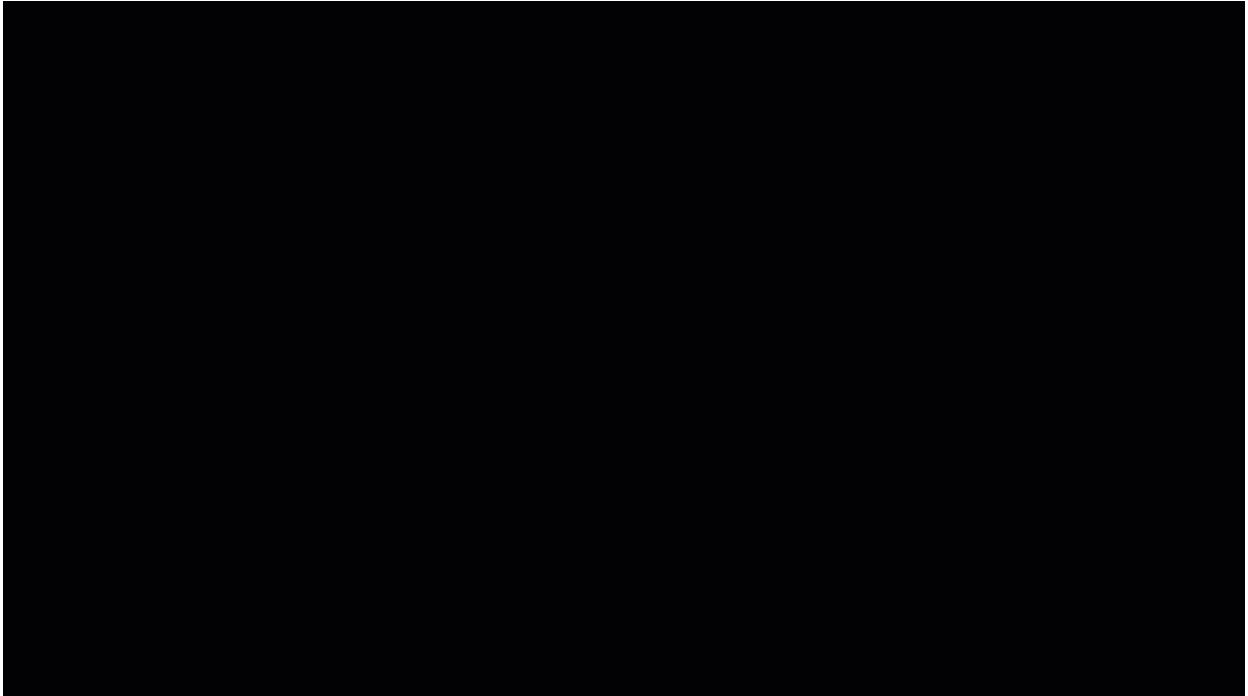
根據預設, 所有已知的開發語言名稱都會顯示在 IntelliSense 中。此設定會覆寫 [顯示常用的開發語言](#) 中所述的 `markdown.docsetLanguages` 屬性。

變更此設定：

1. 選取 [檔案] > [喜好設定] > [設定]，然後依 *Docs Markdown Extension* 篩選。
2. 開啟或關閉 [Markdown: 所有可用的語言] 區段中的設定。

操作實況

以下是這項功能的簡短示範：



影像壓縮

2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [影像](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

所有的文件都是透過網路提供, 只有 PDF 版本的文件除外。提供靜態內容時, 最好能將透過網路傳送的位元組數目降至最低。其中一種方法是壓縮待用影像。

Docs 編寫套件延伸模組包含影像壓縮操作功能表項目。支援下列的影像類型/副檔名:

- *.png
- *.jpg
- *.jpeg
- *.gif
- *.svg
- *.webp

在適用的情況下, 會使用不失真的影像壓縮演算法。

壓縮影像

在 [Explorer] 導覽窗格中, 以滑鼠右鍵按一下影像檔, 然後選取 [壓縮影像] 選項。便可壓縮影像。

壓縮資料夾中的影像

在 [Explorer] 導覽窗格中, 以滑鼠右鍵按一下包含影像的資料夾, 然後選取 [壓縮資料夾中的影像] 選項。便可壓縮資料夾中的所有影像。

考量

解析度大的影像會默默的調整大小。最大尺寸是根據平台建議的最大寬度 `1,200px`。只有在影像大於建議的大小時才會使用最大尺寸, 而且會在自動調整大小時維持外觀比例。

喜好設定

您可設定最大尺寸, 但預設的最大寬度為 `1200` 像素。若要設定最大尺寸, 選取 [檔案-> 喜好設定-> 設定], 並依 `"Docs Image Extension"` 篩選。

Docs Images Extension Configuration

Docs Images: Max Height

The maximum height of an image. When applying image compression, images taller than this will be resized appropriately.

Docs Images: Max Width

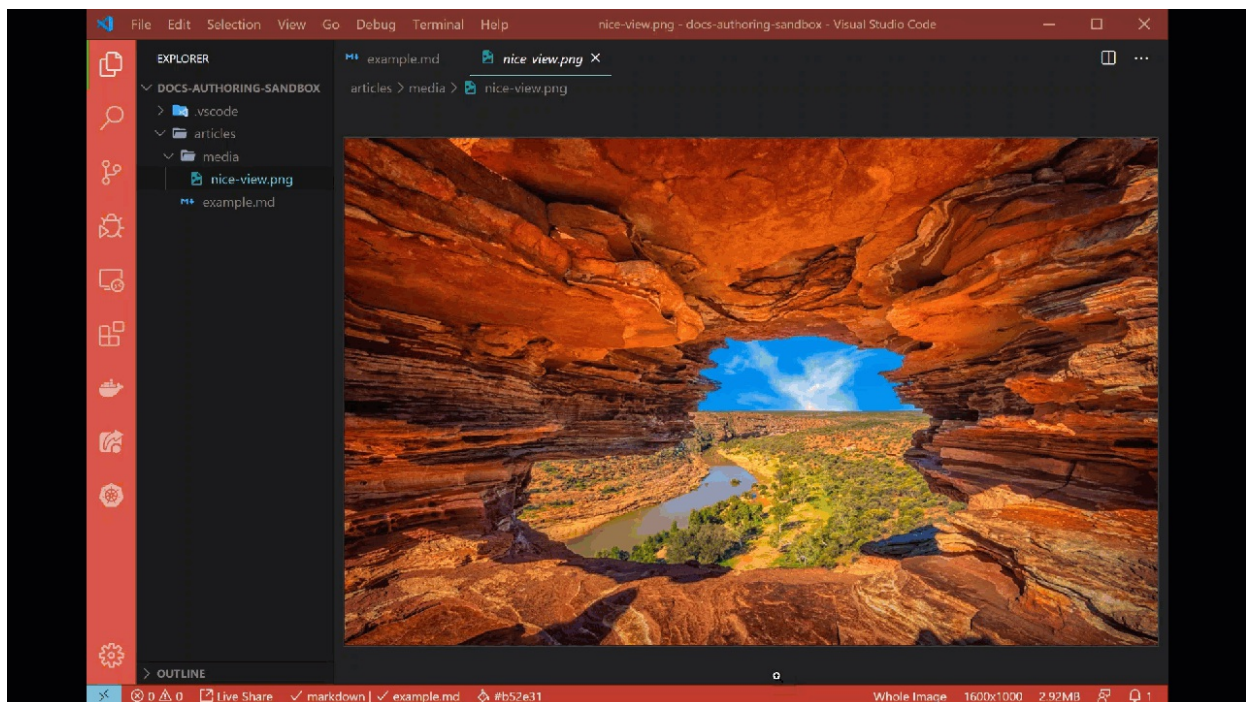
The maximum width of an image. When applying image compression, images wider than this will be resized appropriately.

NOTE

[最大寬度] 或 [最大高度] 的值若為 ，會直接忽略解析度差異。

操作實況

以下是這項功能的簡短示範。



中繼資料瀏覽器

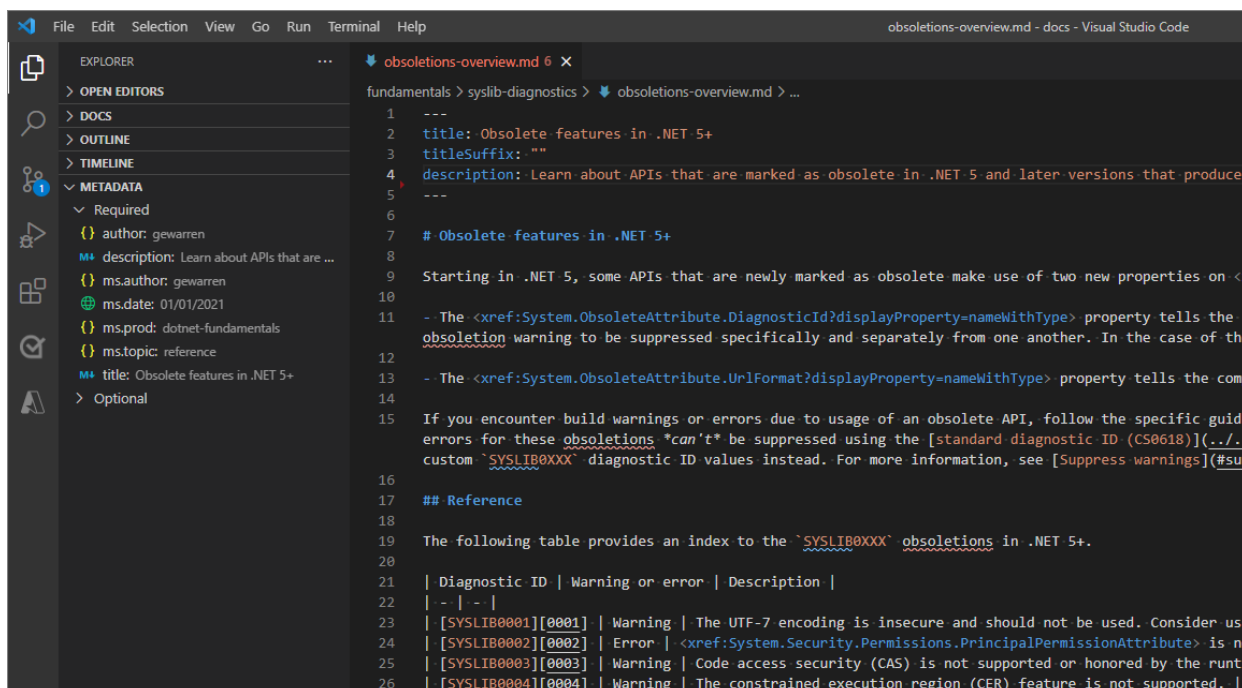
2021/6/23 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

總結

當您開啟檔 Markdown 檔時, 中繼資料瀏覽器會自動出現在 Visual Studio Code 的 Explorer 側邊列中。其中有兩個主要區段: [必要的中繼資料](#) 和 [選擇性中繼資料](#)。



中繼資料來源

中繼資料瀏覽器會顯示來自三個來源的中繼資料: YAML 在編輯器中開啟的 Markdown 檔案, 以及 `fileMetadata` `globalMetadata` docset 之檔案 `docfx.js` 的和區段。圖示有助於快速識別所顯示中繼資料的來源。如果您的文章遺漏任何必要的中繼資料, 該中繼資料索引鍵會出現警告圖示。

| | |
|---|---|
| “ | “““““ |
|  | YAML 使用中的 Markdown 檔案 |
|  | 檔案 <code>docfx.js</code> <code>fileMetadata</code> 區段 |
|  | 檔案 <code>docfx.js</code> <code>globalMetadata</code> 區段 |
|  | 此必要中繼資料已 完全遺失 |

優先順序

中繼資料瀏覽器只會顯示唯一的中繼資料索引鍵。如果有一個以上的值適用於相同的索引鍵，則 explorer 會使用與 DocFx 相同的優先順序設定。優先順序遞減的順序為：

1. YAML 正面重要
2. `fileMetadata`
3. `globalMetadata`

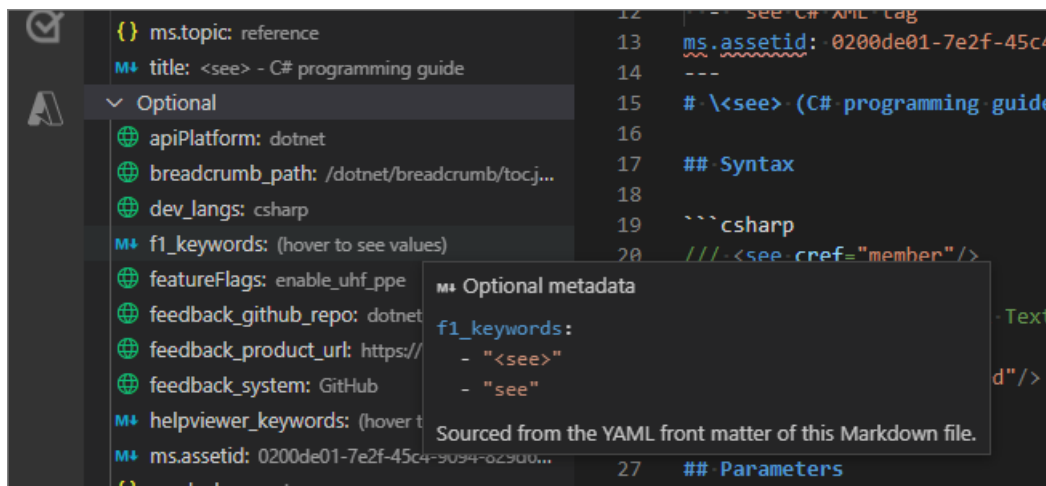
如果有多個以資料夾為基礎的 glob 模式套用至單一發行項，則會選取 [在檔案 `docfx.js` 的最後一個專案]。例如，請考慮位於 `C:/doc/csharp/reference/關鍵字.md` 的檔案，並在檔案的 `docfx.js` 中輸入下列專案：

```
"ms.topic": {
  "csharp/**/*.md": "conceptual",
  "csharp/reference/*.md": "language-reference"
}
```

在此情況下，雖然這兩種 glob 模式都包含檔案的路徑，但已選取的第二個值 `language-reference`。

工具提示

您可以將滑鼠停留在某個值上，以在工具提示中看到展開的顯示。這特別適用於陣列中有多個值的索引鍵。此資訊也包含中繼資料的 [來源](#)。



重新整理

如果您已在 Visual Studio Code 中停用自動儲存，則當您儲存 Markdown 檔案時，中繼資料瀏覽器會自動重新整理。如果您已啟用自動儲存，您可以按一下 explorer 頂端的 [重新整理] 按鈕來重新整理顯示的中繼資料。

當您開啟不同的 Markdown 檔案時，explorer 會自動重新整理。

另請參閱

- [中繼資料](#)

更新中繼資料

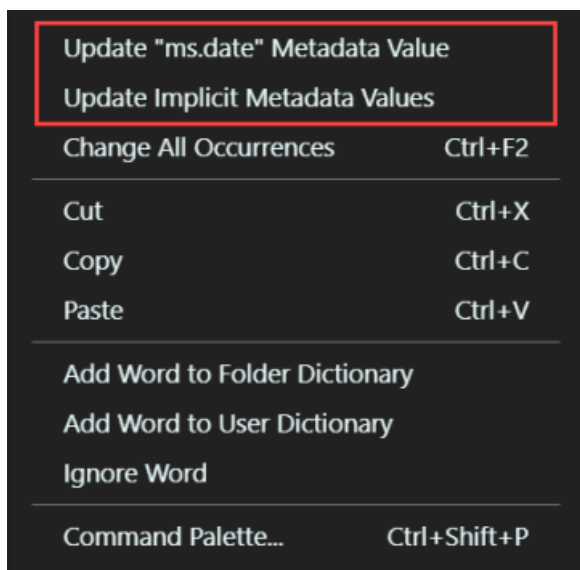
2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

在 Markdown (.md *) 檔案中, 有兩個中繼資料專用的操作功能表項目。當您以滑鼠右鍵按一下文字編輯器中的任何位置, 會看到類似以下的功能表項目:



更新 `ms.date` 中繼資料值

選取 [更新 `ms.date` 中繼資料值] 選項會將目前的 Markdown 檔案 `ms.date` 值設定為今天的日期。如果文件中沒有 `ms.date` 中繼資料欄位, 就不會有任何動作。

更新隱含的中繼資料值

選取 更新隱含中繼資料值 選項將會尋找並取代所有可能隱含指定的中繼資料值。隱含的中繼資料值是在 `build/fileMetadata` 節點下的 `docfx.json` 檔案中指定。 `fileMetadata` 節點中的每個索引鍵值組都代表中繼資料預設值。例如, 在省略 `ms.author` 中繼資料值的 top-level/sub-folder 目錄之中的 Markdown 檔案, 可以隱含地指定要在 `fileMetadata` 節點中使用的預設值。

```
{
  "build": {
    "fileMetadata": {
      "ms.author": {
        "top-level/sub-folder/**/*.md": "dapine"
      }
    }
  }
}
```

在此案例中，所有 Markdown 檔案都會隱含地採用 `ms.author: dapine` 中繼資料值。此功能會作用在 docfx.json 檔案中找到的這些隱含設定。如果 Markdown 檔案包含的中繼資料和值明確設定為隱含值以外的值，便會覆寫隱含值。

請看以下 Markdown 檔案中繼資料 (Markdown 檔案位於 `top-level/sub-folder/includes/example.md`):

```
---
ms.author: someone-else
---

# Content
```

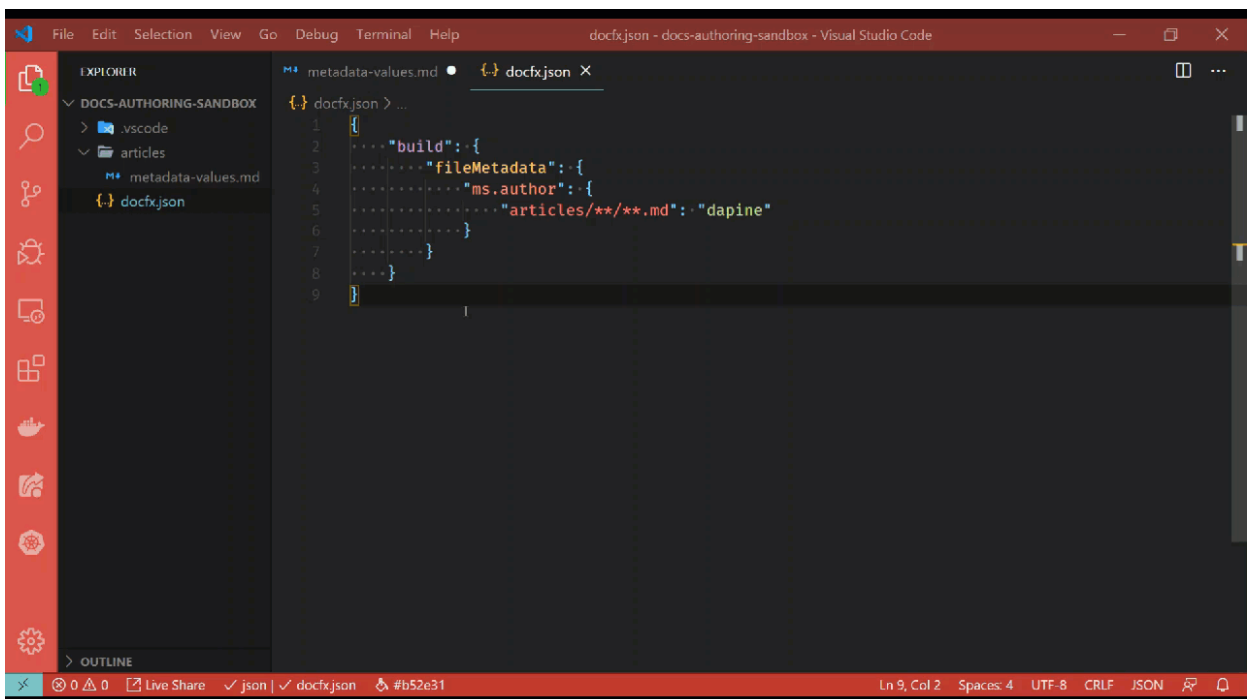
如果已在這個檔案上執行 [更新隱含的中繼資料值] 選項，再加上前述指定的 docfx.json 內容，則中繼資料值會更新為 `ms.author: dapine`。

```
---
ms.author: dapine
---

# Content
```

操作實況

以下是這項功能的簡短示範。



重新格式化 Markdown 資料表

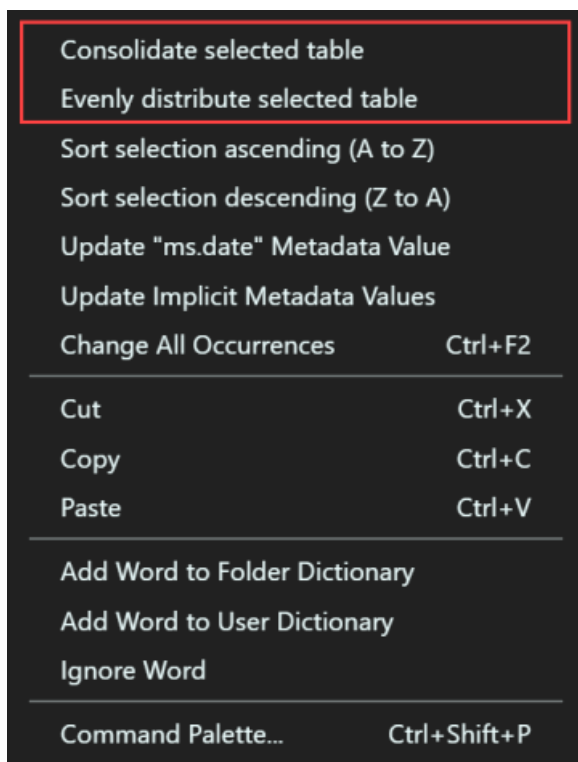
2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

現在, 當您在 Markdown (.md *) 檔案中選取一個完整的資料表時, 在操作功能表中有兩個格式化項目可使用。以滑鼠右鍵按一下選取的 Markdown 資料表即可開啟操作功能表。您會看到類似以下的功能表項目:



TIP

這項功能■用於選取多個資料表, 只能適用於單一 Markdown 資料表。您必須選取整個資料表 (包括標題), 才會同時顯示這兩個項目。

合併選取的資料表

選取 [合併選取的資料表] 選項會摺疊資料表標題和內容, 讓每個值的兩邊只有一個空格。

平均分配選取的資料表

選取 [平均分配選取的資料表] 選項會計算每個資料行中的最長值, 並據此平均分配所有其他值的空間。

考量

此功能不會影響資料表的轉譯, 但有助於改善資料表的可讀性, 進而讓您更容易維護。重新格式化資料表功能可

將資料行對齊。

以下列資料表來說：

| | | | |
|---------|----------------------------|----------|-------------------|
| Column1 | This is a long column name | Column3 | |
| | | | |
| | a value | | |
| | This is a long value | but why? | |
| | | | |
| | | | Here is something |
| | | | |

「平均分配」之後：

| | | | |
|---------|----------------------------|----------------------|-------------------|
| Column1 | This is a long column name | Column3 | |
| | | | |
| | | | a value |
| | | This is a long value | but why? |
| | | | |
| | | | Here is something |
| | | | |

「合併」之後：

| | | | |
|---------|----------------------------|----------|--|
| Column1 | This is a long column name | Column3 | |
| | | | |
| | a value | | |
| | This is a long value | but why? | |
| | | | |
| | Here is something | | |
| | | | |

操作實況

以下是這項功能的簡短示範。

FileEditSelectionViewGoDebugTerminalHelpreformat-table.md - docs-authoring-sandbox - Visual Studio Code

EXPLORER

DOCS-AUTHORING-SANDBOX

- vscode
- articles
 - reformat-table.md

OUTLINE

reformat-table.md

12

Demonstrate table formatting

| | | | |
|---------|----------------------------|---------|-------------------|
| Column1 | This is a long column name | Column3 | |
| | | | |
| | | | |
| | | a value | |
| | | | |
| | This is a long value | | but why? |
| | | | |
| | | | Here is something |
| | | | |

Ln 12, Col 1

Spaces: 4

UTF-8

CRLF

Markdown

智慧引號取代

2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

內容開發人員負責撰寫現代化技術和情報的一些最先進的功能, 然而現今的工具仍偏好在 Markdown 中使用「笨蛋引號」。就怕「智慧引號」聰明反被聰明誤。智慧引號常見於完美的字型設計上, 但在 Markdown 和轉譯 HTML 上並不常用。

例如, 您可能已經注意到, 在處理 Microsoft Word 文件時, 當您按住 Shift 並輸入 " 時, Microsoft Word 會快速地將 " 字元取代為相當於 “ 字元的智慧引號。

| DESCRIPTION | UNICODE | “” | “” |
|-------------|---------|----|----|
| 左雙引號 | \u201c | “ | ” |
| 右雙引號 | \u201d | ” | “ |
| 左單引號 | \u2018 | ‘ | ’ |
| 右單引號 | \u2019 | ’ | ‘ |

在 Markdown (.md**) 檔案中, 當您貼上文字或更新內容時, 這項功能會主動評估內容, 並據此自動取代值。

NOTE

「智慧引號取代」功能也會取代其他字元, 例如但不限於 ©, ™, ®, •、下標和上標字元。從 Word 文件貼入文字時, 這很有用。

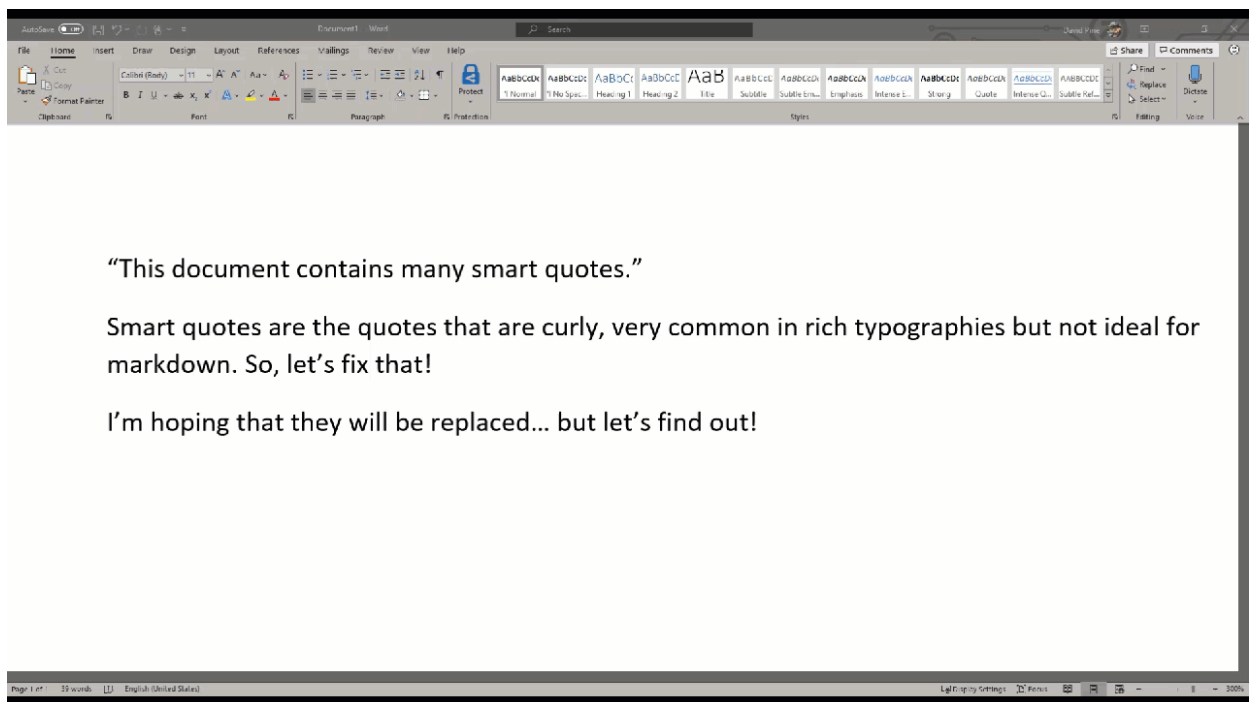
喜好設定

這是選擇性功能, 但預設為 `true`。開啟或關閉此功能:

1. 選取 [檔案] > [喜好設定] > [設定], 然後依 *Docs Markdown Extension* 篩選。
2. 開啟或關閉 [Markdown:取代智慧引號] 區段中的設定。

操作實況

以下是這項功能的簡短示範。



排序重新導向

2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

隨著 docs.microsoft.com 文件集的演進, 部分 Markdown 檔案最後會被刪除。刪除 Markdown 檔案時, 我們必須提供重新導向, 透過重新導向正確地處理指向已刪除文章的任何參考連結。重新導向是在 `.openpublishing.redirection.json` 檔案中指定。

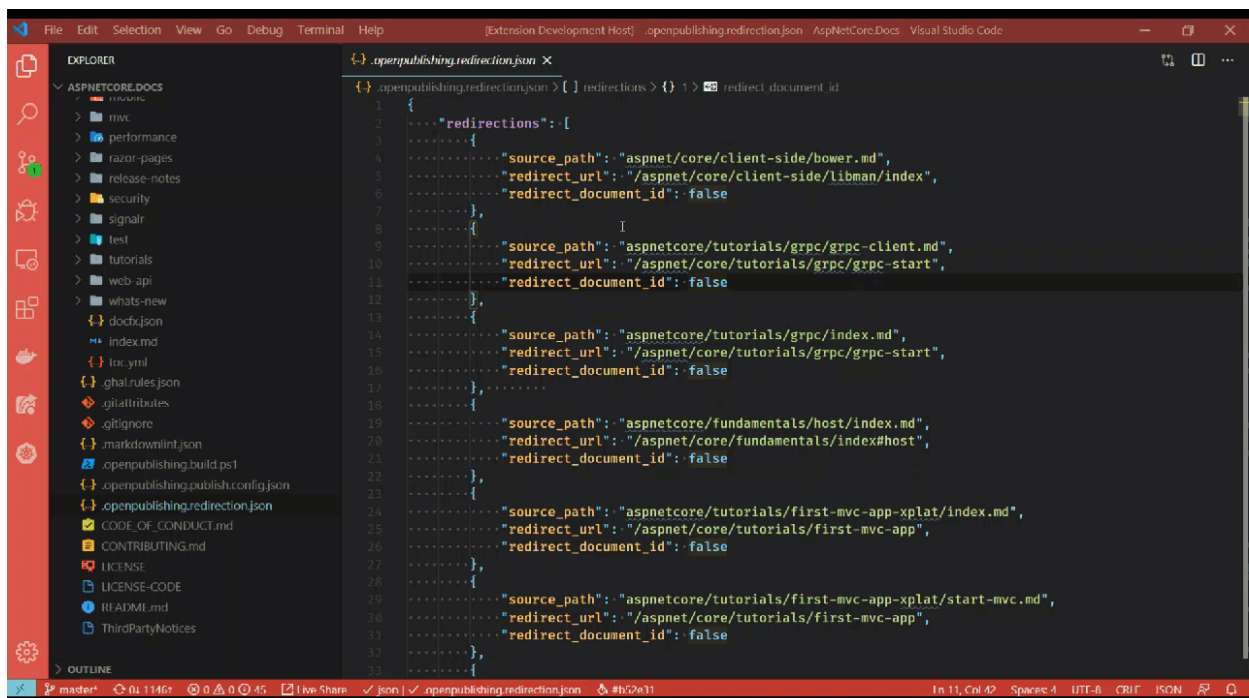
1. 開啟命令選擇區, 按 F1 (或 macOS 上的 `⇧⌘P`)
2. 輸入: `Docs:Sort master redirection file`
3. 選取要執行的命令
4. 觀察 `.openpublishing.redirection.json` 檔案的變更

考量

`.openpublishing.redirection.json` 檔案設計之初便有「菊鍊」的概念。一段時間後, 以重新導向新增的檔案終將過時。刪除檔案 A 需要重新導向至檔案 B, 之後刪除檔案 B 再重新導向至檔案 C, 就是這種情況。理想情況下, 前兩個項目都會指向 C: 讓 A 重新導向至 C, 而 B 保持導向至 C。這是小幅的效能提升, 我們正積極改善此功能。

操作實況

以下是這項功能的簡短示範。



排序選取項目

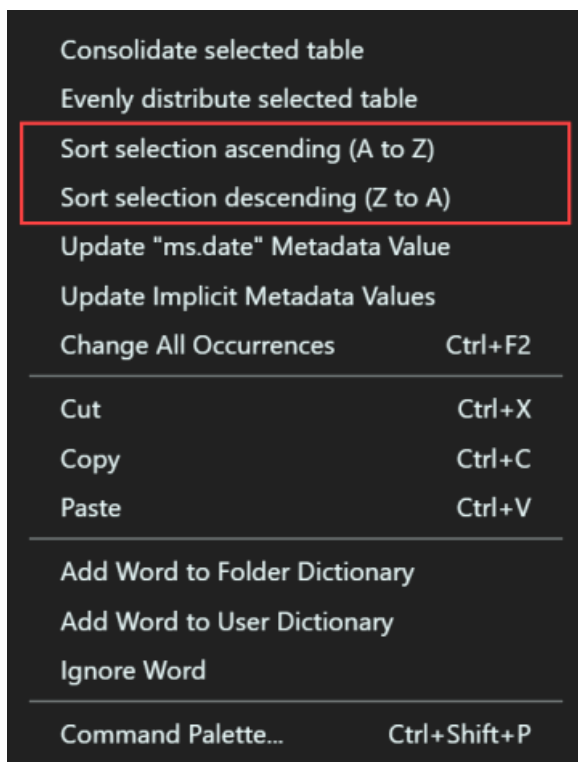
2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

摘要

現在, 當您在 Markdown (.md *) 檔案中選取部分內容時, 在操作功能表中有兩個排序項目可使用。以滑鼠右鍵按一下選取的文字即可開啟操作功能表。您會看到類似以下的功能表項目:



TIP

排序操作功能表的項目會隱藏, 直到 Visual Studio Code 文字編輯器中有有效的選取項目為止。

遞增排序選取項目 (A 到 Z)

選取 [遞增排序選取項目 (A 到 Z)] 選項會將整個選取項目以遞增順序排序, 從 A 到 Z 依字母順序排列。

遞減排序選取項目 (Z 到 A)

選取 [遞減排序選取項目 (A 到 Z)] 選項會將整個選取項目以遞減順序排序, 從 Z 到 A 依字母順序排列。

考量

功能底層的排序機制使用「自然語言」排序。使得它比標準排序更強大且完整。以下列資料表來說:

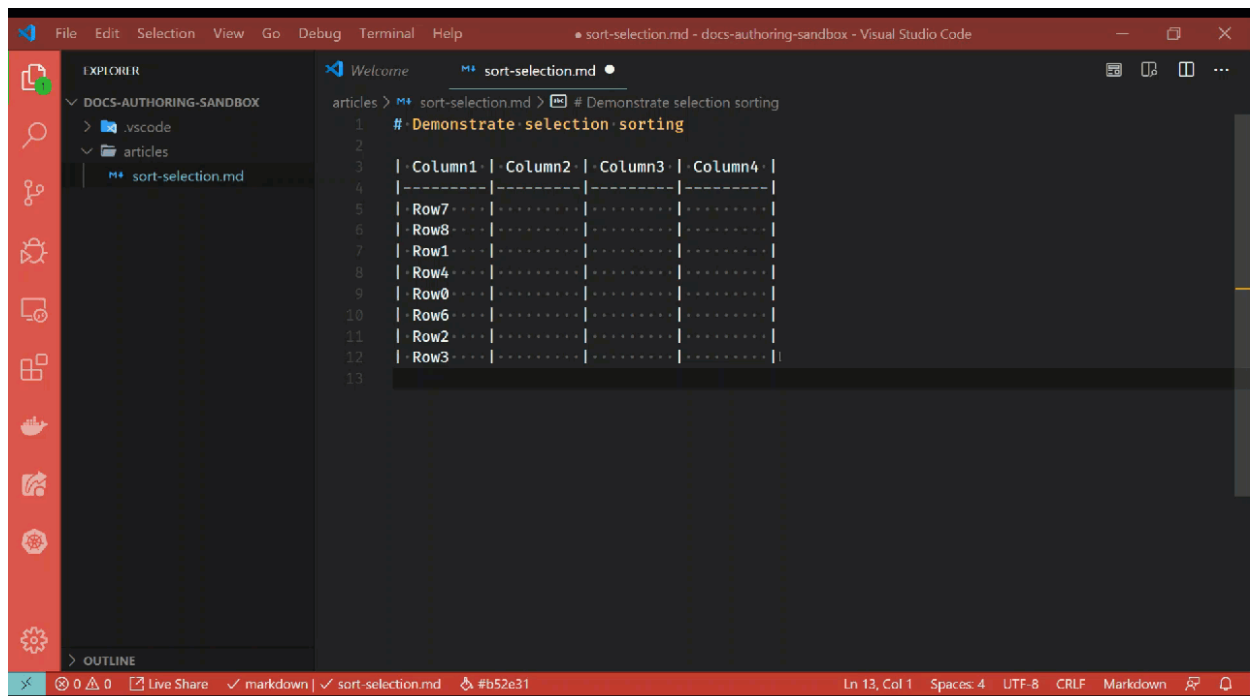
| Column1 | Column2 |
|---------|--|
| 1 | Number 1 |
| Aa | The first letter in the alphabet |
| Ab | The first letter in the alphabet |
| C | The a letter after A in the alphabet |
| M | Somewhere in the middle? |
| 2 | Number 2 |
| X | The alphabet letter is towards the end |
| Z | The last letter in the alphabet |
| 11 | Number 11 |

如果沒有自然語言排序, `Column1` 的順序會是 1、11、2 等。然而實際操作則是 11 大於 2, 故產生下列遞增順序:

| Column1 | Column2 |
|---------|--|
| 1 | Number 1 |
| 2 | Number 2 |
| 11 | Number 11 |
| Aa | The first letter in the alphabet |
| Ab | The first letter in the alphabet |
| C | The a letter after A in the alphabet |
| M | Somewhere in the middle? |
| X | The alphabet letter is towards the end |
| Z | The last letter in the alphabet |

操作實況

以下是這項功能的簡短示範。



從 Jupyter 筆記本插入和更新內容

2021/5/13 •

擴充功能名稱

檔撰寫套件, Visual Studio Code 中繼延伸模組是由多個子延伸模組所組成。這項功能包含在檔 [Markdown](#) 延伸模組中。檔 Markdown 副檔名是檔撰寫套件的一部分, 不需要另外安裝。

IMPORTANT

Jupyter 筆記本功能的插入和更新目前無法在 Mac 上運作。

摘要

Jupyter 筆記本是在 Python 世界中建立和共用程式碼的標準互動式方法。筆記本包含 Python 程式碼的組合、markdown, 以及選擇性的程式碼輸出。

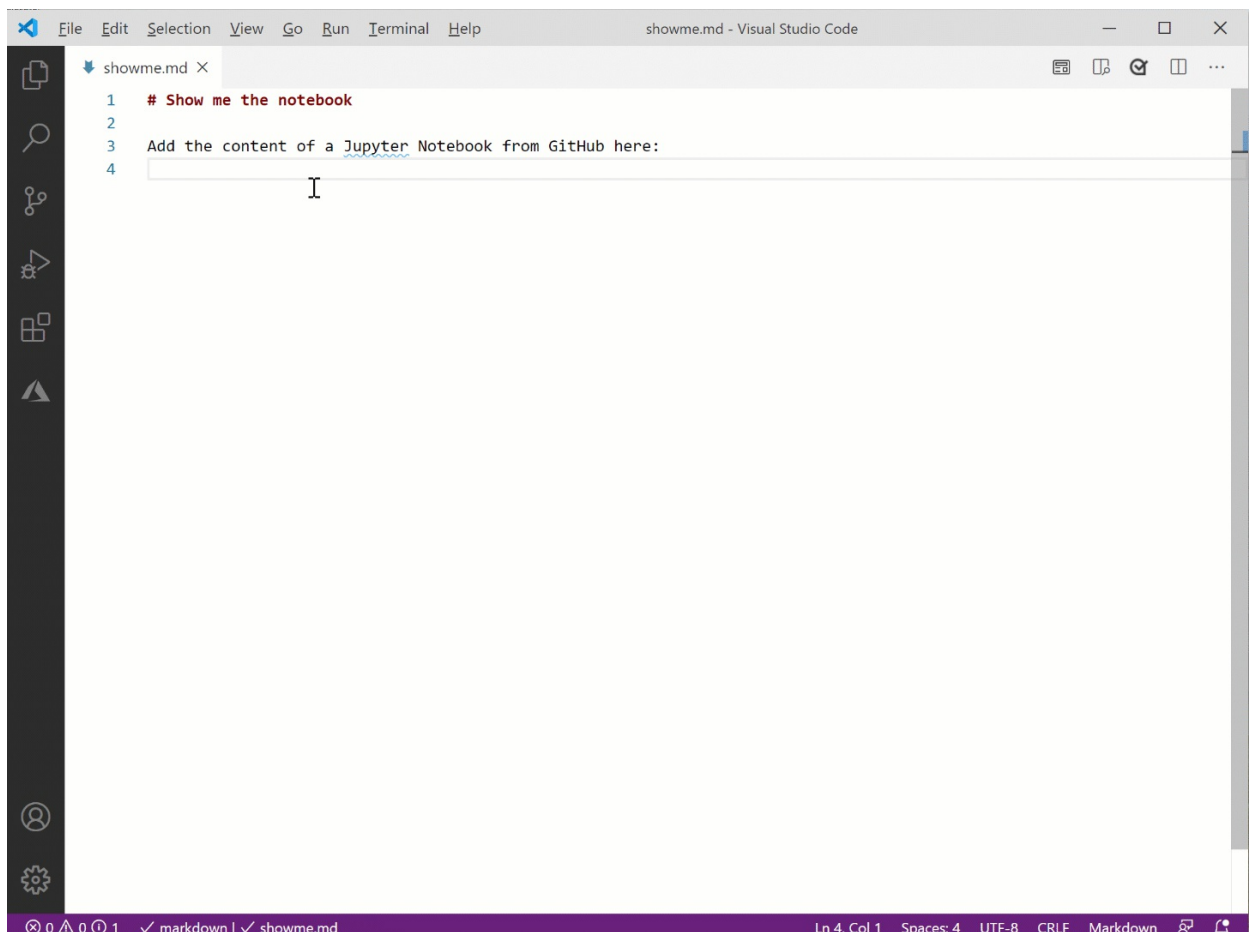
檔編寫套件延伸模組包含將靜態 markdown 版本的 Jupyter 筆記本放入檔中的功能：

檔：**插入 Jupyter 筆記本**：輸入筆記本的 URL。筆記本的 markdown 版本會新增至檔中的游標位置。請勿修改開始或結束標記；這是下一個函式用來更新筆記本的功能。

檔：**更新 Jupyter 筆記本**：此函式會以最新版本的開頭和結尾，取代先前插入的筆記本內容。不需要輸入 URL，它會記錄在開始標記中。更新功能假設檔中有一個筆記本。請勿將多個筆記本新增至單一檔。

實際操作

以下是這項功能的簡短示範。



疑難排解

IMPORTANT

Jupyter 筆記本功能的插入和更新目前無法在 Mac 上運作。

這些函式需要在 `jupyter` 您的 `nbconvert` 電腦上安裝 Python、和。

若要查看是否已安裝 Python，請開啟 VS Code 終端機，然後執行：

- Windows - `where python`
- Linux/Mac- `which python`

如果傳回的是一或多個路徑，則會安裝 Python。如果沒有，請立即 [安裝 Python](#)。

接下來請確定 `jupyter` 已安裝：

- Windows - `where jupyter`
- Linux/Mac- `which jupyter`

如果未傳回路徑，請安裝 `jupyter`：

```
pip install --upgrade jupyter
```

Python 和 `jupyter` 都安裝好之後，請安裝 `nbconvert`：

```
pip install --upgrade nbconvert
```

了解如何參與 .NET 文件存放庫

2021/7/16 •

感謝您對於參與 .NET 文件感到興趣！

本文件涵蓋參與 [.NET 文件網站](#) 上所裝載文章和程式碼範例的程序。這些參與可以是簡單的錯字校正，也可以是複雜的新文章。

.NET 文件網站是從多個存放庫建置而來：

- [.NET 概念文章和程式碼片段](#)
- [程式碼範例和程式碼片段](#)
- [.NET Standard、.NET Core、.NET Framework API 參考](#)
- [.NET Compiler Platform SDK 參考](#)
- [ML.NET API 參考](#)

這些存放庫的問題都已在 [dotnet/docs](#) 存放庫中進行追蹤。

參與方針

感謝社群對文件的參與。下列清單顯示參與 .NET 文件時應注意的一些指導規則：

- 請勿出其不意地提交大量提取要求。相反地，請提出一個問題並開始討論，先讓我們就大方向取得共識再投入大量時間。
- 務必 遵循這些指示，以及 [語態和語調](#) 方針。
- 務必 使用 [範本](#) 檔案作為您工作的起點。
- 務必 先在您的分支上建立個別分支，再參與文章。
- 務必 遵循 [GitHub flow](#) (GitHub 流程)。
- 若想要的話，請 務必 針對參與來撰寫部落格文章和推文 (或透過其他方式)！

遵循這些方針將有助於確保您與我們都有更佳的體驗。

參與 .NET 文件

步驟 1：如果您對撰寫新內容或全面回顧現有內容感興趣，請開啟描述您要執行之作業的 [問題](#)。文件 資料夾的內容會組織成章節並反映在目錄 (TOC) 中。請定義主題在 TOC 中的位置。然後取得此提案的意見反應。

-或-

選擇現有問題並加以解決。您可查看 [open issues](#) (開啟的問題) 清單，並自願參與任何感興趣的問題：

- 依照 [good-first-issue](#) 標籤進行篩選，以取得良好或優先處理的問題。
- 依照 [up-for-grabs](#) 標籤進行篩選，以取得適合由社群進行貢獻的問題。這些問題通常僅需要最少的內容。
- 具備經驗的參與者可處理任何其感興趣的問題。

當找到要處理的問題時，請新增註解來詢問該問題是否處於開啟狀態。

一旦選擇了要參與的工作，請遵循 [入門](#) 指南以建立 GitHub 帳戶並設定環境。

步驟 2：視需要派生 `/dotnet/docs`、`dotnet/samples`、`dotnet/dotnet-api-docs`、`dotnet/roslyn-api-docs` 或 `dotnet/ml-api-docs` 存放庫，並為您的變更建立分支。

若變更不大，請參閱參與者指南 [首頁](#) 上有關在 GitHub 中編輯的指示。

步驟 3：在此新分支上進行變更。

如果這是新主題，您可以使用此[範本檔案](#)作為起點。其中包含撰寫方針，並同時說明每篇文章所需的中繼資料，例如作者資訊。如需 docs.microsoft.com 網站中所使用 Markdown 語法的詳細資訊，請參閱 [Markdown 參考](#)。

請巡覽至步驟 1 中針對您文章所決定 TOC 位置的相對應資料夾。該資料夾包含該區段中所有文章的 Markdown 檔案。如有需要，請建立新的資料夾來放置您的內容檔案。該區段的主要文章名為 *index.md*。

針對影像和其他靜態資源，請在包含您文章的資料夾中建立名為 **media** 的子資料夾 (如果尚未存在)。在 **media** 資料夾中，建立具有文章名稱的子資料夾 (索引檔除外)。如需有關如何放置檔案的詳細資訊，請參閱 [範例資料夾結構](#) 一節。

針對 **程式碼片段**，請在包含您文章的資料夾中建立名為 **snippets** 的子資料夾 (如果尚未存在)。在 [snippets] 資料夾內，使用文章名稱建立子資料夾。在大部分情況下，您會有三種主要 .NET 語言的程式碼片段：C#、F# 與 Visual Basic。在該情況下，請分別建立名為 **chsharp**、**fsharp** 和 **vb** 的子資料夾，其分別適用於三種專案。若正在為位於 [docs/csharp](#)、[docs/fsharp](#) 或 [docs/visual-basic](#) 資料夾下的文章建立程式碼片段，由於程式碼片段只會以一種語言呈現，因此可忽略語言子資料夾。如需有關如何放置檔案的詳細資訊，請參閱 [範例資料夾結構](#) 一節。

程式碼片段為一種小篇幅、重點式的程式碼範例，用以示範文章中提及的概念。大型程式 (供使用者下載與探索) 應該放在 [dotnet/samples](#) 存放庫中。完整範例涵蓋在 [參與範例](#) 一節中。

步驟 4：從您的分支 (PR) 提交提取要求至預設分支。

IMPORTANT

目前無法在任何 .NET 文件存放庫上使用[註解自動化](#)功能。.NET 文件小組成員將會審查並合併您的 PR。

除非有多個問題與相同的 PR 修正相關，否則每個 PR 通常應該會一次處理一個問題。PR 可修改一或多個檔案。如果您要解決不同檔案的多個修正，最好使用個別 PR。

如果您的 PR 修正了現有的問題，請將 `Fixes #Issue_Number` 關鍵字新增至 PR 描述。如此一來，當 PR 合併之後，就會自動關閉問題。如需詳細資訊，請參閱 [使用關鍵字將提取要求連結到問題](#)。

.NET 小組將會審查您的 PR，並讓您知道是否需要任何其他更新/變更以通過核准。

步驟 5：根據與小組的討論結果，對您的分支進行任何必要的更新。

一旦套用意見反應並核准您的變更，維護人員就會將您的 PR 合併到預設分支中。

我們會定期將所有認可從預設分支推送到即時分支，然後您就可以看到您的投稿 <https://docs.microsoft.com/dotnet/>。我們一般會在工作週期間每天發佈。

範例資料夾結構

```
docs
  /about
  /core
  /porting
    porting-overview.md
  /media
    /porting-overview
      portability_report.png
    /shared ...
  /snippets
    /porting-overview
      /csharp
        porting.csproj
        porting-overview.cs
        Program.cs
      /fsharp
        porting.fsproj
        porting-overview.fs
        Program.fs
      /vb
        porting.vbproj
        porting-overview.vb
        Program.vb
    /shared
      /csharp ...
      /fsharp ...
      /vb ...
```

NOTE

假設只有一個語言，語言指南區域中的程式碼片段底下不需要語言資料夾。例如，在 c # 指南中，假設所有程式碼片段都是 c #。

如上所示的結構包含一個影像 *portability_report.png*，以及三個程式碼專案，專案中含有 *porting-overview.md* 文章中的 **程式碼片段**。

程式碼片段/共用 資料夾用於可能橫跨相同父資料夾內多篇文章的程式碼片段，例如上一個範例中的 *移植* 資料夾。只有當您有特定原因要這麼做時，才使用 **共用** 資料夾，例如多個 articles 所參考的 XAML 程式碼，但是無法在發行項 **特定** 資料夾中進行編譯。

當這些文章位於相同的上層資料夾(例如上述範例中的 *移植* 資料夾)時，也可以在發行項之間共用媒體。如果可能的話，應該避免使用這個 **共用** 資料夾，而且只有在有意義時才使用。例如，共用常見的載入畫面來顯示所示範的應用程式，或共用在多個發行項中重複使用的 Visual Studio 對話可能是合理的。

IMPORTANT

為了留下歷程記錄，許多包含的程式碼片段會儲存在 *dotnet/docs* 存放庫中的 */samples* 資料夾底下。如果要大幅變更文章內容，則這些程式碼片段應移至新的結構。不過，請不要擔心移動小變更的程式碼片段。

參與範例

我們對程式碼進行了下列區別，以支援我們的內容：

- **範例**：讀者可以下載並執行範例。所有範例都應該是完整的應用程式或程式庫。使用範例建立程式庫時，應該包含單元測試或應用程式，讓讀者執行程式碼。這些範例通常會使用多項技術、功能或工具組。每個範例的 readme.md 檔案會參考一篇文章，您可以閱讀以深入了解每個範例所包含的概念。
- **程式碼片段**：說明較小的概念或工作。它們可供編譯但並非用作完整的應用程式。它們應該會正確執行，但不是典型案例的範例應用程式。相反地，它們設計成盡可能小到足以說明單一概念或功能。這些程式碼片段不

應該超過一個螢幕的程式碼。

範例儲存在 [dotnet/samples](#) 存放庫中。我們將致力於建立 samples 資料夾結構符合文件資料夾結構的模型。我們遵循下列標準：

- 最上層資料夾會對應到 docs 存放庫中的最上層資料夾。例如，文件存放庫具有 *machine-learning/tutorials* 資料夾，而機器學習服務教學課程的範例會位於 *samples/machine-learning/tutorials* 資料夾中。

此外，*core* 和 *standard* 資料夾下所有範例都應該在 .NET Core 支援的所有平台上建置和執行。我們的 CI 建置系統將會強制執行該項作業。最上層 *framework* 資料夾包含只能在 Windows 上建置和驗證的範例。

請盡可能在適用於指定範例的一組最廣泛平台上建置和執行範例專案。實際上，這表示盡可能建置以 .NET Core 為基礎的主控台應用程式。Web 或 UI 架構特定的範例應該視需要新增這些工具。範例包括 Web 應用程式、行動應用程式、WPF 或 WinForms 應用程式等。

我們將致力於開發適用於所有程式碼的 CI 系統。當您對範例進行任何更新時，請確定每項更新都是可建置專案的一部分。在理想情況下，也可針對範例的正確性新增測試。

您所建立的每個完整範例都應該包含一個 *readme.md* 檔案。此檔案應該包含範例的簡短描述（一或兩個段落）。您的 *readme.md* 應該告訴讀者探索此範例將會學到的內容。*readme.md* 檔案也應該包含 [.NET 文件網站](#) 上即時文件的連結。若要判斷存放庫中指定檔案對應至該網站的位置，請將存放庫路徑中的 `/docs` 取代為

```
https://docs.microsoft.com/dotnet
```

。

您的主題也會包含範例連結。這會直接連結至 GitHub 上的範例資料夾。

撰寫新的範例

範例為可供下載的完整程式與程式庫。範例的範圍可能很小，但會以可讓使用者自行探索與實驗的方式來示範概念。範例的指導方針確保讀者可下載與探索。檢查 [Parallel LINQ \(PLINQ\)](#) 範例，以作為每個指導方針的範例。

1. 範例 **必須是**可建置專案的一部分。請盡可能在 .NET Core 支援的所有平台上建置專案。但示範平台特定功能或平台特定工具的範例則除外。
2. 您的範例必須符合 [執行時間編碼樣式](#) 才能維持一致性。

此外，當示範不需要具現化新物件的內容時，我們偏好使用 `static` 方法，而不是執行個體方法。

3. 您的範例應該包含 **適當的例外狀況處理**。它應該處理範例內容中可能擲回的所有例外狀況。例如，呼叫 `Console.ReadLine` 方法以擷取使用者輸入的範例應該在輸入字串作為引數傳遞至方法時，使用適當的例外狀況處理。同樣地，如果您的範例預期方法呼叫失敗，則必須處理產生的例外狀況。請務必處理方法所擲回的特定例外狀況，而不是 `Exception` 或 `SystemException` 等基底類別例外狀況。

若要建立範例：

1. 提出 [問題](#)，或對您目前參與的現有問題新增註解。
2. 撰寫主題，以說明您範例中所示範的概念（例如：`docs/standard/linq/where-clause.md`）。
3. 撰寫您的範例（例如：`WhereClause-Sample1.cs`）。
4. 建立 `Program.cs`，其中包含呼叫您範例的主要進入點。如果已有此檔案，請將呼叫新增至您的範例：

```
public class Program
{
    public void Main(string[] args)
    {
        WhereClause1.QuerySyntaxExample();

        // Add the method syntax as an example.
        WhereClause1.MethodSyntaxExample();
    }
}
```

請使用可透過 [.NET Core SDK](#) 安裝的 .NET Core CLI，來建置任何 .NET Core 程式碼片段或範例。若要建置並執行您的範例：

- 移至範例資料夾和組建以檢查是否有錯誤：

```
dotnet build
```

- 執行您的範例：

```
dotnet run
```

- 將 readme.md 新增至您範例的根目錄。

這應該包含程式碼的簡短描述，並指示使用者前往參考該範例的文章。*readme.md* 的最上層，必須具有 [範例瀏覽器](#) 所需的中繼資料。標頭區塊應包含下欄欄位：

```
---
name: "really cool sample"
description: "Learn everything about this really cool sample."
page_type: sample
languages:
  - csharp
  - fsharp
  - vbnet
products:
  - dotnet-core
  - dotnet
  - dotnet-standard
  - aspnet
  - aspnet-core
  - ef-core
---
```

- `languages` 集合應包含僅適用於您範例的語言。
- `products` 集合應包含僅與您範例相關的產品。

除非另有註明，否則所有範例都是在 .NET Core 支援的任何平台上透過命令列所建置。有些 Visual Studio 特定的範例需要 Visual Studio 2017 或更新版本。此外，有些顯示平台特定功能的範例需要特定平台。其他範例和程式碼片段需要 .NET Framework 並將在 Windows 平台上執行，而且需要適用於目標 Framework 版本的開發人員套件。

C# 互動式體驗

文章中包含的所有程式碼片段都會使用 [語言標記](#) 來表示來源語言。C# 中的簡短程式碼片段可以使用

`csharp-interactive` language 標記來指定在瀏覽器中執行的 c# 程式碼片段。(內嵌程式碼片段使用 `csharp-interactive` 標記，針對來源所包含的程式碼片段，請使用 `code-csharp-interactive` 標記。) 這些程式碼

片段會在文章中顯示程式代碼視窗和輸出視窗。[輸出]視窗會在使用者執行程式碼片段後，顯示任何執行互動式程式碼的輸出。

C# 互動式體驗會變更我們使用程式碼片段的方式。訪客可以執行程式碼片段以查看結果。有幾個因素有助於判斷程式碼片段或對應的文字是否應該包含輸出的相關資訊。

何時顯示預期的輸出，而不執行程式碼片段

- 適用於初學者的文章應該提供輸出，讓讀者可以比較其工作輸出與預期的回應。
- 在主題中輸出為整數的程式碼片段應該會顯示該輸出。例如，格式化文字上的文章應該會顯示文字格式，而不會執行程式碼片段。
- 當代碼段和預期的輸出都很短時，請考慮顯示輸出。這會省下一些時間。
- 說明目前或不因文化特性而異的文化特性 (Culture) 如何影響輸出的文章，應該說明預期的輸出。互動式 REPL (「讀取、求值、輸出」迴圈) 會在 Linux 主機上執行。預設及不因文化特性而異的文化特性 (Culture) 會在不同作業系統和電腦上產生不同輸出。該文應該說明 Windows、Linux 和 Mac 系統中的輸出。

何時從程式碼片段排除預期的輸出

- 程式碼片段產生較大輸出的文章不應包含在批註中。它會在執行程式碼片段之後模糊程式碼。
- 程式碼片段示範主題的文章，但輸出並不是瞭解該主題的整數。例如，若程式碼會執行 LINQ 查詢來說明查詢語法，然後顯示輸出集合中的每個項目。

NOTE

您可能會注意到某些主題目前未遵循這裡所指定的方針。我們將致力於達成整個網站的一致性。請查看我們目前針對該特定目標所追蹤的[開啟問題](#)清單。

參與國際內容

目前不接受機器翻譯 (MT) 的內容。為了改善 MT 內容的品質，我們已轉換成神經 MT 引擎。我們接受並鼓勵人工翻譯 (HT) 內容的貢獻，這可用於訓練神經 MT 引擎。經過一段時間，HT 內容的貢獻將可同時改善 HT 與 MT 的品質。MT 主題將有免責聲明，表示部分主題可能是 MT，而當編輯功能停用時，不會顯示 [編輯] 按鈕。

NOTE

大部分當地語系化的檔不提供透過 GitHub 編輯或提供意見反應的功能。若要針對當地語系化的內容提供意見反應，請使用 [aka.ms/DocSiteLocFeedback](#) 提供的電子郵件範本。Azure 檔有一個例外狀況。Azure 檔有公開當地語系化的 GitHub 存放庫，適用於 6 種語言的社區貢獻：簡體中文、法文、德文、日文、韓文和西班牙文。針對所有其他語言和內容集，請使用 [電子郵件範本](#) (以任何語言接收您的輸入) 來回報翻譯問題或提供技術意見反應。

參與者授權合約

您必須簽署 [.NET Foundation 貢獻授權合約 \(CLA\)](#) 才能合併 PR。這是 .NET Foundation 中專案的一次性要求。您可以在 Wikipedia 上深入了解 [貢獻授權合約 \(CLA\)](#)。

合約：[.Net Foundation 參與者授權合約 \(CLA\)](#)

您不需要事先簽署合約。您可以如往常般複製、派生及提交 PR。當您的 PR 建立之後，會由 CLA Bot 進行分類。如果變更不大 (例如修正錯字)，則 PR 會標記為 `cla-not-required`。否則，它會分類為 `cla-required`。一旦您簽署了 CLA，目前和所有未來提取要求都會標記為 `cla-signed`。

參與 .NET 文件存放庫中的 .NET 程式碼分析規則文件編審

2021/5/13 •

.NET 編譯器平台 (Roslyn) 分析器會檢查 C# 或 Visual Basic 程式碼以找出程式碼品質與程式碼樣式問題。從 .NET 5.0 開始, 這些分析器即隨附於 .NET SDK。

- **程式碼品質分析 ("CAxxxx" 規則):**
 - 已在 `dotnet/roslyn-analyzers` 存放庫中於[這裡](#)實作。
 - 已在 `dotnet/docs` 存放庫中記載在[這裡](#)。請參閱[參與 'CAxxxx' 規則相關文件的編審](#)。
- **程式碼樣式分析 ("IDExxxx" 規則):**
 - 已在 `dotnet/roslyn` 存放庫中於[這裡](#)實作。
 - 已在 `dotnet/docs` 存放庫中記載在[這裡](#)。請參閱[參與 'IDExxxx' 規則相關文件的編審](#)。

參與 'CAxxxx' 規則相關文件的編審

請依照下列步驟在 `dotnet/docs` 存放庫中參與程式碼品質分析規則相關文件的編審:

1. 判斷 `Rule ID` 與 `Category`: 確定您知道要記載之規則的 'CAxxxx' 規則識別碼與類別。這表示您的 CA 分析器已合併到 `dotnet/roslyn-analyzers` 存放庫, 或您有未決的 PR 且該 PR 具有指派到該規則的已核准識別碼與類別。
2. 新增規則文件:
 - a. 在 `root` 資料夾下複製現有的 CA 規則檔案, 假設其為 `ca1000.md`, 並將其重新命名。
 - b. 適當地更新該檔案的內容。
3. 將規則的項目新增到下列資料表:
 - 在規則類別清單下的[目錄](#)檔案中。例如, 針對具有 `Performance` 類別的 CA 規則, 請在檔案中搜尋字詞
 - name: Performance rules 並新增項目到清單中。若沒有任何類別清單在, 請在
 - name: Code quality rules 下建立新的類別清單。
 - 在[頂層規則索引](#)檔案中。
 - 在 `category` 型規則索引檔案中:
 - a. 在 `root` 資料夾下識別類別型索引檔案。例如, 針對具有 `Performance` 類別的 CA 規則, 此檔案的名稱為 `performance-warnings.md`。若沒有任何類別型索引檔案存在, 請透過複製現有檔案來建立新檔案。
 - b. 在此檔案中新增規則識別碼的項目。

TIP

在 `dotnet/docs` 存放庫中執行存放庫搜尋以尋找已知的現有 CA 規則識別碼, 以識別可能需要更新的潛在位置。如需範例, 請參閱 <https://github.com/dotnet/docs/search?q=CA2000>。

參與 'IDExxxx' 規則相關文件的編審

請依照下列步驟在 `dotnet/docs` 存放庫中參與程式碼樣式分析規則相關文件的編審:

1. 判斷 `Rule ID` 與程式碼樣式選項 (如果有的話): 確定您知道要記載之規則的 'IDExxxx' 規則識別碼與程式碼樣式選項。這表示 IDE 分析器已合併到 `dotnet/roslyn` 存放庫, 或您有未決的 PR 且該 PR 具有指派到該規則的已核准識別碼與程式碼樣式選項。

2. 判斷規則 `subcategory` : IDE 規則具有類別 `Style`。透過完整閱讀[程式碼樣式規則](#)的簡介小節，來識別規則子類別。大部分的 IDE 程式碼樣式規則都位於 `Language` 子類別下。
3. 判斷規則 `preference group` : 透過在[這裡](#)閱讀喜好設定標頭，來識別規則的 `preference group`。
 - 若規則具有已知、關聯的程式碼樣式選項，則喜好設定群組將會符合程式碼樣式選項宣告中指定的 `OptionGroup`。
 - 否則，判斷規則是否屬於現有的喜好設定群組或需要新增喜好設定群組。
4. 新增規則文件：
 - a. 在 `root` 資料夾下複製現有的 IDE 規則檔案，假設其為 `ide0011.md`，並將其重新命名。
 - b. 適當地更新該檔案的內容。
5. 新增 IDE 規則的項目和/或程式碼樣式選項到下列資料表：
 - 在規則子類別與喜好設定清單下的[目錄](#)檔案中。例如，針對具有 `Language` 子類別與 `Modifier preferences` 群組的 IDE 程式碼樣式規則，在檔案中搜尋字詞 `- name: Language rules` 與子節點 `- name: Modifier preferences` 並新增項目到清單中。若子類別下的子類別清單或喜好設定清單不存在，請在 `- name: Code style rules` 下建立新項目。
 - 在[程式碼樣式選項型索引](#)檔案中。
 - 在[規則識別碼型索引](#)檔案中。
 - 在 `preferences group` 型規則索引檔案中：
 - a. 在 `root` 資料夾下識別喜好設定群組型索引檔案。例如，針對 `Modifier preferences` 的 IDE 規則，此檔案的名稱為 `modifier-preferences.md`。若沒有任何喜好設定群組型索引檔案存在，請透過複製現有檔案來建立新檔案。
 - b. 在此檔案中新增規則識別碼的項目。

TIP

在 `dotnet/docs` 存放庫中執行存放庫搜尋以尋找已知的現有 IDE 規則和/或程式碼樣式選項，以識別可能需要更新的潛在位置。如需範例，請參閱 <https://github.com/dotnet/docs/search?q=IDE0011> 與 https://github.com/dotnet/docs/search?q=csharp_prefer_braces。

另請參閱

- [參與 .NET 文件存放庫](#)

標籤、專案與里程碑藍圖

2021/5/13 •

.NET 文件小組廣泛使用 [GitHub 標籤](#) 來管理工作。藉由篩選標籤組合，即可快速專注於 [.NET 文件網站](#) 上感興趣的章節。例如，我們可以使用下列查詢來篩選架構指南上的所有開啟問題：[問題是：開啟標籤：「dotnet 架構/生產」](#)。

我們會使用 [GitHub 專案](#) (英文) 來管理短期衝刺與其他目標導向的 Epic。我們也會使用 [GitHub 里程碑](#) (英文) 來追蹤工作。最好將專案視為用於規劃 (問題)，並將里程碑視為用於工作 (提取要求)。

此藍圖說明如何使用這些管理工具，並包含便利的篩選連結，以便用來尋找感興趣的區域。

標籤

如果這是第一次參與 [dotnet/docs](#)，建議您從 [up-for-grabs](#) 問題開始。這些是範圍較集中的問題。也是您第一次參與的絕佳方式。從 [up-for-grabs](#) 檢視，您可根據區域和優先順序進一步篩選問題。如果想要在第一次參與時嘗試較不嚴重的問題，我們在 [good-first-issue](#) 中列出了適合初學者的不錯問題。

我們使用標籤，以許多不同的方式來分類問題：

- [.NET 指南](#)和[指南的章節](#)。
- [目標版本](#)
- [優先順序](#)

您可從每個集合 (指南、版本、優先順序) 合併成一個標籤來建立範圍較窄的焦點，以尋找想要解決的問題。

尋找單一 .NET 指南的問題

我們針對每個架構電子書和每個 .NET 指南使用標籤。所有電子書都會以 [dotnet 架構/生產](#) 標籤來注明。每一本書都有一個以結尾的唯一標籤 `/tech`。

每個 .NET 指南都會加上後置詞 `/prod`，並具有藍色灰色背景。以下是針對每個 .NET 指南篩選的目前問題。

- [.NET 指南](#) - `dotnet/prod`
- [.NET 基礎指南 \(先前 .NET Standard 指南\)](#) - `dotnet-fundamentals/prod`
- [.NET 基礎指南 \(先前的 .NET Core 指南\)](#) - `dotnet-core/prod`
- [.NET Framework 指南](#) - `dotnet-framework/prod`
- [API 參考](#) - `dotnet-api/prod`
- [C # 指南](#) - `dotnet-csharp/prod`
- [F # 指南](#) - `dotnet-fsharp/prod`
- [Visual Basic 指南](#) - `dotnet-的/生產`
- [ML.NET 指南](#) - `dotnet-ml/prod`
- [Azure .NET SDK](#) - `azure-dotnet/prod`
- [適用於 Apache Spark 指南的 .NET](#) - `dotnet-spark/prod`
- [.NET 桌面指南](#) - `dotnet-desktop/prod`

其他產品標籤則是針對跨存放庫的區域而定義。

尋找指南的一個區段問題

.NET 指南很大，因此這些標籤會依指南章節進一步限制範圍。每個 .NET 協助工具子領域都會以 `/tech` 尾碼標示，並具有淺藍色背景。這些標籤中有許多適用於多個指南，而其他標籤則只適用於一個指南。依區域篩選之後，請新增下列其中一個標籤，以進一步限制問題的範圍。



針對特定版本戳記的問題會以前置詞標示 `:checkered_flag: Release:`，並具有深黃色的背景。

優先順序

所有優先順序標籤都是 `Pri` 後面接著一位數。數字越低則表示優先順序越高：

- Pri0-重大優先順序

安全性問題或合規性的法律要求。我們盡全力修正。

- Pri1-高優先順序

常見情節的基本項目。或頁面閱讀次數高的文章中的明顯錯誤。我們會在 P2 或 P3 工作之前先完成此優先順序的工作。

- Pri2-中優先順序

對於一般情節有幫助，但不會造成無法執行工作。如果能以快又簡單的方式修正，我們會執行這些工作，或在於相同的文章中處理 P1 問題時一併處理此優先順序的問題。

- Pri3-低優先順序

對於邊緣案例、一般情節的非重要性修正、頁面閱讀次數低的文章或過時的技術有幫助。不值得我們花時間處理，不過可接受社群貢獻。P3 問題在兩個月後若未處理可以關閉。

那麼其他標籤呢

內容小組使用許多其他標籤來管理不同分類的問題。如果您不在內容小組上，則可以忽略這些其他標籤。

專案

專案適用於規劃目的，其中具有較高優先順序的工作會透過工作流程看板自動化。專案只應該包含 GitHub 問題，而「非」提取要求。專案與里程碑的不同之處，在於里程碑只會包含提取要求。

我們透過兩種方式來使用專案：

- `Month YYYY` 專案類型：這些是每月工作計畫的工作流程看板。
 - 例如 [2020 年 7 月 \(英文\)](#)、[2020 年 8 月 \(英文\)](#) 等等。
- 長時間執行的 Epic：這些會在工作持續數個月時，用來管理工作達成目標的進度。
 - 範例：[.NET 5 Wave - 重新組織 \(英文\)](#)、[.NET 語言 \(.NET 5 wave\) \(英文\)](#) 等等。

里程碑

里程碑通常會遵循與專案相同的命名慣例 (`Month YYYY`)，但其與專案不同。我們會使用里程碑來追蹤已完成的工作。里程碑「不應該」包含問題 (潛在工作)，而只應包含提取要求。目前的里程碑會自動套用到新的提取要求。

.NET 文件的中繼資料和 Markdown 範本

2021/5/13 •

此 dotnet/docs 範本包含 Markdown 語法的範例，以及設定中繼資料的指導。

建立 Markdown 檔案時，應將包含的範本複製到新檔案中、依照如下指定來填寫中繼資料，並將以上 H1 標題設為文章標題。

中繼資料

所需的中繼資料區塊位於下列範例中繼資料區塊中：

```
---
title: [ARTICLE TITLE]
description: [usually a summary of your first paragraph. It gets displayed in search results, and can help
drive the correct traffic if well written.]
author: [GITHUB USERNAME]
ms.date: [CREATION/UPDATE DATE - mm/dd/yyyy]
---
# The H1 should not be the same as the title, but should describe the article contents
```

- 冒號 (:) 和中繼資料項目的值之間必須有空格。
- 值中的冒號 (例如標題) 會中斷中繼資料解析器。在此情況下，請使用雙引號來括住標題 (例如 `title: "Writing .NET Core console apps: An advanced step-by-step guide"`)。
- 標題：會出現在搜尋引擎結果中。標題不應與 H1 標題中的標題相同，且不應超過 60 個字元。
- 描述：會摘要文章的內容。通常會顯示於搜尋結果頁面，但不會用於搜尋排名。長度應為 115-145 個字元，包括空格。
- 作者：作者欄位應包含作者的 GitHub 使用者名稱。
- ms.date：上次重大更新的日期。如果您已檢閱並更新整篇文章，請在現有文章中對此進行更新。錯字或類似內容的小修正，不保證會更新。

其他中繼資料會附加至每篇文章，但我們通常會在資料夾層級套用大多數中繼資料值 (在 `docfx.json` 中指定)。

基本 Markdown、GFM 和特殊字元

您可透過 [Markdown 參考](#) 一文了解 Markdown、GitHub Flavored Markdown (GFM) 和 OPS 特定延伸模組的基本知識。

Markdown 使用 *、` 和 # 等特殊字元來格式化。如果您希望將其中一個字元包含在內容中，則必須執行以下兩項作業之一：

- 在特殊字元前加上一個反斜線以將其「逸出」(例如：針對 `*` 使用 `*`)。
- 為字元使用 [HTML 實體程式碼](#) (例如，`*` 對於 `*`)。

檔案名稱

檔案名稱使用下列規則：

- 只包含小寫字母、數字和連字號。
- 沒有空格或標點符號字元。檔案名稱中使用連字號來分隔文字和數字。

- 使用特定的動作動詞，例如 develop、buy、build、troubleshoot。沒有 -ing 字詞。
- 不包含 a、and、the、in、or 等短字。
- 必須是 Markdown 格式且使用 .md 副檔名。
- 保持檔案名稱合理簡短。它們是您文章 URL 的一部分。

標題

使用句型大寫。標題的第一個單字一律為大寫。

文字樣式

斜體

用於檔案、資料夾、路徑 (針對較長的項目，請將其拆成自己的行)、新字詞。

粗體

用於 UI 項目。

Code

用於內嵌程式碼、語言關鍵字，NuGet 套件名稱、命令列命令、資料庫資料表和資料列名稱，以及不希望可點選的 URL。

連結

如需錨點、內部連結、其他文件的連結、程式碼包含項目和外部連結的資訊，請參閱[連結](#)的一般文章。

.NET 文件小組使用下列慣例：

- 在大多數情況下，我們使用相對連結且不建議在連結中使用 `~/`，因為相對連結會在 GitHub 上的來源中解析。但是，每當連結到相依存放庫中的檔案時，我們會使用 `~/` 字元來提供路徑。由於相依存放庫中的檔案位於 GitHub 中不同位置，因此無論撰寫方式如何，連結都無法使用相對連結來正確解析。
- C# 語言規格和 Visual Basic 語言規格包含在 .NET 文件中，包含語言存放庫中的來源。Markdown 來源在 [csharp](#) 和 [vb](#) 存放庫中管理。

規格的連結，必須指向包含這些規格的來源目錄。若為 C#，其為 `~/_csharp/spec`，若為 VB，則為 `~/_vb/spec`，如下範例所示：

```
[C# Query Expressions](~/_csharp/spec/expressions.md#query-expressions)
```

API 的連結

組建系統具有一些延伸模組，可讓我們連結到 .NET API 而無需使用外部連結。請使用下列語法之一：

1. 自動連結： `<xref:UID>` 或 `<xref:UID?displayProperty=nameWithType>`

`displayProperty` 查詢參數會產生完整的連結文字。根據預設，連結文字只會顯示成員或型別名稱。

2. Markdown 連結： `[link text](xref:UID)`

用於您要自訂顯示的連結文字時。

範例：

- `<xref:System.String>` 會轉譯為 [String](#) (英文)
- `<xref:System.String?displayProperty=nameWithType>` 會轉譯為 [System.String](#) (英文)
- `[String class](xref:System.String)` 會轉譯為 [String 類別](#) (英文)

如需如何使用此標記法的詳細資訊，請參閱[使用交叉參考](#) (英文)。

UID 包含特殊字元 `、# 或 *，UID 值必須分別以 HTML 編碼為 `%60`、`%23` 和 `%2A`。有時候您會看到括號也會被編碼，但這並非必要。

範例：

- `System.Threading.Tasks.Task`1` 變成 `System.Threading.Tasks.Task%601`
- `System.Exception.#ctor` 變成 `System.Exception.%23ctor`
- `System.Lazy`1.#ctor(System.Threading.LazyThreadSafetyMode)` 變成 `System.Lazy%601.%23ctor%28System.Threading.LazyThreadSafetyMode%29`

您可以在 <https://xref.docs.microsoft.com/autocomplete> 中找到類型的 UID、成員多載清單或特定多載成員。查詢字元字串 `?text=*<type-member-name>*` 可識別出您要查看其 UID 的類型或成員。例如，<https://xref.docs.microsoft.com/autocomplete?text=string.format> 會擷取 [String.Format](#) 多載。該工具會在 UID 的任何部分中，搜尋所提供的 `text` 查詢參數。例如，您可以搜尋成員名稱 (ToString)、部分成員名稱 (ToStri)、類型和成員名稱 (Double.ToString) 等。

如果您在 UID 之後新增 * (或 `%2A`)，則連結代表多載頁面，而不是特定的 API。例如，當想要以一般方法連結到 [List<T>.BinarySearch](#) 方法 (機器翻譯) 頁面，而不是如 [List<T>.BinarySearch\(T, IComparer<T>\)](#) (機器翻譯) 的特定多載時，即可使用該方式。當成員未多載時，您也可以使用 * 連結到成員頁面；這可讓您無需在 UID 中包含參數清單。

要連結到特定方法多載，必須包含每個方法參數的完整類型名稱。例如，`<xref:System.DateTime.ToString>` 連結至無參數的 [DateTime.ToString](#) 方法，`<xref:System.DateTime.ToString(System.String,System.IFormatProvider)>` 則連結至 [DateTime.ToString\(String, IFormatProvider\)](#) 方法。

若要連結至泛型型別 (例如 [System.Collections.Generic.List<T>](#))，則可使用 ``(` (%60) 字元，其後跟隨泛型型別參數的數量。例如，<xref:System.Nullable%601> 連結至 System.Nullable<T> 型別，<xref:System.Func%602> 則連結到 System.Func<T,TResult> 委派。`

程式碼

包含程式碼的最佳方法，是包含工作範例中的程式碼片段。遵循[參與 .NET](#) 文章中的指令來建立範例。包含完整程式的程式碼片段，可確保所有程式碼都透過我們的持續整合 (CI) 系統執行。但是，如果需要顯示導致編譯時間或執行階段錯誤的內容，則可以使用內嵌程式碼區塊。

如需有關用於在文件中顯示程式碼之 Markdown 語法的詳細資訊，請參閱[如何在文件中包含程式碼](#)。

影像

靜態影像或動畫 GIF

```
![this is the alt text](../images/Logo_DotNet.png)
```

連結的影像

```
![alt text for linked image](../images/Logo_DotNet.png)(https://dot.net)
```

影片

目前，您可以使用下列語法嵌入 Channel 9 和 YouTube 影片：

Channel 9

```
> [!VIDEO <channel9_video_link>]
```

若要取得影片的正确網址，請選取影片框架下方的 [嵌入] 索引標籤，然後從 `<iframe>` 項目中複製網址。例如：

```
> [!VIDEO https://channel9.msdn.com/Blogs/dotnet/NET-Core-20-Released/player]
```

YouTube

若要取得影片的正确網址，請以右鍵按一下影片，選取 [複製內嵌程式碼]，然後從 `<iframe>` 項目複製網址。

```
> [!VIDEO <youtube_video_link>]
```

例如：

```
> [!VIDEO https://www.youtube.com/embed/Q2mMbjw6cLA]
```

docs.microsoft 延伸模組

docs.microsoft 為 GitHub Flavored Markdown 提供一些額外的延伸模組。

檢查清單

自訂樣式適用於清單。您可以轉譯具有綠色核取記號的清單。

```
> [!div class="checklist"]
> * How to create a .NET Core app
> * How to add a reference to the Microsoft.XmlSerializer.Generator package
> * How to edit your MyApp.csproj to add dependencies
> * How to add a class and an XmlSerializer
> * How to build and run the application
```

這會呈現為：

- 如何建立 .NET Core 應用程式
- 如何將參考新增至 Microsoft.XmlSerializer.Generator 套件
- 如何編輯您的 MyApp.csproj 以新增相依性
- 如何新增類別和 XmlSerializer
- 如何建置並執行應用程式

您可以在 [.NET Core 文件](#) 中查看檢查清單的範例。

按鈕

按鈕連結：

```
> [!div class="button"]
> [button links](dotnet-contribute.md)
```

這會呈現為：



您可以在 [Visual Studio 文件](#) 中查看按鈕的範例。

逐步

```
>[!div class="step-by-step"]  
> [Pre](../docs/csharp/expression-trees-interpreting.md)  
> [Next](../docs/csharp/expression-trees-translating.md)
```

您可以在 [C# 指南](#) 中查看動作中的逐步範例。

語態和語調方針

2021/5/13 •

閱讀 .NET 文件的人形形色色，包括 IT 專業人員和開發人員，其想要了解並在日常工作中使用 .NET。您的目標是建立可協助讀者展開旅程的絕佳文件。我們的方針有助於達成此目標。我們的風格指南包含下列建議：

使用交談語調

下列段落是以交談的風格撰寫。而後續的段落則是以較具學術風格的方式撰寫。

適當的風格

我們希望文件使用交談語調。當您閱讀任何教學課程或說明時，應該感覺彷彿是在與作者交談。對您而言，這應該是非正式交談且提供資訊的體驗。讀者應該感覺彷彿是聽到作者向他們說明概念。

不適當的風格

讀者可能會發現交談風格與處理技術性主題的學術風格之間有所對比。這些資源很實用，但作者以與我們文件不同的風格來撰寫這些文章。當讀者在閱讀學術期刊時，他們會發現書寫語調和風格相當不同。感覺起來彷彿是在閱讀乏味主題的無趣陳述。

以第二人稱撰寫

下列段落使用第二人稱。而後續的段落則使用第三人稱。請以第二人稱來撰寫。

適當的風格

當您撰寫文章時，請彷彿是自己直接與讀者對話一般。請經常使用第二人稱 (如同這兩句所示)。第二人稱不一定表示使用「您」一字。請以讀者為直接撰寫對象。並撰寫祈使句。告訴讀者您想要他們學到的內容。

不適當的風格

作者也可選擇以第三人稱來撰寫。在該模型中，作者必須尋找一些用來指涉讀者的代名詞或名詞。讀者通常可能會發現此第三人稱風格互動性較差，而且讀起來較無趣。

使用主動語態

請以主動語態撰寫您的文章。主動語態表示句子的主詞會執行該句子的動作 (動詞)。它與被動語態相反，在被動語態中，句子排列方式是讓句子的主詞被動。請比較這兩個範例：

編譯器將原始程式碼轉換成可執行檔。

原始程式碼被編譯器轉換成可執行檔。

第一句使用主動語態。第二句則是以被動語態來撰寫 (這兩句針對每種風格提供另一個範例)。

建議使用主動語態，因為更容易閱讀。被動語態可能較難閱讀。

在撰寫時考量對詞彙所知有限的讀者

您想要透過文章接觸國際讀者。許多讀者可能都不是以英文作為母語，且可能尚未學習到您所掌握的英文詞彙。

不過，您仍以技術專業人員為撰寫對象。您可以假設讀者具備程式設計知識，且理解程式設計術語的特定詞彙。物件導向程式設計、類別與物件、函式與方法都是熟悉的術語。不過，並非所有閱讀您文章的人都有正式電腦科技學位。若使用「等冪」之類的術語時，您可能需要加以定義，例如：

Close() 方法是等幕的，這表示您可以呼叫它多次，其效果會相當於呼叫它一次。

避免使用未來式

在某些非英文的語言中，未來式的概念不同於英文。使用未來式可能會使您的文件更難閱讀。此外，使用未來式時，首當其衝的問題是發生時機。因此，如果您指出「學習 PowerShell 將有益於您」，對讀者而言，首當其衝的問題是何時才會受益？相反地，直接指出「學習 PowerShell 有益於您」即可。

這是什麼，又有何作用？

當您向讀者介紹新概念時，請先定義概念，再說明概念的實用之處。讀者必須了解這是什麼，才能了解其優點(或缺點)。

.NET 文件存放庫的提取要求檢閱程序

2021/5/13 •

某些存放庫 (包括 .NET 存放庫) 啟用「PR 合併」Webhook, 這種 Webhook 會自動合併較小的 PR。本文說明針對那些存放庫的 PR 檢閱程序。PR 檢閱程序是基於這些目標而設計：

- 發佈來自我們小組、產品小組成員及社群成員的高品質內容。
- 以一致的方式為作者提供可採取動作的即時意見反應。
- 促進作者及檢閱者之間的討論。

隨著小組持續創新且平台持續成熟, 這些程序將會繼續進化。

檢閱者

其中一個內容小組成員會檢閱每一個 PR。內容小組成員可以要求特定產品小組成員進行檢閱, 以確認技術上的正確性。內容小組會使用 GitHub 的程式碼擁有權功能來自動要求內容小組成員進行檢閱。作為此程序的一部分, 檢閱者可以標記其他小組成員以檢閱內部 PR。小組成員會在相同的佇列中看見來自小組成員和社群成員的所要求檢閱。

社群成員也可以檢閱 PR 並提供意見反應。不過, 在該 PR 被合併之前, 必須至少由一個核心內容小組的成員進行核准。

檢閱程序

會根據 PR 檢閱下列三個路徑中的其中一個：

- **小型 PR**：小型 PR 為單一錯誤修正：錯字、無效連結、小型程式碼變更或類似的變更。
- **主要貢獻**：主要貢獻為針對現有文章或新文章的大量編輯, 或是針對許多文章進行編輯。
- **正在進行的工作** - 作者可透過開啟「草稿」提取要求, 以建立標記為尚未準備好進行檢閱的 PR。

「參與者授權合約」(CLA) 機器人所使用的處理方式是一個可區別「小型」與「大型」貢獻的良好指導方針。不需要簽署 CLA 的 PR, 通常是「小型」的。需要 CLA 的 PR 則通常是「大型」的。

小型 PR

小型 PR 中的變更會針對正確性進行檢閱, 並使用檢閱網站上的組建進行檢查。由於這些 PR 較小, 它們並不會觸發針對整篇文章的檢閱。

檢閱者可能會注意到其他能改善文章的小型變更。若那些變更的幅度也相當小, 請以檢閱註解的方式建議它們。若建議的變更會觸發較大規模的檢閱, 請提出問題並於未來解決它們。

大型變更

大型 PR 需要進行更徹底的檢閱。會檢查下列所有項目：

- 範例程式碼必須包含在 PR、來源, 以及可下載的 zip 檔案中。
- 範例程式碼必須編譯並正常運作。
- 文章必須清楚地為讀者描述目標, 並達成那些目標。
- 文章符合[樣式和文法指導方針](#), 並且依循我們的[語態和語調原則](#)。
- 所有連結都必須能正確解析。

內容小組成員將會檢閱 PR, 並搭配註解提交檢閱。若僅要求次要變更, 小組成員可以透過意見反應「核准」該 PR。作者可以接著處理意見反應內容並合併 PR。大部分的檢閱都會要求變更, 而當那些變更都被執行之後, 檢閱者便會再次檢閱。

若編輯需要技術性檢閱，內容小組檢閱者將會要求適當的產品小組成員進行檢閱。

檢閱草稿提取要求

您可以會想要在流程初期便取得意見反應。開啟草稿 PR，並新增要求早期檢閱的註解。這些早期檢閱的重點在於文章的結構：大綱、整體內容及範例。這些檢閱並不會針對文法或連結進行徹底的檢查。

說明建議

GitHub 可讓您在類型為 `suggestion` 的三接續符號區塊中輸入註解，這些會顯示為差異並可透過按一下按鈕來合併。在較短的行上，GitHub 能有效地反白顯示變更。在較長的行 (例如一行文字中的長段落) 上，GitHub 不會反白顯示變更。針對較長的行輸入建議時，請注意您的變更是否清楚地反白顯示。若變更未反白顯示，請納入建議區塊外的註解並說明變更為何。如果沒有說明，後續的檢閱者或 PR 作者通常會花很多時間來找出變更的內容。

回應檢閱

作者可以更新 PR 以回應註解，並在每個已處理的註解上標記 "+1" 反應，以指出該註解已處理完畢。若作者不同意註解，或是以不同的方式處理該註解，作者便會新增註解以解釋其變更。

作者會在註解中對原始檢閱者進行 @-mentions 以要求新的檢閱。

預期回應時間

內容小組成員通常會在兩個工作日以內對新的 PR 進行檢閱。若檢閱時間較長，其中一個小組成員將會加入註解以確認該 PR 並設立預期時程。

在檢閱者提交檢閱之後，作者應該嘗試在一週以內對註解做出回應。志工可能會因為自己工作或生活上的其他承諾，而無法達成這些預期的時程。在那些情況下，小組成員會詢問社群作者是否能更新該 PR。若不行，小組成員將會代替他們更新並合併該 PR。

如何在檔中參考 .NET

2021/5/13 •

建議術語

在 [doc 標題] 和 [first] 標題 (h1)：

- 「.NET」(沒有提及 5 或核心)

文章文字中的第一篇參考：

- 「.NET」適用於核心/5 + 的檔，並將其呈現給 .NET 新手的人員。
- ".NET 5 (和 .NET Core) 和更新版本" (如果需要) 強調核心/5 + 作為特定的 .NET 執行。

後續參考：

- ".NET" 適用於提供核心/5 + 的檔給 .NET 的新手，或 Core/5 + 的參考在內容中很清楚。
- 「.NET 5 和更新版本」，需要用來區別核心/5 + 與其他 .NET 的實現。

注意：

- 其他我們考慮的選項：
 - 「.NET 5 (和 .NET Core 2.1/3.1) 和更新版本」- 核心版本號碼似乎不必要，且片語的長度超過其需要的長度。
 - 「.NET 5 和更新版本 (先前稱為 .NET Core)」-- 可能會導致「5 和更新版本」的混淆不會排除 2.1-3.1。
 - 「.NET 5 和更新版本 (包括 .NET Core 2.1-3.1)」- 類似的問題在 2.1-3.1 音效中，它們應該會在「更新版本」中
 - 「.NET 5 +」是「.NET 5 和更新版本」的較短替代方式，但不是普遍瞭解的。
- 在版本 6 和更新版本發行之後，「.NET 5 和更新版本」仍會維持正確的描述。但最終會將「.NET」視為 5 +，而且我們可以在某些內容中排除「5 和更新版本」。

範例

影響相容性的變更

| 「.NET 5」 | 「.NET 5 和更新版本」 |
|---|--|
| 縱觀其歷程記錄，.NET 一直嘗試維護 .NET 各版本之間以及各種變體之間的相容性。在 .NET Core 中仍是如此。雖然 .NET Core 可以視為是獨立於 .NET Framework 之外的新技術，但是有兩個主要因素限制了 .NET Core 從 .NET Framework 分離的能力： | 在整個歷程記錄中，.NET 已嘗試維持從版本到版本的高階相容性，以及跨 .NET 的各項執行。相較于 .NET Framework，雖然 .NET 5 (和 .NET Core) 和更新版本可視為新的技術，但是有兩個主要因素會限制這項 .NET 的執行能力與 .NET Framework 相互分離： |
| .NET Standard 程式庫專案可讓開發人員建立以 .NET Core 和 .NET Framework 共用的通用 Api 為目標的程式庫。 | .NET Standard 程式庫專案可讓開發人員建立以 .NET Framework 和 .NET 5 (和 .NET Core) 和更新版本所共用的通用 Api 為目標的程式庫。 |
| 本文不是 .NET Framework 和 .NET Core 之間的重大變更完整清單。 | 與 .NET Framework 相較之下，這篇文章並不是 .NET 5 (和 .NET Core) 和更新版本的重大變更的完整清單。 |
| 下列 Api 一律會 <code>PlatformNotSupportedException</code> 在 .NET Core 上擲回。 | 下列 Api 一律會 <code>PlatformNotSupportedException</code> 在 .NET 5 (和 .NET Core) 和更新版本上擲回。 |

在 Windows 上安裝 .NET

| 【.NET 5 ㄟ | |
|---|---|
| 標題/h1: 在 Windows 上安裝 .NET Core | 標題/h1: 在 Windows 上安裝 .NET |
| 在本文中, 您將瞭解如何在 Windows 上安裝 .NET Core。
.NET Core 是由執行時間和 SDK 所組成。執行時間是用來執行 .NET Core 應用程式, 而且不一定會包含在應用程式中。
SDK 是用來建立 .NET Core 應用程式和程式庫。 .NET Core 執行時間一律會與 SDK 一起安裝。 | 在本文中, 您將瞭解如何在 Windows 上安裝 .NET 5 (和 .NET Core) 和更新版本。 .NET 是由執行時間和 SDK 所組成。執行時間是用來執行 .NET 應用程式, 且不一定會包含在應用程式中。 SDK 是用來建立 .NET 應用程式和程式庫。 .NET 執行時間一律會與 SDK 一起安裝。 |

如何檢查是否已安裝 .NET

| 【.NET 5 ㄟ | |
|--|---|
| Title/h1: 如何檢查是否已安裝 .NET Core | Title/h1: 如何檢查是否已安裝 .NET |
| 本文會教您如何檢查電腦上已安裝的 .NET Core 執行時間和 SDK 版本。如果您有整合式開發環境, 例如 Visual Studio 或 Visual Studio for Mac, 可能已經安裝 .NET core。 | 本文將告訴您如何檢查電腦上是否已安裝適用於 .NET 5 (和 .NET Core) 和更新版本的執行時間和 SDK 版本。如果您有整合式開發環境 (例如 Visual Studio 或 Visual Studio for Mac), 則執行時間和 SDK 可能已安裝。 |

教學課程: 使用 Visual Studio 建立 .NET 主控台應用程式

| 【.NET 5 ㄟ | |
|--|---|
| 標題/h1: 教學課程: 使用 Visual Studio 建立 .NET Core 主控台應用程式 | 標題/h1: 教學課程: 使用 Visual Studio 建立 .NET 主控台應用程式 |
| 本教學課程說明如何在 Visual Studio 2019 中建立和執行 .NET Core 主控台應用程式。 | 本教學課程說明如何在 Visual Studio 2019 中建立和執行 .NET 主控台應用程式。 |
| 必要條件: 已安裝 .NET Core 跨平臺開發工作負載的 Visual Studio 2019 16.6 版或更新版本。當您選取此工作負載時, 會自動安裝 .NET Core 3.1 SDK。 | 已安裝 .NET 跨平臺開發工作負載的 PrerequisitesVisual Studio 2019 16.6 版或更新版本。當您選取此工作負載時, 會自動安裝 .NET SDK。 |

教學課程: 建立專案範本

| 【.NET 5 ㄟ | |
|---|--|
| 標題/h1: 教學課程: 建立專案範本 | 標題/h1: 教學課程: 建立專案範本 |
| 透過 .NET Core, 您可以建立及部署能產生專案、檔案, 甚至是資源的範本。 | 您可以使用 .NET 來建立和部署範本, 以產生專案、檔案, 甚至是資源。 |
| 必要條件: .NET Core 2.2 SDK 或更新版本。 | 必要條件: .NET Core 2.2 SDK 或更新版本 (包括 .NET 5 和更新版本)。 |

.NET SDK 總覽

| 【.NET 5 ㄟ | |
|-------------------------|--------------------|
| 標題/h1: .NET Core SDK 總覽 | 標題/h1: .NET SDK 總覽 |
| 文字中的 "SDK" | 無變更 |

.NET CLI 總覽

| | |
|---|---|
| 「.NET 5」 | |
| 標題/h1: .NET Core CLI 總覽 | 標題/h1: .NET CLI 總覽 |
| .NET Core 命令列介面 (CLI) 是用於開發、建立、執行及發佈 .NET Core 應用程式的跨平臺工具鏈。 | .NET 命令列介面 (CLI) 是用於開發、建立、執行和發佈 .NET 應用程式的跨平臺工具鏈。 |

.NET 應用程式發行總覽

| | |
|---|--|
| 「.NET 5」 | |
| 標題/h1: .NET Core 應用程式發行總覽 | Title/h1: .NET 應用程式發行總覽 |
| 您使用 .NET Core 建立的應用程式可以在兩種不同的模式下發行, 而此模式會影響使用者執行您應用程式的方式。 | 您使用 .NET 5 建立的應用程式 (和 .NET Core) 和更新版本可以在兩種不同的模式下發行, 而此模式會影響使用者執行您應用程式的方式。 |
| 將您的應用程式發佈為獨立應用程式時, 會產生包含 .NET Core 執行時間和程式庫的應用程式, 以及您的應用程式及其相依性。應用程式的使用者可以在未安裝 .NET Core 執行時間的電腦上執行它。 | 將您的應用程式發佈為獨立應用程式, 會產生包含 .NET 執行時間和程式庫的應用程式, 以及您的應用程式及其相依性。應用程式的使用者可以在未安裝 .NET 執行時間的電腦上執行它。 |

dotnet new

| | |
|--|--|
| 「.NET 5」 | |
| 本文  .net CORE 2.1 SDK 和更新版本 | 本文  .net CORE 2.1 sdk 和更新版本 (包括 .NET 5 SDK 和更新版本) |
| <code>--dry-run</code> 顯示當指定的命令執行時, 如果指定的命令會產生範本, 會發生什麼事的摘要。自 .NET Core 2.2 SDK 起提供。 | (保持「自 .NET Core 2.2 SDK 起可供使用」的變更) |

API 參考檔中的「適用於」連結文字

| | |
|---|--|
| 「.NET 5」 | |
| <exception cref="T:System.PlatformNotSupportedException">僅限 .NET Core: 在所有情況下</exception> | <exception cref="T:System.PlatformNotSupportedException">.NET 5 (和 .NET Core) 和更新版本: 在所有情況下都是如此。</exception> |

如何參與 PowerShell 文件

2021/5/13 •

參與者指南是一系列文章，其說明在 Microsoft 建立文件時所使用的工具和程序。某些指南涵蓋了所有發佈至 docs.microsoft.com 的文件集通用資訊。某些指南是關於如何撰寫 PowerShell 文件的特定內容。

您可在此集中式參與者指南中找到一般文章。[PowerShell 參與者指南](#)是 PowerShell 文件的可用社群區段。

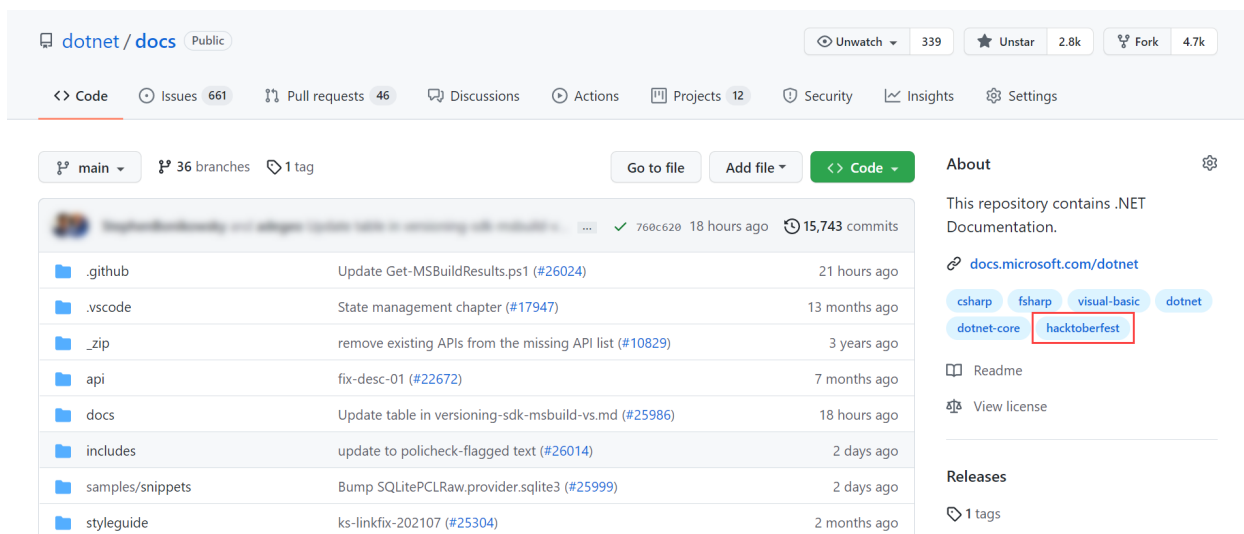
Hacktoberfest 和 Microsoft Docs

2021/9/25 •

Hacktoberfest 是在10月期間舉辦的每年全球活動。此事件鼓勵開放原始碼開發人員透過提取要求來參與存放庫，(PR)。GitHub 裝載許多 Microsoft Docs 的開放原始碼存放庫，這些存放庫全都參與docs.microsoft.com內容。某些存放庫會主動參與 Hacktoberfest 事件。在本文中，您將瞭解如何探索哪些存放庫接受 Pr，以及您可以預期的參與者為何。

尋找存放庫

若要探索檔存放庫是否參與 Hacktoberfest，您會在專案上看到 Hacktoberfest 主題。



若要篩選所有具有 **hacktoberfest** 主題的 Microsoft Docs 和 .net 存放庫，請參閱 [GitHub 主題: hacktoberfest](#)。

或者，儲存機制也可以選擇改用 **Hacktoberfest** 標籤。這個標籤很方便篩選問題。如需詳細資訊，請參閱 [依標籤篩選問題和提取要求](#)。

參與

若要參與開放原始碼檔存放庫，您必須先設定您的帳戶來參與 Microsoft Docs。如果您從未完成此程式，請先[註冊 GitHub 帳戶](#)來開始。

設定您的帳戶之後，請先閱讀並遵循您要參與的存放庫根目錄中的 *CONTRIBUTING.md* 檔案。這些檔案是參與時的指南。以下是一些熱門檔存放庫的一些範例參與者指南：

- [參與 .NET 文件存放庫](#)
- [參與 Microsoft Azure 檔](#)

除了參與的 Markdown 檔之外，如果存放庫有 *CODE_OF_CONDUCT.md* 檔案，則必須遵守該社區的預期行為。同樣地，以下是一些常見的範例：

- [.NET 檔: 管理辦法](#)
- [Azure 檔: 管理辦法](#)

選擇問題

若要在參與的存放庫中找出要處理的問題，請篩選 **up-for-grabs** 或 **help-wanted** GitHub 標籤的問題。雖然您可以解決其他問題，但更容易專注於具有妥善定義範圍且獨立的問題。除了檔存放庫之外，您還可以針對初學者

使用下列網站：

- [適用於初學者的絕佳](#)
- [見者有份](#)
- [僅限第一個計時器](#)

如需詳細資訊，請參閱 [Hacktoberfest: 初學者](#)。

品質期望

若要成功地參與開放原始碼檔案存放庫，請 [建立有意義且具影響力的 PR](#)。下列官方 Hacktoberfest 網站的範例視為 **低品質的投稿**：

- (自動化的 Pr，例如，編寫開啟 Pr 的腳本，以移除空白字元、修正輸入錯誤，或將影像優化)。
- Pr 為干擾性 (例如，將其他人的分支或認可，以及進行 PR)。
- 專案維護程式視為阻礙的 Pr，與協助。
- 這項提交內容顯然會嘗試在10月1日的 PR 計數。

最後一個修正錯誤的 PR 是正常的，但移除偏離空白字元的五個 Pr 則否。

如需詳細資訊，請參閱 [Hacktoberfest: 品質標準](#)。

開啟 PR

PR 提供便利的方式讓參與者提出一組變更。開啟 PR 時，請在原始批註中指定它要貢獻給 *hacktoberfest*。成功的 Pr 具有下列一般特性：

- PR 會增加價值。
- 參與者會願意意見反應。
- 預定的變更清清楚楚。
- 這些變更與現有的問題有關。

如果您在沒有對應問題的情況下建議 PR，請先建立問題。如需詳細資訊，請參閱[GitHub:關於提取要求](#)。

另請參閱

- [GitHub 帳戶設定](#)
- [適用於 Docs 的 Git 和 GitHub 基本資訊](#)
- [官方 Hacktoberfest 網站](#)

其他的 Git 和 GitHub 資源

2021/5/13 •

如果您不熟悉 Git 或 GitHub, 這些資源能有助於您學習、提高生產力, 或是回答您的問題。

Git 原始檔控制資源

- [Git 基本概念](#) (英文): 這涵蓋 Git 運作方式的基本概觀。
- [專業 Git 電子書 \(網頁版\)](#) (英文): 這是完整的 Git 參考資料, 以 HTML 格式提供。
- [專業 Git 電子書 \(PDF\)](#) (英文): 內容與上述連結相同, 以 PDF 形式提供。
- [Codecademy 的 Learn Git 課程](#) (英文): 取得 Codecademy 提供的教學課程。
- [Code School 的 Try Git 課程 \(位於 Pluralsight\)](#) (英文): Code School 的 Git 課程 (位於 Pluralsight)。
- [Udacity 的 Git 與 Github 課程](#) (英文): Udacity 的 Git 與 Github 課程。

GitHub 資源

- [15 分鐘互動式入門](#) (英文): 這是線上 Git 教學課程。課程會向您展示 Git 的基本概念。
- [速查表](#) (英文): 常見的 GitHub 命令與工作流程的快速參考。
- [GitHub 指南](#) (英文): GitHub 文件的首頁。
- [GitHub 學習資源](#) (英文): 其他實用的 GitHub 資源。
- [GitHub 訓練服務](#) (英文): GitHub 的教學課程和訓練供應項目清單。
- [詞彙](#) (英文): 方便的 Git 與 GitHub 術語的詞彙。
- [GitHub 學生開發人員套件](#) (英文): 可讓學生免費存取最佳開發人員工具。