環境篇

PicGO 實現圖床的圖片上傳功能搭配 Github

1. GitHUB 設置倉儲
2. Picgo 套件下載
    1. 至 Github Release 網站下載
    2. 強制安裝與點擊右擊上方小圖示啟動它，選擇打開主窗口
        - picgo 設置 選擇顯示的圖床只需 GitHub
        - 圖床設置 (Github)
            - 設定倉庫名稱 username/repo
            - 設定分支名稱 main
            - 設定 Token
            - 設定存儲路徑
            - 設定自定義域名: https://xxx.xxx

        設定完成點擊 確定 與 設為預設圖床
3. VSCode 外掛套件下載
    1. 點擊左側 延伸模組 安裝 picgo 套件
    2. 強制安裝與點擊右擊上方小圖示啟動它，選擇打開主窗口
        - picgo 設置

Homebrew 安裝

使用指令 `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"` 安裝 Homebrew.

安裝 Wget

使用指令如下:

```
brew install wget
```

3. Brew 指令基本操作

1. `brew services list` # 檢查用 `brew` 安裝了哪些服務 ·
2. `brew services start/restart/stop php@7.4` # 開啟/重啟/關閉 php 7.4 服務 ·
3. `brew search php` # 搜尋可用指定套件名稱來源

`Apache Web Service` 的服務

1. 假設 `Mac OS <=11` 時:

```
sudo nano /etc/apache2/httpd.conf  # 將下列文字前#移除後，儲存檔案，重新啟動
```

```
LoadModule php7_module libexec/apache2/libphp7.so
```

```
sudo apachectl restart
```

2. 假設 Mac OS >11 時:
    1. 先停掉 Apache 預設的服務 sudo apachectl stop
    2. 關閉原生 httpd 服務 sudo launchctl unload -w /System/Library/LaunchDaemons/org.apache.httpd.plist 2>/dev/null
    3. 再使用 brew 安裝 httpd brew install httpd

    ```
    DocumentRoot is /usr/local/var/www
    :
    /usr/local/etc/httpd/httpd.conf to 8080
    /usr/local/etc/httpd/extra/httpd-ssl.conf to 8443
    ```

    4. 啟動 httpd 服務 brew services start httpd

PHP 環境

1. 檢查目前 php 環境設定:
    - php --ini
    - php -version
2. 搜尋可用 php 資源 brew serach php
3. 安裝指定 php 版本 brew install shivammathur/php/php@8.3
4. 添加路徑 PATH 設定

```
echo 'export PATH="/usr/local/opt/php@8.3/bin:$PATH"' >> ~/.zshrc
echo 'export PATH="/usr/local/opt/php@8.3/sbin:$PATH"' >> ~/.zshrc
```

```
To enable PHP in Apache add the following to httpd.conf and restart Apache:
    LoadModule php_module /usr/local/opt/php@8.3/lib/httpd/modules/libphp.so

    <FilesMatch \.php$>
        SetHandler application/x-httpd-php
    </FilesMatch>

Finally, check DirectoryIndex includes index.php
    DirectoryIndex index.php index.html

The php.ini and php-fpm.ini file can be found in:
    /usr/local/etc/php/8.3/

php@8.3 is keg-only, which means it was not symlinked into /usr/local,
because this is an alternate version of another formula.
```

```
If you need to have php@8.3 first in your PATH, run:
  echo 'export PATH="/usr/local/opt/php@8.3/bin:$PATH"' >> ~/.zshrc
  echo 'export PATH="/usr/local/opt/php@8.3/sbin:$PATH"' >> ~/.zshrc

For compilers to find php@8.3 you may need to set:
  export LDFLAGS="-L/usr/local/opt/php@8.3/lib"
  export CPPFLAGS="-I/usr/local/opt/php@8.3/include"


To restart shivammathur/php/php@8.3 after an upgrade:
  brew services restart shivammathur/php/php@8.3
Or, if you don't want/need a background service you can just run:
  /usr/local/opt/php@8.3/sbin/php-fpm --nodaemonize
==> Summary
🍺  /usr/local/Cellar/php@8.3/8.3.0: 528 files, 86.6MB
==> Running `brew cleanup php@8.3`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
```

XAMPP

1. XAMPP官方下載套件與安裝 ·
2. 虛擬主機設定
    1. 位置與 `xampp` 套件安裝位置相同時

虛擬主機名稱	URL	位置
blog.test	http://blog.test:6080	/Applications/XAMPP/xamppfiles/htdocs/blog
event.test	http://event.test:6080	/Applications/XAMPP/xamppfiles/htdocs/event
gallery.test	http://gallery.test:6080	/Applications/XAMPP/xamppfiles/htdocs/gallery
church.test	http://church.test:6080	/Applications/XAMPP/xamppfiles/htdocs/church

`nano /Applications/XAMPP/etc/extra/httpd-vhosts.conf`

```
:
<VirtualHost *:6080>
    #ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot "/Applications/XAMPP/htdocs/blog"
    ServerName blog.test

    <Directory "/Applications/XAMPP/htdocs/blog">
       Options Indexes FollowSymLinks Includes ExecCGI
       AllowOverride All
       Require all granted
    </Directory>

    #ServerAlias www.dummy-host.example.com
    #ErrorLog "logs/dummy-host.example.com-error_log"
    #CustomLog "logs/dummy-host.example.com-access_log" common
</VirtualHost>
```html
```
```

2. 位置與 xampp 套件安裝位置不相同時（使用者自建目錄 Sites）

虛擬主機名稱	URL	位置
blog.test	http://blog.test:6080	~/Sites/blog
event.test	http://event.test:6080	~/Sites/event
gallery.test	http://gallery.test:6080	~/Sites/gallery
church.test	http://church.test:6080	~/Sites/church

nano /Applications/XAMPP/etc/httpd.conf

```
:
User your_name
Group your_group
```

nano /Applications/XAMPP/etc/extra/httpd-vhosts.conf

```
<VirtualHost *:6080>
    #ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot "/Users/yourname/Sites/blog"
    ServerName blog.test

    <Directory "/Users/yourname/Sites/blog">
       Options Indexes FollowSymLinks Includes ExecCGI
       AllowOverride All
       Require all granted
    </Directory>

    #ServerAlias www.dummy-host.example.com
    #ErrorLog "logs/dummy-host.example.com-error_log"
    #CustomLog "logs/dummy-host.example.com-access_log" common
</VirtualHost>
```html
```

3. 使用者家目錄之設置: (http://localhost/~username)

#啟用模組和配置 #您需要啟用（取消註釋該行）並修改/更新 httpd.conf 檔案中的配置。 #要啟用模組，首先，您需要找到要啟用的模組。 #用於調 control + w 出搜尋功能，查詢以下模組並確保取消註釋。（刪除#每行前面的。）

sudo nano /Applications/XAMPP/etc/httpd.conf

```
:
LoadModule authn_core_module lib/httpd/modules/mod_authn_core.so
LoadModule authz_host_module lib/httpd/modules/mod_authz_host.so
LoadModule userdir_module lib/httpd/modules/mod_userdir.so
LoadModule include_module lib/httpd/modules/mod_include.so
LoadModule rewrite_module lib/httpd/modules/mod_rewrite.so
# User home directories
Include etc/extra/httpd-userdir.conf
```
```

`sudo nano /Applications/XAMPP/etc/extra/httpd-userdir.conf`

```
# UserDir: The name of the directory that is appended onto a user's home
:
UserDir Sites
#
# Control access to UserDir    directories.  The following is an    example
# for a site where these    directories are restricted to    read-only.
#
<Directory "/Users/allen/Sites/">
    AllowOverride all
    Options Indexes FollowSymLinks    Multiviews
    MultiviewsMatch Any
    Require all granted
</Directory>
```

#添加虛擬主機 `localhost` 設定 `sudo nano /Applications/XAMPP/etc/extra/httpd-vhosts.conf`

```
<VirtualHost *:6080>
   #ServerAdmin webmaster@dummy-host.example.com
   DocumentRoot "/Users/allen/Sites"
   ServerName localhost

   <Directory "/Users/allen/Sites">
     Options Indexes FollowSymLinks Includes ExecCGI
     AllowOverride All
     Require all granted
   </Directory>

   #ServerAlias www.dummy-host.example.com
   #ErrorLog "logs/dummy-host.example.com-error_log"
   #CustomLog "logs/dummy-host.example.com-access_log" common
</VirtualHost>
```

4. 修改 `php` 為 `XAMPP` 套件同樣的版本

`nano ~/.zshrc`

```
#export PATH="/usr/local/opt/php@8.3/bin:$PATH"
#export PATH="/usr/local/opt/php@8.3/sbin:$PATH"
export PATH="/Applications/XAMPP/xamppfiles/bin:$PATH"
```

> 重新登入後，才會生效

DNSmasq

`Chrome 63` 強制所有 `.dev` 域都使用 `SSL`，上面設置的示例虛擬主機中，定義了一個 ServerNameof `blog.test。默認情況下，這不會解析到您的本地計算機， 但能夠為開發目的設置各種虛擬主機通常非常有用。

可以通過手動向 `/etc/host` 添加項目來完成此操作，或者您可以安裝和配置 `Dnsmasq` 以自動處理通配符 `*.test` 名稱並將它們全部轉發到 `localhost`(127.0.0.1)。

1. 使用 `brew` 安裝 dnsmasq `brew install dnsmasq`

2. 設置 `*.test` 主機： `echo 'address=/.test/127.0.0.1' > /usr/local/etc/dnsmasq.conf`
3. 啟動它並確保它在將來重新啟動時自動啟動： `sudo brew services start dnsmasq`
4. 最後，將其添加到解析器中：

```
sudo mkdir -v /etc/resolver
sudo bash -c 'echo "nameserver 127.0.0.1" > /etc/resolver/test'
```

5. `.test` 現在您可以通過 `ping` 一些虛假名稱來測試它： `ping bogus.test`

XAMPP 建置

1. 安裝 XAMPP官方下載套件與安裝 ·
2. 虛擬主機 .vs. URL .vs. 位置
    1. 位置與 `xampp` 套件安裝位置相同時

虛擬主機名稱	URL	位置
blog.test	http://blog.test:6080	/Applications/XAMPP/xamppfiles/htdocs/blog
event.test	http://event.test:6080	/Applications/XAMPP/xamppfiles/htdocs/event
gallery.test	http://gallery.test:6080	/Applications/XAMPP/xamppfiles/htdocs/gallery
church.test	http://church.test:6080	/Applications/XAMPP/xamppfiles/htdocs/church

`nano /Applications/XAMPP/xamppfiles/etc/extra/httpd-vhosts.conf`

```
:
<VirtualHost *:6080>
    DocumentRoot "/Applications/XAMPP/xamppfiles/htdocs/blog"
    ServerName blog.test
    #以下忽略，因 httpd.conf 已有設定該內容
    #<Directory "/Applications/XAMPP/htdocs/blog">
       #Options Indexes FollowSymLinks Includes ExecCGI
       #AllowOverride All
       #Require all granted
    #</Directory>

    #ServerAlias www.dummy-host.example.com
    #ErrorLog "logs/dummy-host.example.com-error_log"
    #CustomLog "logs/dummy-host.example.com-access_log" common
</VirtualHost>
```html
```

    2. 位置與 `xampp` 套件安裝位置不相同時（使用者自建目錄 `~/Sites`）

虛擬主機名稱	URL	位置
blog.test	http://blog.test:6080	~/Sites/blog
event.test	http://event.test:6080	~/Sites/event
```

虛擬主機名稱	URL	位置
gallery.test	http://gallery.test:6080	~/Sites/gallery
church.test	http://church.test:6080	~/Sites/church

nano /Applications/XAMPP/xamppfiles/etc/httpd.conf

```
:
User  user_name
Group Sites_group_name
```

nano /Applications/XAMPP/xamppfiles/etc/extra/httpd-vhosts.conf

```
<VirtualHost *:6080>
    DocumentRoot "/Users/user_name/Sites/blog"
    ServerName blog.test

    <Directory "/Users/user_name/Sites/blog">
       Options Indexes FollowSymLinks Includes ExecCGI
       AllowOverride All
       Require all granted
    </Directory>
</VirtualHost>
```html
```

3. 使用者家目錄之設置啟用:

- 目的： (http://localhost/~username)
- 設置： #啟用模組和配置 #您需要啟用（取消註釋該行）並修改/更新 httpd.conf 檔案中的配置。 #要啟用模組，首先，您需要找到要啟用的模組。 #用於調 control + w 出搜尋功能，查詢以下模組並確保取消註釋。（刪除#每行前面的。）

sudo nano /Applications/XAMPP/etc/httpd.conf

```
:
LoadModule authn_core_module lib/httpd/modules/mod_authn_core.so
LoadModule authz_host_module lib/httpd/modules/mod_authz_host.so
LoadModule userdir_module lib/httpd/modules/mod_userdir.so
LoadModule include_module lib/httpd/modules/mod_include.so
LoadModule rewrite_module lib/httpd/modules/mod_rewrite.so
# User home directories
Include etc/extra/httpd-userdir.conf
```

sudo nano /Applications/XAMPP/etc/extra/httpd-userdir.conf

```
# UserDir: The name of the directory that is appended onto a user's home
:
UserDir Sites
#
# Control access to UserDir    directories.  The following is an    example
```
```

```
# for a site where these    directories are restricted to    read-only.
#
<Directory "/Users/allen/Sites/">
   AllowOverride all
   Options Indexes FollowSymLinks Multiviews
   MultiviewsMatch Any
   Require all granted
</Directory>
```

#添加虛擬主機 `localhost` 設定 `sudo nano /Applications/XAMPP/etc/extra/httpd-vhosts.conf`

```
<VirtualHost *:6080>
  DocumentRoot "/Users/allen/Sites"
  ServerName localhost

  <Directory "/Users/allen/Sites">
    Options Indexes FollowSymLinks Includes ExecCGI
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

XDEBUG 建置

1. `VSCode` 之 `php debug` 安裝

    1. 安裝 Xdebug 強烈建議你做一個簡單的 `info.php` 文件，放一個 `phpinfo();` 語句，然後復制輸出並將其粘貼到 Xdebug 安裝嚮導中。它將對其進行分析並為您提供適合您的環境的安裝說明。簡而言之：

        - 在 `Windows` 上：為您的 `PHP` 版本、體系結構（64/32 位）、線程安全 (TS/NTS) 和 `Visual Studio` 編譯器版本下載適當的預編譯 `DLL`，並將其放置在您的 `PHP` 擴展文件夾中。
        - 在 `Linux` 上：下載源代碼作為 tarball 或使用 git clone它，然後編譯它。或者查看您的發行版是否已經提供預構建包。

    2. 通過將 `zend_extension=path/to/xdebug` 添加到您的 `php.ini` 來配置 `PHP` 以使用 `Xdebug`。 `php.ini` 的路徑顯示在 "Loaded Configuration File" 下的 `phpinfo()` 輸出中。

    3. 在 `php.ini` 中啟用遠程測試：

        - for `Xdebug v3.x.x`:

        ```
        xdebug.mode = debug
        xdebug.start_with_request = yes
        ```

        - For `Xdebug v2.x.x`:

        ```
        xdebug.remote_enable = 1
        xdebug.remote_autostart = 1
        xdebug.remote_port = 9000
        ```

    4. 還有其他方法可以告訴 `Xdebug` 連接到遠程調試器，例如 `cookie`、查詢參數或瀏覽器擴展。 推薦 `remote_autostart (Xdebug v2)`/`start_with_request (Xdebug v3)` 因為它 "正常工作"。還有各種

其他選項，如端口，請參閱有關遠程調試的 Xdebug 文檔以獲取更多信息。請注意，默認 Xdebug 端口在 Xdebug v2 到 v3 之間從 9000 更改為 9003。

2. Xdebug Installation Wizard

    1. 下載 xdebug-3.2.0.tgz
    2. 安裝編譯 PHP 擴展的先決條件。 在您的 Mac 上，我們只支持使用 "homebrew" 進行安裝，並且 brew install autoconf 應該會引入正確的包。
    3. 使用 cd cd ~/Downloads & tar -xvzf xdebug-3.2.0.tgz 解壓縮下載的文件
    4. 運行： cd xdebug-3.2.0
    5. 運行： phpize（如果沒有 phpize，請參閱常見問題解答）。 作為其輸出的一部分，它應該顯示：

    ```
    Configuring for:
    ...
    Zend Module Api No:      20220829
    Zend Extension Api No:   420220829
    ```

    > 如果沒有，則說明您使用了錯誤的 phpize。請遵循此 FAQ 條目並跳過下一步。

    6. 運行： ./configure Q1: configure: error: C compiler cannot create executables A1: xcode-select --install 更新 xcode
    7. 運行： make
    8. 運行:

    ```
    sudo cp modules/xdebug.so
    /Applications/XAMPP/xamppfiles/lib/php/extensions/no-debug-non-zts-20220829
    ```

    9. 更新 sudo nano /Applications/XAMPP/xamppfiles/etc/php.ini 和增加這行:

    ```
    :
    [xdebug]
    zend_extension = xdebug
    xdebug.start_with_request = yes
    xdebug.mode= "debug,develop"
    ```

    10. 重啟 Apache Webserver

3. VSCode 建立新組態 執行 > 新增組態，選擇 php 後，產生 launch.json

```
{
 // 使用 IntelliSense 以得知可用的屬性。
 // 暫留以檢視現有屬性的描述。
 // 如需詳細資訊，請瀏覽: <https://go.microsoft.com/fwlink/?linkid=830387>
 "version": "0.2.0",
 "configurations": [
     {
         "name": "Listen for Xdebug",
         "type": "php",
```

```
            "request": "launch",
            "port": 9003
        },
        {
            "name": "Launch currently open script",
            "type": "php",
            "request": "launch",
            "program": "${file}",
            "cwd": "${fileDirname}",
            "port": 9003,
            "runtimeArgs": [
                "-dxdebug.start_with_request=yes"
            ],
            "env": {
                "XDEBUG_MODE": "debug,develop",
                "XDEBUG_CONFIG": "client_port=${port}"
            }
        },
        {
            "name": "Launch Built-in web server",
            "type": "php",
            "request": "launch",
            "runtimeArgs": [
                "-dxdebug.mode=debug",
                "-dxdebug.start_with_request=yes",
                "-S",
                "localhost:0"
            ],
            "program": "",
            "cwd": "${workspaceRoot}",
            "port": 9003,
            "serverReadyAction": {
                "pattern": "Development Server \\(<http://localhost:([0-9>]+)\\)
started",
                "uriFormat": "http://localhost:%s",
                "action": "openExternally"
            }
        }
    ]
}
```

4. 設定中斷點，啟動除錯，選擇組態:

   - `Listen for Xdebug` 後，按下綠色箭頭，當網頁重整(執行 `info.php`)時會觸發 組態：

     `launch.json` 內的組態名稱為 `Listen for Xdebug` 需有

     ```
       :
     "runtimeArgs": [
         "-dxdebug.start_with_request=yes"
     ],
     ```

     否則須定義在 `php.ini` 內，如下:

     ```
     [xdebug]
       :
     ```

```
xdebug.mode= "debug,develop"
```

- Launch currently open script 按下綠色箭頭，直接觸發

GIT 安裝

1. 安裝與測試
    1. 連結到 Git 官網，首頁會有下載 Download for Mac 按鈕。
    2. 按下後會跳轉頁面：
        - Homebrew 如果您還沒有 Homebrew，請安裝它，然後： $ brew install git
        - MacPorts 如果您還沒有 MacPorts ，請安裝它，然後： $ sudo port install git
        - Xcode Apple 隨 Xcode 一起發布了 Git 的二進制包。
        - 二進制安裝程序 Tim Harper 為 Git 提供了一個安裝程序 。最新版本是 2.33.0，於 1 年前於 2021-08-30 發布。
        - 構建從原始碼 如果您更喜歡從源代碼構建，可以 在 kernel.org 上找到 tarball 。最新版本是 2.39.2。
        - 安裝 git-gui 如果你想安裝 git-gui 和 gitk，git 的提交 GUI 和交互式歷史瀏覽器，你可以使用 homebrew $ brew install git-gui
    3. 如何確保安裝成功？ 步驟一： 按下鍵盤上的 Control + 空格 組合鍵來打開 Mac 內建搜尋(Spotlight)功能 步驟二： 輸入關鍵字「終端機」或「terminal」，尋找到對應軟體後，按下 Enter 步驟三： 複製此指令「git --version」，並在終端機點選滑鼠右鍵，選擇「Paste」貼上後，按 Enter
    4. 如果系統有回饋你安裝的版本編號，表示安裝成功
2. Git
    1. 初始化

    ```
    git config --global user.name "Allen Tai"
    git config --global user.email user_name@xxx.xxx
    ```

    2. 設定完成之後可以在本地端先開一個作業用資料夾，並初始化。

    ```
    ~ % mkdir mac_pic
    ~ % cd mac_pic
    mac_pic% git init
    ```

    > 在這個資料夾就可以進行程式的版本控管了

    3. 可以在這邊新增一個程式檔案

    ```
    #將指定檔案（或資料夾）加入版本控制（Staging Area and Tracked）。用.可加入（刪除）全部
    `git add <files or folders>`

    #可以確認本地端檔案的資料及狀態
    `git status`
    ```

4. 新增完之後輸入 `git status` 會出現這個資料夾的所有程式檔案，剛剛新增的資料就會是 `Staged`的狀態。（如下圖）
5. 如果將檔案修改的話，他就會是 `modified` 的狀態
6. 每一次修改檔案的時候記得要 `add` 進 `Staging` 裡

```
#將更改的檔案新增到暫存
git add "<file>"
```

7. 再來就可以將檔案提交到本地的倉庫

```
#提交目前的異動並設定摘要說明
git commit -m "<message>"
```

8. 其他參考： `https://ithelp.ithome.com.tw/articles/10297145`

SSH 免密碼登入

1. 生成公鑰和私鑰（客戶端 `Mac` 上操作） 開啟終端（命令列），執行下邊的命令，一路回車，即可得到公鑰和私鑰。

   `ssh-keygen -t rsa`

   > 公鑰和私鑰就放在 `~/.ssh` 目錄下

   其中，`id_rsa` 為私鑰，`id_rsa.pub` 為公鑰，它們都是文字檔案，可以用任何文字編輯器開啟。 如果在該資料夾下有名為 "known_hosts" 的檔案，把該檔案刪掉。可以把它看做一個快取檔案，目標主機發生變化時，快取可能導致驗證不正確。

2. 在遠端主機上新增信任的公鑰（遠端主機 `Mac` 上操作） 開啟遠端 `Mac` 主機，進入到 `~/.ssh` 目錄，開啟名字為 `authorized_keys` 的文字檔案（沒有的話，就建立它），然後把客戶端建立的公鑰新增去.

   至此，所有操作就算完成了，所有在遠端主機 `authorized_keys` 檔案裡添加了公鑰的Mac客戶端，遠端登入時，都不再需要輸入密碼了。

   > ssh-copy-id -i ~/.ssh/id_rsa.pub -p 22 user_name@xxx.xxx.xxx

3. 建立 `nano ~/.ssh/config`, 內容如下：

```
#allen
Host allen
HostName 127.0.0.1
PORT 6022
User allen
identityfile ~/.ssh/id_rsa
```

4. 登入遠端主機指令
   - 格式: `ssh 用戶SSH名稱@伺服器IP位置`
   - 範例: `ssh allen` 或 `ssh allen@127.0.0.1`