Dokumentation des Screenshotters

1. Installation

Um das Script ausführen zu können brauchst du Python. Ich empfehle die Installation von Python 3.8, da es noch ältere Windows-Versionen unterstützt und wir ohnehin nichts neueres brauchen.

Du findest Versionen für alle Betriebssysteme auf der Website: https://www.python.org/downloads/

Zur automatischen Installation entpackst du das „screenhotter.zip“, in das Verzeichnis, in dem das Script laufen soll. Darin befindet sich das Script selbst „ems_screenshot.py“, eine Textdatei „requirements.txt“, die Konfiguration „config.ini“, separate Installationsskripts für Windows, MacOS und Ubuntu, sowie diese Dokumentation. Nachfolgend die Installation für Windows, die ich getestet habe unter Windows 10:

I) Rechtsklick auf „install_windows.ps1“ → „Mit PowerShell ausführen“

II) Nun sollte im Terminal der Download des Python 3.8.10 Installations-Setups stattfinden.

III) Danach öffnet sich der Python-Installer automatisch in einem Fenster. Dort belässt du alles bei den Standardeinstellungen und klickst dich weiter, bis die Installation abgeschlossen ist.

IV) Am Ende klickst du noch auf „Disable Path Length Limit“. Anschließend werden die nötigen Pythonpackages selbstständig installiert und zuletzt der Headless-Chromium-Broswer.

Die Installation auf MacOS hab ich nicht getestet, mangels Mac. Du kannst „macos_install.sh“ gern testen und sagen, ob es läuft. Wechsel ins Verzeichnis und starte es mit diesem Befehl im Terminal:

```
cd /pfad/zum/ordner/des/scripts && bash macos_install.sh
```

2. Konfiguration

Das Script wird über die „config.ini“ Argumenten konfiguriert oder über Argumente, die beim Start eingegeben werden. Die Argumente haben dabei Vorrang bzw. ersetzen Werte in der „config.ini“. Werden gar keine Argumente eingegeben, wird die komplette Konfiguration übernommen. Um eine Übersicht der Argumente zu bekommen, kannst du die Hilfe aufrufen. Starte dafür die **PowerShell** oder das **Terminal**, wechsel in das Verzeichnis des Scripts (mit **cd**) und führe folgenden Befehl aus.

```
python ems_screenshot.py -h
```

Dann werden alle Argumente genauer erklärt. Ich möchte nun die wichtigsten Einstellungen beschreiben und einige Nutzungsbeispiele geben.

Das Programm nimmt mindestens 2 Argumente: „start_datetime“ und „end_datetime“

Sie können entweder beide „now“ gesetzt werden, dann wird das Script nur einmal ausgeführt und nimmt zur nächsten vollen Minute einen oder mehrere Screenshots auf.

```
python ems_screenshot.py now now
```

Wenn nur der 1. Parameter „now“ ist und der 2. Uhrzeit oder Datum mit Uhrzeit, sieht es so aus:

```
python ems_screenshot.py now 1645
```

oder:

```
python ems_screenshot.py now 202503141300
```

Das Datum mit Uhrzeit ist also im Format „YYYYmmddHHMM“

Natürlich können auch beide Parameter eine Uhrzeit oder Datum + Uhrzeit haben:

```
python ems_screenshot.py 1850 202503140600
```

Die restlichen Argumente sind alle sogenannte Flags, die genauso wie die Hilfe mit „-x“ aufgerufen werden, wobei x für einen Buchstaben steht. Ein Flag wird gesetzt, indem „-x“ gefolgt von einem Leerzeichen und dem Wert an das Ende des Befehls angehängt wird. Wird das Flag dagegen nicht gesetzt, bleibt es bei der Standardeinstellung, die in der „config.ini“ festgelegt wurde. Beispiel:

```
python ems_screenshot.py 1850 202503140600 -i 5
```

Manche Flags brauchen auch keinen Wert am Ende, sondern aktivieren eine Option. Dann wird einfach nur „-x“ angehängt. Zum Beispiel bei watermark / Wasserzeichen:

```
python ems_screenshot.py 1850 202503140600 -w
```

Ausführliche Erklärung der aller Flags:

-i → **Intervall** in Minuten, in dem das Skript laufen soll. (Standard = „1“)

-s → **Sites** also Websites, von denen Screenshots aufgenommen werden sollen. Möglich sind „a“=ALLE „d“=DWD, „u“=UWZ und m=MetMaps. Es sind auch Kombinationen davon. ZB: „-s du“ für DWD und UWZ. (Standard = „a“)

-o → **Output**-Verzeichnis. Wohin sollen die Screenshots gespeichert werden? Hier kann entweder der relative Pfad stehen, also innerhalb des Ordners, wo ausgeführt wird oder ein absoluter Pfad wie „C:\Users\Uwe\Bilder\Screenshots\EMS“. (Standard = „screenshots“)

-U → **URL** von MetMaps. Wird nur für MetMaps benötigt; die komplette URL von der ein Screenshot erstellt werden soll. (Standard = „https://metmaps.eu“)

-w → **Wasserzeichen** des Datums + Uhrzeit auf den Screenshot schreiben. Wird nur erstellt, wenn „-w“ angehängt wird. (Standard =““ also kein Wasserzeichen)

-u → **username**. Der Benutzername von MetMaps. (Standard = „sd_juri“)

-p → **password**. Wird auch nur für Metmaps gebraucht. (Standard = „mein Passwort“)

-a → User **Agent**. Die Browserkennung, d.h. als welcher Browser geben wir uns aus. Als Standard ist Chrome eingestellt und das sollte gehen. Muss in der Regel nicht angepasst werden.

-l → **logging**. Sollte aktiviert bleiben, damit ich im Absturzfall dein „error.log“ checken kann.

-v → **verbose**. Ausführlicher Output in der PowerShell während das Programm läuft.

-j → **join**. Die Browseraufrufe der Websites erfolgen in separaten Prozessen. Join sorgt dafür, dass das Hauptprogramm erst beendet wird, wenn alle Unterprozesse abgeschlossen sind.

Ein Nutzungsbeispiel:

```
python ems_screenshot.py 202504141200 202504161800 -i 10 -o „"C:\
pfad\zu\einem\ordner" -w -s dm -u user -p password -U
https://metmaps.eu/[…]
```

Starte am 14.03.2025 um 12:00 UTC und beende das Script am 15.04. um 18 UTC. Nehme alle 10 Minute Screenshots von DWD und MetMaps auf. MetMaps mit user+password und URL

3. Automatisches Ausführen

Um das Script automatisch laufen zu lassen kann es entweder mit Datum + Uhrzeit gestartet werden oder über einen Cronjob oder sonstigen Scheduler regelmäßig mit den Parametern „now now“ + weitere gewünschte Argumente / Flags aufgerufen werden.

Wie ich von Bernd Richter weiß, gibt es Cron auch für Windows. Syntax sollte identisch sein und ich kann dir gern bei der Einrichtung helfen Die Einstellung kann mittels des Tools https://crontab.guru/ getestet werden. Wenn der Crontab nicht mehr benötigt wird, kann er über ein # vor der Zeile in der Crontab-Datei auskommentiert werden. Dann wird die Ausführung gestoppt.

Ich habe das Programm nun recht ausführlich getestet und einige Verbesserung vorgenommen, sowie Vorkehrungen zur Stabilität und damit Sicherheit vor Abstürzen. Ich kann aber dennoch nicht 100%ig garantieren, dass bei der Nutzung mit „start_datetime end_datetime“ nicht doch irgendetwas schief läuft... Deshalb würde ich, besonders bei längeren Laufzeiten, immer zu Crontab oder einer Windows-Alternative raten. Crontab für Windows wird hier beschrieben: https://www.powershell-user.de/crontab-und-aufgabenplanung-mit-der-powershell/

Falls dir Crontab zu kompliziert zu lernen ist: Auf Windows gibt es die „Aufgabenplanung“ (auf englisch „Task Scheduler“) und auf MacOS auch ein Pendant dazu, was sich „Automator“ nennt. Dokumentationen dazu finden sich reichlich im Netz und bei Fragen kann ich dir auch weiterhelfen.

Ich wünsche viel Freude und Erfolg mit dem Script!

Juri