Dokumentation des Screenshotters

1. Installation

Windows

- Unter Windows muss das Script `windows_install.ps1` ausgeführt werden, um die notwendigen Abhängigkeiten zu installieren.
- Falls die Ausführung durch eine Systemrichtlinie blockiert wird, kann die Ausführung der PowerShell-Skripte temporär mit folgendem Befehl erlaubt werden:

  ```
  Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
  ```

- Danach die PowerShell neu starten und das Skript erneut ausführen.

  ```
  .\windows_install.ps1
  ```

- Die Installation installiert die Abhängigkeiten und legt die Umgebungsvariablen an, die für die Ausführung des Screenshotters notwendig sind.
- Zuerst wird Python 3.13 installiert und der Nutzer muss den Anweisungen folgen, um die Installation abzuschließen.
- Danach wird die Python-Executable in den Umgebungsvariablen eingetragen.
- Anschließend werden die benötigen Python-Pakete installiert, die in der `requirements.txt` definiert sind.
- Dann wird noch der Chromium-Headless-Browser installiert, der für die Screenshots benötigt wird.
- Die Installation des Browsers kann einige Zeit in Anspruch nehmen, da er heruntergeladen und entpackt werden muss.
- Als letzter Schritt wird das Skript `ems_screenshotter.py` als eine .pyc-Datei kompiliert, um die Ausführung zu beschleunigen.
- Die Installation ist damit abgeschlossen und der Screenshotter kann verwendet werden.

macOS

- Unter MacOS muss das Script `macos_install.sh` ausgeführt werden, um die notwendigen Abhängigkeiten zu installieren.

  ```
  sudo bash macos_install.sh
  ```

- Die Installation installiert die Abhängigkeiten und legt die Umgebungsvariablen an, die für die Ausführung des Screenshotters notwendig sind.
- Zuerst wird Python 3.13 installiert und der Nutzer muss den Anweisungen folgen, um die Installation abzuschließen.
- Danach wird die Python-Executable in den Umgebungsvariablen eingetragen.
- Anschließend werden die benötigten Python-Pakete installiert, die in der `requirements.txt` definiert sind.
- Dann wird noch der Chromium-Headless-Browser installiert, der für die Screenshots benötigt wird.
- Die Installation des Browsers kann einige Zeit in Anspruch nehmen, da er heruntergeladen und entpackt werden muss.
- Als letzter Schritt wird das Skript `ems_screenshotter.py` als eine .pyc-Datei kompiliert, um die Ausführung zu beschleunigen.
- Die Installation ist damit abgeschlossen und der Screenshotter kann verwendet werden.

Ubuntu

- Unter Ubuntu muss das Script `ubuntu_install.sh` ausgeführt werden, um die notwendigen Abhängigkeiten zu installieren.

  ```
  sudo bash ubuntu_install.sh
  ```

- Die Installation installiert die Abhängigkeiten und legt die Umgebungsvariablen an, die für die Ausführung des Screenshotters notwendig sind.
- Zuerst wird Python 3.13 installiert und der Nutzer muss den Anweisungen folgen, um die Installation abzuschließen.
- Die Python-Executable sollte automatisch in den Umgebungsvariablen eingetragen werden, falls nicht, muss dies manuell erfolgen. Liegt Python im Standardinstalltions-Pfad von Ubuntu, kann dies mit folgendem Befehl erfolgen:

  ```
  export PATH="/usr/bin/python3.13:$PATH"
  ```

- Anschließend werden die benötigten Python-Pakete installiert, die in der `requirements.txt` definiert sind.
- Dann wird noch der Chromium-Headless-Browser installiert, der für die Screenshots benötigt wird.
- Die Installation des Browsers kann einige Zeit in Anspruch nehmen, da er heruntergeladen und entpackt werden muss.
- Als letzter Schritt wird das Skript `ems_screenshotter.py` als eine .pyc-Datei kompiliert, um die Ausführung zu beschleunigen.
- Die Installation ist damit abgeschlossen und der Screenshotter kann verwendet werden.

2. Konfiguration

- Die Konfiguration des Screenshotters erfolgt über die Datei `config.ini`.
- Diese Datei enthält alle notwendigen Einstellungen, die für die Ausführung des Screenshotters benötigt werden.

- Die Datei ist in verschiedene Abschnitte unterteilt, die jeweils eine bestimmte Funktion haben.
- Die Abschnitte sind:
    - `[general]`: Allgemeine Einstellungen, wie das Intervall zwischen den Screenshots.
    - `[playwright]`: Alles, was das `playwright`-Package betrifft, wie z.B. der Browser, der User-Agent und das Timeout.
    - `[debug]`: Einstellungen für den Debug-Modus, wie z.B. die Aktivierung des Loggings (log) und Debug-Ausgaben (verbose).
    - `[metmaps]`: Einstellungen für die Metamaps, wie z.B. die URL der Metamap-API.
- Die Konfiguration kann manuell angepasst werden, um die gewünschten Einstellungen vorzunehmen.
- Es ist wichtig, die Konfiguration vor der ersten Ausführung des Screenshotters anzupassen, um sicherzustellen, dass alle Einstellungen korrekt sind.
- Die Konfiguration kann auch über die Kommandozeilen-Argumente angepasst werden, die beim Start des Screenshotters übergeben werden.
- Kommandozeilen-Argumente haben Vorrang vor den Einstellungen in der `config.ini`-Datei, sodass sie zur Laufzeit angepasst werden können.

3. Benutzung und Kommandozeilen-Argumente

- Der Screenshotter kann über die Kommandozeile (Bash bzw. PowerShell) gestartet werden.
- Wird er ohne Argumente gestartet, werden die in der `config.ini` definierten Einstellungen verwendet.
- Über --help oder -h kann eine Übersicht der verfügbaren Kommandozeilen-Argumente angezeigt werden.
- unter Windows:

  ```
  python ems_screenshotter.py --help
  ```

- unter MacOS und Ubuntu:

  ```
  python3 ems_screenshotter.py --help
  ```

- Die wichtigsten Kommandozeilen-Argumente sind:

  `start_end`: Start- und Enddatum für die Screenshots im Format `YYYYMMDDhhmm`. Wird nur ein Datum angegeben, wird der Screenshotter zu diesem Zeitpunkt gestartet und läuft so lange weiter, wie in der Konfiguration angegeben (`end_datetime`). Steht dort `end_datetime = max` läuft der Screenshotter (theoretisch) unendlich weiter!
  `--sites/-s`: Eine Liste von Websites, die gescreenshotet werden sollen. Diese Liste kann durch Kommata getrennt werden. Wenn keine Websites angegeben sind, werden alle in der Konfiguration definierten Websites gescreenshotet.
  `--output_dir/-o`: Der Pfad zum Verzeichnis, in dem die Screenshots gespeichert werden sollen. Wenn kein Pfad angegeben ist, wird das aktuelle Verzeichnis verwendet.
  `--verbose/-v`: Aktiviert den ausführlichen Modus, der zusätzliche Informationen während der Ausführung des Screenshotters ausgibt.

`--interval/-i`: Das Intervall in Sekunden, in dem die Screenshots erstellt werden sollen. Wenn kein Intervall angegeben ist, wird das in der Konfiguration definierte Intervall verwendet.
`--user_agent/-u`: Der User-Agent, der für die Screenshots verwendet werden soll. Wenn kein User-Agent angegeben ist, wird der Standard-User-Agent aus der Konfiguration verwendet.
`--watermark/-w`: Fügt einen Wasserzeichen (Datum und Uhrzeit) zu den Screenshots hinzu.
`--join/-j`: Nutze den Befehl Process.join() aus dem multiprocessing-Modul, um sicherzustellen, dass der Screenshotter wartet, bis alle Prozesse abgeschlossen sind, bevor er fortfährt. Dies ist besonders nützlich, wenn mehrere Screenshots gleichzeitig erstellt werden.
`--browser/-b`: Der Browser, der für die Screenshots verwendet werden soll. Standardmäßig wird der in der Konfiguration definierte Browser verwendet. Es kann jedoch auch ein anderer Browser angegeben werden, z.B. `chromium` oder `firefox`.
`--timeout/-t`: Die maximale Wartezeit in Sekunden, bevor der Screenshot erstellt wird. Wenn kein Timeout angegeben ist, wird das in der Konfiguration definierte Timeout verwendet.
`--network_idle/-n`: Aktiviert den Netzwerk-Leerlauf-Modus, der sicherstellt, dass der Screenshot erst erstellt wird, wenn keine Netzwerkaktivität mehr stattfindet. Dies kann nützlich sein, um sicherzustellen, dass alle Inhalte der Seite vollständig geladen sind, bevor der Screenshot erstellt wird.

4. Automatisierung (cronjob)

[macOS und Linux](#)
[Windows](#)

macOS und Linux

- Der Screenshotter kann automatisiert über einen Cronjob ausgeführt werden.
- Ein Cronjob ist ein zeitgesteuertes Skript, das regelmäßig ausgeführt wird.
- Um einen Cronjob zu erstellen, muss die Crontab-Datei bearbeitet werden.
- Unter Linux und MacOS kann die Crontab-Datei mit folgendem Befehl bearbeitet werden:

```
crontab -e
```

- Ein Beispiel für einen Cronjob, der den Screenshotter alle 5 Minuten ausführt, könnte wie folgt aussehen:

```
*/5 * * * * /usr/bin/python3 /path/to/ems_screenshotter.py
```

- Dieser Cronjob führt das Skript `ems_screenshotter.py` alle 5 Minuten aus.
- Es ist wichtig, den vollständigen Pfad zum Skript anzugeben, damit der Cronjob es finden kann.
- Es ist auch möglich, den Cronjob so zu konfigurieren, dass er nur an bestimmten Tagen oder zu bestimmten Zeiten ausgeführt wird.

- Um den Cronjob zu testen, kann er manuell ausgeführt werden, um sicherzustellen, dass er korrekt funktioniert.
- Es ist ratsam, die Log-Datei zu überprüfen, um sicherzustellen, dass der Cronjob erfolgreich ausgeführt wurde und keine Fehler aufgetreten sind.
- Die Log-Datei `error.log` wird im gleichen Verzeichnis wie das Skript gespeichert und kann mit einem Texteditor geöffnet werden. Sie enthält Informationen über die Ausführung des Screenshotters, einschließlich etwaiger Fehler oder Warnungen.
- Um sicher zu gehen, dass der Cronjob richtig funktioniert, sollte die Log-Datei regelmäßig überprüft werden.
- Die Log-Datei kann auch verwendet werden, um die Leistung des Screenshotters zu überwachen und sicherzustellen, dass er die erwarteten Ergebnisse liefert.
- Die Konfiguration des Cronjobs kann je nach Betriebssystem und Version variieren, daher ist es wichtig, die Dokumentation des jeweiligen Systems zu konsultieren, um sicherzustellen, dass der Cronjob korrekt eingerichtet ist.

Windows

- Unter Windows kann die Aufgabenplanung verwendet werden, um einen zeitgesteuerten Task zu erstellen.
- Alternativ lässt sich folgende Syntax verwenden, um den Screenshotter zu automatisieren. Hier ein Beispiel für einen Task, der alle 5 Minuten ausgeführt wird:

```
New-ScheduledTask -Action (New-ScheduledTaskAction -Execute "python" -
Argument "C:\path\to\ems_screenshotter.py") -Trigger (New-
ScheduledTaskTrigger -AtStartup -RepetitionInterval (New-TimeSpan -Minutes
5) -RepetitionDuration ([TimeSpan]::MaxValue)) -RunLevel Highest |
Register-ScheduledTask -TaskName "EMS_Screenshotter"
```

- Dieser Befehl erstellt einen neuen geplanten Task, der das Skript `ems_screenshotter.py` alle 5 Minuten ausführt.

5. Fehlerbehebung

- Wenn der Screenshotter nicht wie erwartet funktioniert, kann es an einer fehlerhaften Konfiguration liegen.
- Überprüfe die `config.ini`-Datei auf Fehler oder fehlende Einstellungen.
- Stelle sicher, dass alle notwendigen Abhängigkeiten installiert sind und die Umgebungsvariablen korrekt gesetzt sind.
- Überprüfe die Log-Datei `error.log` auf Fehlermeldungen oder Warnungen, die auf Probleme hinweisen könnten.