Kommandozeilentool für den Zugriff auf Confluence

Automatisiertes Herunterladen, Aktualisieren und Klonen von Confluence-Seiten über die Browser-Session des angemeldeten Benutzers (Playwright/Chromium). Dieses Toolkit besteht aus zwei Node.js-Skripten: atlassian-download.js und atlassian-upload.js.

Beim Start öffnet das Skript einen Chromium-Browser mit persistentem Profil. Nach der ersten Anmeldung in Confluence bleibt die Session im Profil bestehen.

Voraussetzungen

Installation von Scoop

Scoop ist ein einfacher Paket-Manager. Falls nicht vorhanden, kann Scoop unter Windows über die PowerShell installiert werden:

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
irm get.scoop.sh | iex
```

Installation von Git

Falls nicht bereits installiert, kann unter Windows Git einfach mit Scoop nachinstalliert werden:

```
scoop install git
```

Alternativ kann Git von der offiziellen Webseite herunterladen und installiert werden: https://git-scm.com.

Installation von Node.js und npm

Node.js und npm werden für die Ausführung der Skripte benötigt. Unter Windows empfiehlt sich die Installation über Scoop:

```
scoop install nodejs-lts
# Version prüfen (erwartet Node.js 18+)
node -v
npm -v
```

Alternativ kann Node.js (inkl. npm) über die offizielle Webseite installiert werden:
https://nodejs.org/en/download/

Hinweis: Falls bereits eine ältere Version installiert ist, kann über Scoop ein Update durchgeführt werden:

```
scoop update nodejs-lts
```

Klonen des Repositories

Der ausführbare Code aus dem Repository muss lokal zur Verfügung gestellt werden.

```
git clone git@github.com:danielfrey63/atlassian-connect.git
cd atlassian-connect
```

Projektcode fertig installieren

Zum Ausführen des Codes benötigen wir:

- Node.js 18+ und npm
- Internetzugang auf die Confluence-Instanz (z. B. https://confluence.my-company.ch)

```
npm install
npx playwright install chromium
```

Das Browserprofil wird im temporären OS-Verzeichnis unter `chromium-playwright-profile` abgelegt und beim nächsten Lauf wiederverwendet.

Download: Seiten und Bäume aus Confluence exportieren

Skript: atlassian-download.js

Aufruf (empfohlen):

```
node atlassian-download.js <base_url> <page_id> [--recursive=true|false] [--
destination=./output] [--format=xml|md|both] [--ask]
```

Parameter:

- `<base_url>` Basis-URL der Confluence-Site (z. B. https://confluence.my-company.ch)
- `<page_id>` Seiten-ID, die exportiert werden soll. Die Seiten-ID ist im URL der Seite ersichtlich oder über die Seiteninformation abrufbar.

Optionen:

- `--ask`, `-a` Interaktiver Modus: fragt alle Parameter ab
- `--recursive`, `-r true`/`false` Ob Kindseiten rekursiv geladen werden (Standard: true)
- `--destination`, `-d` Zielverzeichnis für Dateien (Standard: aktuelles Verzeichnis)
- `--format`, `-f` Exportformat: `xml` | `md` | `both` (Standard: `xml`). Bei `md` werden referenzierte Bilder/Anhänge heruntergeladen und Links im MD auf lokale Assets umgeschrieben.
- `--help`, `-h` Hilfe anzeigen

Beispiele:

```
# XML nur
node atlassian-download.js https://confluence.my-company.ch 3273443858 --
recursive=false --destination=./output

# Markdown + Assets
node atlassian-download.js https://confluence.my-company.ch 3273443858 -r false -d
"./output" -f md

# Beide Formate
node atlassian-download.js https://confluence.my-company.ch 3273443858 --
recursive=false --destination=./output --format=both

# Interaktiv
node atlassian-download.js --ask
```

Ergebnisstruktur:

- Für jede Seite wird die Confluence-„storage“-Repräsentation als XML gespeichert: `page_<ID>_<bis_zu_5_Begriffe>.xml`
- Optional wird zusätzlich eine Markdown-Datei erzeugt: `page_<ID>_<bis_zu_5_Begriffe>.md`
- Bei MD-Export werden referenzierte Bilder/Anhänge im Ordner `page_<ID>_<bis_zu_5_Begriffe>_assets` abgelegt; Verweise im MD werden auf relative Pfade (`./page_<...>_assets/<datei>`) umgeschrieben
- Bei rekursivem Export wird für jeden Knoten zusätzlich ein Ordner mit dem gleichen Basenamen angelegt, der die Kindseiten enthält
- Die Manifestdatei `tree.json` beschreibt die komplette Seitenhierarchie inkl. Dateipfade; der Eintrag `filepath` verweist weiterhin auf die XML-Datei

Technische Details:

- Die Inhalte werden im Browserkontext mit fetch gegen `/rest/api/content/...` gelesen
- Nach Navigation auf die Zielseite wird durch eine Wartebedingung die korrekte Seite erkannt (Timeout 60s)

Upload: Inhalte aktualisieren oder als neuen Baum klonen

Skript: atlassian-upload.js

Grundaufruf:

```
node atlassian-upload.js --base_url <base_url> [weitere Optionen]
```

Modi:

- update (Standard) Aktualisiert bestehende Seiten
- clone Erstellt eine Kopie eines Baumes unter einer Zielseite

Gemeinsame Optionen:

- `--ask`, `-a` Interaktiver Modus
- `--dry-run` Keine Änderungen, nur geplante Aktionen ausgeben
- `--help`, `-h` Hilfe anzeigen

Update-Modus:

- `--manifest`, `-f` Pfad zur Manifestdatei tree.json (Standard: `./output/tree.json`)
- `--source`, `-s` Verzeichnis der XML-Dateien (Standard: Ordner der Manifestdatei)
- `--page_id`, `-p` Aktualisiert nur eine einzelne Seite und erwartet bei `--source` den Pfad zur konkreten XML-Datei

Beispiele Update:

```
# gesamten Baum laut Manifest aktualisieren
node atlassian-upload.js -b https://confluence.my-company.ch --mode=update -f
./output/tree.json -s ./output

# einzelne Seite aktualisieren (XML-Datei angeben)
node atlassian-upload.js -b https://confluence.my-company.ch --page_id=2393644957
-s ./output/page_1336019187_How-
to_OpenShift/page_2393644957_High_Availability_Ingress_Traffic_Routing.xml
```

Ergebnisse Update:

- Pro Lauf wird eine `upload-summary.json` erzeugt (bei Einzel-Update im aktuellen Verzeichnis, sonst neben dem Manifest)

Clone-Modus:

- `--root_id`, `-r` ID der Zielseite, unter der der neue Baum angelegt wird (Pflicht)
- `--manifest`, `-f` Pfad zur Manifestdatei tree.json (Standard: `./output/tree.json`)
- `--source`, `-s` Verzeichnis der XML-Dateien (Standard: Ordner der Manifestdatei)
- `--suffix <suffix>` für doppelte Titel (Standard: `Clone`) – wird nur verwendet, wenn der Titel bereits existiert

Beispiel Clone:

```
node atlassian-upload.js -b https://confluence.my-company.ch --mode=clone --
root_id=123456789 -f ./output/tree.json -s ./output --suffix=Clone
```

Ergebnisse Clone:

- `upload-summary.json` Liste der erstellten Seiten
- `upload-map.json` Zuordnung Alt-ID → Neu-ID
- `new-tree.json` Manifest des neu erzeugten Baumes (oberste Ebene ist die neu erstellte Wurzel unterhalb von `--root_id`)

Hinweise zur Authentisierung und Laufzeit

- Die REST-Aufrufe erfolgen relativ (`/rest/api/content/...`) innerhalb der angemeldeten Browser-Session
- Beim ersten Lauf im geöffneten Chromium anmelden; danach bleibt die Session im persistenten Profil erhalten
- Standard-Timeouts für das Finden/Laden einer Seite betragen 60s

Fehlerbehebung

- 401/403: Session abgelaufen oder fehlende Berechtigungen – Browserfenster offen lassen und neu anmelden
- 404: Seite existiert nicht oder ist nicht sichtbar – ID prüfen
- Duplikate beim Clone: Das Skript hängt ein Suffix an oder nummeriert weiter, bis ein eindeutiger Titel gefunden ist
- Proxy/VPN: Sicherstellen, dass Confluence erreichbar ist

Entwicklung

- Abhängigkeiten und Skripte sind in package.json definiert
- Das Browserprofil wird in einem OS-Temp-Ordner wiederverwendet, sodass lokale Logins erhalten bleiben