OpenSchoolMaps: Data Cleansing und Integration Apachehop Fortsetzung

Ein Arbeitsblatt für Interessierte und Studierende in Data Engineering und Data Science

Einleitung

Ziele

Dieses Arbeitsblatt baut auf dem ersten Tutorial "Daten sichten, bereinigen und integrieren mit Apache Hop" von OpenSchoolMaps auf und zeigt fortgeschrittene Konzepte für die Automatisierung von Datenverarbeitungsprozessen. Es erläutert weitere wichtige Apache Hop-Funktionalitäten und Tools, insbesondere Workflows, Hop Run und Monitoring.

Die Ziele dieses Arbeitsblattes sind:

- Apache Hop **Workflows** verstehen und erstellen können.
- Apache Hop **Pipelines** zu komplexeren Abläufen orchestrieren können.
- **Hop Run** für die automatisierte Kommandozeilen-Ausführung verwenden können.
- Apache Hop **Pipelines mit Echtzeit-Datenquellen** kennenlernen.
- **Monitoring** und Fehlerbehandlung in Workflows aufsetzen können.

Zeitplanung

Die benötigte Zeit beträgt etwa 10 Minuten für den Leseteil und rund eine Stunde für die Aufgaben, abhängig vom Wissensstand.

Voraussetzungen

Um das Arbeitsblatt durcharbeiten zu können, brauchen Sie folgende Dinge:

- Abgeschlossenes erstes Apache Hop Tutorial von OpenSchoolMaps.
- Internetzugang zum Herunterladen von Echtzeit-Daten.
- Java Version 17, verfügbar bei OpenJDK.
- Software: Apache Hop Version 2.12 (oder neuer), bereits installiert.

Folgende Themen sollten bereits bekannt sein:

- Grundlagen von Apache Hop (zum Beispiel aus dem ersten Tutorial).
- Apache Hop Pipelines erstellen und ausführen.
- Apache Hop Transforms und Hops verwenden.

Struktur des Arbeitsblatts

Das folgende Arbeitsblatt ist in die folgenden Kapitel eingeteilt:

- *Workflows verstehen*: Konzepte und Unterschiede zu Pipelines.
- *Actions und Workflow-Orchestrierung*: Steuerung komplexer Abläufe.
- *Hop Run*: Automatisierung ohne GUI.
- *Echtzeit-Datenverarbeitung*: Integration von Live-Datenquellen.

- *Übungen*: Praktische Aufgaben mit Workflows und Automatisierung.
- *Monitoring und Fehlerbehandlung*: Robuste Produktionsumgebungen.
- *Abschluss*: Zusammenfassung und Best Practices.

Workflows verstehen

Was sind Workflows?

Während Pipelines die eigentliche Datenverarbeitung durchführen, orchestrieren **Workflows** diese Pipelines und andere Aktionen in einer logischen Reihenfolge. Pipelines und Workflows werden in Apache Hop über eine einheitliche grafische Oberfläche verwaltet. Sie sind über Symbole klar gekennzeichnet und werden in eigenen JSON-Dateien (.hpl/.hwf) abgelegt.

Ein Workflow besteht aus:

- **Actions**: Einzelne Aktionen wie das Starten einer Pipeline, Datei-Operationen oder Benachrichtigungen.
- **Hops**: Verbindungen zwischen Actions (ähnlich Verbindungen zwischen Transforms), die den Ablauf steuern.
- **Bedingungen**: Logik für bedingte Ausführung (Success/Failure-Pfade).

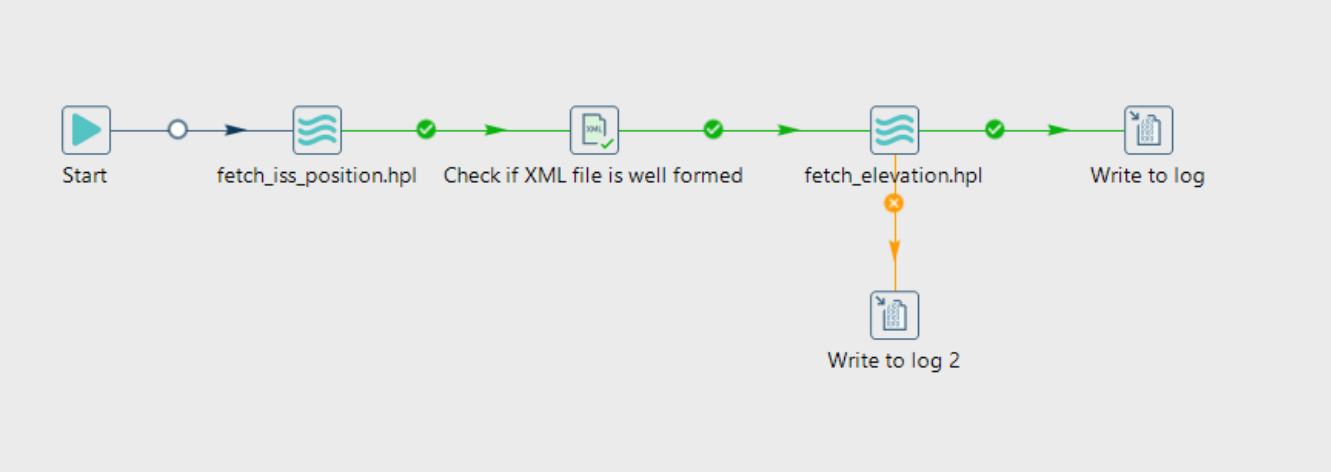

Abbildung 1. Beispiel eines Workflows in Apache Hop (aus Übung 2 unten).

Workflow-Actions

Im Gegensatz zu Pipeline-Transforms arbeiten Workflow-Actions nicht mit Datenströmen, sondern führen Steuerungsaufgaben aus:

- **Pipeline Action**: Startet eine bestehende Pipeline als Teil des Workflows.
- **File Operations**:
  - **Copy Files**: Kopiert Dateien von einem Ort zum anderen.
  - **Delete Files**: Löscht Dateien nach erfolgreicher Verarbeitung.
  - **File Exists**: Prüft, ob eine Datei vorhanden ist.

- **Write to log Action**: Schreibt Nachrichten in das Workflow-Log bei Erfolg oder Fehlern.
- **Wait Action**: Pausiert den Workflow für eine bestimmte Zeit oder wartet auf eine Bedingung.
- **SQL Action**: Führt SQL-Befehle auf Datenbanken aus (z.B. für Abfragen, Cleanup oder Backup).

Bedingte Ausführung

Workflows unterstützen bedingte Ausführung basierend auf dem Erfolg oder Misserfolg vorheriger Actions:

- **Unbedingter Hop (schwarzer Hop)**: Action wird immer ausgeführt.
- **Erfolg (grüner Hop)**: Action wird nur ausgeführt, wenn die vorherige erfolgreich war.
- **Fehler (oranger Hop)**: Action wird nur bei einem Fehler der vorherigen ausgeführt.

Hop Run - Kommandozeilen-Ausführung

Apache Hop kann nicht nur über die grafische Oberfläche, sondern auch über die Kommandozeile ausgeführt werden. Dies ist besonders wichtig für automatisierte Prozesse und Produktionsumgebungen.

Hop Run Grundlagen

`Hop Run` ist das Kommandozeilen-Tool von Apache Hop. Es ermöglicht die Ausführung von Pipelines und Workflows ohne das GUI.

Grundlegende Syntax:

```
hop-run.bat --file=<pfad-zur-datei> --project=<projektname> [optionen]
```

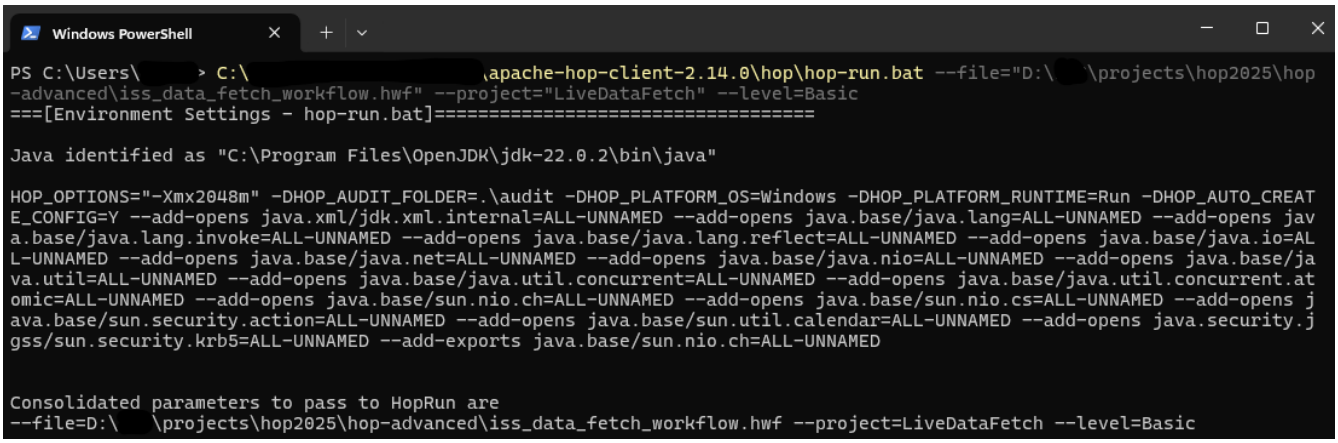

Abbildung 2. Hop Run Ausführung im Terminal.

Parameter und Variablen

Hop Run unterstützt die Übergabe von Parametern zur Laufzeit:

```
hop-run.bat --file=workflow.hwf --project=MyProject --parameter:INPUT_FILE=data.csv
--parameter:OUTPUT_DIR=/output
```

Diese Parameter können in Workflows und Pipelines als Variablen verwendet werden: `${INPUT_FILE}`, `${OUTPUT_DIR}`.

Logging und Monitoring

Hop Run bietet umfangreiche Logging-Optionen:

```
hop-run.bat --file=workflow.hwf --project=MyProject --level=Basic
--logfile=logfile.txt
```

Log-Level:

- **Nothing**: Kein Logging
- **Error**: Nur Fehler
- **Minimal**: Grundlegende Informationen
- **Basic**: Standard-Logging
- **Detailed**: Detaillierte Informationen
- **Debug**: Alle verfügbaren Informationen

Echtzeit-Datenverarbeitung

Streaming-Datenquellen

Apache Hop unterstützt verschiedene Echtzeit-Datenquellen:

- **Datei-Watcher**: Überwachung von Ordnern auf neue Dateien
- **REST APIs**: HTTP-basierte Datenquellen
- **Kafka**: Message-Streaming-Plattform

REST Client Transform

Mit dem REST Web Service Transform können Sie APIs abfragen:

Parameter:

- **URL**: Schnittstellen-Endpunkt (API-Endpoint)
- **HTTP Method**: GET, POST, PUT, DELETE
- **Headers**: Authentifizierung und Content-Type

- **Parameters**: Request Body für POST/PUT/DELETE

Übungen

Übung 1: Einlesen von Echtzeit-Daten

In dieser Übung erstellen Sie zwei einfach Pipelines, die Daten über eine API abfragen und diese in eine Output-Datei schreiben.

Die erste Pipeline soll die aktuelle Position der Internationalen Raumstation (ISS) abfragen und in eine XML-Datei schreiben. Die zweite Pipeline soll die Höhe des Terrains an der aktuellen Position der ISS abfragen und die Ergebnisse in eine Excel-Datei schreiben.

Schritt 1.1: Neues Projekt anlegen

1. Öffnen Sie Apache Hop.
2. Erstellen Sie ein neues Projekt namens "LiveDataFetch".
3. Setzen Sie als Home-Ordner einen neuen Ordner z.B. "hop-advanced".

Schritt 1.2: Pipeline 1 erstellen

1. Klicken Sie auf das **[+]** und wählen Sie `New Pipeline`.
2. Speichern Sie die Pipeline als `fetch_iss_position.hpl` im Projekt-Ordner.

Schritt 1.3: Pipeline-Transforms hinzufügen

Fügen Sie folgende Transforms hinzu:

Generate Rows Transform

Beim Ausführen der Pipeline wollen wir eine einzige Zeile gefüllt haben, die den aktuellen Zeitstempel enthält. Erstellen Sie einen `Generate Rows` Transform, der eine einzelne leere Zeile generiert.

Formula Transform

Fügen Sie einen `Formula` Transform hinzu, um den aktuellen Zeitstempel in das Feld `timestamp` zu schreiben. Verwenden Sie die Formel `NOW()`.

REST Client Transform

Fügen Sie einen `REST Client` Transform hinzu, um Daten von einer öffentlichen Service (API) abzurufen. Verwenden Sie die URL `https://api.wheretheiss.at/v1/satellites/25544` um die aktuellen Koordinaten der ISS abzurufen. Hier werden keine weitere Parameter benötigt. Die API-Antwort soll in einem Feld `response_json` gespeichert werden.

JSON Input Transform

Fügen Sie einen `JSON Input` Transform hinzu, um die Koordinaten aus der API-Antwort zu extrahieren. Wir intressieren uns nur für die Felder `altitude`, `velocity`, `latitude` und `longitude`. Um die JSON-Pfade im Tab "Fields" zu definieren, können sie die Funktion "Select fields from snippet" verwenden. Machen Sie dazu zuerst eine Abfrage an die API, um die Struktur des JSON zu erhalten. Diese kann in "Select fields from snippet" eingefügt werden, um die Felder automatisch in die Tabelle zu laden. Entfernen Sie die Zeilen, die für diese Übung nicht benötigt werden.

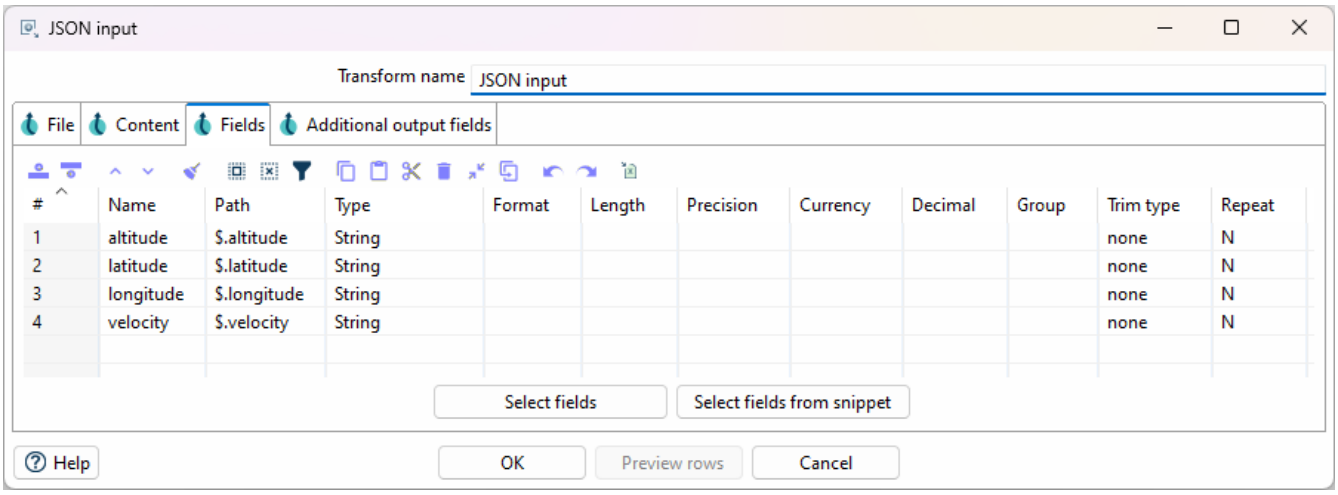

Abbildung 3. Beispiel-Konfiguration des JSON Input Transforms zum Extrahieren der ISS-Koordinaten.

Output Transform

Wir möchten die Daten in eine XML-Datei schreiben. Fügen Sie einen `XML Output` Transform hinzu und konfigurieren Sie diesen so, dass die Daten in eine Datei `iss.xml` im Projekt-Ordner geschrieben werden. Achten Sie darauf, dass die Datentypen stimmen, um Fehler beim Auslesen zu verhindern. `altitude`, `velocity`, `latitude` und `longitude` werden als 'Number' gespeichert.

Das Feld `response_json` möchten wir nicht in die XML-Datei schreiben. Verwenden Sie dazu entweder einen `Select Values` Transform vor dem `XML Output` Transform, um das Feld zu entfernen, oder konfigurieren Sie den `XML Output` Transform so, dass nur die gewünschten Felder geschrieben werden.

Speichern und Ausführen

Speichern Sie die Pipeline und führen Sie sie aus. Überprüfen Sie die generierte `iss.xml` Datei, ob die Daten korrekt geschrieben wurden.

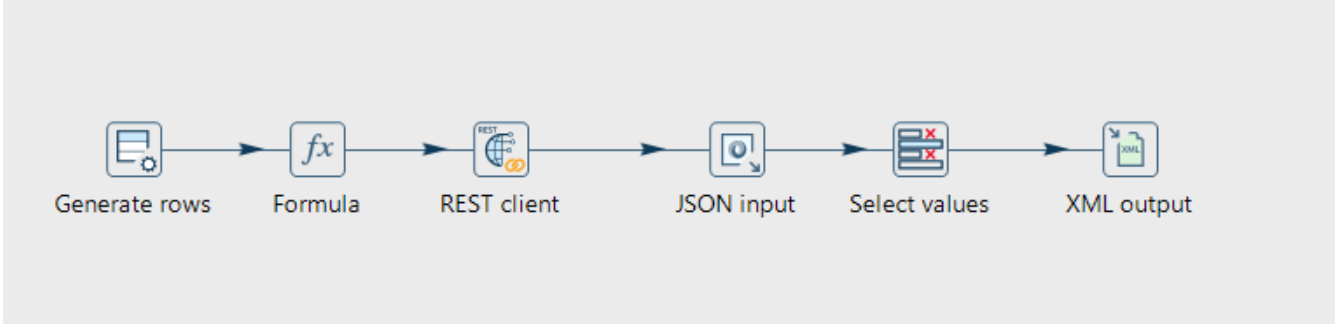

Abbildung 4. Pipeline zur Abfrage der aktuellen ISS-Position

Schritt 1.4: Pipeline 2 erstellen

Erstellen Sie eine zweite Pipeline namens `fetch_elevation.hpl` im selben Projekt-Ordner.

Diese Pipeline soll die `iss.xml` Datei einlesen und die Höhe des Terrains an der aktuellen Position der ISS abfragen. Dazu verwenden wir eine weitere öffentliche API `https://api.open-meteo.com/v1/elevation`.

Schritt 1.5: Transforms hinzufügen

Fügen Sie folgende Transforms hinzu:

XML Input Transform

Fügen Sie einen `XML Input` Transform hinzu, um die `iss.xml` Datei einzulesen. Konfigurieren Sie ihn so, dass alle Felder eingelesen werden.

Formula Transform

Die Query-Parameter für einen GET Request können nicht direkt in dem `REST Client` Transform definiert werden. Deshalb müssen wir die URL für die Elevation API dynamisch generieren. Fügen Sie einen `Formula` Transform hinzu, um die URL zu erstellen.

Erstellen Sie ein neues String Feld `request_url` mit der Formel:

```
"https://api.open-meteo.com/v1/elevation?latitude=" & [latitude] & "&longitude=" & [longitude]
```

REST Client Transform

Fügen Sie einen `REST Client` Transform hinzu, um die Höhe des Terrains abzufragen. Verwenden Sie die dynamisch generierte URL aus dem vorherigen Transform. Speichern Sie die Antwort in einem Feld `response_json`.

JSON Input Transform

Nutzen Sie wieder einen `JSON Input` Transform, um die Höhe aus der API-Antwort zu extrahieren. Beachten Sie, dass die API-Antwort ein Array enthält. Sie können das erste Element des Arrays extrahieren, indem Sie den Pfad `$.elevation[0]` verwenden.

Microsoft Excel writer Transform

Fügen Sie einen `Microsoft Excel writer` Transform hinzu, um die Ergebnisse in eine Excel-Datei zu schreiben. Konfigurieren Sie ihn so, das die Felder `timestamp`, `altitude`, `latitude`, `longitude`, `velocity` und `elevation` in eine Datei `output.xlsx` im Projekt-Ordner geschrieben werden.

Konfigurieren Sie den Excel Writer Transform so, dass keine neue Datei erstellt wird, sondern die Daten an eine bestehende Datei angehängt werden (append). Dafür müssen die entsprechenden Optionen im Tab "File" sowie "Sheet & Template" und "Content" angepasst werden.

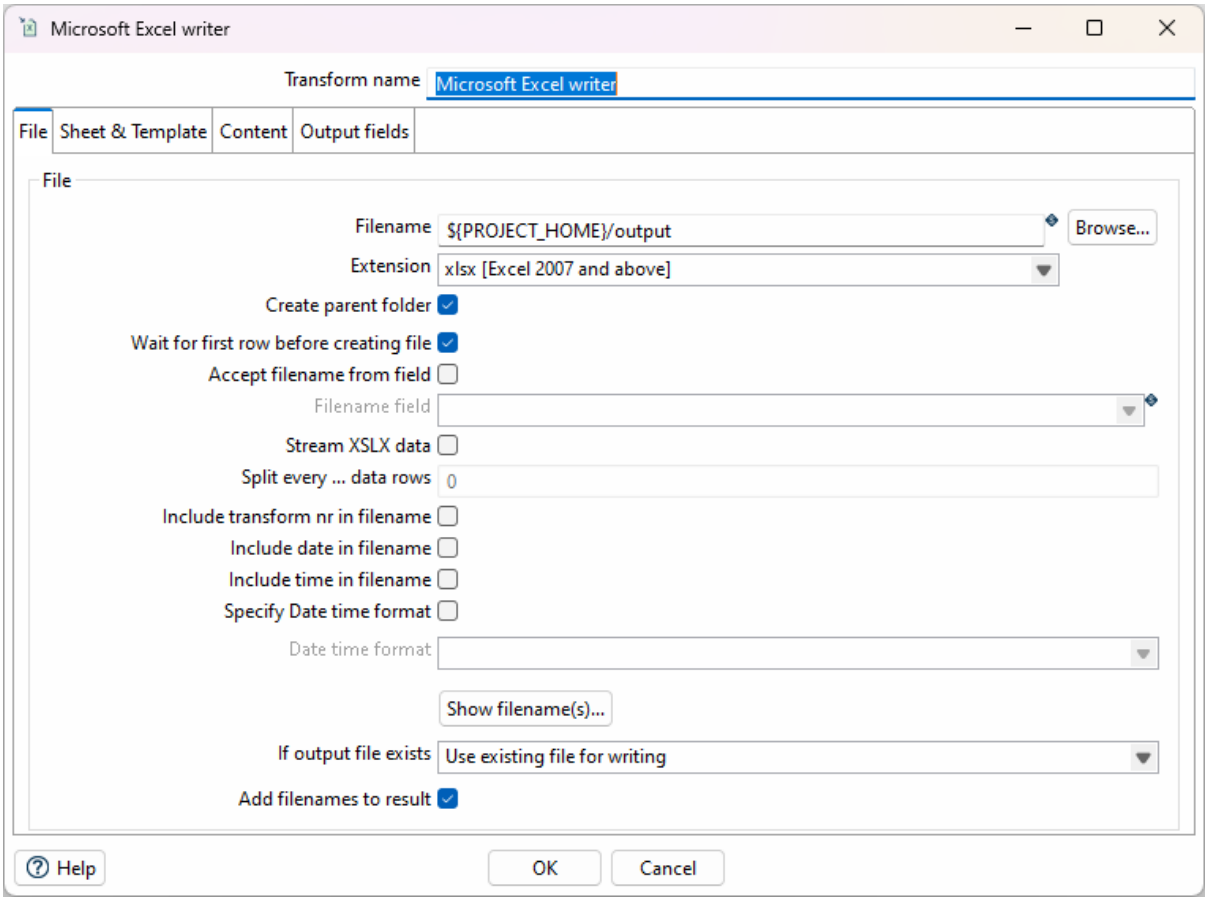

Abbildung 5. Konfiguration des Excel Writer Transforms - Tab "File"

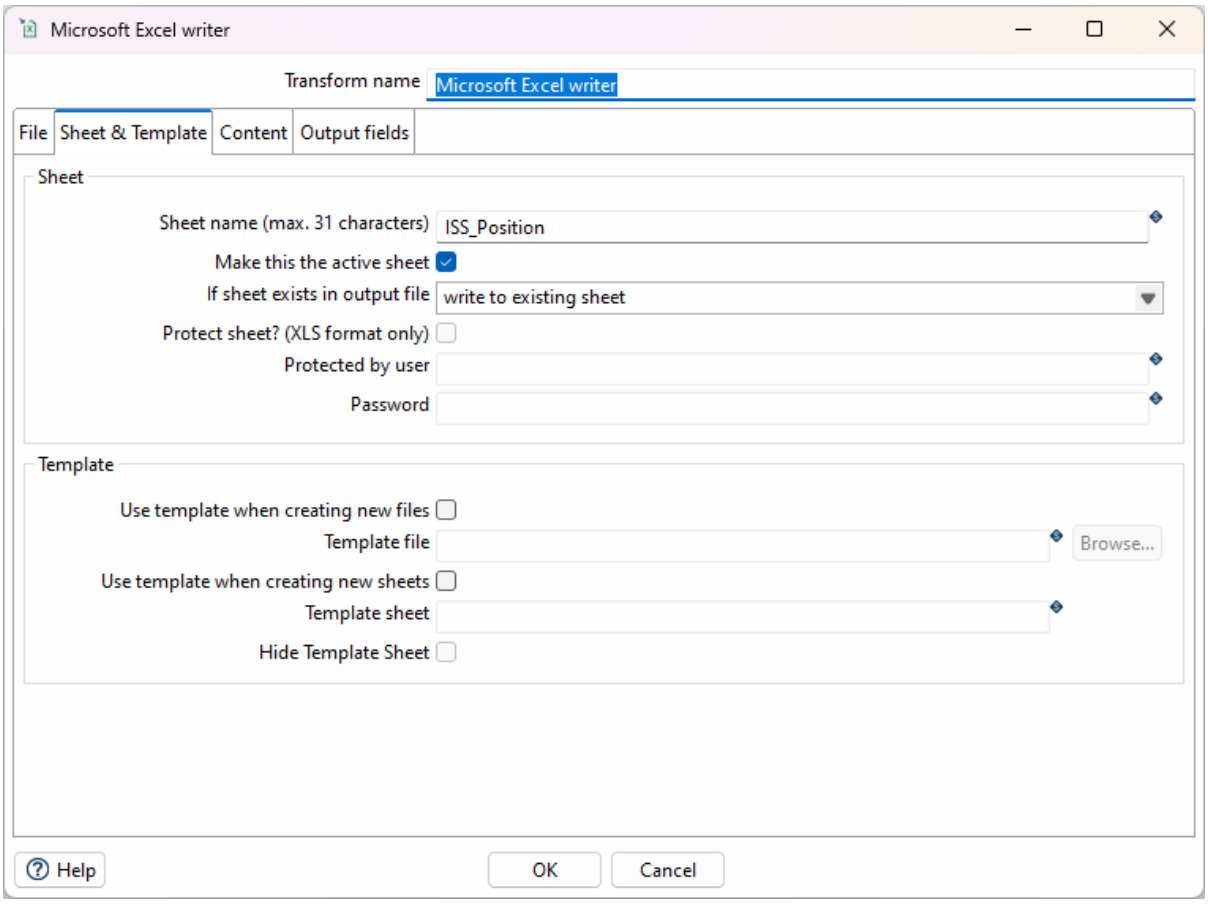

Abbildung 6. Konfiguration des Excel Writer Transforms - Tab "Sheet & Template"

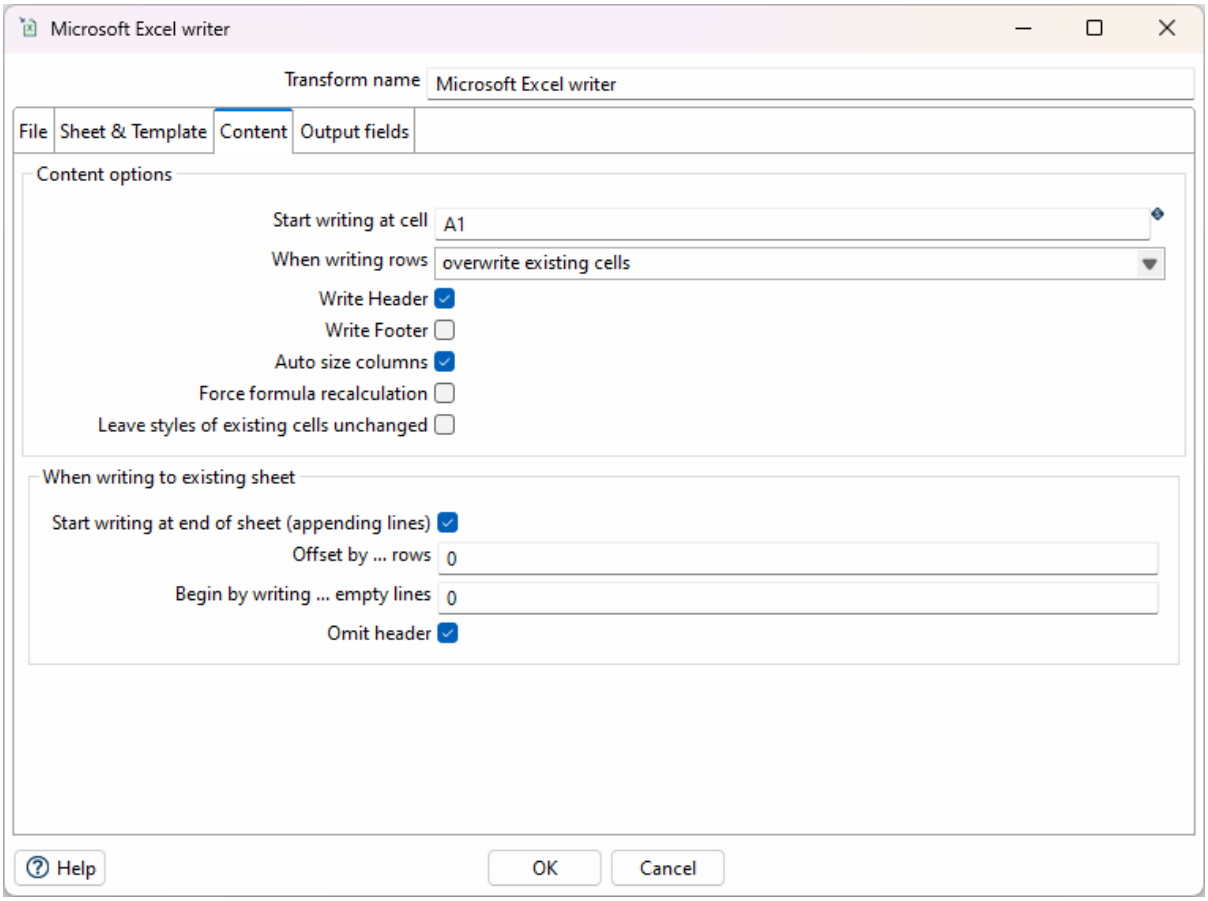

Abbildung 7. Konfiguration des Excel Writer Transforms - Tab "Content"

Speichern und Ausführen

Speichern Sie die Pipeline und führen Sie sie aus. Überprüfen Sie in der generierten ` output.xlsx`-Datei, ob die Daten korrekt an die Datei angehängt wurden.

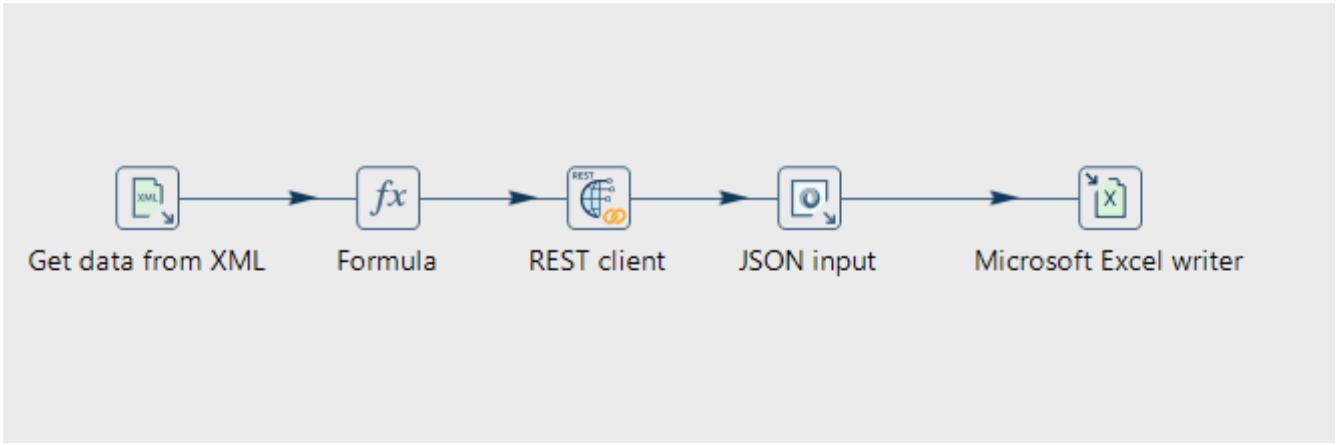

Abbildung 8. Pipeline zur Abfrage der Terrain-Höhe basierend auf ISS-Position

Übung 2: Workflow zur Automatisierung

In dieser Übung erstellen Sie einen Workflow, der die beiden Pipelines aus Übung 1 automatisiert ausführt und zeitgesteuert startet.

Schritt 2.1: Workflow erstellen

1. Klicken Sie auf das **[+]** und wählen Sie `New Workflow`.
2. Speichern Sie den Workflow als `iss_data_fetch_workflow.hwf` im Projekt-Ordner.

Schritt 2.2: Actions hinzufügen

Fügen Sie folgende Actions hinzu:

Pipeline Action 1

Fügen Sie eine `Pipeline` Action hinzu, um die erste Pipeline `fetch_iss_position.hpl` auszuführen. Konfigurieren Sie die Action so, dass sie im selben Projekt-Ordner gespeichert wird.

Check if XML file is well formed Action

Fügen Sie eine `Check if XML file is well formed` Action hinzu, um sicherzustellen, dass die `iss.xml` Datei korrekt erstellt wurde. Konfigurieren Sie sie so, dass sie die Datei `iss.xml` überprüft.

Pipeline Action 2

Fügen Sie eine `Pipeline` Action hinzu, um die zweite Pipeline `fetch_elevation.hpl` auszuführen. Konfigurieren Sie die Action so, dass sie im selben Projekt-Ordner gespeichert wird.

Verbinden Sie die Actions mit einem grünen Hop, um den Ablauf zu steuern.

Write to log Actions

Fügen Sie zwei `Write to log` Actions hinzu: * Eine Action nach der zweiten Pipeline Action, die eine Erfolgsmeldung schreibt. z.B. Log subject: "Status": "successfully fetched ISS data and elevation". * Eine weitere Action nach der zweiten Pipeline Action, die eine Meldung schreibt, wenn der Workflow fehlschlägt. z.B. Log subject: "Status": "failed to fetch ISS data and elevation".

Verbinden Sie die Erfolgsmeldung mit einem Hop (der dann grün erscheint) und die Fehlermeldung mit einem Hop (der dann orange erscheint). (Mit klicken auf den Hop ändern Sie die Art des Hops.)

Speichern und Ausführen

Verbinden Sie die Actions mit einem grünen Hop, um den Ablauf zu steuern. Speichern Sie den Workflow und führen Sie ihn aus. Überprüfen Sie die generierte `output.xlsx` Datei, ob die Daten korrekt an die Datei angehängt wurden.

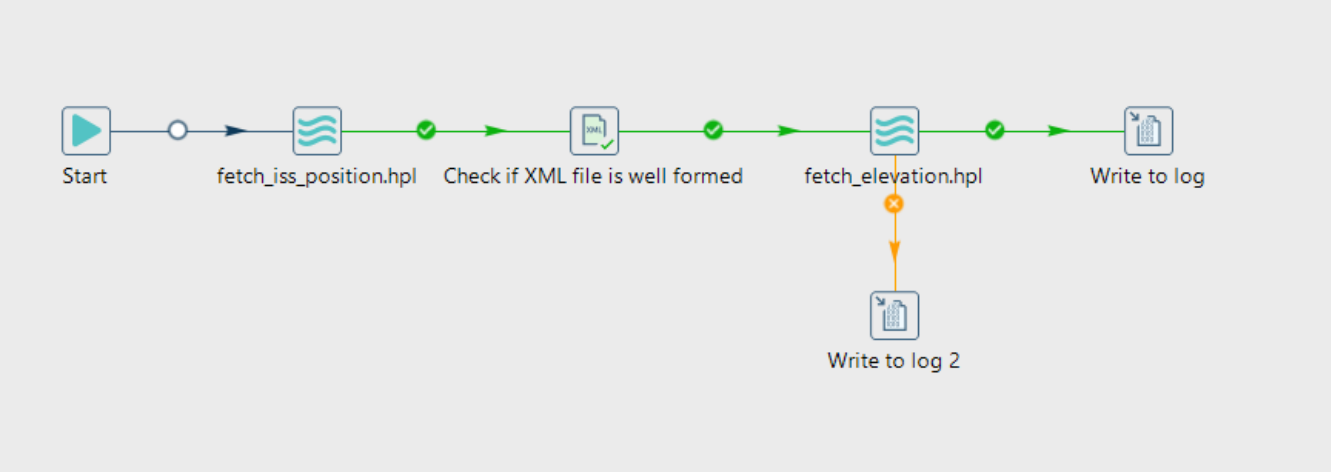

Abbildung 9. Workflow zur automatisierten ISS-Datensammlung mit Fehlerbehandlung

Übung 3: Zyklische Ausführung des Workflows mit einer Prozesszeitsteuerung und Hop Run

In dieser Übung lernen Sie, wie Sie den erstellten Workflow mit einer Prozesszeitsteuerung, d.h. automatisch in regelmässigen Abständen ausführen können. Dies ist besonders nützlich für die kontinuierliche Datensammlung, wie in unserem Beispiel die ISS-Positionsdaten.

Wir verwenden dazu `Hop Run` für die Kommandozeilen-Ausführung in Kombination mit einer betriebssystemspezifischen Prozesszeitsteuerung.

Schritt 3.1: Vorbereitung der Hop Run Ausführung

Bevor Sie den Workflow in den Scheduler einbinden, testen Sie zunächst die Kommandozeilen-Ausführung:

Windows PowerShell

Öffnen Sie die PowerShell. Navigieren Sie zum Apache Hop Installationsordner und führen Sie `hop-run` mit den entsprechenden Parametern aus:

```
cd "C:\Apache-Hop-2.12\"
hop-run.bat --file="D:\projects\hop2025\hop-advanced\iss_data_fetch_workflow.hwf"
--project="LiveDataFetch" --level=Basic --logfile="D:\projects\hop2025\hop
-advanced\iss_data_fetch_workflow.log"
```

Linux Terminal

Öffnen Sie ein Terminal. Navigieren Sie zum Apache Hop Installationsordner und führen Sie `hop-run` mit den entsprechenden Parametern aus:

```
cd /opt/apache-hop-2.12/
./hop-run.sh --file="/home/user/hop-advanced/iss_data_fetch_workflow.hwf"
```

```
--project="LiveDataFetch" --level=Basic --logfile="/home/user/hop
-advanced/iss_data_fetch_workflow.log"
```

Stellen Sie sicher, dass der Workflow erfolgreich ausgeführt wird und die Ausgabedateien erstellt werden.

Schritt 3.2a: Windows - Task Scheduler konfigurieren

Für Windows verwenden wir für die Prozesszeitsteuerung den Task Scheduler zur automatischen Ausführung.

Task Scheduler öffnen

1. Drücken Sie `Win + R` und geben Sie `taskschd.msc` ein.
2. Klicken sie rechts unter "Aufgabenplanung" auf "Aufgabe erstellen" ("Create Task").

Aufgabe konfigurieren

1. **Name**: `ISS Data Collection`
2. **Trigger**: Erstellen Sie einen neuen Trigger der Täglich um 00:00:00 startet. Unter "Erweitert" können Sie die Aufgabe so konfigurieren, dass sie jede Minute wiederholt wird.
3. **Aktion**: "Programm starten"

Programm-Details

- **Programm/Skript**: `C:\Apache-Hop-2.12\hop-run.bat`
- **Argumente hinzufügen**:

```
--file="D:\projects\hop2025\hop-advanced\iss_data_fetch_workflow.hwf"
--project="LiveDataFetch" --level=Basic
--logfile="D:\projects\hop2025\hop-advanced\iss_data_fetch_workflow.log"
```

Passen Sie die Pfade entsprechend Ihrer Umgebung an.

Schritt 3.2b: Linux - Cron Job konfigurieren

Für Linux-Systeme verwenden wir für die Prozesszeitsteuerung `cron`.

Cron Job erstellen

Öffnen Sie die für `cron` notwendige `Crontab`-Datei:

```
crontab -e
```

Fügen Sie folgende Zeile hinzu, um den Workflow jede Minute auszuführen:

```
# ISS Data Collection - jede Minute
* * * * * /opt/apache-hop-2.12/hop-run.sh --file="/home/user/hop
-advanced/iss_data_fetch_workflow.hwf" --project="LiveDataFetch" --level=Basic
--logfile="/home/user/hop-advanced/iss_data_fetch_workflow.log"
```

Im Web gibt es einige frei verfügbare Applikationen, mit denen sich solche Cron-Expressions editieren lassen.

Schritt 3.3: Testen der Automatisierung

1. Lassen Sie die Automatisierung für ein paar Minuten laufen.
2. Überprüfen Sie, ob neue Einträge in der `output.xlsx` Datei erstellt wurden.
3. Kontrollieren Sie die Log-Dateien auf eventuelle Fehler.
4. Validieren Sie die Datenqualität der automatisch gesammelten ISS-Positionen.

Monitoring und Fehlerbehandlung

Log-Analyse

Apache Hop generiert ausführliche Logs, die für das Monitoring – insbesondere in produktiven Umgebungen – wichtig sind:

Log-Levels

- **Error**: Kritische Fehler, die die Ausführung stoppen
- **Warning**: Potentielle Probleme, die beachtet werden sollten
- **Info**: Allgemeine Informationen über den Ausführungsstatus
- **Debug**: Detaillierte Informationen für die Fehlersuche

Standardmässig sind diese Logdaten aus dem GUI-Betrieb im Ordner "audit/" gespeichert in der Datei "hopui.log" gespeichert. Alternativ kann der Speicherort auch über die Umgebungsvariable HOP_AUDIT_FOLDER gesetzt werden. Zusätzlich lassen sich in sogenannte "Pipeline Log" und "Workflow Log" Metadatenobjekte konfigurieren, mit deren Hilfe ausführliche Logdaten von Pipelines oder Workflows in Form von JSON-Objekten generiert und an eigene Pipelines weitergegeben werden können. Diese Logs sind dann über die definierten Logging-Pipelines zugänglich.

Best Practices für Logging

- Verwenden Sie konsistente Log-Level für verschiedene Umgebungen.
- Implementieren Sie strukturierte Logs für automatisierte Analyse.
- Archivieren Sie Logs regelmässig, um Speicherplatz zu sparen.

Fehlerbehandlung in Workflows

Error Handling Actions

Workflows unterstützen verschiedene Strategien zur Fehlerbehandlung:

- **Success Hops (grün)**: Werden nur bei erfolgreicher Ausführung durchlaufen
- **Failure Hops (orange)**: Werden nur bei Fehlern ausgeführt
- **Unconditional Hops (schwarz)**: Werden immer ausgeführt

Retry-Mechanismen

Für instabile Datenquellen können sogenannte Retry-Mechanismen implementiert werden: * Verwendung von `Wait` Actions zwischen Wiederholungsversuchen * Implementierung von Zählern für maximale Wiederholungen * Benachrichtigungen bei dauerhaften Fehlern

Performance-Optimierung

Pipeline-Optimierung

- **Parallelisierung**: Nutzen Sie mehrere Threads für datenintensive Transforms
- **Batch-Grösse**: Optimieren Sie die Batch-Grösse für bessere Performance
- **Memory-Management**: Überwachen Sie Speicherverbrauch bei grossen Datensätzen

Resource-Monitoring

- CPU-Auslastung während der Ausführung überwachen
- Arbeitsspeicher-Verbrauch kontrollieren
- Netzwerk-Bandbreite bei Remote-Datenquellen beachten

Abschluss

Zusammenfassung

In diesem fortgeschrittenen Tutorial haben Sie gelernt:

- **Workflows** als Orchestrierungswerkzeug für komplexe Datenverarbeitungsprozesse zu verstehen und zu erstellen.
- **Actions** und deren Orchestrierung in komplexen Abläufen kennen.
- **Hop Run** für die Kommandozeilen-basierte Automatisierung zu verwenden.
- Mit **REST APIs als Echtzeit-Datenquellen** zu arbeiten und diese in Datenverarbeitungspipelines zu integrieren.
- **Automatisierung** über Prozesszeitsteuerungen implementieren

- **Monitoring und Fehlerbehandlung** für robuste Produktionsumgebungen aufzusetzen

Best Practices

Beachten Sie auch die Best Practices im Anhang des ersten Apache Hop-Tutorials auf OpenSchoolMaps. Darüber hinaus lassen sich insbesondere für die Produktion folgende Best Practices anführen:

- **Namenskonventionen**: Benennen Sie die projektspezifische Variablen mit einem Präfix, z.B. "WORKFLOWS_TUTORIAL_...".
- **Versionskontrolle**: Verwenden Sie Git oder ähnliche Tools für Ihre Hop-Projekte.
- **Dokumentation**: Dokumentieren Sie Ihre Workflows und Pipelines ausführlich.
- **Backup-Strategien**: Implementieren Sie regelmässige Backups Ihrer Hop-Konfigurationen.
- **Security**: Verwenden Sie sichere Verbindungen und verschlüsselte Passwörter.

Noch Fragen? Sehen Sie auch "Kontakt" auf OpenSchoolMaps!