# TraceHistoryDownloader

Das Programm „TradeHistoryDownloader“ ermöglicht es dir, automatisch die Handelshistorie von mehreren Anbietern auf der Website MQL5.com herunterzuladen. Hier ist eine einfache Erklärung, was das Programm macht:

1. **Einstellen des Speicherortes**: Das Programm legt fest, dass alle Daten im Verzeichnis C:\tmp\mql5 gespeichert werden sollen. Es erstellt dieses Verzeichnis und auch ein Unterverzeichnis für Konfigurationen (conf), falls diese nicht vorhanden sind.
2. **Speichern der Login-Daten**: Beim ersten Start fragt das Programm nach deinem Benutzernamen und Passwort für MQL5.com und speichert diese Informationen sicher in einer Konfigurationsdatei (conf.txt). Das heißt, du musst die Login-Daten nur einmal eingeben.
3. **Login bei MQL5.com**: Das Programm verwendet die gespeicherten Login-Daten, um sich automatisch bei MQL5.com anzumelden. Es wartet, bis die Login-Seite vollständig geladen ist, und gibt dann deinen Benutzernamen und dein Passwort ein.
4. **Liste der Handelssignale durchgehen**: Nach erfolgreichem Login geht das Programm auf die Seite mit den Handelssignalen, auf der alle Signal-Anbieter aufgelistet sind. Es findet die Links zu allen Anbietern und geht diese nacheinander durch.
5. **Handelshistorie herunterladen**: Für jeden Anbieter navigiert das Programm zur Seite mit der Handelshistorie und klickt auf den „Export to CSV“-Link, um die Handelshistorie als CSV-Datei herunterzuladen. Die Datei wird im C:\tmp\mql5-Verzeichnis gespeichert.
6. **Speicherung der Dateien**: Jede heruntergeladene Datei wird so umbenannt, dass sie den Namen des entsprechenden Signal-Anbieters enthält. Zusätzlich wird eine Kopie der Datei im Unterverzeichnis „download“ im selben Hauptverzeichnis abgelegt.
7. **Automatisierung der nächsten Anbieter**: Nachdem die Datei für einen Anbieter heruntergeladen wurde, kehrt das Programm zur Hauptseite zurück, um den nächsten Anbieter zu verarbeiten.
8. **Beenden des Programms**: Nachdem alle Anbieter bearbeitet wurden, schließt das Programm den Browser. Alle heruntergeladenen Dateien bleiben erhalten.

Dieses Programm hilft dir, die Handelshistorie von mehreren Signal-Anbietern automatisch herunterzuladen und zu speichern, ohne dass du dies manuell für jeden Anbieter tun musst.

Hier die Beschreibung von Claude:

Das Programm ist ein Java-basierter Web-Scraper, der automatisch Handelsdaten (Trade History) von der MQL5-Plattform herunterlädt. Hier sind die Hauptfunktionen:

1. Anmeldung und Konfiguration:

- Speichert Benutzername und Passwort für MQL5 in einer Konfigurationsdatei

- Wenn keine Konfigurationsdatei existiert, fragt es nach den Login-Daten

- Richtet notwendige Verzeichnisse für Downloads ein

2. Webautomatisierung:

- Verwendet Selenium WebDriver mit Chrome, um die MQL5-Website zu steuern

- Führt einen automatischen Login auf MQL5.com durch

- Navigiert zur Liste der Signal-Provider (Handelssignal-Anbieter)

3. Datensammlung:

- Geht durch alle Signal-Provider auf der Seite

- Für jeden Provider:

- Öffnet die Detailseite des Providers

- Navigiert zum "Trading history" Tab

- Lädt die Handelshistorie als CSV-Datei herunter

- Speichert die Datei mit dem Namen des Providers

- Erstellt eine Kopie im Download-Unterverzeichnis

4. Dateiverwaltung:

- Organisiert die heruntergeladenen Dateien in zwei Verzeichnissen:

- Ein Hauptverzeichnis für die aktuellen Dateien

- Ein "download" Unterverzeichnis für Backup-Kopien

- Behandelt Dateinamen sicher, indem ungültige Zeichen entfernt werden

Das Programm ist besonders nützlich für:

- Trader, die Handelsdaten von verschiedenen Signal-Providern analysieren möchten

- Automatische Datensammlung für Marktanalysen

- Backup und Archivierung von Handelshistorien

Das Programm enthält auch verschiedene Sicherheitsmaßnahmen wie:

- Wartezeiten für das Laden von Seiten

- Fehlerbehandlung für nicht gefundene Elemente

- Verschiedene Methoden, um UI-Elemente zu finden

- Sicheres Herunterfahren des Browsers

Um das Programm mit Selenium-Integration zu erstellen, kannst du den folgenden Prompt verwenden:

**Prompt für Java-Programmierung mit Selenium-Integration**

Erstelle ein Java-Programm, das mit Selenium arbeitet, um automatisch die Handelshistorie von mehreren Anbietern von der Webseite MQL5.com herunterzuladen. Folgende Anforderungen sind zu beachten:

1. **Einstellungen zum Root-Verzeichnis:**
   * Das Hauptverzeichnis wird unter C:\tmp\mql5 gesetzt.
   * Ein Unterverzeichnis conf wird erstellt, um eine Konfigurationsdatei (conf.txt) zu speichern.
   * Falls conf noch nicht existiert, muss es erstellt werden.
2. **Login-Daten:**
   * Das Programm liest Login-Daten (Benutzername und Passwort) aus der Datei conf.txt, die im Verzeichnis C:\tmp\mql5\conf gespeichert ist.
   * Falls die Datei conf.txt noch nicht existiert, sollen die Login-Daten interaktiv abgefragt und anschließend in der Datei gespeichert werden.
3. **Selenium-Integration:**
   * Verwende Selenium, um auf MQL5.com zu navigieren, dich anzumelden und die Daten herunterzuladen.
   * Nutze den ChromeDriver mit angepassten Downloadpfaden.
   * Pfad zum ChromeDriver ist C:\tools\chromedriver.exe.
   * ChromeOptions sollen verwendet werden, um den Download-Ordner zu konfigurieren, damit alle heruntergeladenen Dateien im Verzeichnis C:\tmp\mql5 gespeichert werden.
4. **Automatisierter Ablauf:**
   * Das Programm soll die Root-Seite mit den Signal-Providern öffnen und alle Anbieter-Links nacheinander durchgehen.
   * Bei jedem Anbieter soll der "Trade History"-Tab angeklickt und der "Export to CSV"-Link gefunden werden.
   * Die CSV-Datei mit der Handelshistorie soll heruntergeladen und im Root-Verzeichnis unter einem Namen gespeichert werden, der den Signal-Providernamen beinhaltet.
   * Eine zusätzliche Kopie der Datei soll im Unterverzeichnis C:\tmp\mql5\download gespeichert werden.
5. **Endverarbeitung:**
   * Der Browser soll nach der Durchführung aller Aktionen geschlossen werden, ohne dass die heruntergeladenen Dateien gelöscht werden.

Das Programm sollte mit der neuesten Selenium-Version und dem entsprechenden ChromeDriver arbeiten. Es sollte klar strukturiert und benutzerfreundlich sein, insbesondere bei der ersten Einrichtung der Login-Daten.

# MqlAnalyser

Die Anforderungen an den MqlAnalyser, die Sie ursprünglich gestellt hatten, lauteten wie folgt:

* Das Programm soll einen Rootpfad haben, der auf c:\tmp\mql5 gesetzt wird.
* Im Verzeichnis c:\tmp\mql5 befindet sich ein Unterverzeichnis download.
* In diesem download-Verzeichnis befinden sich verschiedene Tradelisten von Signalprovidern.
* Das Programm soll alle Tradelisten einlesen und die Daten in einer übersichtlichen Tabelle darstellen.
* In der Tabelle sollen folgende Spalten angezeigt werden:
  1. Eine fortlaufende Nummer.
  2. Der Name des Signalproviders.
  3. Die Anzahl der Trades.
* Die Tabelle soll scrollbar sein und die Daten der einzelnen Signalprovider anzeigen.
* Sie haben vorgeschlagen, die Bibliothek SWT zu verwenden, um die Tabelle darzustellen.

Darüber hinaus haben Sie eine Beispieldatei hochgeladen, damit ich das Format der Tradelisten analysieren kann, um das Programm entsprechend zu gestalten. Ich habe basierend auf diesen Anforderungen einen Java-Code geschrieben, der die Dateien liest und eine GUI-Tabelle mit SWT erstellt.

com.tradinganalyzer/

│

├── main/

│ └── SignalProviderTable.java // Hauptklasse mit main-Methode

│

├── data/

│ ├── DataManager.java // Verwaltet das Laden und Speichern der Daten

│ └── ProviderStats.java // Datenmodell für einen Signal Provider (bereits vorhanden)

│

├── ui/

│ ├── MainFrame.java // Hauptfenster der Anwendung

│ ├── DetailFrame.java // Detailansicht eines Signal Providers

│ ├── TradeListFrame.java // Fenster für die Trade-Liste

│ └── EquityCurvesFrame.java // Vergleichsfenster für Equity-Kurven

│

├── components/

│ ├── MainTable.java // Haupttabelle mit allen Signal Providern

│ ├── SearchPanel.java // Suchleiste

│ ├── StatsPanel.java // Panel für Statistiken

│ ├── ChartsPanel.java // Panel für Charts

│ └── TradeListTable.java // Tabelle für die Trade-Liste

│

├── models/

│ └── HighlightTableModel.java // Tabellenmodell mit Suchhervorhebung

│

├── renderers/

│ └── HighlightRenderer.java // Renderer für Suchhervorhebung

│

└── utils/

└── ChartFactory.java // Factory für Chart-Erstellung

Benötigte externe Bibliotheken:

- JFreeChart (jfreechart-1.5.3.jar)

- JCommon (jcommon-1.0.24.jar)

Hauptfunktionen der einzelnen Pakete:

1. main/

- Enthält den Einstiegspunkt der Anwendung

- Koordiniert die Hauptkomponenten

2. data/

- Datenverarbeitung und -verwaltung

- Modellklassen

3. ui/

- Alle Hauptfenster (JFrame-Klassen)

- Verantwortlich für das Layout der Fenster

4. components/

- Wiederverwendbare UI-Komponenten

- Spezialisierte Panels und Tabellen

5. models/

- Datenmodelle für UI-Komponenten

- Tabellenmodelle

6. renderers/

- Spezielle Renderer für Tabellen

- Verantwortlich für die Darstellung der Daten

7. utils/

- Hilfsfunktionen und Utilities

- Factories für komplexe Objekte

Package-Beziehungen:

- main verwendet data und ui

- ui verwendet components, models und utils

- components verwendet models, renderers und utils

- models und renderers sind weitgehend unabhängig

- alle Pakete können auf data zugreifen

Zur Trendwertberechnung:

### **Was wurde geändert?**

✅ **Sanftere Berechnung:**

* Statt einer **strikten Steigerung** (profit1 < profit2 < profit3) wird geprüft, ob **der Durchschnitt der letzten beiden Monate höher ist als der älteste Wert**.

✅ **Toleranz von 0.5% eingeführt:**

* Falls die Werte leicht schwanken, aber dennoch tendenziell steigen, wird trotzdem ein **Trendwert berechnet**.

✅ **Fehlertoleranz verbessert:**

* Falls ein Wert nicht richtig geparst werden kann, bricht die Methode nicht mehr ab, sondern gibt eine Warnung aus.

### **Beispielwerte vorher/nachher**

| **Monat** | **Gewinn (%)** | **Trendwert vorher** | **Trendwert jetzt** |
| --- | --- | --- | --- |
| Monat 1 | 2.5 | 0.0 | 0.0 |
| Monat 2 | 3.1 | 0.0 | 0.0 |
| Monat 3 | 2.8 | 0.0 | **berechnet!** |
| Monat 4 | 3.2 | 0.0 | **berechnet!** |

**Vorher:** Sobald eine kleine Schwankung auftritt, wird **immer 0.0** zurückgegeben.  
**Jetzt:** Kleine Schwankungen sind erlaubt, solange der **durchschnittliche Trend positiv ist**.

Diese Änderung sollte dazu führen, dass **der Trendwert nicht mehr ständig 0.0 ist**. Teste es aus! 😊