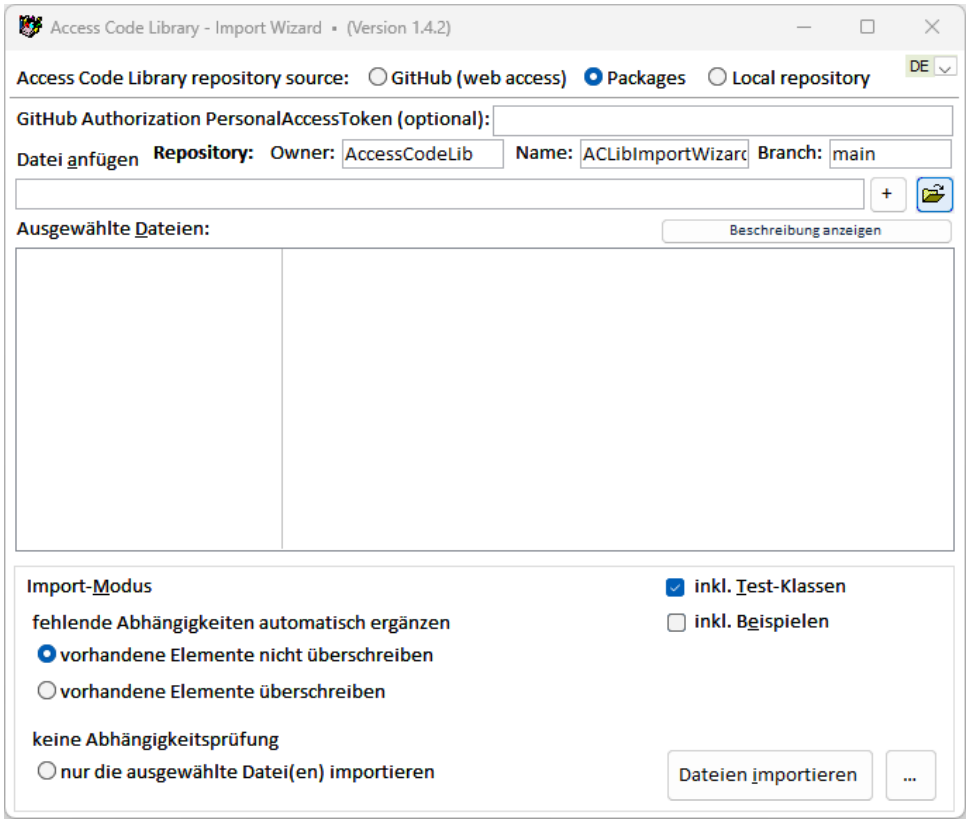
ACLib Import Wizard

Wiederverwendbaren Code nutzen



https://github.com/AccessCodeLib/ACLibImportWizard

Aufgaben

- Import der benötigten Codemodule
  - Abhängigkeiten auflösen (auch mehrstufig)
- Erforderliche Verweise setzen
- Code ausführen
  - z. B. Tabelle erzeugen, Add-In aufrufen, …

Geschichte

2010 Lokal gespeicherte Dateien
Synchronisation mit Subversion über Windows-Client

2023 Direkter Download über GitHub-API
Import aus AccessCodeLib Repository

2025 Package File – **Katalog-Idee reloaded**
Import von Codemodulen aus verschiedenen GitHub Repositories

Konzept

Ein Kommentarblock im Codemodul liefert die Daten für den Import-Vorgang.

CodeLib-Block

```
'<codelib>
'  <file>data/FilterStringBuilder.cls</file>
'  <license>_codelib/license.bas</license>
'  <use>text/StringCollection.cls</use>
'  <use>data/SqlTools.cls</use>
'  <test>_test/data/FilterStringBuilderTests.cls</test>
'  <example>data/FilterStringBuilder_Examples.bas</example>
'</codelib>
```

Tags

`file`	Export-Pfad
`description`	Beschreibung des Moduls
`use`	Benötigtes Codemodule
`ref`	Benötigter Verweis auf COM-dll/tlb
`test`	Unit-Test
`example`	Beispiel-Codemodul
`execute`	Prozedur wird am Ende ausgeführt
`package`	Kennzeichnung einer Package-Datei – ähnlich <file>, Package-Datei wird nicht in die Anwendung importiert

Katalog-Idee (Stand vor AEK)

Auslöser für Add-In-Erweiterung:
AWF - Code Catalog for Access / VBA - Making Code Sharing Easy

Package File

- Katalog Package File
  Verweis auf Package des jeweiligen Repository
- Repository Package File
  Liste der benötigen Codemodule

Katalog Package File

Wird von „Katalog-Team“ verwaltet. Ersteintrag erfolgt durch Repository-Betreiber über Pull-Request.
Datenbasis für Katalog: Es enthält Namen, Verweis auf Repository Package, Beschreibung, Stichwörter für Suche usw.

```
<codelib>
  <package>VBA-MicrosoftGraph</package>
  <description>…</description>
  <use>
      %GITHUB%:…/VBA-MicrosoftGraph@master/VBA-
      MicrosoftGraph.package
  </use>
</codelib>
```

Repository Package File

Liegt im Repository des Code-Anbieters.

Enthält eine Liste aller erforderlichen Codemodule.

```
<codelib>
  <package>VBA-MicrosoftGraph</package>
  <description>…</description>
  <use>/Src/AttachmentHelpers.bas</use>
  …
  <use>/authenticators/IWebAuthenticator.cls</use>
  <use>/authenticators/OAuth2Authenticator.cls</use>
  <ref>
      <name>MSXML2</name><major>6</major><minor>0</minor>
      <guid>{F5078F18-C551-11D3-89B9-0000F81FE221}</guid>
  </ref>
</codelib>
```

Katalog-Idee (nach der AEK)

Hauptfragestellungen

- Wie wird man im Katalog etwas finden? (Nutzer)
- Wie funktioniert das mit GitHub? (Anbieter)
- Wer macht mit?

Ideen für Katalog-Aufbau

- Package files einlesen und in lokaler Tabelle speichern
  - Suche über Access-Formular (falls Access-Add-In verwendet wird)
- Markdown-Datei aus GitHub für Detailbeschreibung verwenden
  - Vielleicht Tag <md> innerhalb <description>?
- Extra GitHub-Organisation verwenden
  - Mehrere Administratoren (Ausfallssicherheit, Verfügbarkeit)
- 2 Branches
  - Main: „geprüfter“ Zweig
    - Grundregeln wie option explicit, kompilierbar, einheitliche Benennung innerhalb der Module aus dem Package (Anm.: durchgehend einheitliche Benennungsregeln über alle Anbieter ist niemals umsetzbar.)
    - Team prüft im 2. „Antrags“-Branch und gibt dann über Pull-Request frei
      ⇨ Nachvollziehbar, GitHub Actions können laufen (Automatisierte Prüfung möglich?)
  - Staging (oder wie auch immer benannt): ungeprüfter Zweig, Zugriff aus Katalog-Add-In möglich (extra Checkbox o.ä. für Aktivierung + Kennzeichnung der Pakete)
    Code-Anbieter macht Pull-Request auf staging-Branch.
- Mit Open Source starten
  - Einfacher, da keine Downloadrechte o. ä. zu regeln sind
- Bewertungssystem
  - Könnten „GitHub-Sterne“ verwendet werden?

Weitere Fragen

- Wie werden zu bezahlende Codesammlungen o. ä. bereitgestellt?
- Wer legt fest, was in den Katalog darf?
- Wie verhindert man Schadcode?