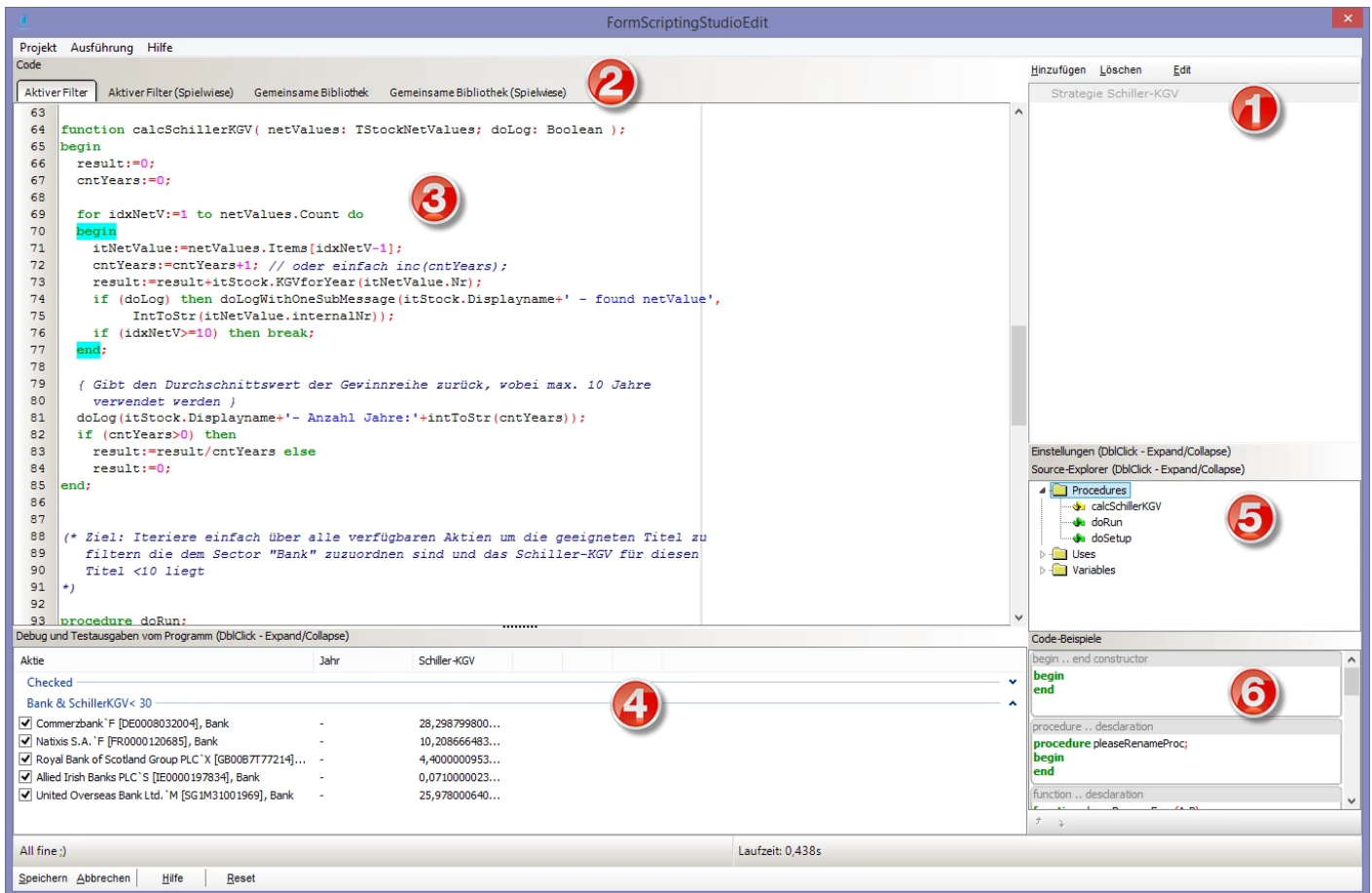


# Scripting-Studio-Editor und Bedienung



## (1) Auswahl des aktiven Scriptes

Es werden beim Laden eines Filter immer gleichzeitig drei Skripte geladen, um das spätere Handling deutlich zu vereinfachen:

Speicherung unter	Titel	Zielstellung	Sichtbarkeit
-------------------	-------	--------------	--------------

\Daten\Scripts\<Name>.script	Aktiver Filter	Produktiver Filter	Ja (ab 2.6) in den Filtermasken im Frontend außerhalb des Scripting-Studios
\Daten\Scripts\<Name>.script-dev	Aktiver Filter ("Spielwiese")	Test/Entwicklungsversion des produktiven Filters für Weiterentwicklungen, ohne den produktiven "Filter" kaputt zu entwickeln. Jeder Filter besitzt immer einen produktive und eine Entwicklungs(Sandbox)-Version. Eine Entwicklungsversion kann zur Produktiv-Version gemacht werden (wird hier kopiert) über das Hauptmenü "Projekte / Entwicklungsversion live nehmen".	Nein nur im Scripting-Studio
\Daten\Scripts\Common.script	Gemeinsame Bibliothek	Hier sollte gemeinsam genutzte Funktionen und Hilfsroutinen ausgelagert werden, um effektiv in allen Filter zu arbeiten und Redundanzen zu vermeiden. Da auch das gegenseitige Einbinden der Standard-Scripte erlaubt ist, dient diese Sonderbehandlung aber der Effizienzsteigerung, um hier gezielte wiederverwendbare Funktionen und Procedures zu verlagern, um die eigentlichen Filter schlank und effizient zu halten.	Nein nur im Scripting-Studio
\Daten\Scripts\Common.script-dev	Gemeinsame Bibliothek ("Spielwiese")	Test/Entwicklungsversion der gemeinsam genutzten Bibliothek	Nein nur im Scripting-Studio

## Hinweise zu Datensicherung und -Ablage

Da in Scripten sehr viel Zeit und Aufwand gesteckt werden kann, ist die Datensicherung nicht zu unterschätzen. Aktuell umgesetzt ist daher die Erstellung einer automatisches Backup-Version mit <Name>-backup-YYYYMMDD-HHMM bei jedem Speichervorgang (<F2>). Zu einem späteren Zeitpunkt ist auch die Synchronisation mit eigenen Dropbox-Instanzen vorgesehen. Aktuell werden diese Sicherungsdateien zu einem Script wieder gelöscht, wenn eine Script-Version "Verifiziert" wird (siehe Kontextmenü zu einem Filtereintrag (r.Maustaste über einen Filternamen)).

# (1a) Einstellungen

Unterhalb der Script-Auswahl findet sich ein kleines ein/ausklappbares Pannel "Einstellungen".

Für weitere Ausbaustufen wurde sofort ein Einstellungs-Grid angelegt, was praktisch unendlich fortgeführt werden kann durch seine Scrollfähigkeit. Beim Start sind nur wenige Einstellungen vorhanden:

- - Codebeispiele ausführbar: Dies bezieht sich auf die unter (4) gezeigten Code-Beispiele, die so beeinflusst werden. Im Standard sind die Beispiele nur als Hinweise für die abzubildende Mindeststruktur gedacht. Für die Sandbox ist es aber wahrscheinlich interessant auch lauffähige Codebeispiele zu haben. Die Einstellung wird geändert durch Klick in die Wert-Spalte, wo sich dann eine entsprechende Drop-Down-Box öffnet
  - Code-Folding nutzen: Im Programm können dann alle begin..end - Blöcke ein- und ausgeklappt werden. Dies erleichtert das Handling bei längeren Skripten.

## (2) Bibliotheks-Zugriff

Um die Entwicklung zu unterstützen werden zwei Grundkonzepte mit unterstützt:

- Unterscheidung zwischen Entwicklung (Sandbox) und aktuell produktivem Code
- Unterstützung von Bibliotheken insb. auch gemeinsam genutzter Bibliotheken zwischen den unterschiedlichen Skripten

Mit Wechsel zwischen den Reitern wird zwischen den verschiedenen dahinterliegenden Code-Fragmenten gewechselt.

## (3) Code

Hier befindet sich der Quelltexteditor der analog einem Notepad funktioniert mit den vorhandenen Tastenkombinationen (Strg-C-Kopieren, Strg-V-Einfügen etc.). Die Besonderheit ist hier, dass mit dem vorhandenen Code ein automatische Code-Highlighting erfolgt d.h. Schlüsselwörter oder Strukturen werden automatisch hervorgehoben in Schriftart und Form. So werden Kommentare beispielsweise immer kursiv/blau dargestellt.

Für die Entwicklung sind besonders folgende Funktionen relevant:

- Strg + "." zeigt die aktuell gültigen Funktionen/Konstanten/Variablen/Methoden an
- ( ) nach einer Procedure oder Funktion zeigt alle gültigen Werte an

Übergreifend unabhängig vom Quell-Code kann mit:

- F8 - Der Quellcode überprüft werden. Fehler werden in der Statusleiste direkt angezeigt und der Cursor springt automatisch zur Fehlerstelle
- F9 - Der Quellcode wird überprüft und danach ausgeführt. Im Datenverzeichnis wird im Erfolgsfall unter Unitname.PSC der übersetzte Quellcode angelegt und ab diesem Zeitpunkt kann dieser Filter als Bibliothek in anderen benutzt werden über "uses <Unitname>"

## (4) Ausgabe

Dies ist eigentlich eines der Highlights der Umsetzung und ist auch bisher nur eine erste Version, die spätere deutlich ausgebaut wird. Ziel ist es die Ergebnisse aus einem Filter-Lauf übersichtlich und performant und unmittelbar darzustellen.

Die Darstellung wird dabei durch den Code festgelegt d.h. welche Spalten, welcher Detaillierungsgrad, welche Sortierungen und welche Darstellungs-Gruppierungen genutzt werden sollen.

In der Ausgabe wird auf eine spezielle zugreifbare Komponente zugegriffen die folgende Eigenschaften unterstützt:

- Nutzung von Gruppen, um Ergebnisse im Filtervorgang zu gruppieren. Im Standard sind aktuell 3 Gruppen angelegt
- Ein/Ausklappen von Gruppen
- Nutzung von Checkbox-Markierungen
- Mehrspaltiges Layout, womit Informationen anders und strukturiert ausgegeben werden können

## (5) Sourcecode-Explorer

Der Source-Explorer versucht zeitnah synchron zum Quellcode die Struktur des Skriptes wiederzugeben, wobei unterschieden wird nach

- procedures: Alle Funktionen und Procedures im Code, wobei Funktionen mit gelb markiert werden und procedures mit grün

- uses: Alle eingebundenen Fremdbibliotheken. Dies kann die Common-Bibliothek sein (siehe (1)) oder alle anderen vorhandenen Skripte
- variables: Alle nicht lokalen Variablen (innerhalb einer procedure oder function definiert) werden hier gezeigt.

Mit Doppelklick auf einen Eintrag springt der Cursor automatisch an die zugehörige Programmcode-Position.

## (6) Code-Snippet-Beispiele zur Übernahme

Dies sind aktuell nicht erweiterbare Code-Beispiele (TODO: Pflege sollte außerhalb der IDE möglich sein und auch über Updates ermöglicht werden z.B. ScriptStudio.MOD).

Mit Doppelklick auf einen Eintrag wird das Codebeispiel übernommen. Aktuell existiert eine Einstellung unter (1), um die Beispiele entweder nur als Struktur oder lauffähig zu verwenden.

---

Revision #4

Created 19 June 2022 07:49:26 by Jens Werschmoeller

Updated 19 June 2022 07:56:21 by Jens Werschmoeller