# SAS Studio – Eine Oberfläche für Jedermann

Sebastian Reimann
viadee Unternehmensberatung GmbH
Anton-Bruchausen-Str. 8
48147 Münster
sebastian.reimann@viadee.de

**Zusammenfassung**

Seit einigen Jahren bietet SAS neben den klassischen Oberflächen SAS Display Manager und SAS Enterprise Guide eine zusätzliche Oberfläche für die Programmierumgebung SAS an. SAS Studio arbeitet hierbei webbasiert und ist somit plattformunabhängig und auf fast jedem Endgerät lauffähig. In dem Vortrag wird ein Überblick über Funktionsumfang und Nutzungsmöglichkeiten für verschiedene Anwendungsszenarien gegeben.
So kann SAS Studio gleichermaßen vom klassischen SAS Programmierer als Entwicklungsumgebung für SAS Base Code wie auch vom Fachanwender als grafisches Modellierungstool für SAS Anwendungen genutzt werden. Zusätzlich bieten Administrationsmöglichkeiten für Code Snippets und eigene Tasks die Möglichkeit, SAS Studio individuell an die Bedürfnisse der Nutzer einer Organisation anzupassen.

**Schlüsselwörter:** SAS Studio, SAS Enterprise Guide, SAS Display Manager

# 1 Motivation

Mit der Einführung von SAS 9.4M1 wurde im März 2014 auch die erste Version von SAS Studio veröffentlicht. Während am Anfang noch der Fokus auf der Bereitstellung einer browsergestützten Zugriffsmöglichkeit auf SAS Tabellen und SAS Programmen stand, hat sich in der Zwischenzeit das SAS Studio zu einem vollwertigen Entwicklungs- und Analysewerkzeug weiterentwickelt.

Bei einer klassischen, lokalen SAS Base Installation haben inzwischen die meisten Anwender meist völlig unbemerkt auch eine lokale SAS Studio Installation auf dem Rechner. In diesem Fall wird mit dem starten von SAS Studio ein lokaler Application Server gestartet, der die SAS Studio Software auf einer lokalen URL (z.B. http:// http://localhost:63932) bereitstellt.

In einer größeren, verwalteten Umgebung mit Enterprise BI Server und SAS Visual Analytics wird die SAS Studio Software als zusätzliche Mid-Tier Komponente installiert, so dass die Anwendung von allen Anwendern ohne jegliche Client-Installation genutzt werden kann. Dabei kann die Konfiguration der Umgebung zentral verwaltet und an die Bedürfnisse der Organisation angepasst werden.

Und zum Schluss haben die Anwender, die keinen Zugriff auf eine lizensierte SAS Umgebung haben, mit der SAS University Edition die Möglichkeit, SAS unabhängig zu lernen und auszuprobieren. Auch hierbei kommt die Software SAS Studio zum Einsatz. SAS stellt auf der eigenen Homepage ein fertig konfiguriertes VirtualBox Image kostenfrei bereit, auf welches mittels einer integrierten SAS Studio Installation zugegriffen wird. Dieses kann von jedermann zu Evaluations- und Ausbildungszwecken genutzt werden.

Die Verbreitung von SAS Studio zeigt die Relevanz des Tools. Aus diesem Grund soll in diesem Artikel auf grundlegende Funktionen von SAS Studio eingegangen und deren Nutzung gezeigt werden.

## 2 Der Einstieg

Im Rahmen dieser Beschreibung wird die SAS University Edition in der Version 3.71 genutzt. Diese steht über die Homepage der Firma SAS zum Download zur Verfügung. Analog kann natürlich auch jede andere verfügbare Variante des SAS Studio genutzt werden.

Der Start der Anwendung erfolgt über das bereitgestellte VirtualBox Image. Sobald die virtuelle Maschine gestartet ist, kann ein beliebiger Browser genutzt werden, um das SAS Studio über die URL http://localhost:10080 zu öffnen.

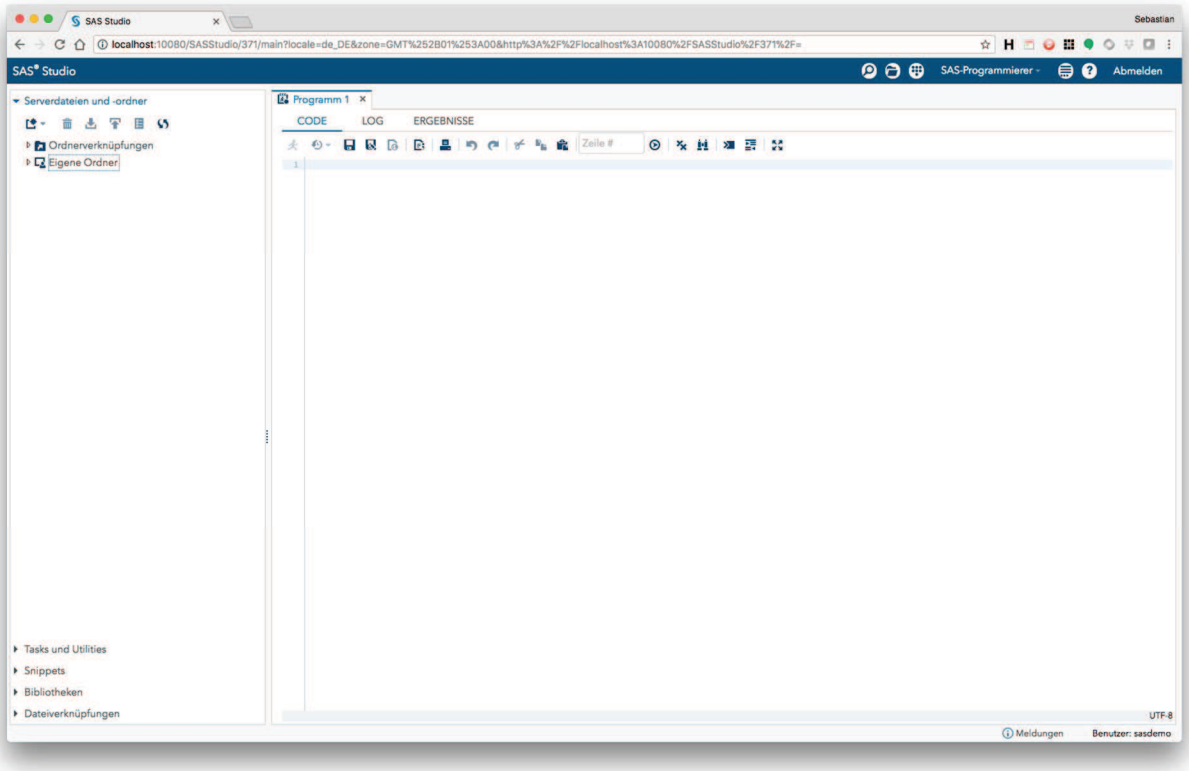
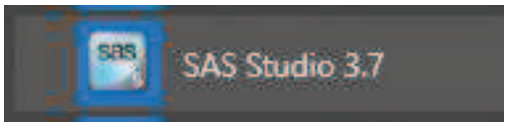



**Abbildung 1:** Startansicht SAS Studio

Bei einer klassischen SAS Base Installation unter Windows findet man das SAS Studio im Startmenü unter folgendem Symbol:




Nach dem Start der Anwendung, welcher durchaus einige Minuten dauern kann, öffnet sich automatisch der Standard-Browser mit dem Link auf die SAS Studio Instanz. Sollte der Browser versehentlich geschlossen worden sein, kann dieser über das Status-Icon rechts unten neben der Uhr problemlos wieder gestartet werden. Dies ist hilfreich, das SAS Studio hier einen zufällig gewählten Port für den Web Application Server nutzt.

## 2.1 Direkteinstieg für Programmierer

Die Standard-Startansicht von SAS Studio ist die Perspektive das SAS Programmierers. In dieser Perspektive wird beim Start automatisch ein neues Programmfenster geöffnet, in welchem direkt mit der SAS Programmierung begonnen werden kann. Über die drei Reiter „Code“, „Log“ und „Ergebnisse“ kann schnell zwischen den jeweiligen Ansichten hin- und hergesprungen werden.

Über die Symbolleiste lassen sich die Standardfunktionen eines Code-Editors aufrufen:


Das laufende Männchen (#1) schickt den Code (bzw. falls nur ein Teilbereich markiert ist, nur den Teil-Code) ab und öffnet danach die Ergebnisansicht. Im Fehlerfall wird das SAS Log geöffnet. Die kleine Uhr (#2) bietet die Möglichkeit, auf bereits ausgeführte Programmläufe im schreibgeschützten Modus nochmals zuzugreifen. Dies ist schön, da sowohl der abgeschickte Programmcode, als auch das SAS Log und das Ausführungsergebnis noch einmal angezeigt werden. Anschließend folgen Symbole zum Speichern (#3), Speichern unter (#4) oder zum Erstellen einer Programmzusammenfassung im HTML Format (#5).

Über das Blatt mit dem symbolisierten SAS Code (#6) lässt sich der Programmcode als Code Snippet speichern. Snippets haben den Vorteil, dass diese aus dem linken Baum-Menü per Drag&Drop bzw. per Doppelklick in den Code eingefügt werden können. So lassen sich besonders leicht Code-Vorlagen erstellen, die an verschiedenen Stellen im Programm später wieder verwendet werden können.

Die folgenden Symbole 7 bis 13 bedürfen sicher keiner weiteren Erklärung. Hier geht es um die Druckausgabe des SAS Codes und um Standardfunktionen wie Rückgängig machen oder Ausschneiden, Kopieren und Einfügen. Gleiches gilt für die Sprungfunktion zu einer bestimmten Zeile im Code. Über das Doppel-X (#14) lässt sich der gesamte Code-Bereich löschen, das Fernglas öffnet die Suchfunktion auf der aktuellen Seite.

Interessant ist das folgende Symbol mit der spitzen Klammer nach rechts (#14). Dieses startet das Programmfester im interaktiven Modus. Dieser ist für all die Prozeduren von Vorteil, die interaktive Elemente enthalten. Klassisch ist dies die Prozedur IML, aber auch Prozeduren wie PROC DATASETS und PROC SQL können vom interaktiven Modus profitieren. Um die Vorteile des interaktiven Modus zu verstehen, ist es hilfreich, die Arbeitsweise des regulären Übertragungsmodus zu verstehen. Wird ein SAS Programm ausgeführt, so wird im Standard dieser Programmcode immer mit einem QUIT; RUN; abgeschlossen. Dies entspricht übrigens auch der Arbeitsweise des SAS Enterprise Guides. So wird sichergestellt, dass keine Prozedur über einen längeren Zeitraum aktiv bleibt. Vergleicht man dies jedoch mit dem klassischen SAS Display Manager, so unterscheidet sich die Ausführung. Im Display Manager wird bei partieller Ausführung immer nur der markierte Code ausgeführt. Eine Code-Ergänzung findet nicht statt. Der interaktive Modus dient dazu, diese Art der Ausführung des Display Managers auch im SAS Studio zur Verfügung zu stellen.

Am besten wird der Unterschied anhand eines kleinen Beispiel-Programms deutlich. Betrachten wir folgendes Programm:

```
proc sql;
  select * from sashelp.class where sex = 'M';
  select * from sashelp.class where sex = 'F';
quit;
```

Führt man das Programm im klassischen Modus aus, werden zwei Ausgabe-Tabellen als Ergebnis generiert. Möchte man im Rahmen von Analysen aber nur die zweite der beiden Abfragen ausführen, muss man zunächst die erste Abfrage auskommentieren, um das Programm auszuführen. Kommt man (wie vom SAS Display Manger gewohnt) auf die Idee, zunächst nur die erste Zeile zu markieren und auszuführen und danach die dritte und vierte zu markieren und auszuführen, so würde beim zweiten SUBMIT eine Fehlermeldung ausgegeben. Dies liegt daran, dass im klassischen Modus beim ersten SUBMIT neben der markierten Zeile auch ein implizites QUIT; RUN; mit ausgeführt wurde. Damit wurde die Prozedur beendet und beim zweiten SUBMIT beginnt der Code nicht mit einer Prozedur, sondern direkt mit dem gewünschten SQL. Gleichzeitig fällt auf, dass das SAS Log der ersten Ausführung von der zweiten Ausführung automatisch überschrieben wurde.

Führt man das Programm hingegen im interaktiven Modus aus, können einzelne Zeilen markiert und einzeln ausgeführt werden. Die Prozeduren bleiben aktiv und es können interaktiv weitere Statements zur Prozedur hinzugefügt werden. Das SAS Log wird mit jeder weiteren Ausführung ergänzt, so dass die Verarbeitungsweise mit der des SAS Display Managers übereinstimmt. Mit dem Beenden des interaktiven Modus wird auch das SAS Log und die Ergebnisansicht geleert. Über die Einstellungen des SAS Studios lassen sich übrigens auch konfigurieren, dass neue Programme automatisch im interaktiven Modus gestartet werden.

Die letzten beiden Symbole in der Reihe dienen übrigens des automatischen Code-Formatierung und der Maximierung des Editor-Fensters. Die Regeln für die automatische Code-Formatierung lassen sich über die Einstellungen der Anwendung (Editor-Optionen) anpassen. So kann bspw. festgelegt werden, ob Einrückungen mittels Tabulator oder mittels Leerzeichen erfolgen. Genauso lässt sich über die Editoroptionen ein- bzw. ausschalten, ob Autovervollständigungs-Funktionen aktiviert sein sollen, oder nicht.

Das Autovervollständigen unterstützt zum einen durch die direkte Anzeige von Syntax-Beschreibungen zur den SAS Prozeduren. Auf der anderen Seite hilft es aber auch bei der Auswahl von Tabellen, da bspw. auch die Inhalte einer SAS Bibliothek mit vorgeschlagen werden, wenn für eine Prozedur eine Eingabetabelle gewählt werden soll. Im Weiteren erfolgt eine kontextsensitive Unterstützung, indem an bestimmten Prozedurstellen die möglichen Schlüsselwörter vorgeschlagen werden, die an der jeweiligen Stelle zulässig sind.

## 2.2 Programmaufbau

Im ersten Schritt hatten wir den Direkteinstieg eines SAS Programmierers betrachtet. Tritt man jedoch einen Schritt zurück, so stellen sich schnell Fragen zur Organisation der SAS Umgebung. Dies geschieht im SAS Studio über die Menüelemente im linksseitigen Verzeichnisbaum. Hierbei unterscheidet die Anwendung zwischen folgenden Ordnern:

- Serverdateien und –Ordner
- Task und Utilities
- Snippets
- Bibliotheken
- Dateiverknüpfungen

Der Bereich „Serverdateien und –Ordner“ stellt den primären Ablageort für Inhalte dar, die mittels SAS Studio erzeugt werden. Technisch gesehen wird beim Start der Web-Anwendung auf dem Server ein Benutzerkontext des angemeldeten Users erzeugt. In diesem Benutzerkontext wird ein privates „Home-Verzeichnis“ angelegt, auf welches hier zugegriffen wird. Der Ordner „Eigene Dateien“ verweist genau auf dieses lokale Verzeichnis auf dem Server, auf dem SAS Studio ausgeführt wird. Im Spezialfall der SAS University Edition wird bei der Konfiguration der VirtualBox ein Profilordner „mysasfiles“ angelegt, der beim Start der virtuellen Maschine als Benutzerordner eingebunden wird. Von daher zeigt in der SAS University Edition der Ordner „Eigene Dateien“ genau auf diesen konfigurierten „mysasfiles“-Ordner.

In der Single-User Version des SAS Studios, welche mit einer klassischen SAS Base Installation installiert wird, werden an dieser Stelle alle eingebundenen Laufwerke bereitgestellt, so dass Zugriff auf alle lokalen Dateien besteht.

Über den Punkt „Ordnerverknüpfungen“ können weitere externe Ordner eingebunden werden. Diese können entweder lokal auf dem Server außerhalb des Home-Verzeichnisses des Anwenders liegen, oder auf einem externen FTP Server. In einer Windows-Umgebung besteht weiterhin die Möglichkeit, den Pfad zum Serverordner auf einer entfernten Windows-Freigabe anzugeben, so dass auf diese Weise einfach auf zentrale Daten im Windows Netzwerk zugegriffen werden kann.

Der Bereich „Tasks und Utilities“ ist vergleichbar mit den Anwendungsroutinen im SAS Enterprise Guide. Hier werden von der Firma SAS Assistenten bereitgestellt, die einen bei der Erstellung von SAS Programmen und Auswertungen unterstützten. Je nachdem, welche Komponenten in der Anwendung lizensiert sind, kann die Liste der verfügbaren Tasks länger oder auch kürzer ausfallen.

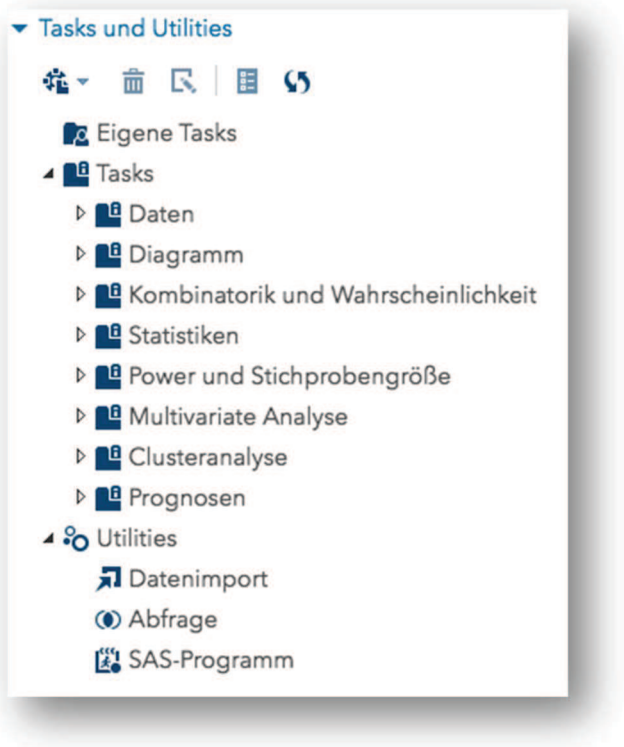



**Abbildung 2:** Übersicht der mitgelieferten Tasks

Betrachtet man beispielsweise den Daten-Task „Tabellenattribute auflisten“, so erkennt man schnell die Art und Weise, wie ein Task als SAS Code-Generator funktioniert:

Im Bereich der Einstellungen können Vorgaben für den Task gemacht werden. Auf dem Reiter „Daten“ wird festgelegt, welche SAS Tabelle von dem Task betrachtet werden soll. Im Beispiel hier SASHELP.CLASS.

Auf dem Reiter Optionen können weitere Einstellungen bezüglich der Ausgabe vorgenommen werden, wie z. B. ob Dateiattribute, Variablenlisten oder Verzeichnisinformationen mit ausgegeben werden sollen. Genauso kann die Erzeugung einer Ausgabetabelle hier mit aktiviert werden.

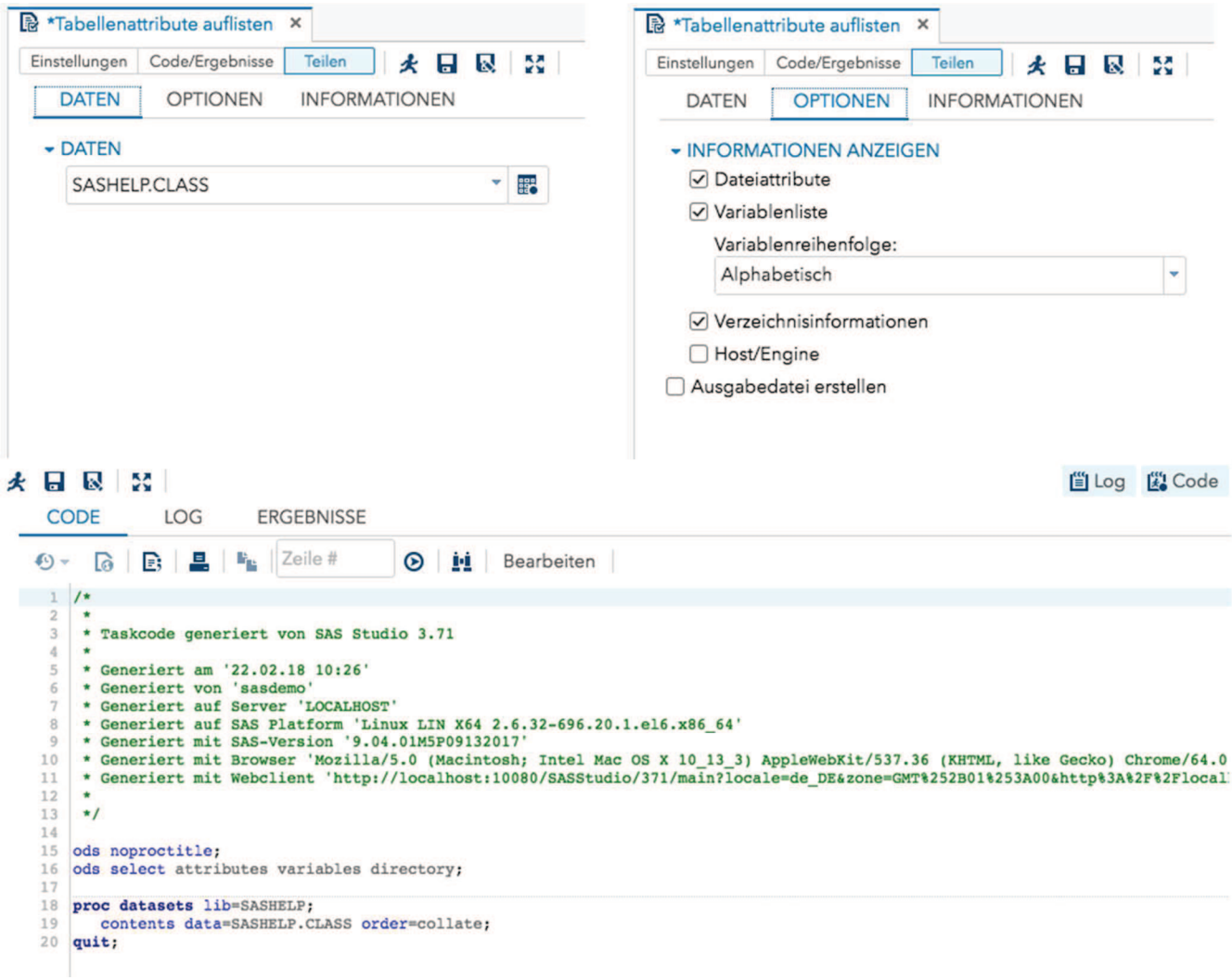



**Abbildung 3:** Code-Generierung durch Nutzung von Tasks

Das Ergebnis des Tasks wird im Code-Fenster angezeigt. Je nach Komplexität des Tasks wird hier mehr oder weniger SAS Code ausgegeben. In diesem einfachen Beispiel führen die gesetzten Optionen zu einem einfachen PROC DATASETS, der eine Contents-Ausgabe auf die gewählte Tabelle produziert. Änderungen an den Optionen werden quasi in Echtzeit in das Code Fenster übertragen, so dass die Tasks auch genutzt werden können, um die Möglichkeiten zur Konfiguration einer SAS Prozedur anhand der Eingabeparameter zu prüfen.

Das wunderbare an den Tasks ist, dass diese durch den Anwender problemlos erweitert werden können. Dies kann jeder Anwender autonom für sich machen, oder in einer größeren Organisation können Tasks über zentrale Repositories bereitgestellt werden. Auch SAS stellt über GitHub einige Open Source Tasks bereit, die sich einfach über die Einstellungen des SAS Studios in die Oberfläche integrieren lassen (vgl. [1]).
Hierzu kann bspw. die Respository-URL
https://sassoftware.github.io/sas-studio-tasks/contributed/tasktuesdays/repository.html
wie folgt eingebunden werden.

In den Programmeinstellungen wird in der Kategorie Repositories über das + Symbol ein neues Repository hinzugefügt. Der Name ist frei wählbar, als URL ist die o.a. URL (ohne Leerzeichen) einzugeben:

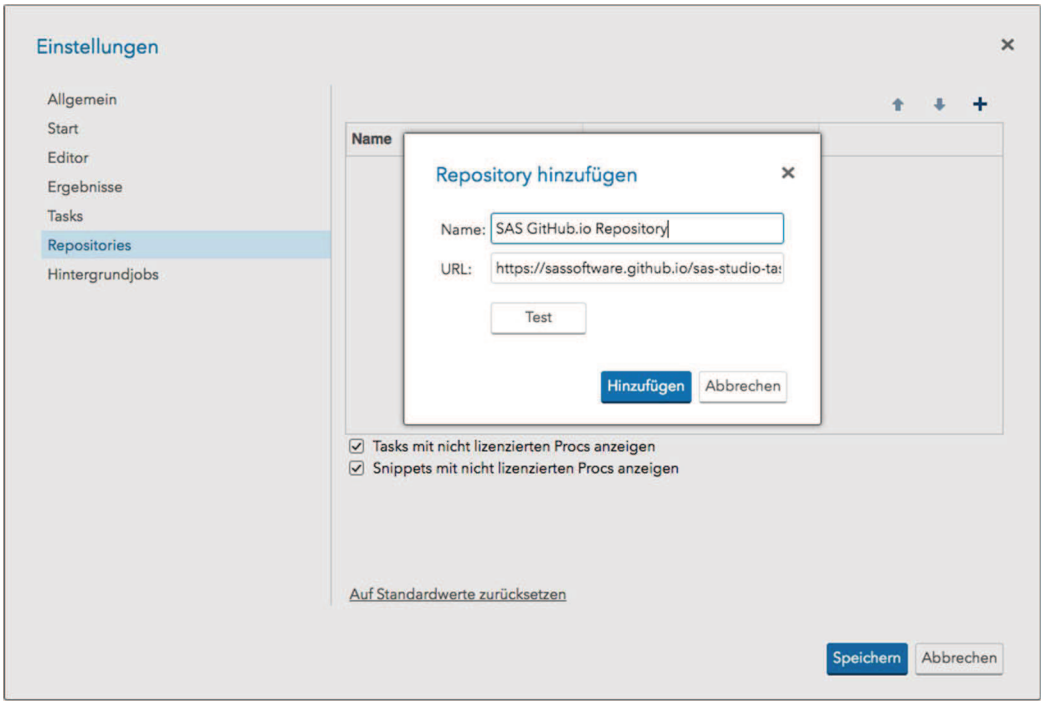



**Abbildung 4:** Hinzufügen eines eigenen Repositories

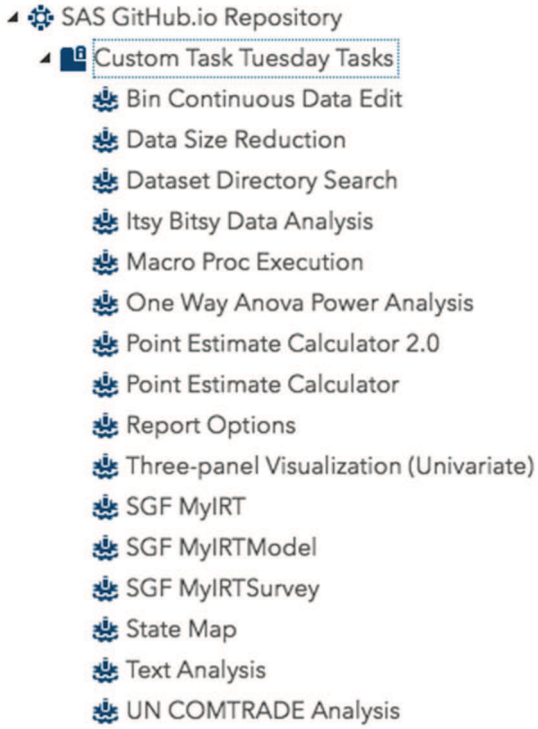



**Abbildung 5:** Ansicht der Tasks aus dem SAS GitHub.io Repository

Der Bereich „Snippets“ sammelt einzelne Code-Fragmente, die per Doppelklick in ein SAS Programm mit eingefügt werden können. Hier geht es nicht darum, einen Assistenten zur Generierung von SAS Code bereitzustellen; vielmehr soll die Tipparbeit von

häufig verwendetem Code gespart werden. Beim Doppelklick auf ein Snippet wird der darin enthaltene Code an die aktuelle Cursor-Position im SAS Programm eingefügt und kann dann individuell angepasst werden.

Doppelklickt man bspw. während der Bearbeitung eines SAS Programms auf das Snippet „Beschreibend – PROC SQL“, so wird folgender Code an der Cursor-Position eingefügt:

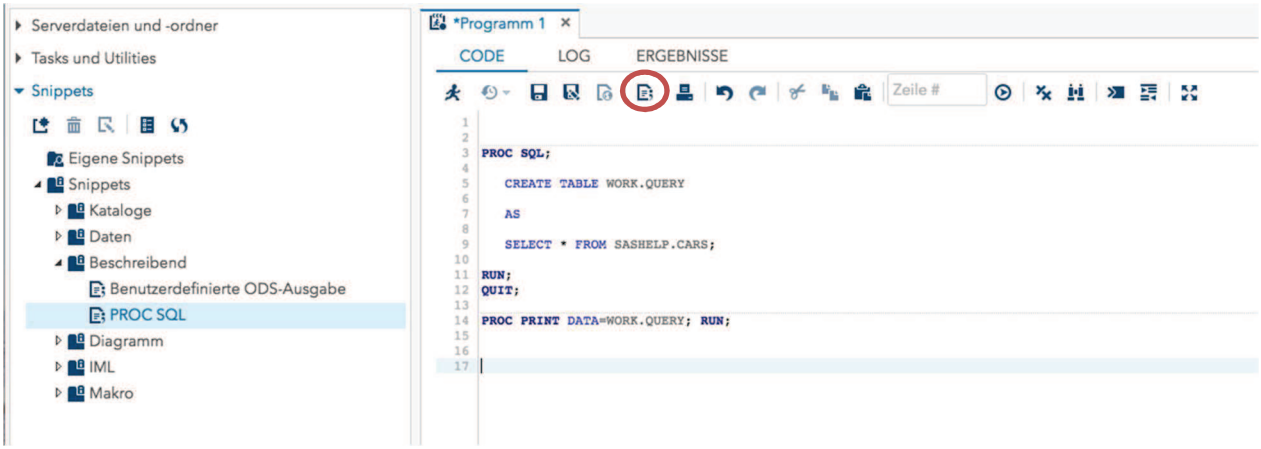



**Abbildung 6:** Nutzung von Snippets

Der SAS Code kann anschließend noch an die eigenen Bedürfnisse angepasst werden.

Auch die Snippets lassen sich ganz leicht individuell abspeichern, um so die Arbeit mit SAS Studio zu optimieren. Im Programmfenster wird hierzu einfach der relevante SAS Code markiert und dann über das oben markierte Symbol als neues, eigenes Snippet abgespeichert. Dies kann dann sofort wieder in weiteren Programmen verwendet werden. Es wird in der Kategorie „Eigene Snippets“ abgelegt; leider ist es in der aktuellen Version von SAS Studio noch nicht möglich, eigene Ordner unterhalb der Snippets anzulegen, um diese besser und übersichtlicher zu strukturieren.

Die letzten beiden Bereiche „Bibliotheken“ und „Dateiverknüpfungen“ sind dazu da, um auf die registrierten SAS Bibliotheken und FileRefs zuzugreifen. Über die Symbolleiste oberhalb der Bibliotheken bzw. FileRefs lassen sich einfach neue Elemente anlegen. Bei Bedarf können diese auch Statements auch der privaten autoexec-Prozedur hinzugefügt werden, so dass beim Start von SAS Studio die Dateien und Bibliotheken automatisch zugewiesen werden.

In einer verwalteten Umgebung mit einer SAS Studio MidTier Installation können Bibliotheksdefinitionen an dieser Stelle auch aus dem zentralen Metadaten-Repository zugewiesen werden. In der Übersicht der Bibliotheken werden aber nur die schon zugewiesenen Bibliotheken angezeigt, was die Übersichtlichkeit der Liste deutlich steigert.

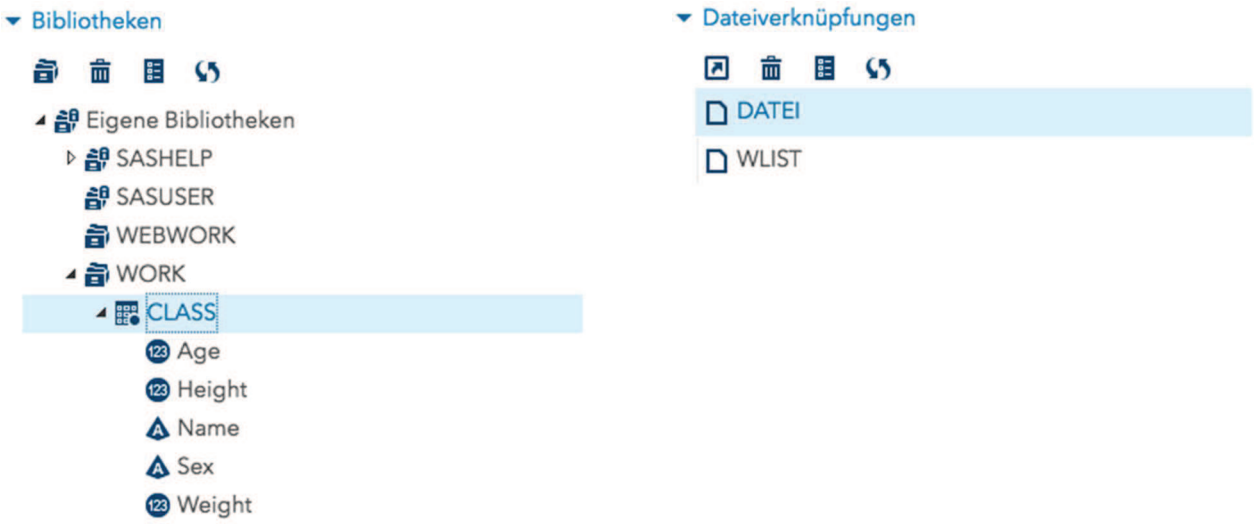



**Abbildung 7:** Bibliotheken und Dateiverknüpfungen

Im Gegensatz zum SAS Enterprise Guide, der alle verfügbaren Bibliotheken unterhalb des Server-Kontextes darstellt, hört beim SAS Studio die Ansicht nicht bei der Tabelle auf. Unterhalb der Tabelle werden zusätzlich die Spalten angezeigt, welche sich im Rahmen der Programmierung einfach per Drag&Drop aus der Bibliotheksansicht in das Programm gezogen werden können. Auf diese Weise entfällt bei der Angabe von Variablen in Prozeduren das Abtippen des Variablennamens.

## 2.3 Einstieg für Fachanwender

Das klassische Analysewerkzeug des Fachanwenders ist nicht der SAS Display Manager, sondern eher der Enterprise Guide. Durch die toolgestützte Entwicklung von Abfrage-Prozessflüssen und die Möglichkeit, Abfragen grafisch zusammenzustellen, anstatt den SQL Code von Hand zu schreiben, ist er bei den Fachanwendern äußerst beliebt.

Mit der SAS Studio Version 3.3 im Februar 2015 wurde das SAS Studio um die Perspektive des „Visual Programmer“ erweitert, welche genau diese Zielgruppe anspricht.

Mit der Version 3.7, welche im September 2017 veröffentlicht wurde, hat SAS zudem die Möglichkeit geschaffen, Enterprise Guide Projekte im SAS Studio zu öffnen und in einen SAS Studio Prozessfluss zu konvertieren. Nach dem Konvertieren eines EG-Projektes wird eine Übersicht über das Konvertierungsergebnis angezeigt. Da noch nicht alle Features des Enterprise Guides im SAS Studio bereitstehen, muss an einzelnen Stellen nach dem Import noch etwas nachgearbeitet werden. Dennoch lässt sich die Prozess-Logik des Projektes in SAS Studio übernehmen.

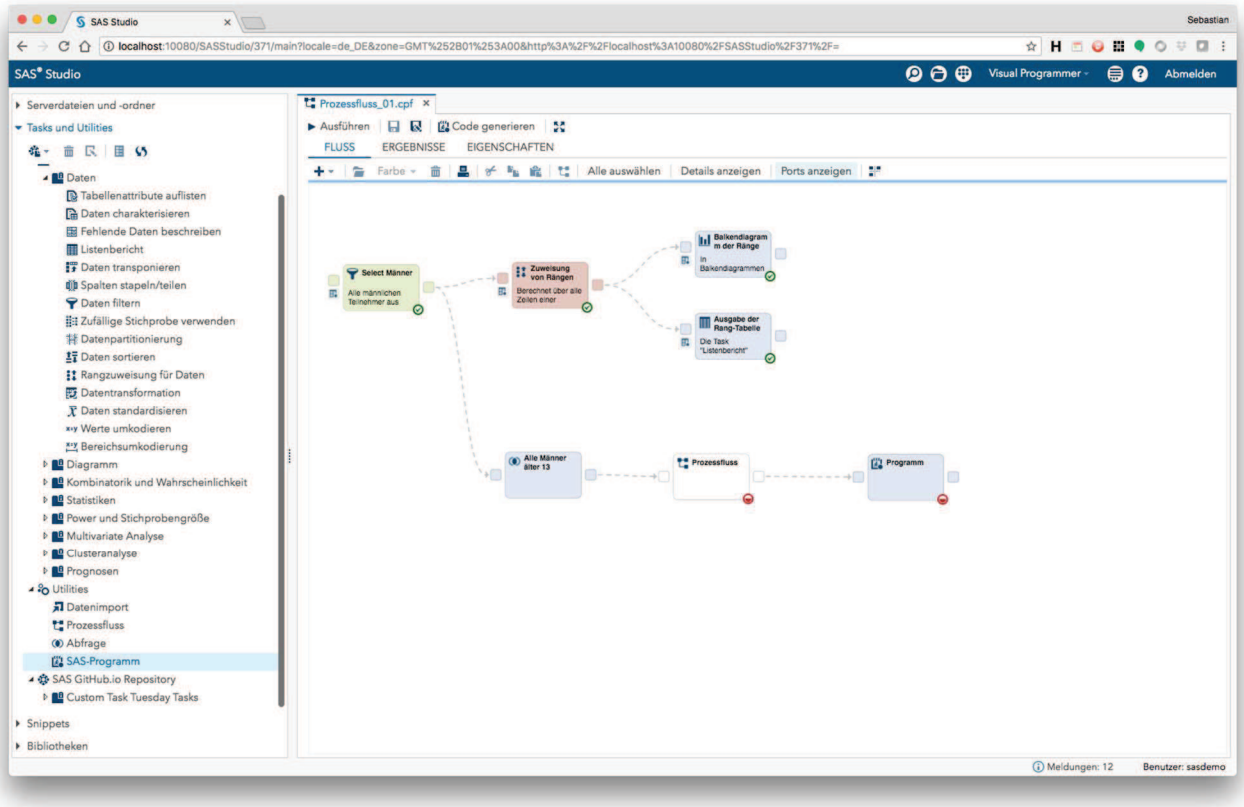



**Abbildung 8:** Ansicht „Visual Programmer“ mit Beispiel-Prozess

Wie beim Enterprise Guide geht es beim Visual Programmer um die Verkettung von einzelnen Code-Elementen. Zunächst wird bspw. eine Datenselektion durchgeführt, danach ein Ranking der Daten und zum Schluss die Aufbereitung der Ergebnisse als HTML Report oder PDF Dokument. Genau diese Schritte werden in Form von Prozessknoten im Prozessfluss modelliert und über die Verkettung in eine vorgegebene Reihenfolge gebracht.

Im Gegensatz zum Enterprise Guide erfolgt die Verkettung der Prozessknoten nicht automatisch über die Eingabe- und Ausgabedateien sondern muss manuell vorgenommen werden. Auch werden die Ausgabetabellen nicht als Prozesselemente im Prozessfluss dargestellt. Zur Einsicht der Ausgabetabelle muss der erstellende Prozessknoten geöffnet werden. Darüber kann die Ergebnistabelle eingesehen werden.

An jedem Prozesselement wird über ein kleines farbiges Symbol der Status des Elements symbolisiert. Mit einem zur Hälfte rot gefüllten Kreis markierte Knoten weisen darauf hin, dass es hier noch inhaltliche Fehler gibt, die eine Ausführung verhindern. So muss bspw. ein Programm-Element mit SAS Code gefüllt werden, bevor es ausgeführt werden kann. Wurde noch kein SAS Code erfasst, verhindert dies eine Ausführung des Knotens und dieser wird entsprechen mit einem zur Hälfte rot gefüllten Kreis markiert.

Mit einem grünen Haken versehene Prozessknoten deuten darauf hin, dass diese bereits fehlerfrei ausgeführt wurden und die Ausgabedaten dazu erstellt wurden. Treten bei der

Ausführung Warnungen aus, erfolgt eine Markierung mit einem gelben Ausrufezeichen. Treten im SAS Code Fehler auf, so wir der Knoten mit einem roten X markiert.

Bei der Erstellung des Prozessflusses können die unter „Tasks und Utilities“ aufgeführten Tasks als Prozessknoten in den Prozessfluss mit integriert werden. So können auch ohne SAS Programmierkenntnisse Abfragen und Auswertungen zusammengestellt werden, die dann auf Knopfdruck ausgeführt werden. Dadurch, dass in einer Organisation über Task-Repositories individuelle Tasks für die Anwender bereitgestellt werden können, stehen hier alle Möglichkeiten offen, den Anwender bei der Nutzung der Anwendung SAS zu unterstützen.

## 2.3 Der erste eigene Task

Der große Vorteil im SAS Studio liegt in der einfachen Erweiterbarkeit der Anwendung um eigene Assistenten. Als Beispiel soll hier das SAS Studio um einen Assistenten zum Verketten von zwei Tabellen erweitert werden. Im Enterprise Guide gibt es hierzu eine Anwendungsroutine „Tabelle anhängen“, die zwei Eingabetabellen erwartet und diese mittels PROC SQL UNION zu einer Tabelle zusammenführt. Diese Anwendungsroutine ist im SAS Studio weder unter den von SAS gelieferten Tasks, noch unter den Snippets verfügbar.

Zum Erstellen eines eigenen Tasks ist lediglich der Button „Neue Task“ in der Symbolleiste auszuwählen.

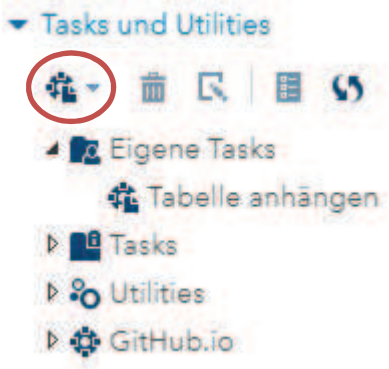



**Abbildung 9:** Erstellung eines eigenen Tasks

Als Ergebnis wird das Grundgerüst eines Tasks als XML Dokument geöffnet, welches anschließend über den Speichern-Button unter „Eigene Tasks“ gespeichert werden kann.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.2" runNLS="never">
  <Registration>
  <Name>New Task</Name>
  <Description>This is a blank task</Description>
  <GUID>726B2B12-4542-4200-96F4-960C645D767F</GUID>
  <Procedures>TBD</Procedures>
```

```
  <Version>3.7</Version>
  <Links>
    <Link
      href="http://documentation.sas.com/?softwareId=STUDIOMID&amp;s
      oftwareVer-
      sion=3.7&amp;softwareContextId=tasks&amp;requestor=inapp">SAS
      Studio Task Reference Guide</Link>
  </Links>
  </Registration>

  <Metadata>
    <DataSources>
    </DataSources>
    <Options>
    </Options>
  </Metadata>

  <UI>
  </UI>

    <CodeTemplate>
    <![CDATA[

proc print data=sashelp.cars;run;

    ]]>
  </CodeTemplate>
</Task>
```

Damit ist der erste eigene Task angelegt. Die Metadaten des Tasks sind nun noch an den eigenen Zweck anzupassen und das CodeTemplate muss um die gewünschte Funktion erweitert werden. Im CodeTemplate wird die Template Sprache Apache Velocity genutzt, damit dort bei der Generierung auf die Eingaben reagiert werden kann. Hinweise zu den einzelnen Knoten im XML Dokument und auch zur Template Sprache Velocity sind im Custom Task Developer Handbuch zum SAS Studio (vgl. [2]) zu finden.

Bei dem zu erstellenden Task sollen zwei Konfigurations-Reiter „EINGABE“ und „AUSGABE“ entstehen. Auf dem Reiter EINGABE sollen die beiden eingehenden Tabellen auszuwählen sein. Optional kann noch ein Filter-Ausdruck angegeben werden, um nur eine Teilmenge der Eingabe in die Ausgabe zu übernehmen. Auf dem Reiter AUSGABE soll der Tabellenname für die Ausgabetabelle erfasst werden können. Wenn gewünscht soll statt einer Tabelle auch nur eine View angelegt werden. Die Eingabemaske soll also in etwa so aussehen:

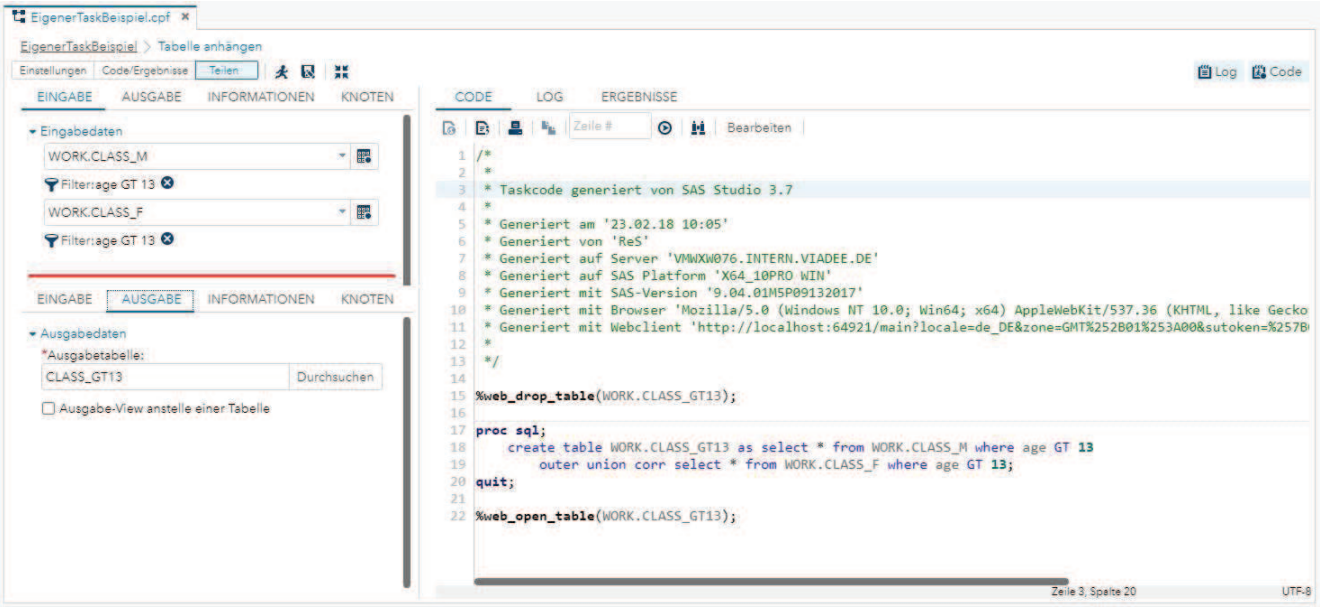



**Abbildung 10:** User Interface für den eigenen Task „Tabelle anhängen“

Um aus dem leeren Task den aus dem SAS Enterprise Guide bekannten „Tabelle anhängen“ Task zu machen sind folgende Anpassungen an dem XML Dokument notwendig:

```
<?xml version="1.0" encoding="UTF-8"?>
<Task runNLS="never" schemaVersion="5.3">
  <Registration>
    <Name>Tabelle anhängen</Name>
    <Description>Es werden zwei Eingabetabellen zu einer
                 Tabelle zusammengefasst</Description>
    <GUID>726B2B12-4542-4200-96F4-960C645D767F</GUID>
    <Procedures>SQL</Procedures>
    <Version>3.71</Version>
    <Links>
      <Link
           href="http://documentation.sas.com/?softwareId=STUDIOMID&a
           mp;softwareVersion=3.71&amp;softwareContextId=tasks&amp;re
           questor=inapp">SAS Studio Task Reference Guide</Link>
      <Link
           href="http://documentation.sas.com/?docsetId=sqlproc&amp;d
           ocsetTar-
           get=titlepage.htm&amp;docsetVersion=9.4&amp;locale=de">SAS
           SQL Procedure User's Guide</Link>
    </Links>
  </Registration>
  <Metadata>
    <DataSources>
      <DataSource name="InDS1" where="true"/>
      <DataSource name="InDS2" where="true"/>
    </DataSources>
    <Options>
      <Option inputType="string" name="intab">EINGABE
      </Option>
      <Option inputType="string" name="ingrp">Eingabedaten
```

```
      </Option>
      <Option inputType="string" name="outtab">AUSGABE
      </Option>
      <Option inputType="string" name="outgrp">Ausgabedaten
      </Option>
      <Option defaultValue="AppendTable"
              inputType="outputdata"
              name="outputDSName" required="true">
              Ausgabetabelle:
      </Option>
      <Option defaultValue="0" inputType="checkbox"
              name="createView">
              Ausgabe-View anstelle einer Tabelle
      </Option>
    </Options>
  </Metadata>

  <UI>
    <Container option="intab">
      <Group open="true" option="ingrp">
        <DataItem data="InDS1"/>
        <DataItem data="InDS2"/>
      </Group>
    </Container>

    <Container option="outtab">
      <Group open="true" option="outgrp">
        <OptionItem option="outputDSName"/>
        <OptionItem option="createView"/>
      </Group>
    </Container>
  </UI>

  <CodeTemplate>
    <![CDATA[

%web_drop_table($outputDSName);

proc sql;
#if ($createView == '0')
  create table
#else
  create view
#end
  $outputDSName as
    select * from $InDS1
    #if($InDS1.getWhereClause() != '')
        where $InDS1.getWhereClause() #end
    outer union corr
    select * from $InDS2
    #if($InDS2.getWhereClause() != '')
        where $InDS2.getWhereClause() #end
    ;
quit;
```

```
%web_open_table($outputDSName);
    ]]>
  </CodeTemplate>
</Task>
```

Zunächst ist in der Rubrik „Registration" der Name des Tasks und die Beschreibung zu ergänzen. Die genutzten Prozeduren sollten im Bereich „Procedures" angegeben werden. Wenn gewünscht kann noch ein Link auf Dokumentation angegeben werden, der im Task später auf dem Reiter „Informationen" angezeigt wird.

Anschließend sind die Metadaten zu ergänzen. Es sollen zwei Eingabedatasets genutzt werden. Diese sind unter „Datasources" anzugeben. Der gewählte Name dient später als Referenz im CodeTemplate. Er kann an dieser Stelle frei gewählt werden; auf Groß-/Kleinschreibung ist zu achten. Weiterhin sind die Optionen zu definieren. Hier sind alle auszugebenden Strings und Checkboxen im User Interface zu definieren. Die jeweils angegebenen Namen dienen als Referenz auf die im Bereich „UI" genutzten Elemente. Es werden die Namen der Reiter, die Überschriften und die Checkboxen an dieser Stelle definiert.

Der Bereich „UI" definiert das User Interface. Es werden zwei Container (intab und outtab) definiert. Innerhalb der Container gibt es jeweils nur eine Gruppe (ingrp bzw. outgrp), die bereits aufgeklappt ist. In der Gruppe der Eingabetabellen werden zwei Verweise auf die Input Datasets InDS1 und InDS2 angegeben, in der Gruppe der Ausgabetabelle werden die Verweise auf die Ausgabetabelle und die Checkbox zur View-Erstellung angegeben. Bei all diesen Angaben handelt es sich um Verweise auf die unter „Options" definierten Optionen. Daher werden an dieser Stelle auch keine Texte angegeben, da diese alle unter den „Options" definiert sind.

Zum Schluss muss noch das Code Template definiert werden. Hierbei handelt es sich im Prinzip um reinen SAS Code. Dieser wird um Variablen ergänzt, die im User Interface Bereich definiert sind. So kann bspw. auf die Eingabe-Datasets mittels $InDS1 und $InDS2 zugegriffen werden. Auch die optional angegeben WHERE-Bedingung kann über die Variablen $InDS1 bzw. $InDS2 ermittelt werden. IF-Bedingungen können über das Konstrukt #if #else #end in das Code Template eingebunden werden. Durch die Nutzung der weit verbreiteten Template Sprache Apache Velocity stehen hier bei der Code-Generierung viele Möglichkeiten offen, den Code dynamisch anhand der Eingaben im User Interface zu generieren.

Nach dem Speichern steht der Task sofort in der Rubrik „Eigene Tasks" zur Verfügung und kann von dort mittels Doppelklick ausprobiert werden. Nutzt man die geteilte Ansicht, so kann man die Änderungen der Eingabe- und Ausgabetabellen direkt im Code-Fenster beobachten und so mögliche Fehler im Code Template schnell erkennen.

In der Ansicht des Visual Programmes kann die Nutzung des Tasks „Tabelle anhängen“ dann wie folgt aussehen:

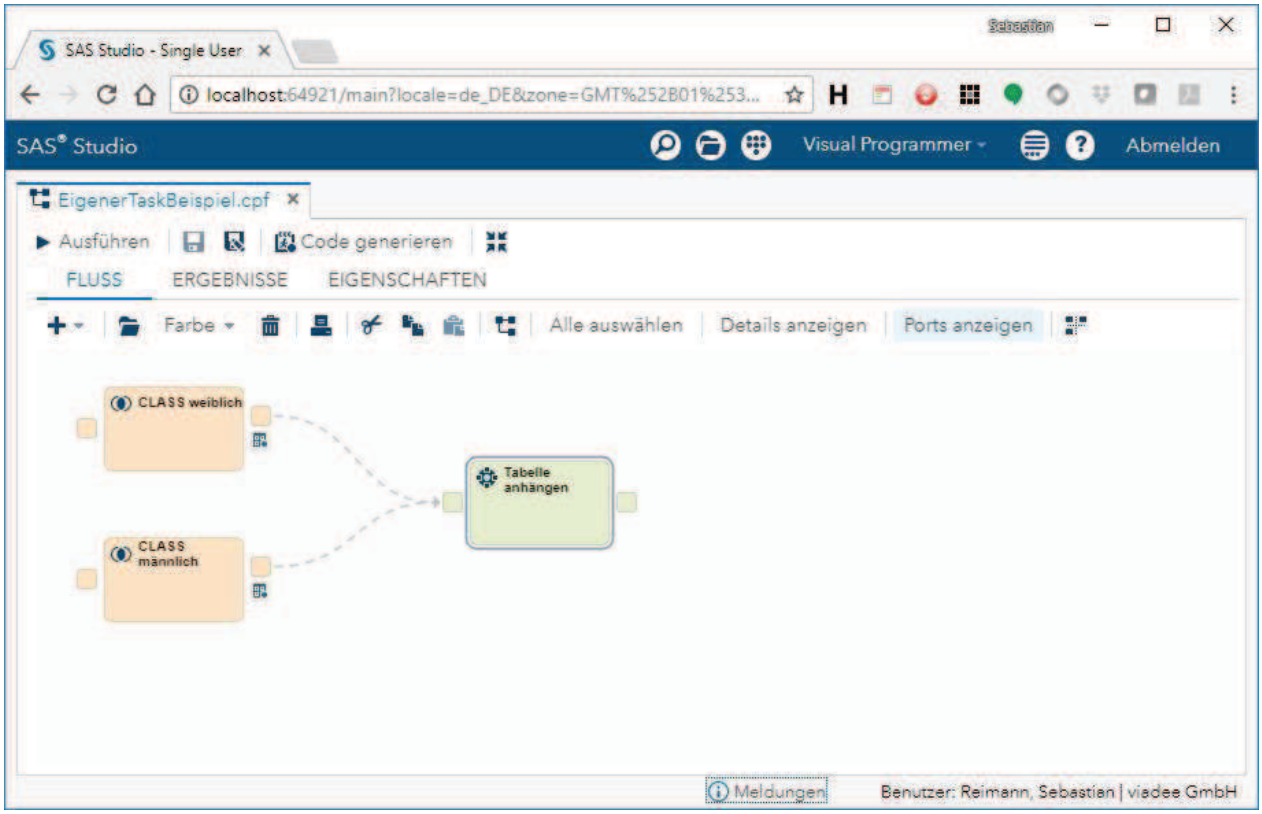



**Abbildung 11:** Eigener Task im Prozessfluss genutzt

# 3 Fazit

Das SAS Studio stellt eine neue, moderne Komponente für den Zugriff auf die Anwendung SAS dar. Durch die Bereitstellung als Web-Komponente bietet sie gerade in einer verwalteten Umgebung mit SAS Metadatenserver die Möglichkeit, auf Client Installationen vollständig zu verzichten. Aber auch bei reinen SAS Base Installationen steht das SAS Studio als alternative Zugriffskomponente zum SAS Display Manager zur Verfügung, da die SAS Base Installation eine Single-User Version mit integriertem Anwendungsserver gleich mitbringt.

Mit seinen Perspektiven „SAS Programmer“ und „Visual Programmer“ spricht das SAS Studio den Nutzer des SAS Display Managers gleichermaßen an, wie den Nutzer des SAS Enterprise Guides. Beide Nutzergruppen finden sich in ihrer jeweiligen Perspektive der Anwendung schnell zurecht.

Gerade für den Visual Programmer stellt die offene Architektur mit der einfachen Integration von eigenen Tasks einen enormen Vorteil gegenüber dem Enterprise Guide dar. Über Standard-Repositories, welche in Standardanwendungen wie GitHub gehostet werden können, lassen sich eigene Task-Bibliotheken einfach in das SAS Studio integrieren. Der versierte SAS Anwender hat zudem die Möglichkeit, für seine Aufgaben

spezielle Tasks selbst zu entwickeln, ohne dabei eine wirkliche Entwicklungsumgebung nutzen zu müssen.

## Literatur

[1] SAS Custom Task Tuesday Examples (https://github.com/sassoftware/sas-studio-tasks/blob/master/contributed/tasktuesdays/README.md), Abruf: 22.02.20218

[2] SAS Studio 3.7 Developers Guide to Writing Custom Tasks (http://documentation.sas.com/?docsetId=webeditordg&docsetTarget=bookinfo.htm&docsetVersion=3.7&locale=de), Abruf: 22.02.2018