mayato

Was wir vorhersagen, soll auch eintreffen!

Web Applikationen in SAS erstellen – Handsontable und H54s-Adapter //

Benno Schultheiss

Zusammenfassung

Dieses Whitepaper zeigt wie mit SAS® 9.4, HTML5 und JavaScript interaktive Web-Applikationen erstellt werden können. Dabei wird demonstriert wie Webapplikationen mit AJAX-Technik als asynchrone JavaScript-Applikationen programmiert werden. Der Ansatz integriert den H54s-Adapter und Handsontable für eine nahtlose Kommunikation zwischen dem SAS-System und der Web-Applikation. Das Ergebnis ist eine Applikation mit nutzerfreundlicher Dateneingabe und -ausgabe im Excel-Feeling, die schnell ausgebaut werden kann.

Mit Hilfe des open-source Boemska Data Adapter (H54s) und der Handsontable (HOT) JS Bibliothek ist es möglich, asynchrone Webapplikationen in SAS® zu erstellen. Das Potential dieser Web-Applikationen wird am Beispiel einer im Excel-Feeling gestalteten Webapplikation demonstriert. Diese einfache aber performante Kombination aus SAS und JavaScript gibt Nutzern die Möglichkeit, zentral gespeicherte SAS-Daten im Browser anzuzeigen oder zu bearbeiten. AJAX-Technik erlauben den asynchronen Austausch der Daten mit einem Webserver. Damit können Teile einer Webseite aktualisiert werden, ohne die ganze Seite erneut laden zu müssen.

SAS-Programmierer können die Applikation mühelos verändern. Zudem wird kein Client-Side Deployment oder zusätzliches Training für Nutzer benötigt. Nach einer kurzen Einführung in die Anforderungen und einer Übersicht über das empfohlene Setup wird das Web-Applikation Beispiel demonstriert und der dazugehörige Code erläutert.

Schlüsselwörter

✓ SAS®, HTML, ✓ JavaScript, H54s, ✓ Handsontable, App, ✓ Web-Applikation, ✓ AJAX

Zielgruppe

Dieses Whitepaper ist an jene SAS-Entwickler gerichtet die neben Plattformadministration auch Frontend-Entwicklung mit HTML und JavaScript betreiben.

Grundlegende Informationen

Was ist eine Web App?

Web Apps sind Browser gestützte Programme und funktionieren auf jedem internetfähigen Gerät. In der Regel werden Web Apps auf Basis von HTML5, JavaScript und CSS entwickelt und funktionieren Browser-übergreifend. Sie benötigen keine Installation, da sie nach dem Client-Server-Modell funktionieren. Der Server übernimmt dabei die Bereitstellung und Verarbeitung der Daten.

Diese Struktur bietet einige Vorteile:

- Web-Apps funktionieren auf allen Betriebssystemen und internetfähigen bzw. mit einem Webbrowser ausgestatteten Endgeräten.
- Web-Apps sind im Vergleich zu Native-Apps effizienter und günstiger zu programmieren.
- Updates der Software sind weniger kompliziert, da Anwender nur auf die Server-Version zugreifen.

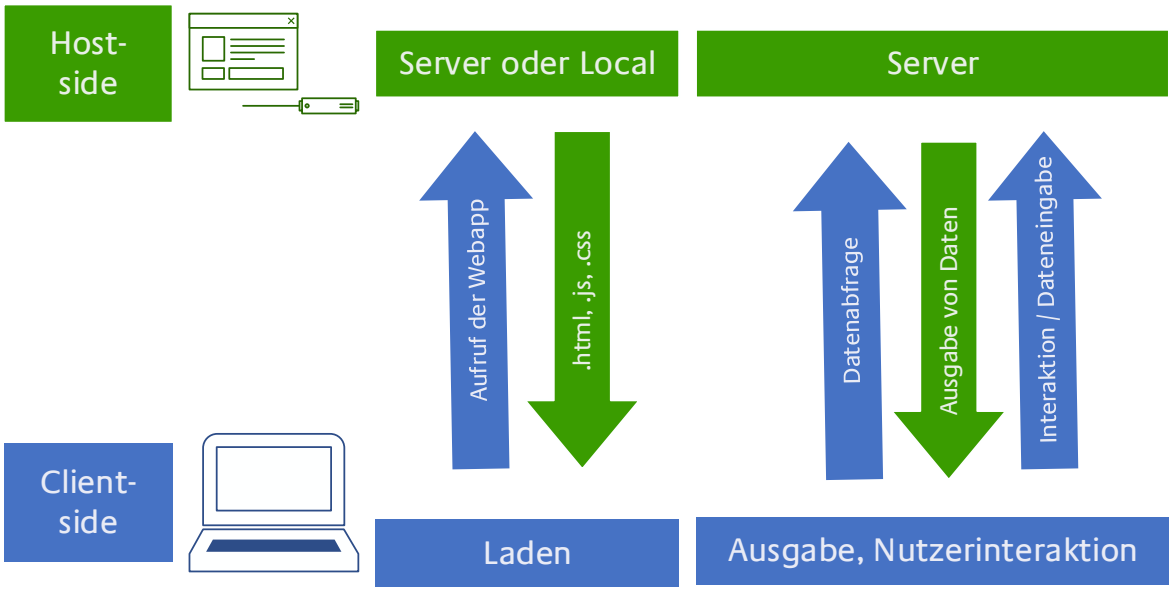

Abbildung 1: Funktionsweise einer HTML App

Was ist AJAX?

AJAX steht für „Asynchronous JavaScript and XML“ und ist eine Technik mit der von Webseiten auf einen Webserver zugegriffen wird. Grundsätzlich werden dabei folgende Komponenten verwendet:

- Ein in den Browser implementiertes XMLHttpRequest-Objekt, mit dem Daten von einem Webserver angefragt werden können.
- JavaScript und HTML DOM, womit Daten angezeigt oder verarbeitet werden.

AJAX ist zum Datenaustausch nicht auf XML beschränkt, denn es können auch Textelemente oder JSON-Text zum Datentransport verwendet werden.

Die AJAX Technik ermöglicht das Updaten einer Webseite ohne diese neu laden zu müssen. Hierdurch können Daten nach dem Laden angefordert und entgegengenommen werden, da die neue Dateneingabe durch den Server im Hintergrund empfangen und verarbeitet wird.

Was ist Handsontable?

Handsontable ist eine JavaScript/HTML5 Spreadsheet Komponente für Web-Applikationen, die eine GestaltungExcel-ähnlicher Kalkulationstabellen mit umfassenden Features erlaubt. Die Komponente lässt sich über JSON mit allen Quellen verbinden und kommt auch mit sehr großen Datenmengen zurecht.

Was ist der Boemska H54s Data Adapter?

Die Bibliothek H54s ermöglicht die Kommunikation zwischen einer HTML5 (JavaScript) basierten Applikation und SAS®. Der Adapter übernimmt die Konvertierung von JavaScript Object Arrays nach SAS® Datasets in beide Richtungen, erfasst SAS® Logs und Fehleranalyse und verwaltet SAS®-Logon Weiterleitungen. Der Back-End-Datenservice wird in SAS® geschrieben und auf der SAS® Enterprise BI Plattform betrieben.

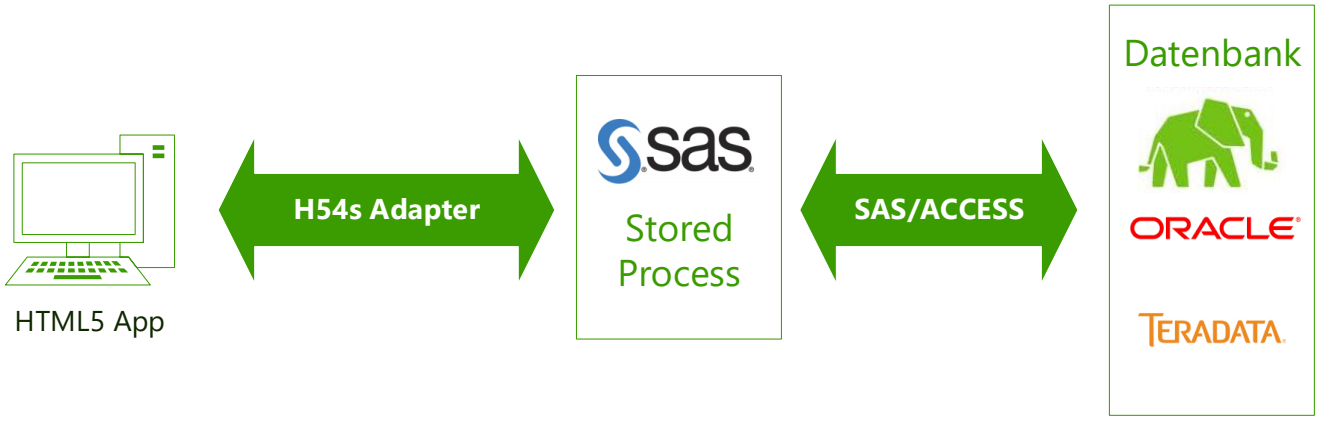

Abbildung 2: Funktion einer SAS®-gestützten HTML5 Applikation.

Was wird für das demonstrierte Beispiel benötigt?

Server Anforderungen:

- SAS® Stored Process Web Application (Integration Technologies)

Client Anforderungen:

- Web Browser
- Web Application Framework oder Library. Bei aktuellen Browsern ist dies stets vorhanden.

Installation des h54s-Adapters und der Handsontable-Komponente:

Installation h54s-Adapter:

Für die Installation des h54s-Adapters müssen zunächst die Installationsdateien aus dem github-Repository geladen werden. https://github.com/Boemska/H54s

Folgende Daten sind in der SAS®-Umgebung abzuspeichern:

- `H54s.min.js -> /js folder (midtier web root)`
- `H54s.sas -> /SASEnvironment/SASCode/Programs (SASApp LevX folder)`

Zusätzlich sollte folgender Code in der autoexec_usermods.sas Datei platziert werden, damit die Macros aus h54s.sas verfügbar sind.

```
/*compile Boemska data connector macros */
%include "/path/to/SASEnvironment/SASCode/Programs/H54s.sas";
```

Nach einem Neustart des Stored Process Servers werden die Änderungen übernommen.

Installation Handsontable: Die Installation der Handsontable Komponente kann durch Ablegen der JavaScript- und CSS.-Dateien (handsontable.full.min.js und handsontable.full.min.css) auf dem SAS-Server und Einbinden der Pfade zu den Dateien im Stored Process erfolgen. Alternativ ist auch die Installation über Package Manager wie z. B. Node Package Manager (npm) möglich:

```
npm install handsontable
```

Außerdem stehen Installationsmöglichkeiten über nuget, bower, yarn etc. und verschiedene Wrapper zur Verfügung (React, Angular, Polymer etc.).

Grundstruktur des SAS Codes

Für den einfachsten Aufbau der Web-Applikationsstruktur werden drei Elemente verwendet:

- **Input Data:** Mit dem Befehl %hfsGetDataset( ); wird das H54s Makro aufgerufen welches zuvor in autoexec_ussermods.sas platziert wurde.
- **Processing:** Nach dem Aufrufen der Daten können diese mit SAS Code bearbeitet werden.
- **Output Data:** Die Ausgabe aus dem Processing kann anschließend zurück zum Client geschickt werden. Hierfür werden die Macros %hfsHeader(); %hfsOutDataset(); und %hfsFooter(); aufgerufen.

Durch das Hinzufügen von Funktionen in Form von Macros kann das Programm beliebig erweitert werden.

Code-Beispiel und Web-Applikations-Demonstration

Dieser Abschnitt zeigt ein Code-Beispiel für eine Web-Applikation, mit der Daten interaktiv aufgerufen, verändert und gespeichert werden können. Zunächst wird die Funktion der Web-Applikation erläutert. Anschließend wird in einer Übersicht der Aufbau des Codes dargestellt, welcher im Anschluss in einzelnen Funktionen detailliert erklärt wird.

Funktion der Applikation:

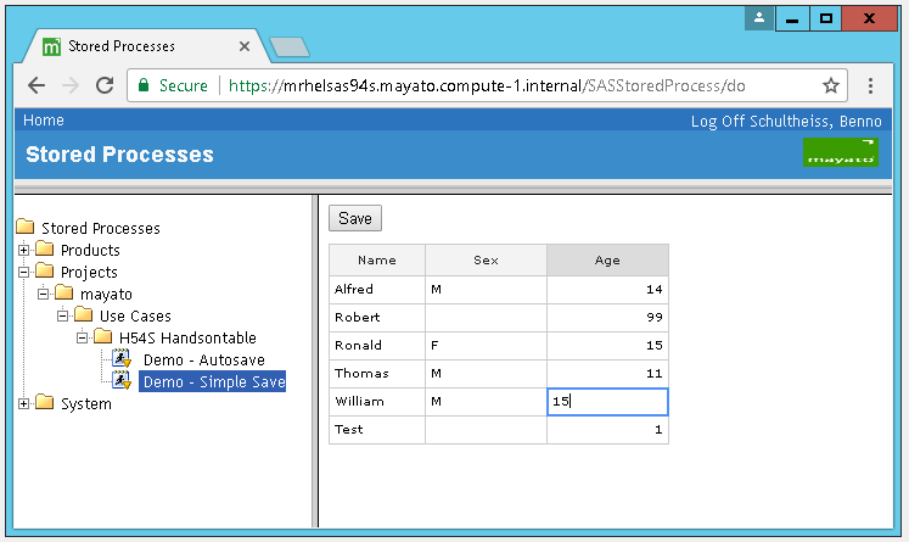

Abbildung 3: Screenshot Beispiel Web-Applikation

Die Applikation ist zu Demonstrationszwecken sehr einfach gehalten. Beim Aufruf des Stored Process wird die Beispieltabelle „UC_H54S_Handsontable“ geladen. Die Einträge der Tabelle können verändert und durch den „Save“-Button in SAS® gespeichert werden.

Übersicht über Code-Aufbau

Der Code besteht aus drei SAS-Macros:

1. stp_run: Das stp_run Macro ist am Ende des Codes platziert. wird aber als einziges Macro direkt ausgeführt. Es lädt zunächst das h54s.sas Macro und ruft das %hfsGetDataset(); Macro auf. Anschließend versucht es eine control-Tabelle zu laden. Wenn diese nicht existiert führt es das stp_output Macro aus. Da die control-Tabelle nur existiert, wenn „save“ geklickt wird, führt es das stp_save Macro nicht aus.
2. stp_output: Dieses Macro führt ein proc sql-Query aus, dessen Resultat in Handsontable geladen wird. Das von SAS® an die Webapplikation übertragene H54s-Objekt wird aus den Handsontable-Daten generiert. Die Ausgabe für den Browser erfolgt durch HTML-Code. Mit JavaScript wird der Save-Button mit dem dazugehörigen Event für einen Klick auf „Save“ definiert und die Tabelle für das HTML-Dokument erzeugt. Ein Klick auf den save-Button generiert die control-Tabelle und ruft den stp_run erneut im Hintergrund auf.
3. stp_save: Wenn der „Save“-Button geklickt wird, lädt dieses Macro die aktuell verwendeten Daten und konvertiert sie von einem H54s-Objekt in eine SAS-Tabelle. In diesem Format werden die Daten anschließend gespeichert. Das Macro wird aufgerufen, wenn stp_output eine control-Tabelle erzeugt hat und stp_run erneut im Hintergrund aufruft. Somit führt stp_run direkt den stp_save aus.

Die Funktionsweise des Codes wird in folgender Grafik visualisiert:

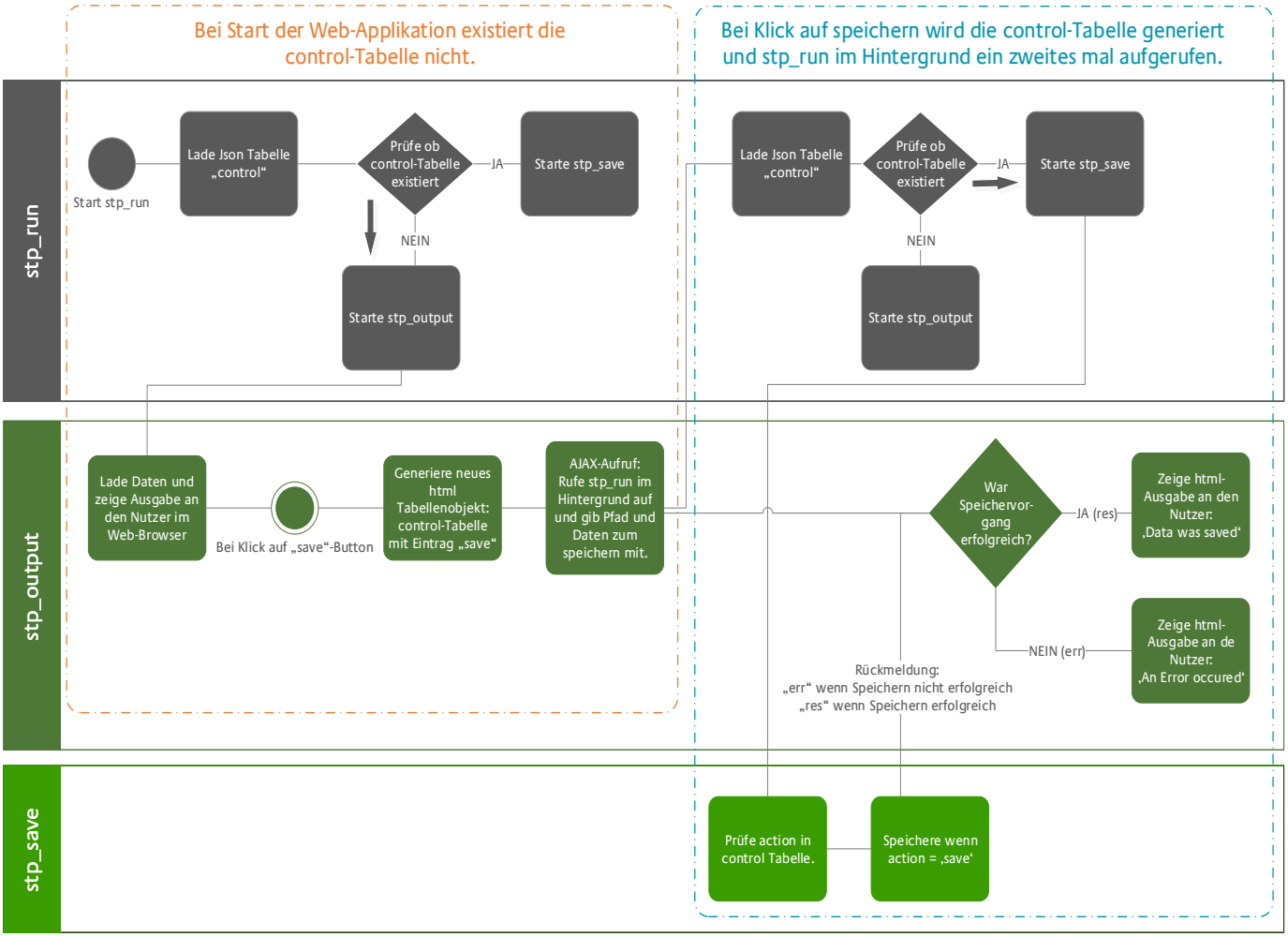

Abbildung 4: Funktionsablauf des Codes

Detaillierte Betrachtung des Codes

Zunächst wird das Macro stp_run erklärt:

```
1.  /* Check if stp is supposed to generate HTML or to save data. */
2.  %macro stp_run();
3.  /* Include h54s source. */
4.  %include "/sas/sasdata/mayato/macros/h54s.sas";
5.  /* Load json table. */
6.  %hfsGetDataset(control, work.control);
7.  /* Check if table exists. */
8.  %if %sysfunc(exist(work.control)) %then %do;
9.  %stp_save;
10. %end;
11. %else %do;
12. %stp_output;
13. %end;
14.
15. %mend;
16. %stp_run;
```

- Zeile 4 nutzt das %include Statement um den H54s.sas Code in das Macro einzubinden.
- Zeile 6 wandelt ein H54s-Objekt in eine SAS-Tabelle um.
- Zeilen 8 bis 13 prüfen ob ein SAS-Dataset aus dem H54s-Objekt extrahiert werden konnte. In diesem Fall wird getestet ob work.control existiert. Beim Ausführen von stp_run existiert keine control-Tabelle. Diese wird nur für den Speichervorgang erzeugt und dabei für einen erneuten Aufruf von stp_run im Hintergrund übergeben. Somit wird beim Aufrufen der Applikation immer zunächst stp_output ausgeführt.
- Zeile 16 führt das stp_run-Macro direkt beim Starten des Stored Process aus.

Im Folgenden ist das Macro stp_save beschrieben:

```
1.  %macro stp_save();
2.
3.  %hfsGetDataset(data, work.data);
4.  %hfsErrorCheck;
5.
6.  /* Library. */
7.  libname data '/sas/sasdata/mayato/data/UC_h54S_Handsontable';
8.
9.  /* Check action. */
10. proc sql noprint;
11. select action into :action from work.control;
12. quit;
13.
14. /* Simple save. */
15. %if "&action" = "save" %then %do;
16. /* Simple save. */
17. data data.class;
18. set work.data;
19. run;
20. %end;
21.
22. /* Send response. */
23. %hfsHeader();
24. %hfsFooter();
25.
26. %mend;
```

- Das Macro wird zwischen Zeile 1 und 26 mit %macro stp_save(); und %mend definiert.
- Zeile 3 führt das H54s Macro auf, um die erhaltenen Daten in einen SAS-Datensatz zu übertragen.
- Zeile 4 überprüft, ob die erhaltenen Daten ok sind.
- Zeile 7 definiert das libname statement für die Datenablage mit einem Pfad.
- Zeilen 10 bis 12 lesen mit proc sql die Variable “action” aus der control-Tabelle und überträgt den Wert mit dem Schlüsselwort „into:“ in die Makrovariable „action“.
- Zeilen 15 bis 20 überprüfen, ob die Makrovariable „&action“ aus Zeile 10-12 den Wert „save“ aufweist. Ist dies der Fall, wird data.class mit work.data überschrieben.
- Zeile 23 bereitet den Output-Stream für Datenobjekte vor.
- Zeile 24 schließt den Output-Stream für Datenobjekte.

Der nächste Abschnitt beschreibt das Macro stp_output. Da es von größerem Umfang ist, wird die Beschreibung in funktionale Abschnitte des Codes unterteilt.

```
1. %macro stp_output();
2.
3. /* Set numeric missing to ''. */
4. options missing='0';
5.
6. /* Library. */
7. libname data '/sas/sasdata/mayato/data/UC_H54S_Handsontable';
8.
```

- Zeile 4 legt die SAS-Systemoption MISSING auf „0“ fest, sodass fehlende Werte entsprechend mit 0 ersetzt werden.
- Zeile 7 definiert das libname statement für die Datenablage mit einem Pfad.

```
9. /* Query data to load handsontable with macro variables. */
10. proc sql noprint;
11. select name,
12. sex,
13. age
14. into  :name1 -,
15. :sex1 -,
16. :age1 -
17. from data.class;
18. select count(*)
19. into :numObs
20. from data.class;
21. quit;
22.
```

- Zeile 10 bis 20 dienen dazu, die Daten mit der SAS-Prozedur „proc sql“ aus der Tabelle „class“ in die jeweiligen Makrovariablen zu laden. Hierzu wird wiederum das Schlüsselwort „into:“ verwendet.

```
21. /* HTML output.*/
22. proc stream outfile=_webout;
23. begin
24.
25. <!doctype html>
26. <html>
27. <head>
28. <link rel="stylesheet"
29. type="text/css" href="/css/handsontable.full.min.css">
30. <script src="/scripts/handsontable.full.min.js"></script>
31. <script src="/scripts/h54s.min.js"></script>
32. <script src="/scripts/jquery-3.2.1.min.js"></script>
33. <style>
34. body { font-family: Verdana, Arial; font-size: 10px; }
35. </style>
36. </head>
37. <body>
38.
39. &streamDelim newline;
40.
41. <div class="controls" style="margin-bottom: 10px;">
42. <button name="psave" id="psave" class="intext-btn">Save</button>
43. </div>
44. <div id="pconsole" class="console"></div>
45. <div id="ptable"></div>
46.
47. &streamDelim newline;
```

- Zeile 22 und 23 markieren den Beginn eines Freitextbereichs, dessen Inhalt an eine externe Datei übertragen werden kann. Mit proc stream und der Filereferenz outfile=„_webout“ wird der HTML Output an den Browser gesendet.
- Zeile 25 leitet die Definition des HTML-Dokuments ein.
- Zeile 28 und 29 importieren ein Stylesheet im CSS-Format.
- Zeilen 30 bis 32 referenzieren die URL von externen Skripten. Diese werden entsprechend für handsontable, H54s und jQuery geladen. Die jQuery-Library wird für das Anzeigen der Konsole verwendet.
- Zeilen 33 bis 35 definieren die Schriftart und Schriftgröße im CSS Format.
- Zeile 39 forciert mit dem &steamDelim newline; einen Zeilenumbruch in der generierten Ausgabe.

```
49. <script type="text/javascript">
50. var
51. $$ = function(id) {return document.getElementById(id);},
52. container = $$('ptable'),
53. tableConsole = $$('ptableconsole'),
54. save = $$('psave'),
55. hot;
56.
57. var data = [
58. /* Add data to javascript array. */
59. %do i=1 %to &numObs. %by 1;
60. &streamDelim newline;
61. %if &i > 1 %then ,;
62. {"name":"&&name&i.",
63. "sex":"&&sex&i.",
64. "age":&&age&i.
65. }
66. %end;
67. ];
68.
69. &streamDelim newline;
70.
```

- Zeilen 41 bis 45 definieren Sektoren im HTML-Dokument, darunter der Save-Button, die Konsole und die Output-
- Zeile 49 definiert den Internet Media Type als JavaScript. Damit wird der Inhalt bis zum Ende des Scripts identifiziert.
- Zeilen 50 bis 55 ordnen HTML-Elemente den JavaScript Variablen zu, um den Zugriff in JavaScript zu ermöglichen.
- Zeilen 57 bis 67 erzeugen eine weitere JavaScript Variable, in der die Daten in ein JavaScript Array gelesen werden. Für jede Zeile (&numObs.) in den Daten wird ein neues Item im Array mit den Objekteigenschaften „name“, „sex“, und „age“ erstellt.

```
71. var container = document.getElementById('ptable');
72. var hot = new Handsontable(container, {
73. data: data,
74. rowHeaders: false,
75. colHeaders: true,
76. sortIndicator: true,
77. fillHandle: false,
78. manualColumnResize: true,
79. autoWrapRow: true,
80. colWidths: [80, 100, 100, 300, 300, 300],
81. contextMenu: ['row_below', 'remove_row'],
82. colHeaders: ['Name', 'Sex', 'Age'],
83. columns: [
84. {data: 'name'},
85. {data: 'sex'},
86. {data: 'age', type: 'numeric'}
87. ]
88. });
89.
90. &streamDelim newline;
```

- Zeile 71 erstellt eine neue Variable „container“ mit der Methode document.getElementByID(), die das Element aufruft, welches als ID hinterlegt ist (hier „ptable“).
- Zeilen 72 bis 88 definieren die Variable „hot“, in der mit der Methode new Handsontable die Einstellungen für die Tabelle eingetragen und den Spalten ihre jeweiligen Namen und Daten

```
91.  Handsontable.dom.addEvent(save, 'click', function() {
92.  var adapter = new h54s();
93.  var controlTable = [{ action: 'save' }];
94.  var h54sTables = new h54s.Tables(data,'data');
95.  H54sTables.add(controlTable, 'control');
96.  adapter.call(%bquote(')&_program.%bquote('),
97.  H54sTables, function(err, res) {
98.  if(err) {
99.  pconsole.innerHTML='An Error occoured!';
100. pconsole.style.color = "#D8000C";
101. pconsole.style.backgroundColor = "#FFBABA";
102. pconsole.style.border = '1px solid';
103. pconsole.style.padding = '10px';
104. pconsole.style.marginBottom = '10px';
105. $('#pconsole').fadeIn().delay(1000).fadeOut();
106. console.log(err);
107. } else {
108. pconsole.innerHTML='Data was saved.';
109. pconsole.style.color = "#4F8A10";
110. pconsole.style.backgroundColor = "#DFF2BF";
111. pconsole.style.border = '1px solid';
112. pconsole.style.padding = '10px';
113. pconsole.style.marginBottom = '10px';
114. $('#pconsole').fadeIn().delay(1000).fadeOut();
115. console.log(res);
116. }
117. });
118. });
119.
120. &streamDelim newline;
121.
122. </script>
123. </body>
124. </html>
125.
126. ;;;;
127. run;
128.
129. %mend stp_output;
```

- zugewiesen werden. Dabei wird auch die Variable „container“ aufgerufen und das Element „ptable“ weitergegeben.
- Zeilen 91 bis 118 definieren die Funktion des „save“-Buttons.

- Zeile 91 definiert den Utility Wrapper für JavaScript Events Handsontable.dom.addEvent. Damit kann das Klick-Event des Save-Buttons für die in den darauffolgenden Zeilen definierte Funktion verwendet werden.
- Zeile 92 öffnet eine Instanz des H54s-Adapters, in diesem Beispiel mit der URL des Servers der Test-Umgebung.
- Zeile 93 erstellt ein neues JavaScript Array „control“ mit dem Eintrag „save“.
- Zeile 94 erstellt ein Tabellenobjekt (h54sTables), an das eines oder mehrere Objektarrays angebunden werden können. In dieses Table-Objekt wird direkt die Daten-Tabelle eingefügt.
- Zeile 95 fügt dem H54sTabellenobjekt das JavaScript Array „control“ hinzu.
- Zeilen 96 bis 97 dienen dazu, mit einem AJAX Aufruf und der &_programm Standardmacrovariable den Stored Process stp_run erneut aufzurufen. Dabei werden die H54s-Tabellenobjekte (data und control) übergeben und eine Funktion definiert, die den Rückgabewert verarbeitet.
- Bei dem erneuten Aufruf von stp_run im Hintergrund wird die data-Tabelle und die control-Tabelle mitgegeben. Da die control-Tabelle nun existiert, wird das stp_save-Macro ausgeführt.
- Zeilen 99 bis 115 erstellen die Rückmeldung des Speichervorgangs mit entsprechenden Style-Properties für das pconsole Objekt. „err“ ist ein speziell gebautes JavaScript Error Objekt mit einem dazugehörigen Typenfeld, „type“, in dem die Fehlerart spezifiziert wird ("loginError", "notLoggedinError", "parseError", "sasError", oder eine http Rückmeldung).
- Damit wird die Rückmeldung für den Speichervorgang erzeugt. Tritt beim Speichern ein Fehler auf, löst dies die Fehlermeldung mit der Nachricht 'An Error occoured!' aus. Ist das Speichern erfolgreich, wird 'Data was saved.' ausgegeben.

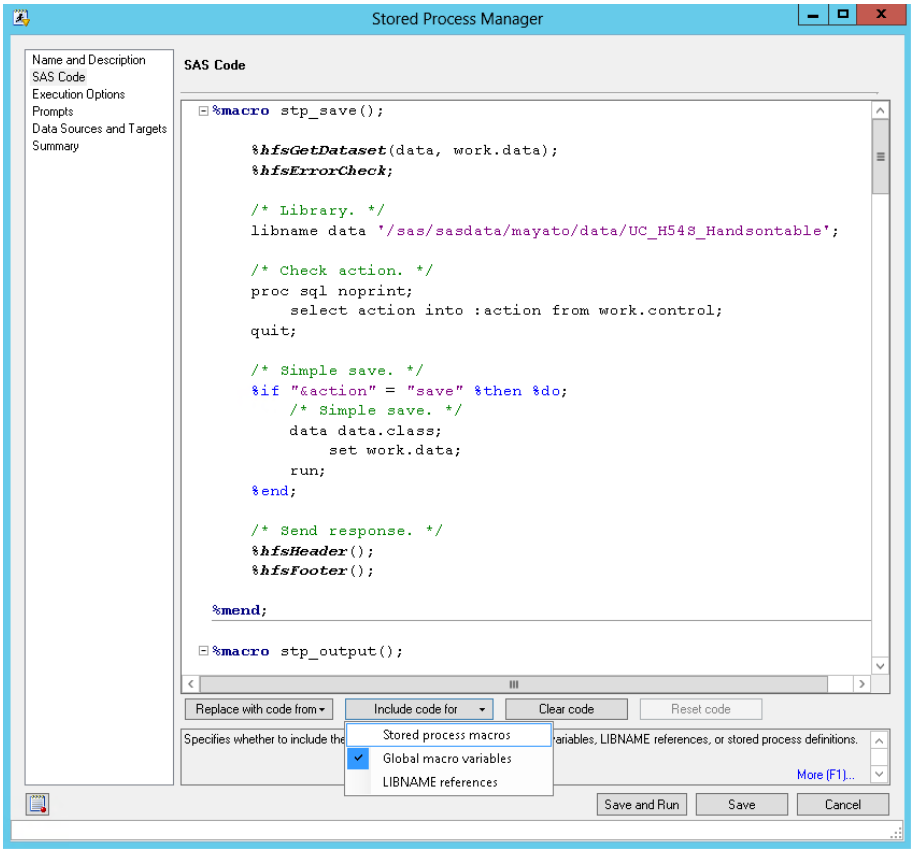

Abbildung 6: Stored Process Einstellungen - SAS Code

Vorteile und Nutzen der verwendeten Tools

Handsontables ist eine beliebte JavaScript Komponente. Im präsentierten Beispiel zeigt sich die einfache Implementierung in Web-Applikationen. Nutzern kann damit eine Oberfläche für Kalkulationstabellen mit bekannter Handhabung geboten werden. Typische Anwendungsfälle können z. B. in der Bearbeitung von Datenbanken, der Finanzanalyse, dem Sales-Reporting und der Personalplanung gefunden werden.

Handsontable bietet in der Community-Edition einen breiten Funktionsumfang mit Excel-Funktionalitäten, bedingte Formatierungen, Checkboxes, anpassbares HTML, Drag-down, Suchfelder, Sortierungsfunktionen uvm. Die Pro-Edition erweitert diesen Umfang um Premium Features wie z. B. Dropdown Menu, Zeilenfilter und Exportfunktionen.

Auch der Boemska H54s-Adapters bietet diverse Vorteile

Seitens der Entwicklung und des täglichen Gebrauchs ist besonders die Unabhängigkeit von der IT Infrastruktur von Nutzen. Neben geringer Entwicklungszeit hat der Adapter außerdem den Vorteil, dass er ohne Lizenzkosten eingesetzt werden kann. Der Adapter ist vollständig HTML 5 standardkonform und benötigt keine zusätzlichen Komponenten.

Sämtliche Nutzerrechte und Rollen können durch Metadaten gesteuert und von SAS® Management Console-Administratoren verwaltet werden. Die Architektur ermöglicht einfache Anmeldevorgänge (Single Sign-On) und ein User- und Rollenmanagement, welches durch Metadaten gesteuert und von SAS® Management Console Administratoren verwaltet werden kann.

Natürlich profitieren die Applikationen auch von den umfassenden Analysemöglichkeiten der SAS® Software.

Fazit

In diesem Whitepaper wurden die JavaScript Komponente Handsontable und der Boemska H54s Adapter mit einem einfachen Beispiel einer interaktiven Web Applikation demonstriert. Neben einer Übersicht über die verwendete Technologie wurde eine Erläuterung des Beispielcodes gegeben.

Es zeigt sich wie mit HTML und JavaScript in kurzer Zeit professionelle Web-Applikationen mit SAS® Unterstützung erstellt werden können. Dabei kann auch die Datenhaltung von SAS® und eine Datenkommunikation zwischen SAS® und der Web Applikation genutzt werden. Dem Nutzer wird dabei eine übersichtliche Oberfläche im Excel-Feeling geboten, mit der Daten vom Webbrowser direkt in SAS eingegeben werden können.

Kontaktieren Sie uns //

mayato GmbH
Am Borsigturm 9
13507 Berlin
info@mayato.com
+49 | 30 4174 4270 10