Daten aus der BI-Plattform - elegant visualisiert im Web

Klaus Kepert
HMS Analytical Software GmbH
Rohrbacher Straße 26
69115 Heidelberg
Klaus.Kepert@analytical-software.de

Zusammenfassung

Wie kombiniert man die beliebig flexible Design-Vielfalt der Web-Welt mit der mächtigen Datentechnologie einer SAS-BI Plattform? Der Beitrag skizziert zum einen, wie ein SAS Stored Process (STP) ein vorbereitetes HTML-Template liest, darin SAS-Makrovariablen durch Daten aus der BI-Plattform ersetzt und das Ergebnis an einen beliebigen Browser sendet. Zum anderen wird veranschaulicht, wie ein Web-Frontend interaktiv Daten mit Stored Processes austauschen kann.

Schlüsselwörter: SAS BI-Plattform, Stored Process, HTML, Excel

1 Motivation

Die SAS BI-Plattform bietet mit dem Stored Process eine sehr einfach zu nutzende Möglichkeit, um Anwendern ein Reporting im Browser zur Verfügung zu stellen. Im SAS Information Delivery Portal können Anwender SAS Stored Processes starten und erhalten die mit SAS-Mitteln erstellten Reports. Auch Fragen an den Anwender können mit dem Prompting Framework den Stored Processes mit gegeben werden.

Wenn ein Report in Gestaltung und Design individuell sein soll oder der Anwender Möglichkeiten der Eingabe von Information braucht, die mit dem Prompting Framework so nicht abbildbar sind, dann eröffnet sich eine nahezu grenzenlose Design Vielfalt bei der Verwendung von HTML. Damit wird möglich, was im Internet möglich ist. Dennoch kann im Hintergrund die mächtige Datenmaschine mit Zugriff auf DWH Daten genutzt werden, indem die SAS BI Stored Processes nicht mehr Reports sondern Daten für HTML-Seiten oder das HTML selbst liefern.

1.1 Report Design mit PROC REPORT

Das Report Design mit Proc Report bietet aufregende Gestaltungsmöglichkeiten. Um die komplexeren Hausforderungen der HTML Welt zu meistern, sollen hier einige Tipps gegeben werden.

2 Wie funktioniert es?

1. Man erstellt eine HTML-Seite oder speichert einen Excel Prototyp als HTML Seite ab und verfügt somit über ein HTML Template.

2. Das HTML Template wird modifiziert, indem man an den Positionen, die Daten zeigen sollen, eine Makrovariable platziert. Die Makrovariablen können auch schon im Excel eingetragen werden, sind dann aber im HTML maskiert (s.u.).

3. Man schreibt ein SAS Programm, das die Daten beschafft. Dieses wird um einen DATA Step am Ende erweitert, der das HTML-Template einliest, die Makrovariablen ersetzt und an den Browser sendet.

4. Das HTML-Template kann mehrere Links enthalten, die weitere Stored Processes starten.

Wenn in einem Stored Process HTML an den Browser gesendet werden soll, erfolgt dies über den filename *_webout*. Dies muss der Benutzer, wie bereits am Unterstrich zu ersehen, nicht selbst setzen. Alles was man in diesen Filename schreibt, kommt im Browser des Anwenders an.

Um jedoch die volle Kontrolle über *_webout* zu erlangen, dürfen die SAS eigenen Makros *%STPBEGIN* und *%STPEND* nicht genutzt werden: Ansonsten ist der Filename *_webout* mit einer Datei auf der Platte verbunden und der Benutzer kann diesen nicht selbst im Code nutzen.

Mit *STP modifizieren* im Kontextmenü, über den Tab *SAS Code* und in der Auswahl *Code einschließen* das Häkchen bei *Stored Process-Makros* entfernen.

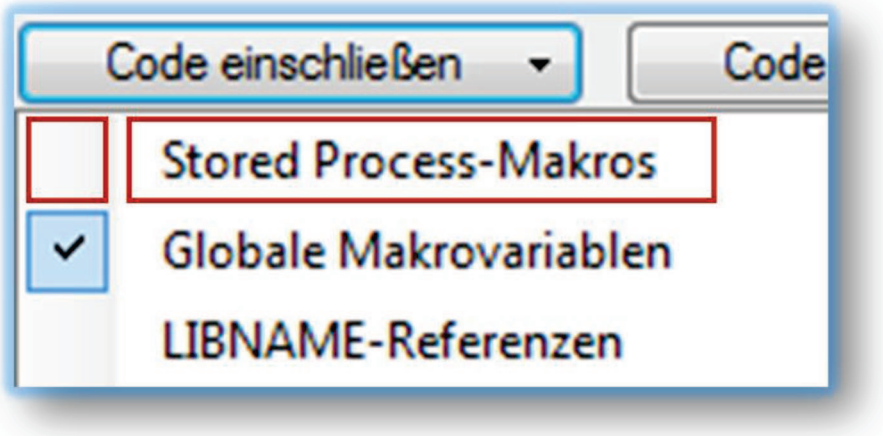


Abbildung 1: Stored Process Makros abschalten

2.1 Szenario: HTML Report mit einer Grafik

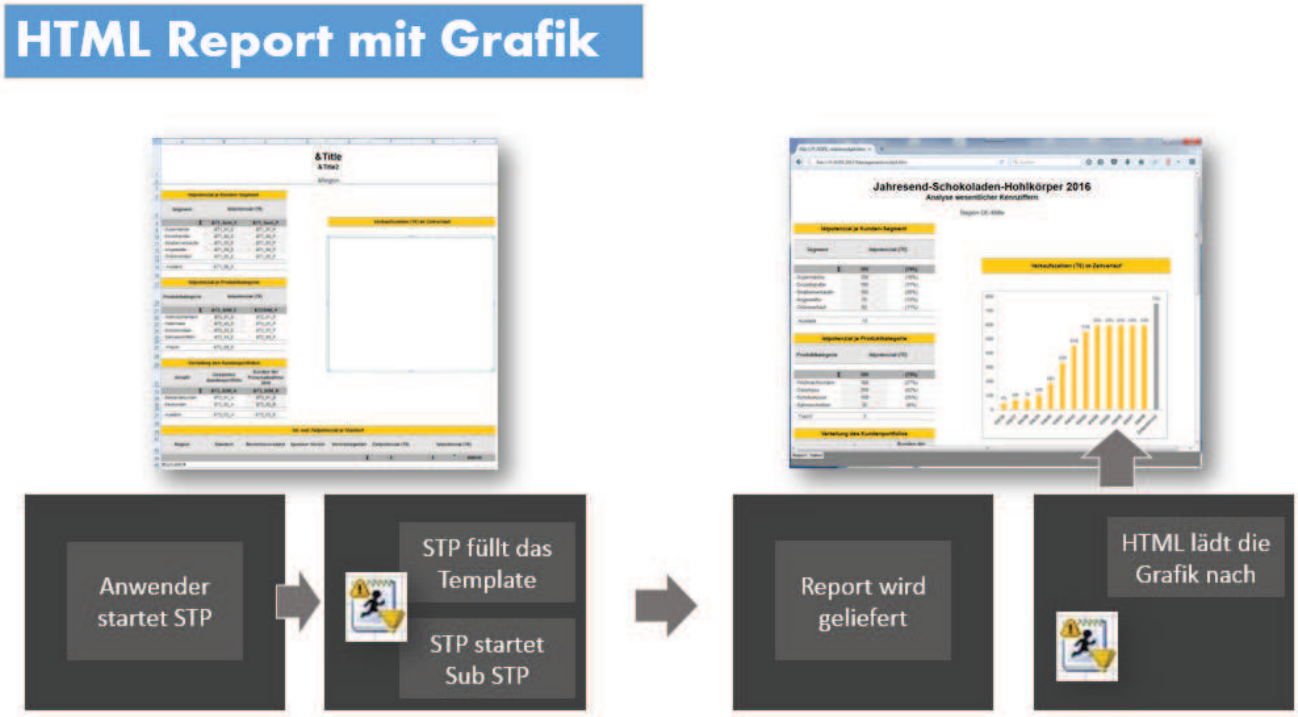


Abbildung 2: Szenario HTML Report mit Grafik

Der gestartete Stored Process startet einen zweiten Sub-Stored Process, berechnet die Daten und sendet die HTML-Seite an den Browser. Die HTML-Seite enthält einen Image-Tag, der den Grafik-Stored Process aufruft. Dieser kapert den Sub-Stored Process und verfügt damit schon über die Daten, welche er benötigt. Er erzeugt die Grafik und liefert diese zurück in die aufrufende HTML-Seite.

2.2 Szenario: HTML Eingabemaske mit Nachverarbeitung

Der Anwender startet einen Stored Process, der die Eingabemaske als HTML an den Browser sendet. In der HTML-Seite ist ein Stored Process Aufruf enthalten, der durch einen Button ausgelöst wird.

Eine Sub-Session kann hier nicht genutzt werden, weil diese mit einem Timeout abläuft und man nicht vorhersehen kann, wie lange der Anwender für die Eingabe benötigt.

Die Übergabe der Makrovariablen funktioniert über HTML-Input-Tags jeder Art.

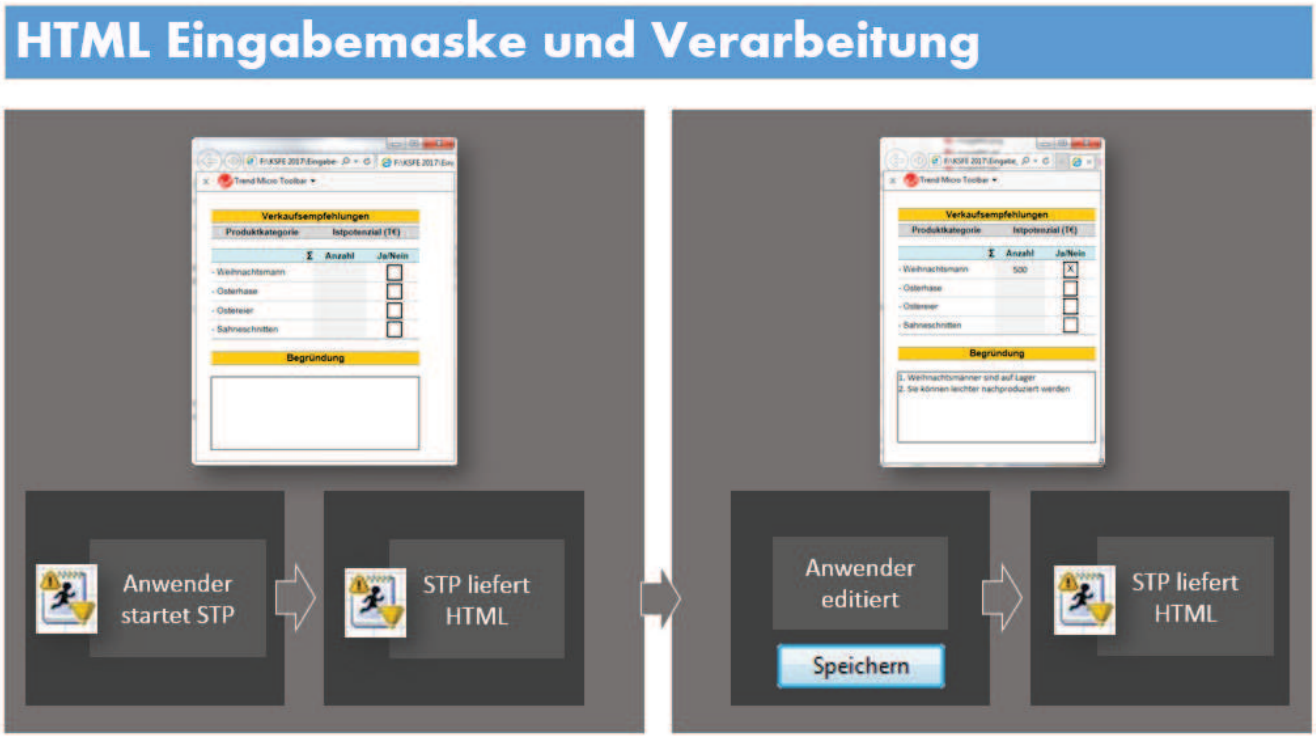


Abbildung 3: Szenario Eingabemaske und Verarbeitung

2.3 Szenario: HTML Eingabemaske mit PDF-Antwort

In diesem Szenario wird zusätzlich ein PDF-Dokument zurück geliefert, wenn der Anwender den Speichern-Button drückt.

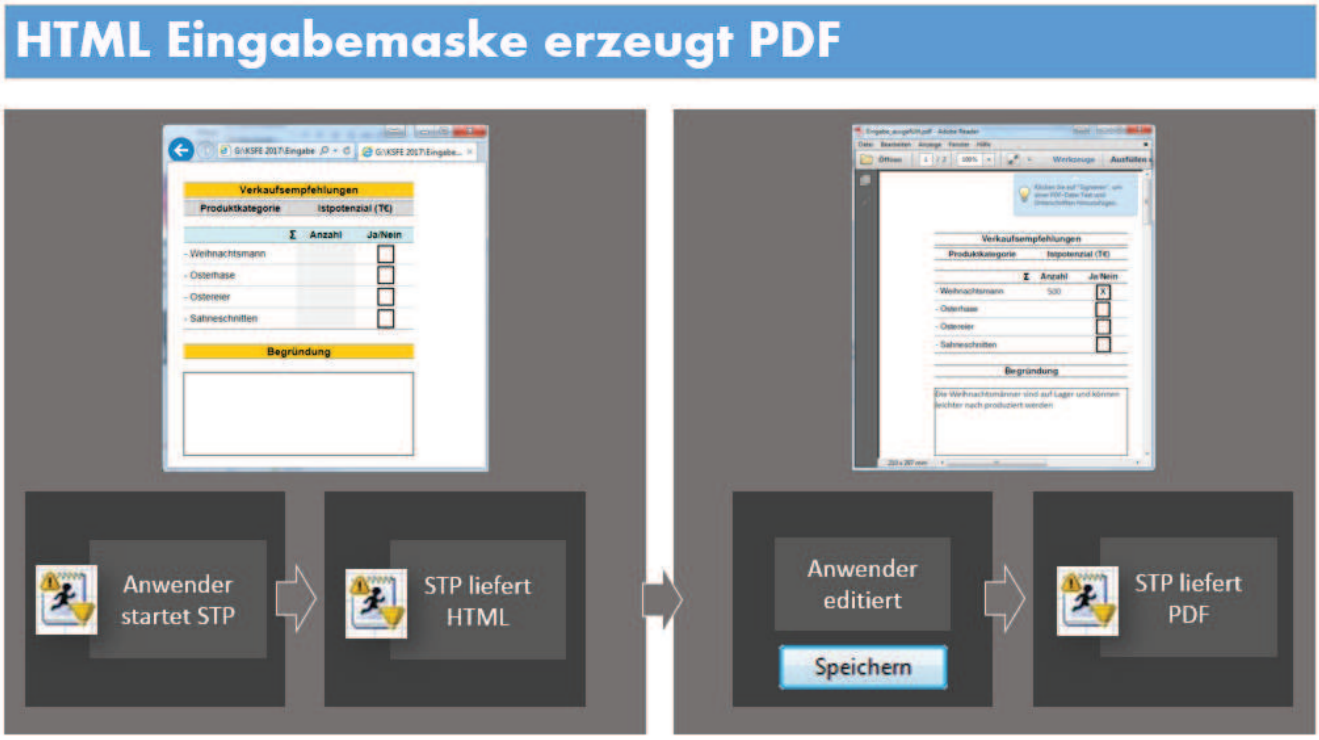


Abbildung 4: Szenario Eingabemaske erzeugt PDF

Beispiel: Ein PDF Dokument in HTML einbetten.

```
<div id="myDiv">
<embed type="application/pdf"
src="https://&Webserver/SASStoredProcess/do?_program=PDF_REPORT"
id="pdfDocument">
</embed>
</div>
```

3 HTML Seite erstellen

Ziel ist es, über ein HTML-Template zu verfügen, das von SAS Stored Processes vervollständigt und zum Anwender gesendet wird.

3.1 HTML programmieren

Die HTML-Sprache ist einfach zu erlernen und der Entwickler verfügt über eine nahezu grenzenlose Bibliothek an Dokumentationen, Beispielen und Designer-Tools im Internet.

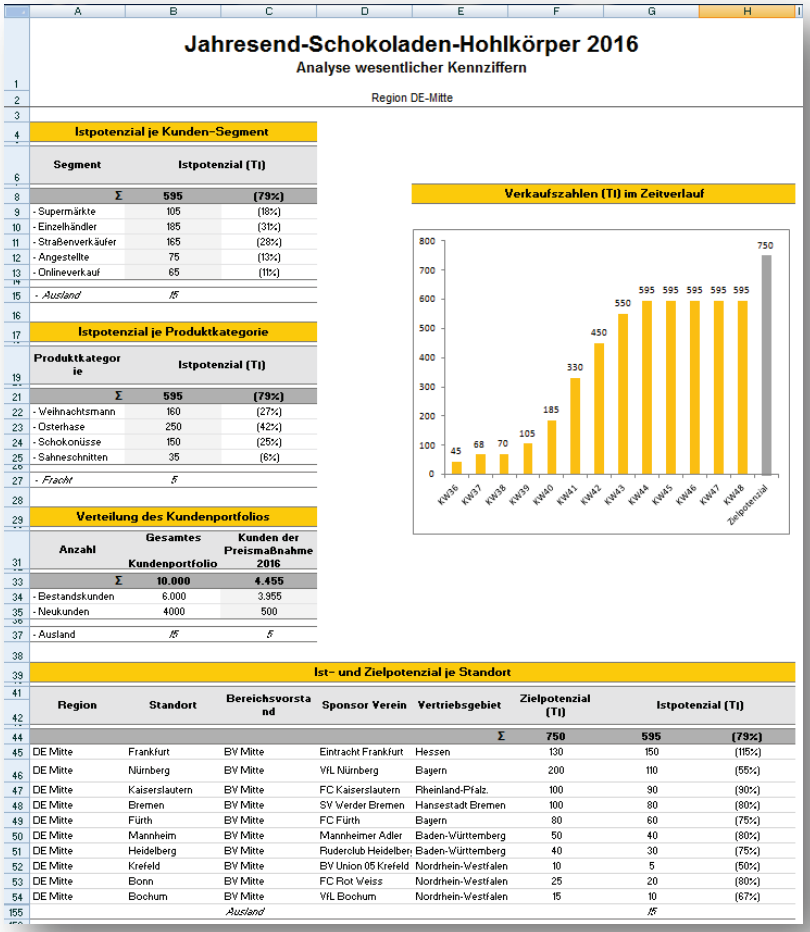
```
<html>
 <head>
 </head>

 <body>
   <!— Hier meine eigene HTML Programmierung -->
</body>
</html>
```

3.2 HTML mit Excel erstellen

Heute, mehr denn je, ist Microsoft Excel das Tool der Wahl, wenn es um die Erstellung von Prototypen geht. In Excel lassen sich sehr individuell Strukturen aufbauen.

Das fertige Ergebnis kann man nun als HTML-Seite abspeichern und nach ein paar kleineren Anpassungen als HTML-Template verwenden. Prinzipiell funktioniert das mit jeder Anwendung, die HTML-Code exportieren kann. Eine vorfertige Homepage kann um dynamische Inhalte ergänzt werden.



Jahresend-Schokoladen-Hohlkörper 2016

Analyse wesentlicher Kennziffern

Region DE-Mitte

Istpotenzial je Kunden-Segment

Segment	Istpotenzial (TI)	
Σ	595	(79%)
- Supermärkte	105	(18%)
- Einzelhändler	185	(31%)
- Straßenverkäufer	165	(28%)
- Angestellte	75	(13%)
- Onlineverkauf	65	(11%)
- *Ausland*	*15*	

Istpotenzial je Produktkategorie

Produktkategorie	Istpotenzial (TI)	
Σ	595	(79%)
- Weihnachtsmann	160	(27%)
- Osterhase	250	(42%)
- Schokonüsse	150	(25%)
- Sahneschnitten	35	(6%)
- *Fracht*	*5*	

Verteilung des Kundenportfolios

Anzahl	Gesamtes Kundenportfolio	Kunden der Preismaßnahme 2016
Σ	10.000	4.455
- Bestandskunden	6.000	3.955
- Neukunden	4000	500
- Ausland	*15*	*5*

Verkaufszahlen (TI) im Zeitverlauf

Ist- und Zielpotenzial je Standort

Region	Standort	Bereichsvorstand	Sponsor Verein	Vertriebsgebiet	Zielpotenzial (TI)	Istpotenzial (TI)	
				Σ	750	595	(79%)
DE Mitte	Frankfurt	BV Mitte	Eintracht Frankfurt	Hessen	130	150	(115%)
DE Mitte	Nürnberg	BV Mitte	VfL Nürnberg	Bayern	200	110	(55%)
DE Mitte	Kaiserslautern	BV Mitte	FC Kaiserslautern	Rheinland-Pfalz.	100	90	(90%)
DE Mitte	Bremen	BV Mitte	SV Werder Bremen	Hansestadt Bremen	100	80	(80%)
DE Mitte	Fürth	BV Mitte	FC Fürth	Bayern	80	60	(75%)
DE Mitte	Mannheim	BV Mitte	Mannheimer Adler	Baden-Württemberg	50	40	(80%)
DE Mitte	Heidelberg	BV Mitte	Ruderclub Heidelber	Baden-Württemberg	40	30	(75%)
DE Mitte	Krefeld	BV Mitte	BV Union 05 Krefeld	Nordrhein-Westfalen	10	5	(50%)
DE Mitte	Bonn	BV Mitte	FC Riot Weiss	Nordrhein-Westfalen	25	20	(80%)
DE Mitte	Bochum	BV Mitte	VfL Bochum	Nordrhein-Westfalen	15	10	(67%)
		Ausland				*15*	

Abbildung 5: Vorbereitung in Excel

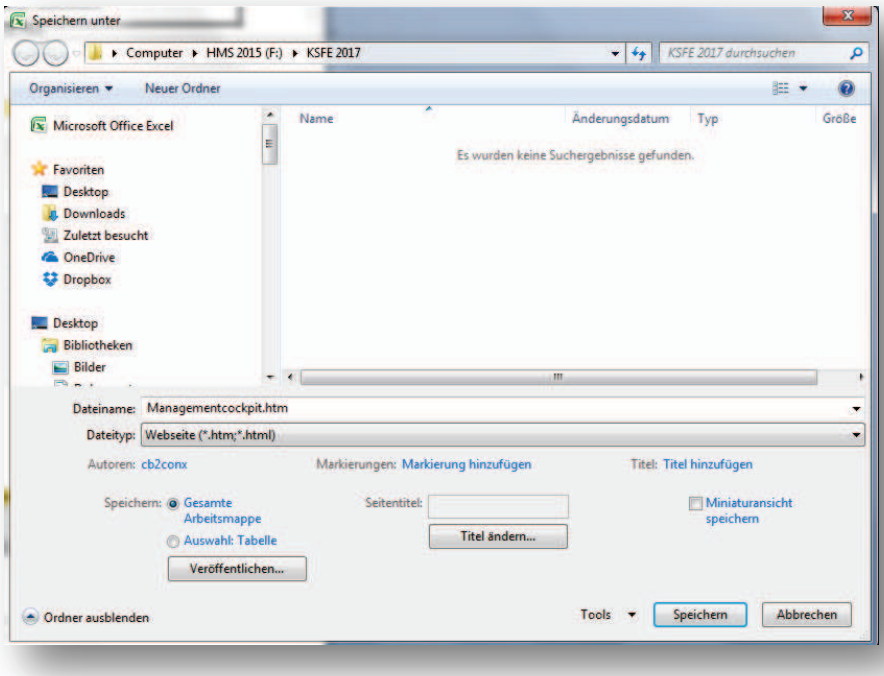
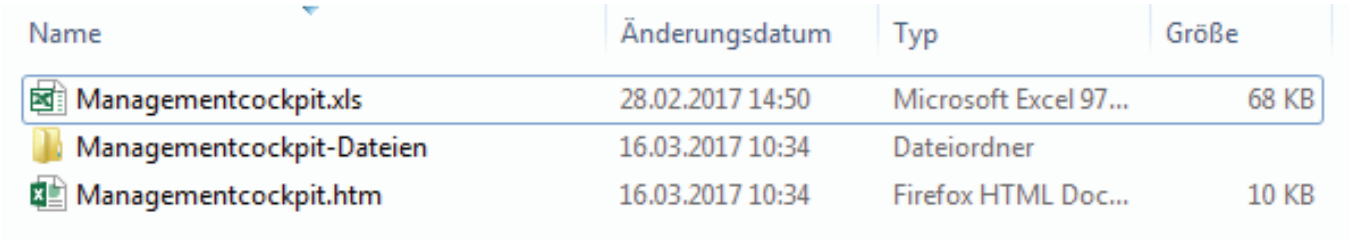


Abbildung 6: HTML erzeugen mit „Datei speichern unter…“

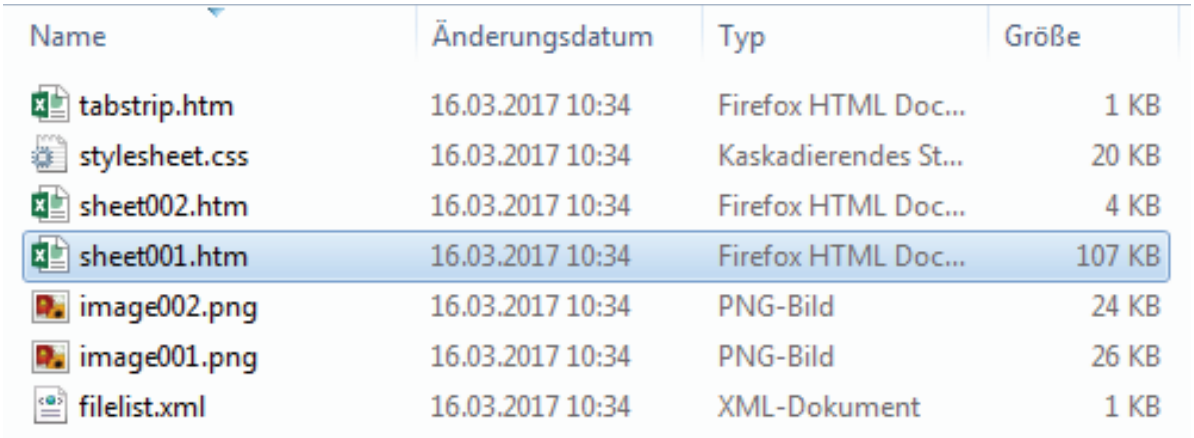
Dabei entstehen eine HTML-Seite und ein gleichnamiges Unterverzeichnis. Die HTML-Seite bildet eine Rahmenseite um die einzelnen Sheets herum und ist deshalb selbst uninteressant.



Name	Änderungsdatum	Typ	Größe
Managementcockpit.xls	28.02.2017 14:50	Microsoft Excel 97...	68 KB
Managementcockpit-Dateien	16.03.2017 10:34	Dateiordner	
Managementcockpit.htm	16.03.2017 10:34	Firefox HTML Doc...	10 KB

Abbildung 7: HTML Struktur

Im gleichnamigen Unterverzeichnis sind die einzelnen Sheets abgelegt, die die Inhalte enthalten. Diese sind als Template verwendbar.



Name	Änderungsdatum	Typ	Größe
tabstrip.htm	16.03.2017 10:34	Firefox HTML Doc...	1 KB
stylesheet.css	16.03.2017 10:34	Kaskadierendes St...	20 KB
sheet002.htm	16.03.2017 10:34	Firefox HTML Doc...	4 KB
sheet001.htm	16.03.2017 10:34	Firefox HTML Doc...	107 KB
image002.png	16.03.2017 10:34	PNG-Bild	24 KB
image001.png	16.03.2017 10:34	PNG-Bild	26 KB
filelist.xml	16.03.2017 10:34	XML-Dokument	1 KB

Abbildung 8: HTML Einzelsheet

Die Grafik ist im HTML binär abgelegt. Finden Sie den Datenblock in der HTML-Seite und löschen Sie ihn heraus (zwischen <td> Tag und </td> Tag). Merken Sie sich die Stelle, wenn die Grafik später an der gleichen Position bleiben soll.

Das HTML-Template ist noch nicht einsatzbereit, weil die in HTML übliche Maskierungen von Sonderzeichen, wie z.B. das *&* in SAS nicht mehr funktioniert.

```
*** & wird zu &amp; ***;

<td class=x169 style='border-top:none'>&amp;T1_sum_E</td>
```

Mit *Suchen, ersetzen, alle* kann man das mit einem Texteditor korrigieren.

```
<td class=x169 style='border-top:none'>&T1_sum_E</td>
```

Wenn im HTML-Template maskierte Sonderzeichen enthalten sind, die nicht zu den SAS Makrovariablen gehören, sollen diese nicht als solche interpretiert werden. Dazu legt man eine Makrovariable an, die die Maskierung mit sich selbst ersetzt.

Beispiel ein *non breakable space* - * *

```
*** Der HTML-Code " "
    soll nicht als Makrovariable interpretiert werden ***;

%LET nbsp=%nrstr( );
```

Fertig ist das HTML-Template.

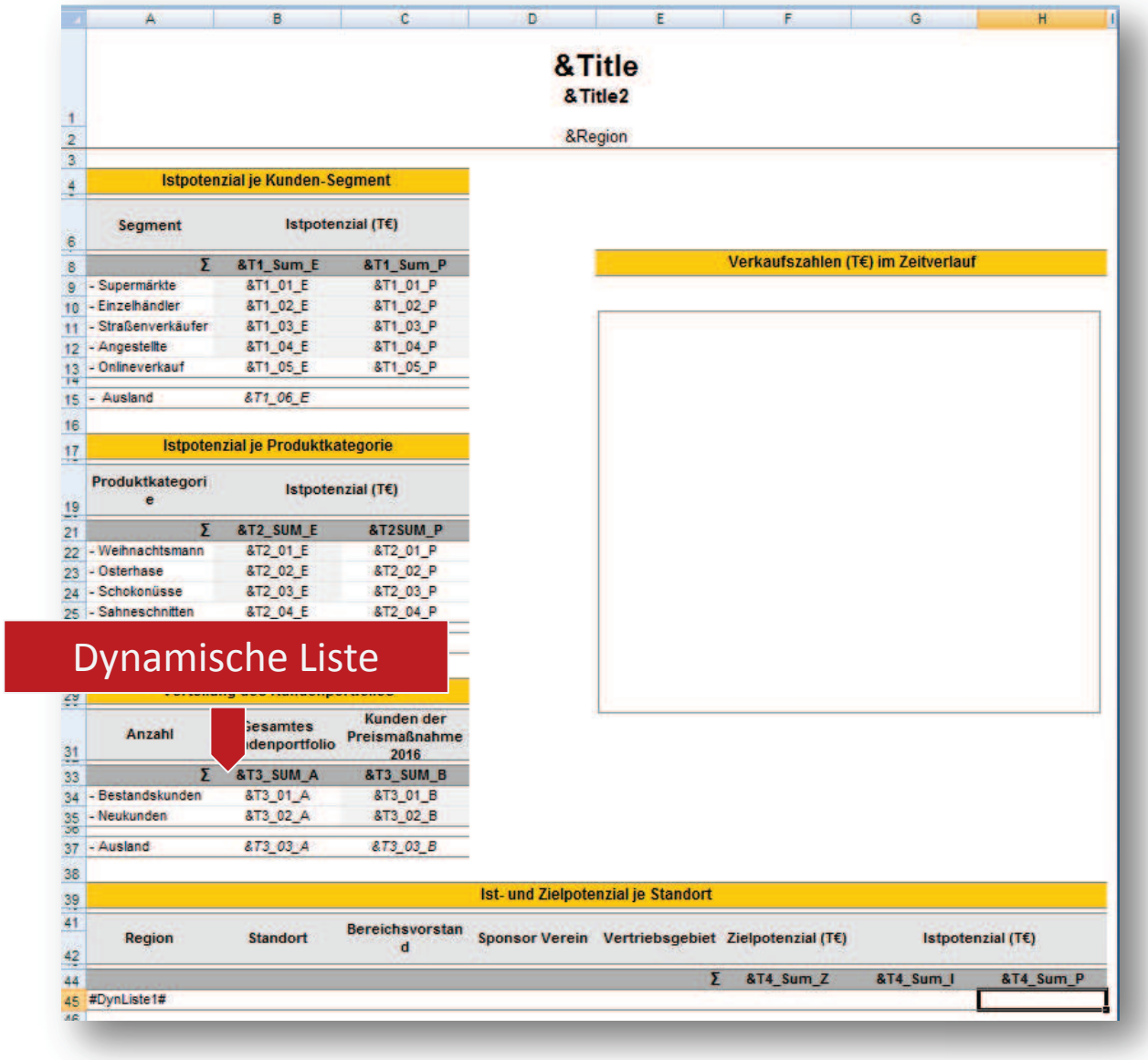


Abbildung 9: HTML Template

3.3 HTML-Template füllen und senden

Das Template wird mit einem *Data _NULL_* Step eingelesen. Mit der *Resolve()-Funktion* werden die Makrovariablen ersetzt und an den Browser gesendet. Die Makrovariablen müssen zuvor erzeugt worden sein.

```
/*-----------------------------------------------------------
  HTML-Template einlesen, Makrovariablen einsetzen und senden
 -----------------------------------------------------------*/
Data _NULL_;
  Infile "/daten/html/template_html.sas";
  Length Line $32000;
  Input;

  Line = Resolve(_infile_); *** Makrovariablen ersetzen ;

  File _webout;
  Put Line;
Run;
```

Das HTML Template hat die Endung .sas, weil es dann im EG geöffnet und editiert werden kann. Beim Einlesen spielt die Endung keine Rolle.

Alternativ kann mit SAS 9.4 auch Proc Stream genutzt werden.

```
/*-----------------------------------------------------------
  HTML-Template einlesen, Makrovariablen einsetzen und senden
 -----------------------------------------------------------*/
 Proc Stream Outfile=_webout;
   Begin
   %include "/daten/html/template_html.sas";
   ;;;;
Run;
```

3.4 Grafik einbinden

Um die Grafik einzubinden muss an der vorgesehenen Stelle ein Image-Tag <IMG> angelegt werden. Im Parameter SRC wird der Aufruf des Stored Process eingetragen, welcher die Grafik erzeugt. Die an den Browser gesendete HTML-Seite lädt die Grafik nach, indem der Stored Process aufgerufen wird.

Um die Daten im Grafik Programm nicht nochmals beschaffen zu müssen, kann der Stored Process einen Sub-Stored Process anlegen, der alle Makrovariablen übernimmt, die mit **save_** beginnen und es steht eine SAS Bibliothek mit dem LIBREF save zur Verfügung.

So wird der Sub-Stored Process angelegt:

```
  %let rc=%sysfunc(stpsrv_session(create));
  %put &rc;
  %put &_sessionid;
```

Dieser Image-Tag kapert den Sub-Stored Process, der die Grafik nachlädt, indem er die Session-ID als Parameter übergibt. Sie ist als Makrovariable verfügbar, sobald eine Sub-Session angelegt wurde.

.

```
<img alt='Grafik wird geladen...'
 align='right'
 src="https://&Server./SASStoredProcess/do?_sessionid=&_sessionid."
    '&_program=/Berichte/Grafik'
 border="0" >
```

Der Parameter *&_program* ist keine Makrovariable, sondern gehört zur Aufrufsyntax. Deshalb steht dieser in einfachen Hochkommata.

3.5 Eingabemasken

Eingaben in HTML erfolgen mit Input Tags. Jedes Input Tag erzeugt ein Eingabefeld in der HTML-Seite. Auch Checkboxen, Listboxen oder Radio Buttons sind möglich und über den Type steuerbar.

Beispiel für eine Checkbox

```
<input type='checkbox' name='drucken' checked ='checked'>"
```

Beispiel für mehrzeilige Eingabefelder, sogenannte Textareas.

```
<textarea cols="100" rows="10" class='taba'
name='Tab05_01'>&Tab05</textarea>
```

Im HTML-Template muss ein STP enthalten sein, der erst beim Klick auf den *Speichern Button* ausgeführt wird. Der SAS Aufruf ist innerhalb des Form-Tags als Action eingetragen. Der auszuführende Stored Process steht in einem Input Tag mit dem Namen *_program*. Das Input Tag ist vom Type *hidden*, damit er am Bildschirm nicht auftaucht.

Beispiel für einen aktionsabhängigen SAS Aufruf:

```
<form action="https://&Webserver./SASStoredProcess/do" method='post'
enctype='multipart/form-data'>
<input type='hidden' name='_program' value="&REPORT" >
```

Beispiel für einen Submit-Button, der den STP aufruft:

```
<input type="image"
src="/SASBIRes/msbrep/resources/button_drucken_pb.gif" alt="Submit">
```

Das Besondere ist nun, dass alle Eingabe Tags automatisch, ohne weiteres Zutun, als Makrovariablen im STP zur Verfügung stehen. Das auch dann, wenn sie vom Type *hidden* und deshalb für den Anwender unsichtbar sind. Solche *hidden Input Tags* kann man verwenden, um Makrovariablen durchzureichen.

```
                          Makrovariable     Ausprägung
<input type="hidden" name="PGM1" value="&PGM1">
```

Besonderheit bei mehrzeiligem Text:
In der Regel soll die Position der Zeilenumbrüche beibehalten werden. Werden mehrzeilige Texte in einem PDFausgegeben, werden die Zeilenumbrüche nicht interpretiert und gehen verloren.
Deshalb müssen diese in *Escape Sequenzen* umgewandelt werden.

Mehrzeiliger Text

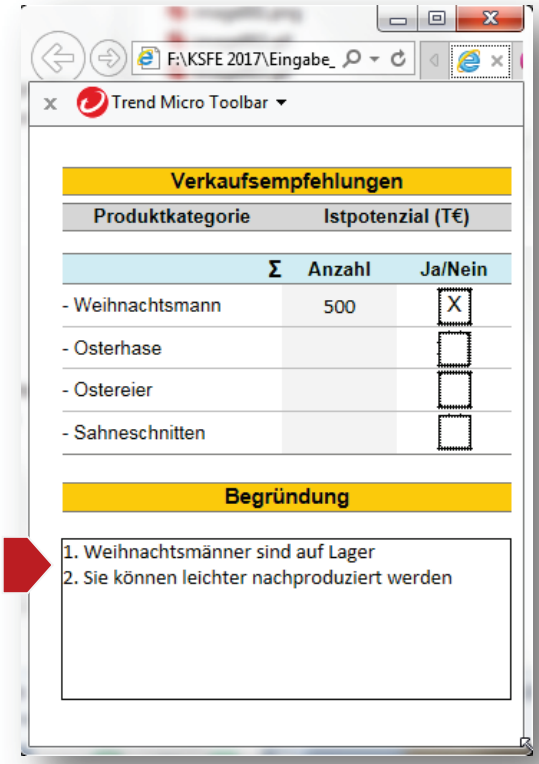


Abbildung 10: Mehrzeiliger Text

Beispiel für die Korrektur des Zeilenumbruches in der Makrovariablen &Tab_01:

```
/*-------------------------------------------------------------------
  Den Zeilenumbruch mit Escape Zeichen maskieren
 -------------------------------------------------------------------*/

ODS escapechar="~";

Data _NULL_;
  var = TRANWRD("&Tab_01",'0D0A'x,'~n');
  Call Symput ('Tab_01',var);
Run;
```

3.6 Dynamische Listen

Wenn die Anzahl der Zeilen einer Tabelle von den Daten abhängt, liest man das HTML-Template bis zum Beginn der Daten ein. Entweder mit einem Teil-Template oder mit einem Marker, nach dem man sucht und dann unterbricht. Die Daten lassen sich in einem Data _NULL_ Step zum Anwender senden.

```
Data _NULL_;
   Set Ergebnis(Keep= Region Standort Bereichsvorstand
                      Sponsor_Verein  Vertriebsgebiet
                      Zielpotential   Istpotential);
   File _webout;
   *** Table-Row öffnen *;
   Put "<tr height=20 style='mso-height-
       source:userset;height:15.0pt'>";

   *** Zellen schreiben
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Region            "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Standort          "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Bereichsvorstand  "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Sponsor_Verein    "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Vertriebsgebiet   "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Zielpotential     "</td>";
   Put "<td height=20 class=xl92 style='height:15.0pt'>"
       Istpotential      "</td>";

   Put "</tr>"; *** Table-Row schließen *;
Run;
```

4 Fazit

Wenn individuell gestaltete Reports gefragt sind, kann man sich die Arbeit erheblich erleichtern indem man ein HTML Template erzeugt. Das geht aus jeder Anwendung heraus, die HTML erzeugen kann. Hier im Beispiel war es Excel. Ein Basis Know-How in der HTML Syntax ist notwendig, aber auch einfach zu erlernen.

Ist eine erweiterte Funktionalität im Report gefragt, kommt JavaScript ins Spiel.

Bei größeren Datenmengen kann man ein Tabellen-Element in HTML verwenden, das von JSON-Objekten (JavaScript Object Notation) befüllt wird.