PHP automatisiert Word

Microsoft Word ist ein zentraler Bestandteil in nahezu allen Büros dieser Welt. Geht es um Einzel- oder Serienbriefe, Tutorials oder ganze Bücher: Word ist immer die Zentrale. Mittels COM kann PHP Ihnen da einiges an Arbeit abnehmen und Word dabei voll ausreizen.

von Thomas Wiedmann

Microsoft Word ist das zentrale Werkzeug, um Dokumente aller Couleur zu erzeugen. Mittels Automatisierung lässt sich Word auch als Textgenerator nutzen. Briefe werden automatisch erzeugt, Formulare selbständig ausgefüllt. Selbst drucken oder versenden per E-Mail ist kein Problem. Um beispielsweise einen Einzelbrief automatisiert mit Word zu erzeugen – und darum geht es im folgenden Artikel – bedarf es Folgendem :

- SQL Zugriff auf eine Datenbank via ODBC
- Korrekt aufbereitete und formatierte Adressdaten
- eine passende Word-Vorlage mit Bookmarks (.DOT)
- COM Automatisierung mit PHP

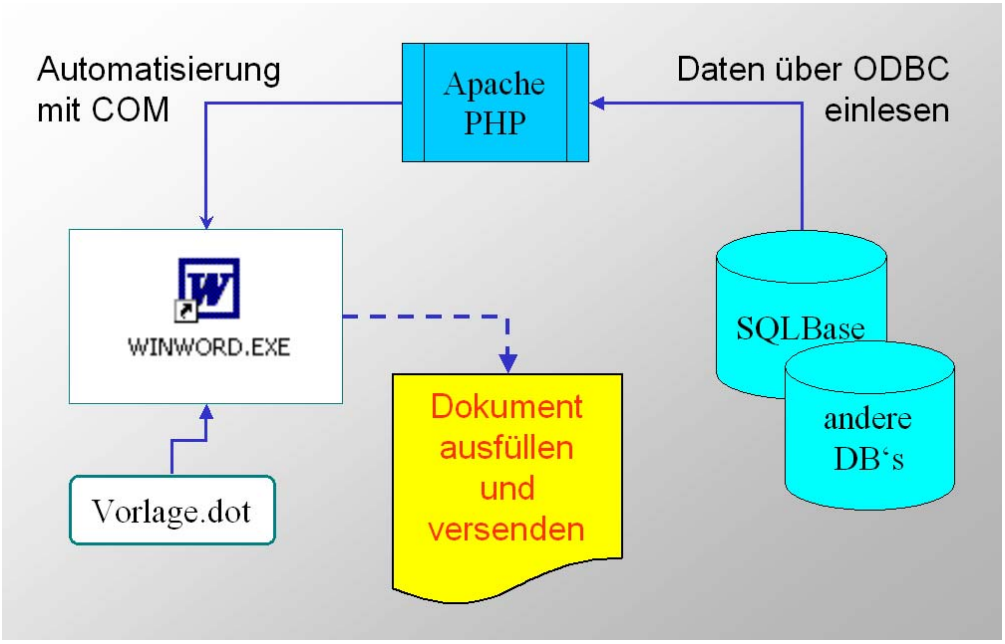

Bild 1: Zeigt das Ablaufschema, wie PHP ein Word-Dokument erstellt

Folgendes ist geplant: Auf einem Web-Server für ein Intranet ist Apache, PHP und MS-Word installiert. Weiterhin gibt es eine Datenbank aus der die Adressen gelesen werden. Für DB2 könnte dies eine (gar nicht so) kleine UDF (User Defined Function) sein, die die Adresse gleich DIN-gerecht aufbaut und an PHP en bloc übergibt. Im Falle von Gupta SQLBase erfolgt die Formatierung der Adresse innerhalb der PHP Klasse. Beginnen wir mit dem obligatorischen Blick auf die eingesetzen Komponenten:

- PHP Version 5.0.4 für Windows
- Apache/1.3.28 (Win32)
- Centura SQLBase 7.5.1 (Developer Edition)
- Microsoft Office 2000
- Browser Firefox 1.0.4, Opera 7.51 und MS-Internet Explorer 6.0

Die Datenbasis - SQLBase

Grundlage für ein Anschreiben sind immer Adressen, und Adressen korrekt zu verwalten ist gar nicht so einfach, wenn es um Titel, Anrede und Formatierungen geht. Die DIN 5008 beschreibt die formalen Details dazu. Beginnen wir mit SQLBase und schauen uns die Tabellenstruktur `SQLBASE_ADRESSE.DDL` an:

```
Listing 01: (SQLBASE_ADRESSE.DDL)

CREATE TABLE adresse (
 id          INTEGER NOT NULL,
 anrede      VARCHAR(40),
 titel       VARCHAR(40),
 name        VARCHAR(40) NOT NULL,
 vorname     VARCHAR(40),
 strasse     VARCHAR(55),
 postfach    VARCHAR(16),
 plz         VARCHAR(16),
 ort         VARCHAR(40),
 pplz        VARCHAR(5),

 PRIMARY KEY (id)
);
```

Die Tabelle ist nicht sonderlich aufregend, zeigt aber bei genauerem Hinsehen bereits das typische Problem. Nicht jede Spalte muss zwingend einen Inhalt haben und gewisse Informationen haben unterschiedliche Prioritäten. Denn das Postfach ist wichtiger als die Hausadresse, jedenfalls was den reinen Schriftverkehr angeht, bei Lieferscheinen geht die Strasse vor, denn zumeist ist das Postfach ungeeignet für die Heizöllieferung. Ein besonderes Feature ist dann auch die Tatsache, dass die Postleitzahl (PLZ) zur Strasse eine andere sein kann, als zum Postfach (PPLZ). Eventuell ist sogar der Ort des Postfaches ein anderer, als der Ort der Hausadresse. Die Komplexität des Problems schreit gerade zu danach, zentral - ein für alle mal - gelöst zu werden.

Die Testdaten für SQLBase liegen in der Datei `SQLBase_INSERT.SQL`. Falls Sie beim Einfügen der Testdaten in SQLBase den Fehler:

```
Error: 01417 DLU TIC Table ADRESSE in incomplete state
```

zu sehen bekommen, dann haben Sie vergessen, zu dem PRIMARY KEY den passenden UNIQUE INDEX zu erzeugen. Ganz nebenbei: Bei DB2 oder ORACLE wird beim Anlegen einer Tabelle mit »PRIMARY KEY« automatisch ein passender INDEX erstellt.

SQLBase Adressen aufbereiten mit PHP

Bevor das Adressenfenster einer Vorlage gefüllt werden kann, ist es sinnvoll, einen kompletten Anschriftblock zu erzeugen. Häufig sieht man, wie dies mit dem Word-Serienbrief-Manager über aufwendige Regeln versucht wird. Bei komplexen Adressregeln werden diese Versuche allerdings scheitern. Dann kommen Sie nicht um eine Funktion oder einen Programmteil herum, die diese Aufgabe erledigen. Generell sinnvoll ist es, diese spezielle Funktion gleich als globale Businessregel in Form eines Stored Procedure in der Datenbank abzulegen. Aber nicht alle Datenbanken unterstützen Stored Procedure, deshalb macht es auch Sinn, eine zentrale PHP-Klasse für die Adressenregeln bereit zu stellen. Eine mögliche Lösung kann wie folgt aussehen (Bild 2).

Bild 2: Zeigt die formatierte SQLBase Adresse (php_AdrBlock_SQLBase.php)

Die Beispieladressen sind in der Tabelle ADRESSE hinterlegt. Wenn beim INSERT der Testdaten alles geklappt hat, sollte der komplette Inhalt wie folgt aussehen (Bild 3). Für dieses Bild habe ich den kompletten Tabelleninhalt mit Hilfe der ODBC-Treiber in eine MS-Access Datenbank eingelesen.

ID	ANREDE	TITEL	NAME	VORNAME	STRASSE	POSTFACH	PLZ	ORT	PPLZ
1	Herr	Dr.	Müller	Ewald		11 12 13	70000	Stuttgart	70001
2			Maier	Hilde	Sonnengasse 5		70000	Stuttgart	
3	Frau	Prof. Dr.	Schmidt	Luise	Gassenhauer 15A	19 20 91	70000	Stuttgart	70001
4	Herr	Dipl. Informatiker	Schalter	Gerd	Zum Bluescreen 4.0		80000	München	
5			Maier		Fluchtweg 7/B		80000	München	
6	Frau		Maier			557	10000	Berlin	10001
7	Firma		Versandhaus XXL					Berlin	10001

Bild 3: Inhalt der SQLBase Tabelle ADRESSE mit MS-Access dargestellt.

Bild 3 zeigt die einzelnen Spalten und den Inhalt der Tabelle ADRESSE. Betrachten Sie bitte die Adresse mit der ID **3**. Obwohl eine Strasse »Gassenhauer 15A« erfaßt ist, wird entsprechend der hinterlegten Geschäftregeln, das Postfach vorgezogen. Dies zeigt sich auch an der eingesetzten Postleitzahl (PLZ). Es wird 70001 anstatt 70000 verwendet (Bild 2). Das Thema »Anschrift« ist offensichtlich gar nicht so einfach und ich garantiere Ihnen, das hier aufgebaute Szenario ist garantiert nicht überzogen, sondern die DIN 5008 bzw. die Praxis hat noch ganz andere Konstruktionen in petto. Damit möchte ich nochmal abschließend auf den Sinn einer zentralen Funktion zur Adressenformatierung verweisen. Nun aber werfen wir einen Blick in die PHP-Klasse, die für obige Formatierung zuständig ist.

Klassentreffen

Damit das PHP-Skript *php_AdrBlock_SQLBase.php* (Bild 2) die Adressen einlesen und formatieren kann, muss es zunächst die zuständige Klasse einbinden.

Listing 02: (**php_AdrBlock_SQLBase.php**)

```
[...]
09 /**
10  * Bibliotheken einbinden
11  */
12  require_once('php_AdrBlock_class.php');
13  $oGetAdr = new GetAdrBlock();
[...]
```

Mit *require_once* wird das Skript *php_AdrBlock_class.php* eingebunden. PHP sucht es dabei zuerst im aktuellen Verzeichnis und anschließend auch im eingestellten *include_path* (siehe PHP.INI) und muss zwingend vorhanden sein. Falls nicht, bricht das Skript hier mit einem *Warning* plus *Fatal Error* in Zeile **12** ab. Alternativ gibt es auch *include_once.* Findet *include* ein gesuchtes File nicht, gibt es keine Fehlermeldung, sondern erst dann, wenn das aufrufende PHP-Skript in Zeile **13** die Klasse instanziieren (oder ableiten) möchte. Ob Sie nun eher *include* oder *require* favorisieren, bleibt letztlich Ihnen überlassen, aber meiner Meinung nach ist *require* der stabilere Programmierstil, der Ihnen so manche Überraschung erspart.

Nun aber wieder zurück zu unserem Programm. In Zeile 13 wird erfolgreich eine Instanz der Klasse *GetAdrBlock* erzeugt und der Variablen *$oGetAdr* zugewiesen. Das vorangestellte »o« nutze ich, um Objektvariablen optisch hervorzuheben. Nachdem »Anzeigen« (Bild 2) ausgelöst worden ist, läuft im wesentlichen dieser Programmteil ab, um den ausgewählten Datensatz einzulesen und zu formatieren (Listing 03).

Listing 03: (**php_AdrBlock_SQLBase.php**)

```
[...]
/**
 * Welche Aktion wurde ausgelöst ?
 */
 if (isset($_POST['read'])) {
  $sAction = 'read';
[...]

[...]
/**
 * gewünschte Aktion ausführen
 */
 switch($sAction) {

  case 'read' :

   /**
   * Datensatz lesen
   */
   $oGetAdr->Init();
   if ($oGetAdr->ReadAdresse($hConnection,$nID)) {

    /* gefunden, jetzt die GUI-Variablen füllen */
    $sAnschrift = $oGetAdr->GetAdresse();
    $sBriefanrede = $oGetAdr->GetBriefanrede();
    $sMessage = $sMessage.' ok!';
    $oGetAdr->Commit($hConnection);
   } else {
[...]
```

Wird »Anzeigen« ausgelöst, erhält die Variable *$sAction* den Wert »read« zugewiesen.

Mittels *switch* und *case* wird der Programmteil »Datensatz lesen« ausgeführt. Mit Hilfe der Objektvariablen *$oGetAdr* erfolgt der Zugriff auf die einzelnen Objektmethoden.

Mit *if ($oGetAdr->ReadAdresse($hConnection,$nID)* wird die Adresse eingelesen. Der erste Parameter *$hConnection* beschreibt dabei die aktuelle Datenbankverbindung. Der zweite Parameter *$nID* enthält die eindeutige Adressen-ID, mit der in der Tabelle gelesen wird. Kann die Klassen-Methode *ReadAdresse()* korrekt ausgeführt werden, liefert diese als Returnwert *true*. Anschließend kann per *$sAnschrift = $oGetAdr->GetAdresse()* die fertig formatierte Anschrift und mit *$sBriefanrede = $oGetAdr->GetBriefanrede()* die aufbereitete Briefanrede abgeholt werden.

Klassenpower

Nun wird es Zeit, hinter die Kulissen zu schauen. Was passiert in der Methode *ReadAdresse()* wirklich. Wie der Name vermuten läßt, es wird die Adresse aus der Datenbank gelesen. Dazu werden zwei Informationen (Parameter) benötigt: Eine gültige Datenbankverbindung und die Adressen ID, die gelesen werden soll.

Listing 04: (**php_AdrBlock_class.php**)

```
[...]
  function ReadAdresse ($p_hConnection, $p_nId) {

   /**
   * numerischen Inhalt prüfen und Datentyp des Parameters umwandeln zu String
   */
   $nId = (INT) $p_nId;
   $sId = strval($nId);

   // Variablen init
   $this->Init();

   // Abfrage ausführen
   $sQuery = '';
   $sQuery = $sQuery .'SELECT titel, anrede, name, vorname, strasse,
                            plz, ort, postfach, pplz ';
   $sQuery = $sQuery .'  FROM adresse ';
   $sQuery = $sQuery .' WHERE id = '.$sId;

   $hCursor = odbc_exec($p_hConnection,$sQuery);
   if ($hCursor) {

    // Select ok, jetzt den 1. Satz der Ergebnismenge einlesen
    if(true === odbc_fetch_row($hCursor)) {
     $this->sTitel = odbc_result($hCursor,'titel');
     $this->sTitel = odbc_result($hCursor,'titel');
     $this->sAnrede = odbc_result($hCursor,'anrede');
     $this->sName = odbc_result($hCursor,'name');
     $this->sVorname = odbc_result($hCursor,'vorname');
     $this->sStrasse = odbc_result($hCursor,'strasse');
     $this->sPLZ = odbc_result($hCursor,'plz');
     $this->sOrt = odbc_result($hCursor,'ort');
     $this->sPostfach = odbc_result($hCursor,'postfach');
     $this->sPPLZ = odbc_result($hCursor,'pplz');
[...]
```

Das Listing 04 zeigt die Parametervalidierung – damit uns kein »faules Ei« in Form von SQL-Injection untergeschoben werden kann – anschließend wird die SQL-Abfrage ($sQuery) zusammengebaut und ausgeführt. Geht das alles glatt, liefert *odbc_exe*c als Rückgabewert den Cursor *($hCursor*), mit dessen Hilfe der erste Satz der Ergebnismenge eingelesen werden kann. Findet *odbc_fetch_row* einen Datensatz, ist der Returnwert *true*. Der Vergleich *(true === odbc_fetch_row)* mag Sie vielleicht verblüffen, aber nur mit den drei Gleichheitszeichen ist in PHP absolut sichergestellt, dass links und rechts vom Gleichheitszeichen der selbe Wert und der selbe Datentyp steht. Nachdem ein neuer Datensatz gelesen werden konnte, lassen sich mit *$this->sTitel =*

odbc_result($hCursor,'titel') die einzelnen Spaltenwerte (z.B. titel) auslesen und eigenen (Klassenvariablen) zuweisen. Damit sind die Werte aus der Datenbank eingelesen und *ReadAdresse()* korrekt abgeschlossen. Im nächsten Schritt werden die einzelnen Teile der Adresse nach unseren Businessregeln zusammen gesetzt.

```
Listing 05: (php_AdrBlock_class.php)
[...]
  /**
  * Erzeugt die komplette Adresse bzw. den Anschriftblock
  * @return Adresse String mit Zeilenumbruch CR/LF
  */
  function GetAdresse () {

   $sAnrede = $this->GetAnrede();

   // keine Postfach, aber Strasse
   if ((empty($this->sPostfach)) and (!empty($this->sStrasse))) {
    $this->sAdresse = '';
    $this->sAdresse = $this->sAdresse.$sAnrede.$this->CRLF;
    $this->sAdresse = $this->sAdresse.$this->sStrasse.$this->CRLF.$this->CRLF;
    $this->sAdresse = $this->sAdresse.$this->sPLZ.' '.$this->sOrt;
   }
[...]
```

Listing 05 zeigt beispielhaft, wie die Methode bzw. Funktion *GetAdresse()* die einzelnen Klassenvariablen analysiert und in eine Gesamtvariable $this->sAdresse zusammen setzt. Aktuell sehen Sie, wie geprüft wird, ob das Postfach oder die Strasse verwendet wird. Aber ich möchte Sie nicht mit endlosen Kontrollstrukturen langweilen. Der komplette Source liegt ja bei. Denn nun kommen wir zum zentralen Thema, erstellen eines Word-Serienbriefes.

COMmunizieren mit Word

Nachdem eine Anschrift eingelesen und aufbereitet ist, geht es nun darum, ein Word-Dokument mit PHP zu öffnen und auszufüllen. Zum Öffnen wird die Microsoft Technologie COM eingesetzt. PHP in der Version 5 ist dafür bestens gerüstet. Ziel ist es nun, die Textmarken des Word-Dokuments automatisch auszufüllen.

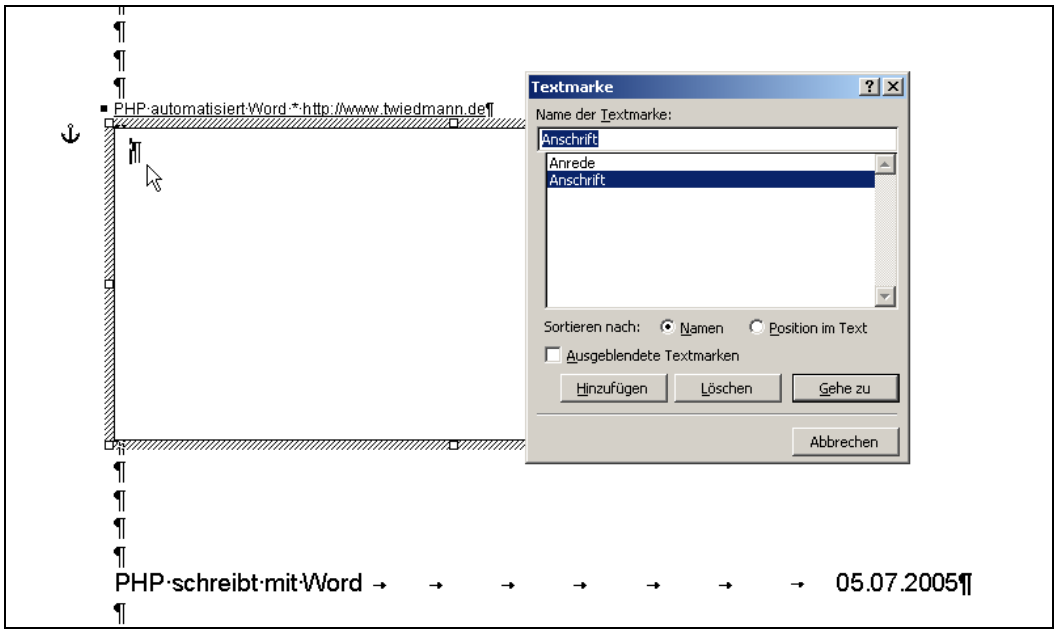

Bild 4: Zeigt die Word-Vorlage mit zwei Textmarken (WordVorlage.dot)

In Bild 4 sehen Sie einen Ausschnitt meiner vorbereiteten Word-Vorlage mit zwei Textmarken (Anrede und Anschrift). Für die Anschrift empfiehlt es sich, die Textmarke in ein eigenes Textfeld zu plazieren. Das Textfeld sorgt dafür, dass bei unterschiedlich langen Briefanschriften, der nachfolgende Text nicht verschoben wird, sondern immer fest verankert ist. Nun geht es darum, die beiden Textmarken (Bookmarks) via PHP auszufüllen und anschließend das Dokument zu speichern und an den Client zu schicken.

Word – Objektmodell

Voraussetzung für das folgende sind Grundkenntnisse zum Word-Objektmodell. Word wird in der Regel mit VBA (Visual Basic for Applications) programmiert. VBA ist damit auch die beste Informationsquelle für das nun folgende Vorhaben, sich mit PHP durch das Word-Objektmodell zu hangeln (Bild 5).

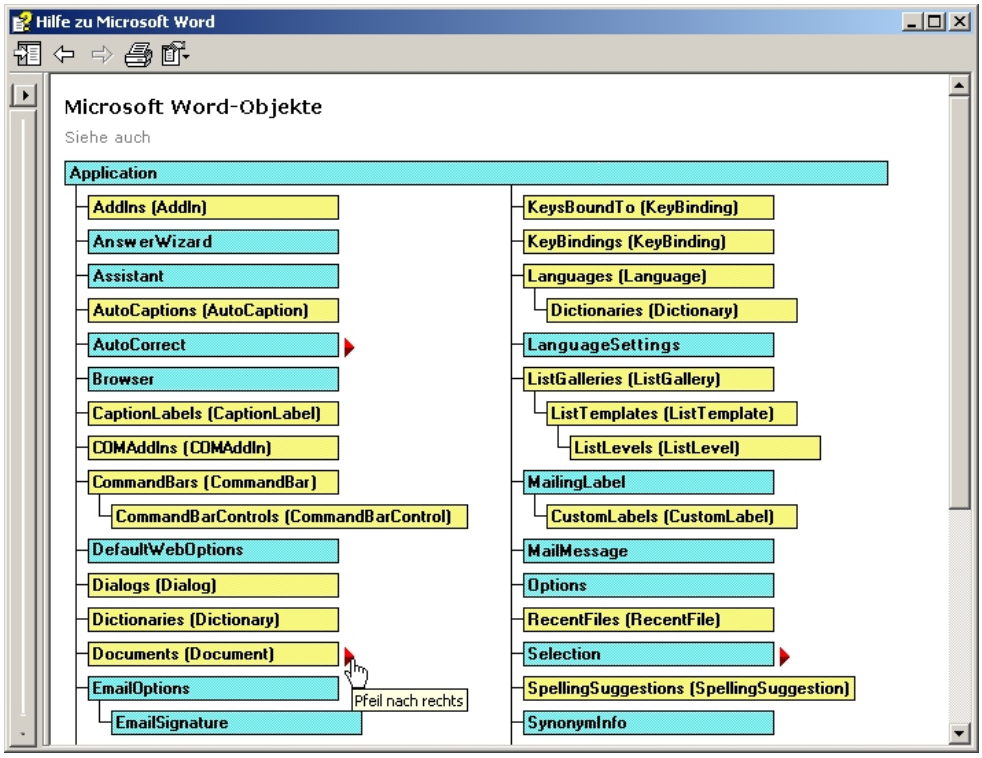

Bild 5: Zeigt einen Ausschnitt des Word-Objektmodelles von Office-2000 (Vbawrd9.chm)

Ist MS-Word installiert, sollten Sie - versionsabhängig - eine der folgenden Hilfedateien finden. Eine Menge zusätzlicher Informationen zu VBA-Programmierung finden Sie hier [1]. Weitere Details natürlich ebenso auch bei Microsoft [2].

Tabelle 1: Hilfedateien für jede Word-Version

Version 97 (oder 8.0)	Version 2000 (oder 9.0)	Version 2002 (oder 10)	Version 2003 (oder 11)
Vbawrd8.hlp	Vbawrd9.chm	Vbawd10.chm	VBAWD10.chm

Auf den ersten Blick wird klar: Das Objektmodell von Word ist gigantisch und komplex. Um alle Möglichkeiten zu verstehen und zu nutzen, gehen sicher Jahre ins Land. Für unser kleines Vorhaben benötigen wir zum Glück nicht alle Objekte, sondern müssen nur den richtigen Pfad durch diesen Objektdschungel finden. Der Startpunkt der Ralley beginnt mit dem Objekt »Application« (siehe Bild 5 links oben). Über *$this->oWord = new COM("word.application")* verbinden wir PHP mit dem Word-Objekt. Wenn dies funktioniert, ist

die erste Hürde übersprungen und wir haben Zugriff auf eine neue Welt, das Word-Objektmodell.

PHP als Objekt-Pfadfinder

Für die Reise durch das Objektmodell habe ich eine eigene Klasse zusammengestellt, um damit wiederkehrende Abläufe zu vereinfachen. Beginnen wir mit *Create().*

Listing 06: (**php_MSWord_class.php**)

```
<?php

class MSWord {
 [...]
  var $oWord = null; // das Wordobjekt
[...]
  function Create () {

   // versuche Word zu laden
   try {
     $this->oWord = new COM("word.application");
     if ($this->oWord === false) {
      // Word konnte nicht instanziiert werden
      return false;
     } else {

      // Word soll unsichbar bleiben
      $this->oWord->Visible = false;
      return true;
     }
   } catch (com_exception $e) {

     echo "COM-Fehler aufgetreten in " . $e->getFile() . "(" .$e->getLine(). ")\n";
     echo $e->getMessage() . "\n";

     $this->oWord = null;
     unset($this->oWord);
     return false;
   }
  }
[...]
```

Die Variable *$this->oWord* ist nun ein COM Objekt und verweist auf Word. Lassen Sie sich nicht von diesem *$this->* verwirren*,* es wird benötigt, da *$oWord* bereits selbst eine Klassenvariable der Klasse *MSWord* ist (siehe Listing 06 oben). Als erste Amtshandlung sorgen wir dafür, dass Word unsichtbar bleibt, indem der Eigenschaft (Property) *Visible* der Wert *false* zugewiesen wird: *$this->oWord->Visible = false*. Warum unsichtbar? Nun, PHP ist Server-Side-Scripting und damit wird Word dort gestartet, wo PHP auch läuft, vermutlich auf dem Server. Da wird wohl kaum jemand dabei zuschauen. Sollte beim Erzeugen des COM Objektes etwas schief gehen, sorgt der *try-catch* Block für ein kontrolliertes Programmende. Auf dem Server läuft nun ein Prozess mit dem Namen WINWORD.EXE (siehe Windows Taskmanager / Prozesse). Gibt es Probleme bei der Automatisation, bleibt dieser Prozess (Word) eventuell im Speicher »hängen« und beim nächsten Skriptaufruf startet PHP einen weiteren Word-Prozess. Dies führt dann unweigerlich zu Problemen. Lesen Sie hierzu unbedingt die Meinung von Micosoft [3] und meine Überlegungen dazu am Schluß dieses Artikels.

Listing 07: (**php_AdrBlock_SQLBase_Word.php**)

```
[...]
 require_once('php_MSWord_class.php');
 $oMSWord = new MSWord();
[...]
   /**
   * Template und Dokument mit vollständiger Pfadangabe
   */
   $sFilename = 'WordDokument.doc';
   $sTemplate = $_SERVER['DOCUMENT_ROOT'] . '\\WordVorlage.dot';
```

```
   $sDokument = $_SERVER['DOCUMENT_ROOT'] . '\\' . $sFilename ;
[...]

[...]
   /**
   * Word starten und Vorlage ausfüllen
   */
   if ($oMSWord->Create()) {

     /**
     * Array mit den notwendigen Bookmarks (Textmarken) und Inhalten erzeugen
     */
     $aBookmarks = array('Briefanrede' => $sBriefanrede,
                         'Anschrift' => $sAnschrift);

     /**
     * Auf Basis der Vorlage wird ein neues Dokument erzeugt
     * und die Textmarken mit den Inhalten (Adresse + Anschrift)
     * ausgefüllt.
     */
     $oMSWord->Build($sTemplate,$sDokument,$aBookmarks);
     $oMSWord->Destroy();
[...]
```

Listing 07 zeigt den weiteren Ablauf aus einer »höheren« Warte aus an. Nachdem *Create()* erfolgreich war, geht es darum die Bookmarks zu benennen und Inhalte bereit zu stellen. In der verwendeten Vorlage sind bereits zwei Textmarken vorbereitet (Bild 4). Die Array-Variable *$aBookmark*s enthält die Werte für die beiden Textmarken »Briefanrede« und »Anschrift«. Mit *$oMSWord->Build($sTemplate,$sDokument,$aBookmarks)* wird die zentrale Methode ausgeführt. Dabei werden drei Parameter an *Build()* übergeben:

- $sTemplate – Die Wordvorlage (WordVorlage.dot) mit komplettem Pfad.
- $sDokument – Das neue Worddokument mit komplettem Pfad.
- $aBookmarks – Ein String-Array mit den Textmarken und Inhalten

Der zentrale Programmteil zum Suchen und Ausfüllen der Textmarken ist in der Methode *Build()* hinterlegt.

Listing 08: (**php_MSWord_class.php**)

```
[...]
function Build($p_sTemplate = null, $p_sDokument = null, $p_aBookmarks = array()) {

 // VARIANTs erzeugen
 $this->vTemplate = new VARIANT($p_sTemplate, VT_BSTR);
 $this->vDokument = new VARIANT($p_sDokument, VT_BSTR);

 // 'optische' Meldungen von Word abschalten
 $this->oWord->DisplayAlerts = false;
 $this->oWord->ScreenUpdating = false;

 // Dokuments-Objekt (Achtung: Mehrzahl)
 $oWordDocs = $this->oWord->Documents();
[...]
```

Zuerst werden die Werte Parameter *$p_sTemplate* und *$p_sDokument* einem VARIANT Container mit dem Datentyp VT_BSTR zugewiesen. VARIANT ist ein spezielles »typenloses« Objekt, das sowohl Zeichen- als auch numerische Datentypen annehmen kann. Innerhalb des Word-Objektmodelles wird es häufig eingesetzt. Weiterhin ist es sinnvoll Word mitzuteilen, dass »optische« Reaktionen (DisplayAlerts und ScreenUpdating) sinnvollerweise zu unterlassen sind. Das bereits angedeutet »hangeln« durch das Objektmodell von Word beginnt bei :

```
$oWordDocs = $this->oWord->Documents()
```

Das Objekt *$oWordDocs* wird von $this->oWord (Application-Ebene) abgeleitet (Bild 5). Der nächste Befehl *$oWordDocs->Add()* erzeugt das neue Dokument.

```
$oWordDoc = $oWordDocs->Add($this->vTemplate)
```

Auf Basis des Template erzeugt Word ein neues Dokument (Achtung: Einzahl!). Das Objekt *$oWordDoc* hat darauf Zugriff. Nun geht es darum, die vorhandenen Bookmarks in dem neuen Dokument zu finden. Dazu gibt es wieder ein neues Objekt *$oBookmarks*, welches alle vorhandenen Textmarken kennt.

```
$oBookmarks = $oWordDoc->Bookmarks()
```

Sind wir soweit, geht es darum, herauszufinden, welchen Namen diese Textmarken haben. Dies ist sinnvoll, um flexibel auf verschiedene Vorlagen reagieren zu können, ohne den Code jedesmal überarbeiten zu müssen.

```
$nAnzBookmarks = $oBookmarks->Count()
```

auf Basis von *$nAnzBookmarks* ist eine Schleife möglich, um alle vorhandenen Textmarken zu finden.

```
for ($i=1;$i<=$nAnzBookmarks;$i++) {

 $this->vIndex = $i;
 $oBookmark = $oBookmarks->Item($this->vIndex);
 $sName = $oBookmark->Name();
```

Über *$oBooksmarks->Item()* erhalten wir Zugriff auf eine einzelne Textmarke *$oBookmark* und können damit endlich den Namen der Textmarke ermitteln. Existiert die gefundene Textmarke in dem eingangs übergebenen Bookmark-Array *$p_aBookmarks*, kann der Zugriff über ein Rangeobjekt erfolgen.

```
// soll diese Textmarke gefüllt werden?
if (array_key_exists($sName,$p_aBookmarks)) {

  // Rangeobjekt zur Textmarke
  $oWordRange = $oBookmark->Range();
```

Im letzten Schritt springen wir zum Rangeobjekt (Goto) und setzen den Wert an die Position der Textmarke.

```
     $this->vName = $sName;
     $oWordRange = $oWordRange->Goto($this->vWhat,
                                     $this->vOptional,
                                     $this->vOptional,
                                     $this->vName);
     $oWordRange->Text = $p_aBookmarks[$sName];
```

Der Name der gesuchten Textmarke wird dafür einem VARIANT zugewiesen. Der Methode *Goto()* wird anhand von vier Parametern mitgeteilt, was zu tun ist.

```
[...]
$this->vWhat        VARIANT (-1,VT_I2);  // What (Bookmarktyp: GoToBookmark = -1)
$this->vOptional    VARIANT (); // leerer Variant
$this->vName        VARIANT('Name der Textmarke', VT_BSTR);
[...]
```

Der gefundenen Textmarke wird der Inhalt des Bookmark-Array *$p_aBookmarks* zugewiesen. Sehen Sie, kaum sind zwei Artikelseiten voll, sind wir (beinahe fertig). Sicherlich wundern Sie sich jetzt über meine aufwendige Vorgehensweise. In den Weiten des Web finden Sie häufig Beispiele wie diese:

```
$word->Documents->Open('C:/Unfilled.DOC');
$word->Selection->GoTo(wdGoToBookmark,$empty,$empty,'YourName');
```

Hierbei wird auf die einzelnen Objekte(stufen) mit dem Pfeil -> *Operator* verwiesen. Auch das ist natürlich korrekt. Aber wie immer gibt es mehrere Wege zum Ruhm und beim bereits angesprochenen »Hangeln« durch das Objektmodell von Word, ist mir persönlich die vorgestellte Variante die übersichtlichere. Bleibt noch zum guten Schluß das ausgefüllte Dokument für die weitere Verwendung (auf dem Server) zu speichern. Bei Multiuseranwendungen, muß hier natürlich ein eindeutiger Dokumentname gefunden werden, ansonsten überschreibt Word bereits vorhandene Dokumente.

```
$oWordDoc->SaveAs($this->vDokument);
```

Word Prozess beenden

Wie bereits in Listing 07 zu sehen ist, nach *Build()* wird die Methode *Destroy()* aufgerufen. Dabei geht es darum, den Word-Prozess (hoffentlich!) wieder zu beenden und den Speicher wieder freizugeben. In der Testphase lohnt es sich hier häufig mit dem Taskmanager zu prüfen, ob noch WINWORD.EXE Prozesse aktiv sind.

```
Listing 08: (php_MSWord_class.php)
[...]
  function Destroy () {
   $this->oWord->quit($this->vOptional, $this->vOptional, $this->vOptional);
   $this->oWord = null;
   unset($this->oWord);

   return true;
  }
[...]
```

Die *Quit()* – Methode veranlasst Word alle Dokumente zu schließen. Die drei Parameter *vOptional* mit dem Wert *false* stellen sicher, dass es dabei keine weiteren Fragen wie »Wollen Sie wirklich..« stellt. Mit *$this->oWord = null* wird das Objekt wieder freigegeben. Ist der (interne) Referenzzähler des Word-Prozesses jetzt auf Null, schmeißt Windows ihn aus dem Speicher.

PHP als Postbote

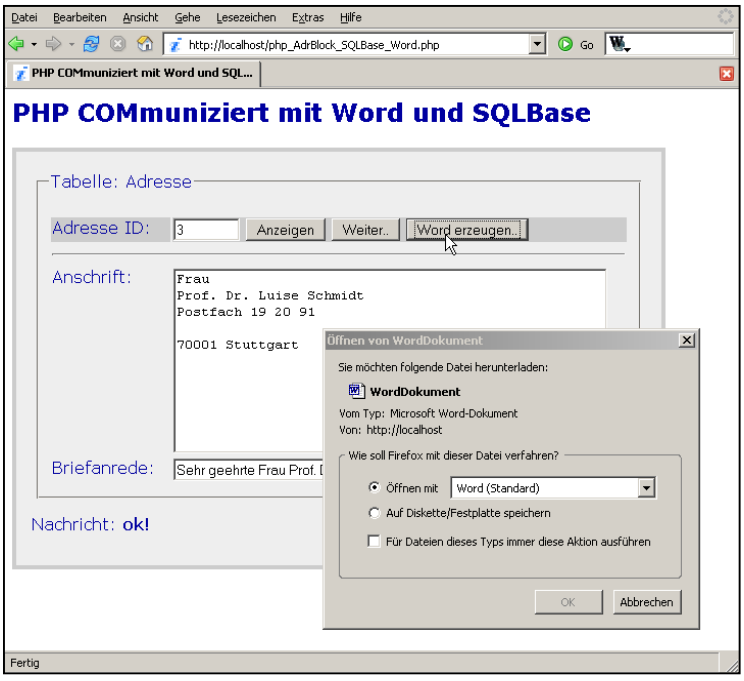

Bild 6:Das ausgefüllte Word-Dokument, wird dem Client zugeschickt (php_AdrBlock_SQLBase_Word.php)

Nachdem das neue Word-Dokument auf dem Server abgespeichert ist, wäre es doch interessant zu wissen, wie das Ergebnis denn nun aussieht. Dazu muß das Dokument aber irgendwie den Weg zum Client finden. Dafür gibt es natürlich mehrere Überlegungen:

1. Sie gehen zum Server oder haben Zugriff auf das Serverlaufwerk?
2. Der Administrator schickt Ihnen das Dokument per E-Mail?
3. PHP schickt Ihnen automatisch das Dokument!

Während die Varianten eins und zwei kaum praktikabel sind, hört sich die dritte Möglichkeit doch sehr vernünftig an. Wie in Bild 6 zu sehen ist, nach dem Auslösen von [Word erzeugen] erscheint automatisch der Dialog »Öffnen von WordDokument« im Browser. Was ist da passiert? PHP hat Ihnen automatisch das ausgefüllte Dokument zugeschickt, es steht nun genau wie Download zur Verfügung. Mit [Ok] wird das Dokument lokal geöffnet. Entweder mit MS-Word oder – falls vorhanden – auch mit OpenOffice.

```
Listing 09: (php_AdrBlock_SQLBase_Word.php)
[...]
 /**
 * Das erzeugte Word-Dokument liegt nun auf dem Server.
 * Das Dokument nun an den Client-Senden mit dem Kurznamen
 * ohne Pfadangabe ($sFilename)
 */
 if ($oMSWord->DokumentSenden($sDokument,$sFilename)) {
  $oMSWord->DokumentDelete($sDokument);
  die();
 }
[...]
```

Listing 09 zeigt den Aufruf der Klassenmethoden *DokumentSenden()* und anschließend *DokumentDelete()*. Nach dem Versenden kann das Dokument auf dem Server gelöscht werden. Nun aber wieder einen Blick hinter die Kulissen und in die Klasse *MSWord*.

```
Listing 10: (php_MSWord_class.php)
[...]
  function DokumentSenden ($p_sDokument = null,$p_sFilename = null) {

   if (is_file($p_sDokument)) {

    // speziell für Internet-Explorer
    header('Pragma: public');
    header('Expires: 0');
    header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
    header('Cache-Control: public');

    // Standard
    header('Content-type: application/msword', true);
    header('Content-Disposition: attachment; filename="'.$p_sFilename.'"' );
    header('Content-Transfer-Encoding: binary');
    header('Content-Length: '.filesize($p_sDokument));
    readfile($p_sDokument);
    flush();
[...]
```

Mit dem Kommando *header()* können Dateien, Bilder oder auch Word-Dokument vom Server zum Client-Browser gesendet werden. Listing 10 zeigt die notwendigen Befehle dazu. Ganz wichtig ist der *'Content-type: application/msword'*. Damit wird definiert, welche Art von Datei verschickt wird. Mit *'Content-Disposition: attachment;* steuern Sie, auf welche Art und Weise die Datei transportiert wird. Als *attachment* oder auch *inline*. Mit *inline* startet Word innerhalb des Browser als ActiveX Addin. Dies unterstützt allerdings nur Microsoft IE. Typischerweise wird deshalb *attachment* eingesetzt, daraufhin öffnet sich das Downloadfenster im Browser und der Anwender kann entscheiden, was damit nun passieren soll. Es gibt übrigens eine ganze Menge weiterer Header Kommandos, mit denen bei Bedarf experimentiert werden kann [4].

Wichtig ist noch der Hinweis, dass vor *header()* keinerlei Ausgaben mit zum Beispiel *echo, print* etc.) erfolgen darf, ansonsten sehen Sie auf dem Client berühmte Fehler wie (Bild 7):

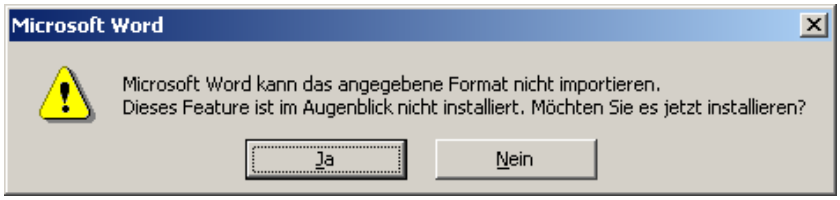

Bild 7: Word Fehlermeldung bei einer echo Ausgabe vor dem header().

Haben wir aber alles richtig gemacht, sollte Word sich öffnen und die übertragene Anschrift und Briefanrede ist an den jeweiligen Textmarken eingefügt. Damit ist das eingangs angesprochene Problem gelöst und dieser Artikel am Ende angelangt. Bleibt als Resumée noch die Tatsache, dass PHP und COM eine hilfreiche Rolle im Büroalltag spielen können. Von Seiten PHP spricht nichts gegen den Einsatz, von Seiten Microsoft leider ein paar Aussagen zur serverseitigen Automatisierung von Office. Bitte beachten sie deshalb noch den nachfolgenden Abschnitt, aber zuvor geniessen wir noch das Bild 8.

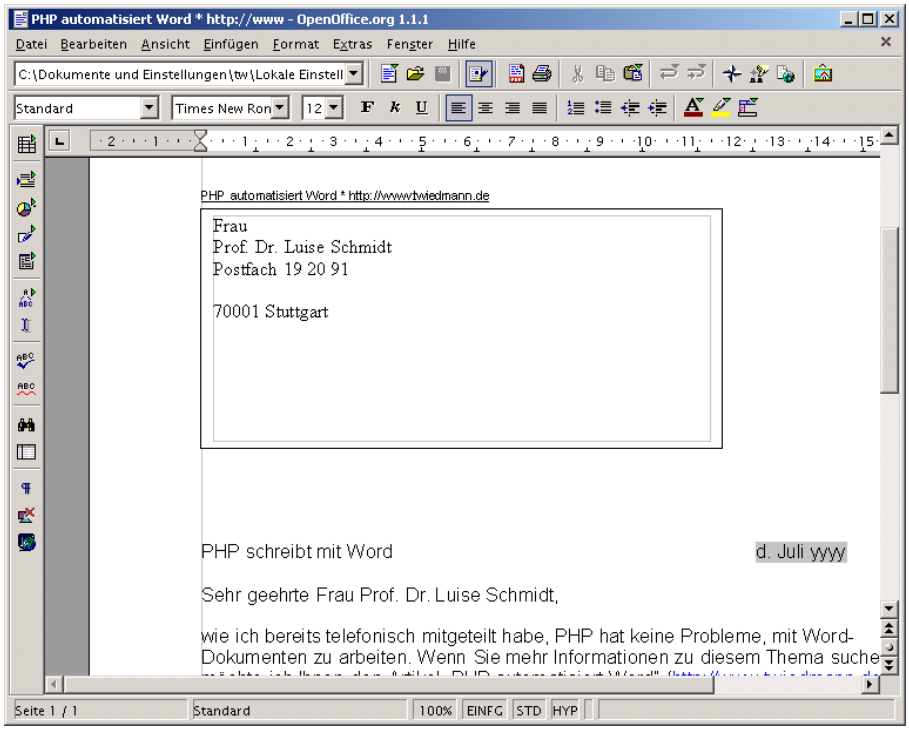

Bild 8: Auch OpenOffice kann mit dem gesendeten Ergebnis leben.

Bei meinen verschiedenen Versuchen hat sich der Firefox oder Opera deutlich hilfsbereiter als der Internet Explorer gezeigt. Der IE in den jetzigen Versionen bereitet vielen Leuten eine Menge Kopfschmerzen bei diesem Thema [5].

COM, Sicherheit und Zugriffsrechte

Falls es beim Testen der Beispiele Probleme gibt, kann es auch an den Zugriffsrechten der einzelnen Komponenten liegen. Erfolgt eine HTTP-Anfrage vom Client beim Server, arbeitet Apache diese ab. Um auf Word zugreifen zu können, muß das eingestellte Systemkonto des Dienstes »Apache« dies auch erlauben. Prüfen Sie deshalb bitte auch unter *Verwaltung / Dienste* / *Eigenschaften* / »Anmelden als « zugewiesene Konto. Im Zweifelsfall liefert auch das Programm DCOMCNFG.EXE »Eigenschaften der DCOM Konfiguration« noch Hinweise zu den Einstellungen für *Standort*, *Sicherheit* und *Identität*.

Überlegungen zur serverseitigen Automatisierung von Office

PHP kann zusammen mit Word Schriftstücke verschiedenster Art erzeugen. Dies ist nicht das eigentliche Problem. Bei der Konzeption eines solchen »Dokumentenserver« müssen Sie beachten, dass Office – laut Aussage von Microsoft [3] – nicht für den serverseitigen Betrieb konzipiert ist. Auf unvorhergesehene Fehler jeder Art, reagiert Word mit modalen Dialogfenstern und wartet auf eine Benutzerreaktion, die es auf dem Server kaum geben kann. Eventuell möchte Word auch mal unbedingt ein fehlendes Modul nachinstallieren. Ein regelmässiger Reboot des »Dokumentenservers« ist folglich notwendig. Trotzdem ist die Idee einer zentralen »Dokumentenfabrik« faszinierend. Einen PDF-Generator einzusetzen anstatt Word ist vermutlich stabiler. Andererseits ist der Markt von MS-Word durchdrungen, ohne Word befindet man sich quasi auf einer Insel. Im kontrollierbaren Rahmen eines Intranet, ist das Ganze auf jeden Fall ein bis zwei Überlegungen wert.

Bonusartikel zur ODBC Verbindung für PHP und SQLBase

Um nicht bereits vorhandene Informationen mehrfach vorzustellen, lege ich hier einen Teil aus meinen früheren Tutorial »PHP und Centura SQLBase« vom 30.05.2005 bei. Darin wird unter anderem das Thema: »Connectivity mit ODBC und SQLBase“ erläutert.

Links und weiterführende Literatur

[1] MS Word-VBA Zyklus http://mypage.bluewin.ch/reprobst/WordFAQ/VBA.htm

[2] Finden und Verwenden der Dokumentation zum Office-Objektmodell.htm
http://support.microsoft.com/?scid=kb;de;222101&spid=2488&sid=251

[3] Considerations for Server-Side Automation of Office.
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q257757

[4] PHP - Sendet einen HTTP-Header
http://de2.php.net/manual/de/function.header.php

[5] Zend - Finding a solution
http://www.zend.com/zend/trick/tricks-august-2001.php

Abkürzungen und Quellen

- Apache Apache Web-Server (http://www.apache.org)
- COM Component Object Model
- CSS Cascading Stylesheets (http://www.w3.org)
- DDL Data Definition Language
- GTD Gupta Team Developer (http://www.guptaworldwide.com)
- HTML Cascading Stylesheets (http://www.w3.org)
- HTTP HyperText Tranfer Protocol
- PHP Hypertext Preprocessor (http://www.php.net)
- VBA Visual Basic for Applications

Kennen Sie schon diese Tutorials?

30.05.2005	PHP und Centura SQLBase (PDF – 15 Seiten)
24.04.2005	Gupta Team Developer als COM-Server für PHP (PDF – 10 Seiten)
10.04.2005	Der Reportbuilder von Gupta im Einsatz. (PDF – 6 Seiten)
03.04.2005	DB2, PHP, JpGraph und SQL treibens bunt (PDF – 7 Seiten)

Stehen zum Download bereit auf: http://www.twiedmann.de/beispiele.php