Das Apache POI-Framework als Reporting-Tool für Oracle Anwendungen

Sven-Olaf Kelbert
MT AG
Ratingen

Schlüsselworte:

Apache, POI, HSSF, Java, Oracle Forms, Report, Excel.

Die Ausgangslage

RWE Power als integraler Teil des RWE-Konzerns ist eines der führenden Unternehmen der Energiegewinnung und -erzeugung in Deutschland. In der Organisationseinheit Strombezug im Bereich Vertragsmanagement beschäftigen sich die Mitarbeiter hauptsächlich mit der Abwicklung von Fremdstrombezugsrechnungen verschiedenen Kraftwerke. Als Unterstützung diente ihnen seit 2002 das System IAIS (Integriertes Abrechnungs- und Informationssystem).

IAIS basierte auf der Entwicklungsplattform Microsoft Visual Basic 6 (VB6), für die seitens Microsoft die Unterstützung eingestellt wurde. Damit war die Möglichkeit von zukünftigen Erweiterungen eingeschränkt. Ferner nutzte die Applikation Komponenten von Drittherstellern, die nicht für die neuen Oracle-Datenbanken zertifiziert sind. Diese Situation machte eine Modernisierung der Applikation unumgänglich.

Die RWE IT GmbH als IT-Dienstleister für die RWE Power analysierte die Rahmenbedingungen und leitete eine Neu-Entwicklung der Anwendung mit der Unterstützung der MT AG in die Wege. Als Entwicklungswerkzeug wurde Oracle Forms 10g ausgewählt, weil dies schon in vielen anderen Konzern-Anwendungen im Einsatz ist und somit Synergie-Effekte genutzt werden konnten.

Ein wesentlicher Bestandteil der Altanwendung war die komplexe Excel-Schnittstelle, die durch die technischen Möglichkeiten von VB6 sehr eng mit dem System verwoben war. Da diese enge Verknüpfung von IAIS und Excel unter Oracle Forms nicht mehr möglich war, musste die Schnittstelle nahezu komplett neu entwickelt werden. Hierfür wurde auf das Apache POI-Framework zurückgegriffen, das schon erfolgreich bei vielen Oracle Forms Projekten der RWE eingesetzt wird.

Der fachliche Hintergrund

Das System IAIS unterstützt seine Anwender bei der Abwicklung und Prognose von Strombezügen und -lieferungen verschiedener Kraftwerke. Hierbei geht es um die Pflege der Geschäftspartner und deren Verträge mit der RWE Power. Neben diesen überschaubaren Stammdaten erhält das System IAIS Strommengendaten verschiedener Stromzähler (sog. Zählpunkte) an den Kraftwerken. Diese Strommengen werden viertelstündlich ermittelt und über verschiedene Schnittstellen nach IAIS importiert. Diese Viertelstundenwerte sind Ausgangsbasis für die Berechnung von Monatsverbräuchen, Anzahl von Kraftwerksstillständen und Wiederanfahrten der Kraftwerke.

Außerdem werden über komplexe Berechnungskomponenten mit speziellen Formeln die Werte der verschiedenen Zählpunkte gegeneinander verrechnet, um mit Hilfe von verschiedenen Preiskomponenten und Tarifen eine Prüfung von Rechnungen bzw. Gutschriften der Geschäftspartner möglich zu machen. Neben Berechnungen von vergangenen Monaten sind dabei auch Prognosen in die Zukunft möglich. Die Ergebnisse der Berechnungskomponenten sowie komplette Zeitreihen von Zählpunkten werden dabei in Form von Auswertungen und Controlling-Berichten in Excel dargestellt und unterstützen damit das interne Berichtswesen.

Der technische Hintergrund

Die Viertelstundenwerte der Strommengen werden über verschiedene Schnittstellen nach IAIS importiert. Hierbei fallen in zwei Tabellen je Monat jeweils etwa 4 Mio. neue Datensätze an. Insgesamt enthalten die Tabellen mittlerweile jeweils ca. 350 Mio. Datensätze.

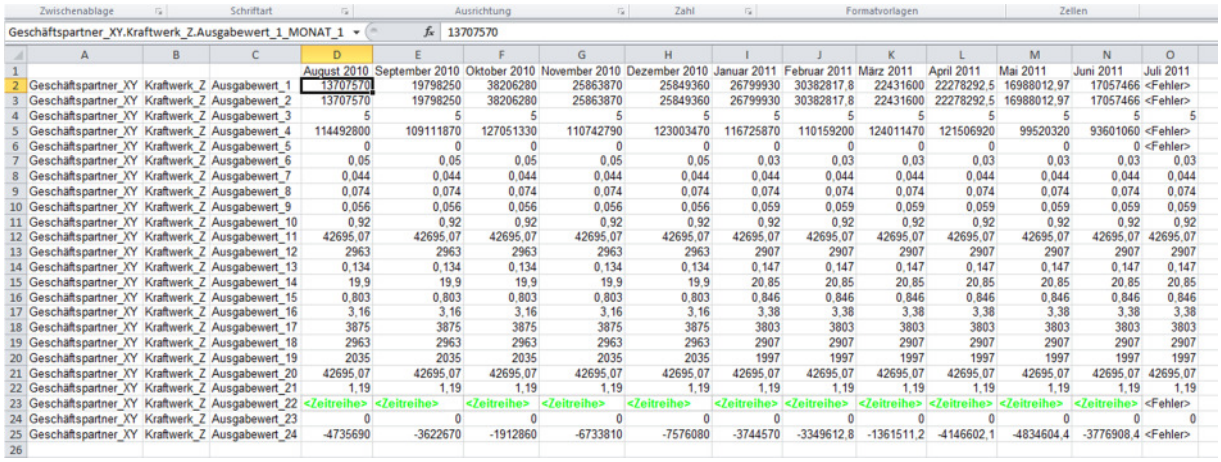
Am Anfang eines jeden Monats werden die Berechnungskomponenten genutzt, um aus den Werten der Zählpunkte des vorigen Monats sog. Ausgabewerte zu berechnen. Die Ergebnisse dieser Ausgabewerte werden dann nach Excel geschrieben, um damit die vorliegenden Rechnungen zu kontrollieren.

Für jeden Geschäftspartner und jeden Vertrag gibt es dabei diverse Excel-Vorlagen, die in der Datenbank gespeichert sind.

	A	B	C	D
1				MONAT_1
2	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_1	
3	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_2	
4	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_3	
5	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_4	
6	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_5	
7	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_6	
8	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_7	
9	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_8	
10	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_9	
11	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_10	
12	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_11	
13	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_12	
14	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_13	
15	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_14	
16	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_15	
17	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_16	
18	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_17	
19	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_18	
20	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_19	
21	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_20	
22	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_21	
23	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_22	
24	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_23	
25	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_24	
26				

Abb. 1: Beispiel für ein Tabellenblatt einer Vorlage einer Auswertung

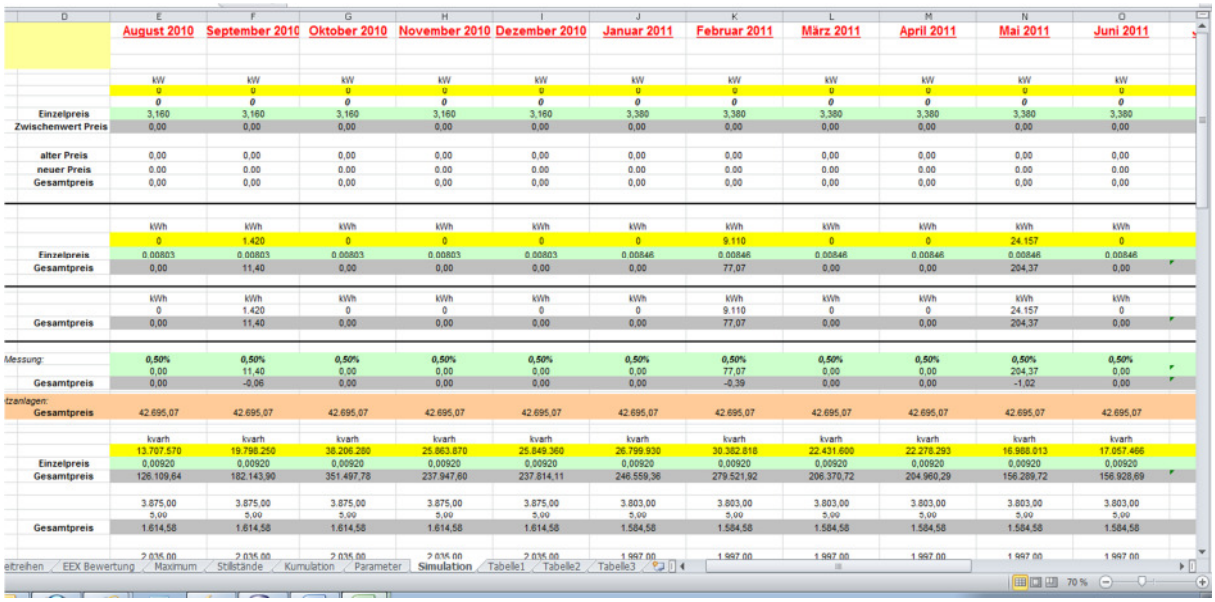
Diese Vorlagen enthalten verschiedene Tabellenblätter, auf denen z.B. die Ergebnisse der Ausgabewerte, eine Liste aller Viertelstundenwerte eines Zählpunktes, eine Liste aller Betriebszeiten und Ausfallzeiten eines Kraftwerks oder Listen der Strom- und CO2-Preise der Leipziger Strombörse zu sehen sind. Die Vorlagen werden dann im Rahmen der Rechnungsprüfung mit den benötigten Daten aus der Datenbank verknüpft, um die gewünschte Auswertung zu erzeugen.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1				August 2010	September 2010	Oktober 2010	November 2010	Dezember 2010	Januar 2011	Februar 2011	März 2011	April 2011	Mai 2011	Juni 2011	Juli 2011
2	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_1	13707570	19798250	38206280	25863870	25849360	26799930	30382817,8	22431600	22278292,5	16988012,97	17057466	<Fehler>
3	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_2	13707570	19798250	38206280	25863870	25849360	26799930	30382817,8	22431600	22278292,5	16988012,97	17057466	<Fehler>
4	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_3	5	5	5	5	5	5	5	5	5	5	5	5
5	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_4	114492800	109111870	127051330	110742790	123003470	116725870	110159200	124011470	121506920	99520320	93601060	<Fehler>
6	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_5	0	0	0	0	0	0	0	0	0	0	0	<Fehler>
7	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_6	0,05	0,05	0,05	0,05	0,05	0,03	0,03	0,03	0,03	0,03	0,03	0,03
8	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_7	0,044	0,044	0,044	0,044	0,044	0,044	0,044	0,044	0,044	0,044	0,044	0,044
9	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_8	0,074	0,074	0,074	0,074	0,074	0,074	0,074	0,074	0,074	0,074	0,074	0,074
10	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_9	0,056	0,056	0,056	0,056	0,056	0,059	0,059	0,059	0,059	0,059	0,059	0,059
11	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_10	0,92	0,92	0,92	0,92	0,92	0,92	0,92	0,92	0,92	0,92	0,92	0,92
12	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_11	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07
13	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_12	2963	2963	2963	2963	2963	2907	2907	2907	2907	2907	2907	2907
14	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_13	0,134	0,134	0,134	0,134	0,134	0,147	0,147	0,147	0,147	0,147	0,147	0,147
15	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_14	19,9	19,9	19,9	19,9	19,9	20,85	20,85	20,85	20,85	20,85	20,85	20,85
16	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_15	0,803	0,803	0,803	0,803	0,803	0,846	0,846	0,846	0,846	0,846	0,846	0,846
17	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_16	3,16	3,16	3,16	3,16	3,16	3,38	3,38	3,38	3,38	3,38	3,38	3,38
18	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_17	3875	3875	3875	3875	3875	3803	3803	3803	3803	3803	3803	3803
19	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_18	2963	2963	2963	2963	2963	2907	2907	2907	2907	2907	2907	2907
20	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_19	2035	2035	2035	2035	2035	1997	1997	1997	1997	1997	1997	1997
21	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_20	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07	42695,07
22	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_21	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19
23	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_22	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Zeitreihe>	<Fehler>
24	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_23	0	0	0	0	0	0	0	0	0	0	0	0
25	Geschäftspartner_XY	Kraftwerk_Z	Ausgabewert_24	-4735690	-3622670	-1912860	-6733810	-7576080	-3744570	-3349612,8	-1361511,2	-4146602,1	-4834604,4	-3776908,4	<Fehler>
26															

Abb. 2: Beispiel für ein Standard-Tabellenblatt einer Auswertung (Vorlage aus Abb. 1 gefüllt mit Daten)

In jeder Vorlage sind dabei 7 fest definierte Tabellenblätter enthalten, während es zusätzlich für jede Auswertung möglich ist, individuelle Tabellenblätter zu definieren, auf denen dann die einzelnen Werte noch grafisch oder tabellarisch fürs Controlling aufbereitet werden können.



	August 2010	September 2010	Oktober 2010	November 2010	Dezember 2010	Januar 2011	Februar 2011	März 2011	April 2011	Mai 2011	Juni 2011
Einzelpreis	3,160	3,160	3,160	3,160	3,160	3,380	3,380	3,380	3,380	3,380	3,380
Zwischenwert Preis	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
alter Preis	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
neuer Preis	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Gesamtpreis	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Einzelpreis	0,00803	0,00803	0,00803	0,00803	0,00803	0,00846	0,00846	0,00846	0,00846	0,00846	0,00846
Gesamtpreis	0,00	11,40	0,00	0,00	0,00	0,00	77,07	0,00	0,00	204,37	0,00
Gesamtpreis	0	1.420	0	0	0	0	9.110	0	0	24.157	0
Gesamtpreis	0,00	11,40	0,00	0,00	0,00	0,00	77,07	0,00	0,00	204,37	0,00
Messung:	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%	0,50%
Gesamtpreis	0,00	11,40	0,00	0,00	0,00	0,00	77,07	0,00	0,00	204,37	0,00
Gesamtpreis	0,00	-0,06	0,00	0,00	0,00	0,00	-0,39	0,00	0,00	-1,02	0,00
tzanlagen:											
Gesamtpreis	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07	42.695,07
Einzelpreis	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920	0,00920
Gesamtpreis	126.109,64	182.143,90	351.497,78	237.947,60	237.814,11	246.559,36	279.521,92	206.370,72	204.960,29	156.289,72	156.928,69
Gesamtpreis	3.875,00	3.875,00	3.875,00	3.875,00	3.875,00	3.803,00	3.803,00	3.803,00	3.803,00	3.803,00	3.803,00
	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00
Gesamtpreis	1.614,58	1.614,58	1.614,58	1.614,58	1.614,58	1.584,58	1.584,58	1.584,58	1.584,58	1.584,58	1.584,58
	2.035,00	2.035,00	2.035,00	2.035,00	2.035,00	1.997,00	1.997,00	1.997,00	1.997,00	1.997,00	1.997,00

Abb. 3: Beispiel für ein individuelles Tabellenblatt einer Auswertung (Vorlage aus Abb. 1 gefüllt mit Daten)

Es ist möglich, die Vorlagen selbst zu ändern und wieder in der Datenbank abzuspeichern. Bei den Auswertungen selbst kann man festlegen, für welchen Zeitraum diese gestartet werden sollen. Dies kann ein einzelner Tag oder ein einzelner Monat, aber auch eine Serie von Tagen oder Monaten sein, um z.B. ein komplettes Jahr in einem Schritt zu prüfen.

Das POI-Framework

Das Apache POI-Projekt besteht aus Java-APIs zum Erstellen und Bearbeiten von Microsoft-Office-Dateien, wie zum Beispiel Excel- und Word-Dateien. POI steht dabei für „Poor Obfuscation Implementation“ (schlechte, verschleiernde Implementierung), womit aber nicht das POI-Framework selbst gemeint ist, sondern das Microsoft-Office-Dateiformat. POI besteht aus mehreren

Komponenten, wobei sich HSSF mit Excel beschäftigt. Einige der wichtigsten Klassen im HSSF-Package sind die folgenden:

Klasse	Funktionalität
HSSFWorkbook	repräsentiert eine Excel-Datei
HSSFSheet	repräsentiert ein Tabellenblatt
HSSFRow	repräsentiert eine Zeile eines Tabellenblatts
HSSFCell	repräsentiert eine Zelle einer Zeile
HSSFCellStyle	repräsentiert das Format einer Zelle
HSSFName	repräsentiert den Namen eines Zellbereichs

In diesen Klassen kann über diverse Methoden auf die gewünschten Funktionalitäten zugegriffen werden. Hier folgt eine Auswahl der Methoden, die im Projekt IAIS genutzt wurden:

Methode aus HSSFWorkbook	Funktionalität
`HSSFSheet createSheet(String sheetName)`	erstellt ein neues Tabellenblatt
`removeSheetAt(int index)`	löscht ein Tabellenblatt
`int getNumberOfSheets()`	gibt die Anzahl der Tabellenblätter zurück
`String getSheetName(int index)`	gibt den Namen des Tabellenblattes zurück
`setSelectedTab(int index)`	selektiert ein Tabellenblatt
`HSSFName createName()`	definiert einen neuen Zellbereich
`removeName(int index)`	entfernt einen Zellbereich
`write(FileOutputStream fos)`	schreibt die Excel-Datei ins Filesystem

Methode aus HSSFSheet	Funktionalität
`boolean getProtect()`	gibt an, ob das Tabellenblatt geschützt ist
`setProtect(boolean protection)`	schützt das Tabellenblatt bzw. gibt es frei
`HSSFRow createRow(int rownum)`	erstellt eine neue Zeile
`removeRow(HSSFRow row)`	löscht eine Zeile

Methode aus HSSFRow	Funktionalität
`HSSFCell createCell(short column)`	Erstellt eine neue Zelle

Methode aus HSSFCell	Funktionalität
`HSSFCellStyle getCellStyle()`	gibt das Aussehen der Zelle zurück
`setCellStyle(HSSFCellStyle style)`	setzt das Aussehen der Zelle
`setCellValue(String wert)`	schreibt einen Wert in eine Zelle
`setCellFormula(String formula)`	schreibt eine Formel in eine Zelle

Methode aus HSSFCellStyle	Funktionalität
`setDataFormat(short format)`	setzt das Format der Zelle

Methode aus HSSFName	Funktionalität
`setNameName(String wert)`	setzt den Namen eines Zellbereichs
`setReference(String cellArea)`	definiert das Ausmaß des Zellbereichs

Darüber hinaus gibt es natürlich noch wesentlich mehr Funktionalitäten, um sämtliche Möglichkeiten von Excel auch über POI nutzen zu können. In Abb. 3 ist ein kleines Beispiel zu sehen, welche Möglichkeiten man mit POI hat, Zellen in Excel zu füllen und zu formatieren.

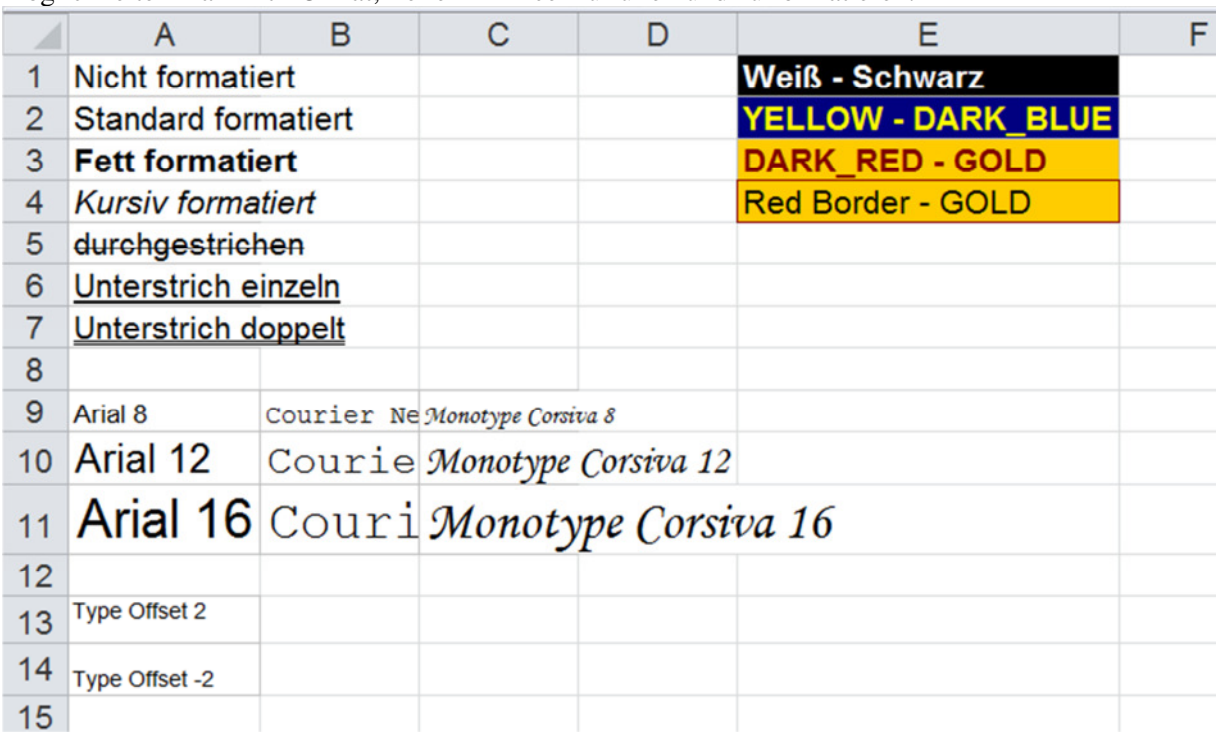


Abb. 4: Beispiele für die Möglichkeiten von POI

Die Umsetzung

Wie wurde das POI-Framework nun konkret eingebunden? Als erstes wurde eine neue Java-Klasse geschrieben, die als Wrapper für die POI-Funktionalitäten diente und die Methoden enthielt, die im Projekt benötigt wurden, z.B. die folgende:

```
public int schreibeNumber(double wert) {
  int result = RETURN_OK;
  if (m_cell != null) {
    try {
      if (m_cell.getCellType() != HSSFCell.CELL_TYPE_NUMERIC) {
        setDataformat(STYLE_STANDARD);
      }
      m_cell.setCellValue(wert);
      result = RETURN_OK;
    } catch (Exception e) {
      simpleErrors(e, "schreibeNumber");
      result = RETURN_FEHLER;
    }
  } else {
    result = RETURN_ZELL_NICHT_SELEKTIERT;
  }
  return result;
}
```

Diese neue Java-Klasse wurde nun in Forms über den Java Importer eingebunden, so dass die Methoden als generiertes PL/SQL-Package in der Form zur Verfügung standen.

```
FUNCTION schreibeNumber(
  obj   ORA_JAVA.JOBJECT,
  a0    NUMBER) RETURN NUMBER IS
BEGIN
  args := JNI.CREATE_ARG_LIST(1);
  JNI.ADD_DOUBLE_ARG(args, a0);
  RETURN JNI.CALL_INT_METHOD(FALSE,
                             obj,
                             'com/rwe/iais/excel/IaisExport',
                             'schreibeNumber',
                             '(D)I',
                             args);
END;
```

Beim Start einer Auswertung lief nun die folgende Programmlogik ab: Als erstes wurden sämtliche Ausgabewerte, die in irgendeiner Form in der Excel-Datei erscheinen mussten, mit ihren zugrunde liegenden Formeln in der Datenbank berechnet. Die Ergebnisse (Zahlen, Texte oder Zeitreihen) wurden in einem Object Type in der Datenbank abgelegt. Dieser Object Type hatte in etwa die Struktur wie die Excel-Datei. Es gab also für die 7 fest vorgegebenen Reiter jeweils unterschiedliche Strukturen mit Untertypen im Object Type, in denen die Ergebnisse dort abgelegt wurden.

Anschließend wurde per Webutil die zur Auswertung passende Excel-Vorlage aus der Datenbank auf den Application Server geladen. In diese Vorlage wurden dann mit Hilfe der POI-Funktionalitäten die Ergebnisse der Auswertung auf den jeweils passenden Reiter geschrieben. Die Datei wurde gespeichert und per Webutil auf den Client kopiert.

Außerdem wurde eine weitere spezielle Excel-Datei aus der Datenbank auf den Client kopiert. Diese Datei enthielt ein Makro, welches ein paar Nacharbeiten in der fertigen Auswertung durchführen musste. Diese Excel-Datei mit dem Makro wurde auf dem Client gestartet. Das Makro öffnete dann die eigentliche Auswertungs-Datei, führte die Nacharbeiten durch und wurde anschließend wieder gelöscht. Die Auswertungs-Datei blieb offen und konnte anschließend vom Anwender angepasst oder ergänzt werden. Anpassungen konnten allerdings nur auf den individuellen Reitern vorgenommen werden, die 7 fest vorgegebenen Reiter sind per Blattschutz gegen Änderungen geschützt.

Probleme und ihre Lösungen

Im letzten Abschnitt klang es schon an: Ein Visual Basic-Makro musste Nacharbeiten in der fertigen Auswertung durchführen. Warum? Aufgrund von Kompatibilitätsvorgaben musste im Projekt die POI-Version 2.5 eingesetzt werden und nicht die aktuellste. Das hatte verschiedene Konsequenzen.

Als erstes stießen wir auf einen Bug, durch den Zellkommentare und grafische Elemente wie Rechtecke oder Linien nicht verarbeitet werden konnten. Excel-Vorlagen, die solche Elemente enthielten, wurden durch die Bearbeitung mit POI korrupt und ließen sich anschließend nicht mehr öffnen. Glücklicherweise waren diese Elemente in den Auswertungen nicht zwingend notwendig, und der Fachbereich konnte diese Elemente aus den Auswertungen entfernen, so dass es an dieser Stelle keine Probleme mehr gab.

Ein weiteres Problem war die Nutzung von benannten Zellbereichen in Excel. Wie man in Abb. 2 an der gerade selektierten Zelle erkennen kann, sind alle Ergebnis-Zellen des ersten Standard-Reiters mit

einem Zellnamen versehen, um diese Werte auf den individuellen Reitern einfacher referenzieren zu können. Dies sollte normalerweise und laut Dokumentation wie folgt möglich sein:

```
// benannten Zellbereich erzeugen
HSSFName namedCell = m_workbook.createName();
// Name des Tabellenblattes herausfinden
String sheetName = m_workbook.getSheetName(sheetIndex);
// Name des Zellbereichs setzen
namedCell.setNameName(wert);
// Referenz erstellen für die Zelle, so dass der Zellbereich auf die Zelle
// verweist
namedCell.setReference(sheetName+"!"+m_row.getRowNum()+m_cell.getCellNum();
```

Aber so einfach war das nicht. Als erstes liefern getRowNum() und getCellNum() die Zellposition in der Form „3,2“ zurück, also 3. Zeile, 2. Spalte, und nicht als „B3“, wie es in Excel gebräuchlicher ist. Die Methode setReference() benötigt den Input für die Zelle aber im Format „B3“, weswegen das Format erst mal umgewandelt werden musst. Dies geschah mit Hilfe von

```
// aktuelle Zelle umwandeln von Format 0,0 in Excel-Format A1
String cellName = new CellReference(m_row.getRowNum(),
                                    m_cell.getCellNum()).toString();
```

Aber auch die anschließende Umformulierung des setReference()-Aufrufs brachte keinen Erfolg.

```
namedCell.setReference(sheetName+"!"+cellName);
```

setzte den Zellbereich nicht fest auf eine Zelle, z.B. „B3“, sondern interpretierte intern die Information „B3“ als relative Position zur aktuell selektierten Zelle. War die Zelle A1 momentan selektiert, wurde der Zellbereich korrekt auf die Zelle B3, also die 3. Zeile und 2. Spalte gesetzt. War aber aktuell z.B. die Zelle D6 selektiert, wurde der Zellbereich auf die Zelle E8 gesetzt, also 3. Zeile und 2. Spalte von D6 aus gesehen. Also durfte an setReference() nicht „B3“ übergeben werden, sondern „B3“. Dafür musste der Zellname aber erst mal über eine reguläre Expression aufgespalten werden, um herauszufinden, wo das $-Zeichen, welches zwischen Buchstaben und Zahlen hin muss, genau hinkommt. Denn es gibt ja in Excel auch Zellen der Form „AC34“. Der Code wuchs weiter an:

```
// benannten Zellbereich erzeugen
HSSFName namedCell = m_workbook.createName();
// Name des Tabellenblattes herausfinden
String sheetName = m_workbook.getSheetName(sheetIndex);
// Name des Zellbereichs setzen
namedCell.setNameName(wert);
// aktuelle Zelle umwandeln von Format 0,0 in Excel-Format A1
String cellName = new CellReference(m_row.getRowNum(),
                                    m_cell.getCellNum()).toString();
// Buchstabe heraussuchen, um $-Zeichen einfügen zu können
Pattern patt = Pattern.compile("[0-9]");
Matcher m = patt.matcher(cellName);
String cellLetter = m.replaceAll("");
int cellNumber = m_row.getRowNum()+1;
// Referenz erstellen für die Zelle, so dass der Zellbereich auf die Zelle
// verweist,
// $-Zeichen werden benötigt, da sonst die Region relativ und nicht absolut
// angelegt wird
namedCell.setReference(sheetName+"!$"+cellLetter+"$"+cellNumber);
```

Aber auch das war noch nicht das Ende. Durch einen weiteren Bug in POI war es nicht möglich, einen Zellbereich für eine einzelne Zelle zu definieren. Wir mussten also einen echten Bereich definieren, auch wenn dieser nur eine Zelle umfasste. Somit sah die letzte Zeile dann folgendermaßen aus:

```
// Referenz erstellen für die Zelle, so dass der Zellbereich auf die Zelle
// verweist, wegen Bug nicht einzelne Zelle, sondern Bereich
// $-Zeichen werden benötigt, da sonst die Region relativ und nicht absolut
// angelegt wird
namedCell.setReference(sheetName+"!$"+cellLetter+"$"+cellNumber+":$"+
                       cellLetter+"$"+cellNumber);
```

Eine weitere Einschränkung brachte die Nutzung der POI-Version 2.5 mit sich, da es dort noch nicht möglich ist, Spalten auf ihre optimale Breite zu setzen. Dies war aber nötig, weil die 7 festen Tabellenblätter mit Blattschutz geschützt werden, und deshalb die Spaltenbreite nicht manuell von den Anwendern verändert werden könnte, sofern die Breite einer Spalte nicht ausreicht. Deshalb wurde der Umweg über das Visual Basic Makro gewählt. Eine spezielle Excel-Vorlage mit dem Makro wurde aus der Datenbank geladen. In die Datei wurde der Dateiname der eigentlichen Auswertung geschrieben. Dann wurde auf dem Client die Datei mit dem Makro geöffnet. Das Makro macht im Wesentlichen nichts anderes, als die eigentliche Auswertung zu öffnen und dort die optimale Breite einer jeden Spalte zu setzen. Anschließend wird die Datei mit dem Makro wieder gelöscht, so dass nur noch die eigentliche Auswertung vorhanden ist.

Ein weiteres Problem gab es mit den benannten Zellbereichen in Excel. Die Auswertungen können ja in der Form geändert werden, dass die Reihenfolge der Ausgabewerte vertauscht wird. Geht man von der Auswertung in Abb. 2 aus, hieße dies z.B., dass in Zukunft „Ausgabewert 1“ in der dritten Zeile und „Ausgabewert 2“ in der zweiten Zeile stehen soll. Dementsprechend muss dann die Zelle D2 den Namen „Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_2_MONAT_1“ bekommen und die Zelle D3 den Namen „Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_1_MONAT_1“. Wenn man nun die Namen über POI in der neuen Reihenfolge nach Excel schreibt, bekommt Excel nicht mit, dass sich die Reihenfolge geändert hat. Warum das?

Ist auf einem anderen Reiter ein Verweis auf die Zelle D2 in der Form „=Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_1_MONAT_1“ hinterlegt, hat sich nach der Änderung der Reihenfolge dieser Verweis geändert zu „=Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_2_MONAT_1“. Der Verweis schien sich also nicht den Namen der Zelle zu merken, sondern irgendetwas anderes, was scheinbar immer noch auf die Zelle D2 verwies. Es stellte sich dann heraus, dass Zell-Namen in Excel auch jeweils eine ID haben. Alle Namen, die in einer Excel-Datei definiert sind, und die z.B. über den Namen-Manager zu sehen sind, haben eine interne ID und zwar in der Reihenfolge, wie sie angelegt wurden. Und über diese ID wird der Verweis in einer anderen Zelle geregelt. Löscht man nun händisch in Excel einen Namen, werden die IDs aller folgenden Namen um 1 heruntergesetzt, aber auch die Verweise entsprechend angepasst. Ändert man über POI etwas an den Namen, werden die IDs der Verweise scheinbar nicht mitkorrigiert. Am Beispiel war also Folgendes passiert: Der Zellname „Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_1_MONAT_1“ hatte ursprünglich die ID 1, der Zellname „Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_2_MONAT_1“ hatte die ID 2. In der Zelle, die auf einem der individuellen Reiter in der Excel-Vorlage den Verweis „=Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_1_MONAT_1“ hinterlegt hat, war intern hinterlegt, dass der Name mit der ID 1 referenziert wird. Als nun nach der Reihenfolge-Änderung „Ausgabewert 2“ vor „Ausgabewert 1“ stand und somit auch die entsprechenden Zellnamen in umgekehrter Reihenfolge angelegt wurden, hatte nun der Name

„Geschäftspartner_XY.Kraftwerk_Z.Ausgabewert_2_MONAT_1“ die ID 1 bekommen, weil er als erstes angelegt wurde. Und die Zelle auf dem individuellen Reiter verwies auf einmal auf „Ausgabewert 2“ statt auf „Ausgabewert 1“, weil dieser nun die ID 1 bekommen hatte.
Lösen ließ sich dieses Problem nur dadurch, dass im Vorfeld geprüft wurde, in welcher Reihenfolge die Namen in der Excel-Vorlage angelegt worden waren. Es wurde also die Reihenfolge über die ID herausgesucht. Anschließend musste beim Laden der Daten nach Excel sichergestellt werden, dass die Namen der Ergebnis-Zellen eines jeden Ausgabewertes nicht zeitgleich mit dem Füllen seiner Daten angelegt wurden, sondern dass die Werte in einer anderen Reihenfolge als die zugehörigen Zellnamen nach Excel geschrieben wurden. Auf diese Weise blieben die IDs in der richtigen Reihenfolge erhalten und die Verweise wurden von der Excel-Vorlage korrekt in die eigentliche Auswertung übernommen.
Blieb noch als kleines Randthema zu klären, was mit Ausgabewerten passiert, die aus der Auswertung gelöscht werden sollten. Die zugehörigen Namen durften natürlich nicht gelöscht werden, da sonst die Reihenfolge der IDs wieder durcheinander geraten würde. Also wurden diese Namen umbenannt, indem sie ein „gelöscht-Prefix“ bekamen. Dies sollte dafür sorgen, dass diese Namen nicht versehentlich vom Fachbereich auf einem individuellen Reiter neu zugeordnet würden. Außerdem wurde die Referenz dieses Namen auf eine Zelle fernab der Ergebnisliste gesetzt, so dass nur noch leere Zellen referenziert würden und auf diese Weise evtl. noch vorhandene Verweise auf diese Zellen auf den individuellen Reitern schnell auffallen würden.

Nachdem dieses Problem auch gemeistert war, machten wir noch die Bekanntschaft mit java.lang.OutOfMemoryError. Auf dem zweiten festen Tabellenblatt werden ja ganze Zeitreihen ausgegeben. Da diese Zeitreihen Viertelstundenwerte sind, gibt es also je Zeitreihe und Monat bis zu 2976 Werte. Hat man nun eine Auswertung, die 20 Zeitreihen enthält und auch noch Werte für 12 Monate ausgibt, ist man schnell bei mehr als 700.000 Werten, die nach Excel gebracht werden müssen. Dies brachte POI an seine Grenzen. Und mehr Speicher zuzuweisen war nicht möglich, da POI ja in der JVM von Forms mit läuft und somit darüber begrenzt ist. Eine Idee war, die POI-Funktionalitäten aus der Form auszulagern in einen separaten Thread, dem man dann einen größeren Speicher mitgeben könnte. Allerdings war glücklicherweise noch eine einfachere Lösung möglich. Der Fachbereich stellte fest, dass in diesen umfangreichen Auswertungen die kompletten Zeitreihen überhaupt nicht benötigt werden. Dies erscheint auch logisch, wenn man sich vorstellt, dass 700.000 Werte ja auch nicht auf die Schnelle geprüft werden könnten. Also wurden die entsprechenden Zeitreihen aus den Auswertungen entfernt, was per Zuordnung in der Form möglich ist. Das hatte natürlich auch den angenehmen Seiteneffekt, dass diese Auswertungen dann noch wesentlich performanter waren, weil wesentlich weniger Daten benötigt wurden.

Vergleich alt-neu

Apropos Performance: Dies war auch ein wesentlicher Punkt bei der Neuentwicklung des Systems. Das Altsystem war ein Client/Server-System, wodurch auch alle Auswertungen auf dem Client durchgeführt wurden. Da dort viele Auswertungen sehr lange brauchten, hatte man eine Funktionalität eingebaut, Auswertungen speziell auf dem Datenbank-Server laufen zu lassen. Allerdings musste dann regelmäßig geprüft werden, ob diese Auswertungen bereits fertig waren. Und nach erfolgreicher Durchführung mussten diese dann noch abgeholt werden.

Mit der Umstellung auf Forms im Web wurden diese Auswertungen automatisch auf dem Application Server erstellt und nach Erstellung auf den Client transferiert. Dadurch ließen sie sich wesentlich schneller und komfortabler als vorher ausführen.

Die Programmierung der Auswertungs-Schnittstelle war aufwändiger, da Excel und Visual Basic im Vorsystem wesentlich integrierter genutzt werden konnten. Allerdings hat man nun die Schnittstelle

sinnvoller gekapselt und hat es bei weiteren Technologiewechseln oder Ergänzung von Funktionalitäten nun wesentlich leichter.

Ausblick

Gerade Technologiewechsel zum Beispiel zu Java oder ADF wären nun wesentlich leichter möglich. Denn die Schnittstelle liegt ja gekapselt als Java-Funktionalität vor und kann somit natürlich auch ohne Probleme aus anderen Technologien heraus aufgerufen werden.

Wie sieht das aber aus, wenn man POI noch nicht im Einsatz hat, sondern z.B. Oracle Reports? Hier besteht die Möglichkeit, die Reports umzuformen in Excel-Auswertungen. Dies ginge am Einfachsten dadurch, dass man für die entsprechenden Reports Object Types in der Datenbank definiert, welche die Struktur der Reports abbilden. Diese könnten dann in der Datenbank mit Werten gefüllt werden. Außerdem benötigt man eine Excel-Vorlage, die vom Layout her dem bisherigen Report entspricht. Diese Vorlage könnte man dann mit den in der Datenbank zusammengestellten Werten verknüpfen und erhielte die neue Auswertung.

Anschließend hat man zwei Optionen. Man könnte den Vorteil nutzen, dass Excel bearbeitbar ist, und die Auswertung freigeben zum weiteren Bearbeiten, sofern das vom Anwender gewünscht ist. Aber man kann natürlich auch über die Blattschutz-Funktionalität dafür sorgen, dass die Reports auch weiterhin schreibgeschützt sind. Oder man verbindet beide Optionen wie bei IAIS und schützt bestimmte Bestandteile und lässt andere frei änderbar.

Im Fall einer größeren Oracle Forms/Reports-Anwendung hätte man so die Möglichkeit, schon mal Teile der Anwendung, nämlich die Reporting-Funktionalität auf eine neue Technologie-Ebene zu stellen, ohne die Anwendung selbst ändern zu müssen. Anschließend könnte man sich dann mit einer Migration der Anwendung beschäftigen oder sich auch erst mal damit zufrieden geben, dass zumindest die Reports im neuen Stil und wesentlich leichter wartbar sind.

Kontaktadresse:

Sven-Olaf Kelbert
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon: +49 (0) 2102 30961-0
Fax: +49 (0) 2102 30961-10
E-Mail sven-olaf.kelbert@mt-ag.com
Internet: www.mt-ag.com