Erfahrungen aus einem Forms-APEX Konvertierungsprojekt

Hertha Rettinger
up to data professional services GmbH
Wörrstadt

Schlüsselworte:

Forms, APEX, Migration, Konvertierung

Einleitung:

Bei der Migration von Forms-Anwendungen nach APEX ist es mit dem Durchführen der in der APEX-Dokumentation vorgegebenen Schritte nicht getan. Die generierte Anwendung hat in Aussehen und Funktionalität wenig mit der ursprünglichen Forms-Anwendung zu tun. Die eigentliche Arbeit kommt erst danach. Das Konvertierungsprojekt in APEX kann jedoch mit seinem integrierten Projekt-Statusmanager eine große Unterstützung sein. Kennt man die Stolpersteine die einem auf diesem Weg begegnen, so kann man diese gut umgehen oder überspringen.

Von der Forms-Anwendung zur generierten APEX-Applikation – und was kommt danach?

Schauen wir uns kurz die Konvertierung an.

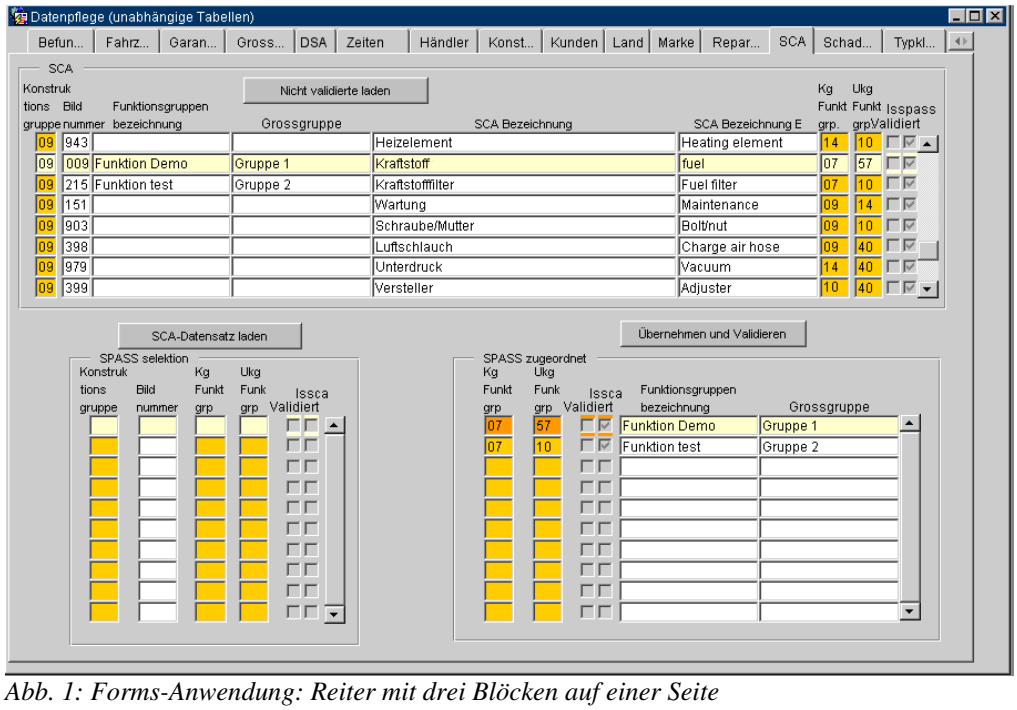

Abb. 1: Forms-Anwendung: Reiter mit drei Blöcken auf einer Seite

Wir gehen von einer Forms-Anwendung aus, die für die Stammdatenpflege entwickelt wurde.
Jeder Reiter enthält einen Datenblock. Eine Ausnahme bildet ein Reiter mit drei Blöcken bei dem Daten zwischen Tabellen abgeglichen werden:

Nun beginnen wir mit der Konvertierung.
Wie sieht unsere Erwartung aus? Vielleicht so?

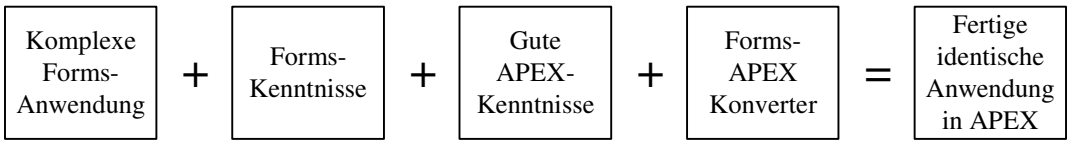

Abb. 2 Falsche Erwartung an den Konverter

Wir befolgen die Schritte in der Anleitung in APEX:

Erzeugen der Metadaten: Generieren der XML-Dateien.
Unsere Anwendung besteht aus den beiden Forms-Modulen Stammdaten.fmb und Password.fmb, aus dem Menümodul Menu_Admin.mmb und der Forms-Bibliothek util.pll.
Um die Metadateien zu erzeugen verwenden wir die Werkzeuge aus der Oracle Developer Suite (in dem Beispiel wird die Version 10.1.2 verwendet).
Unsere Forms-, und Menüdateien konvertieren wir mit dem Forms2XML Konvertierungswerkzeug hierfür rufen wir die Datei `frmf2xml` auf und übergeben die Dateinamen, der zu konvertierenden Module:

```
C:\Formsfiles>frmf2xml overwrite=yes Stammdaten.fmb
Oracle Forms 10.1.2 Forms to XML Tool
Copyright(c) 2001, 2005, Oracle.  All rights reserved.

Processing module Stammdaten.fmb
XML Module saved as Stammdaten_fmb.xml

C:\Formsfiles>
```
Abb. 3: Generieren der Metadaten mit dem Forms2XML Konvertierungswerkzeug

Auf diese Weise erzeugen wir die Dateien Stammdaten_fmb.xml, Password_fmb.xml und Menu_Admin_mmb.xml.
Um die pll-Datei zu konvertieren, verwenden wir den zu Report Builder gehörende rwconverter.
Mit dem Kommando
rwconverter stype=pllfile source=util.pll dtype=pldfile dest=util.pld
öffnet sich der entsprechende Dialog, in dem wir noch die Konvertierung bestätigen.

Erstellen des APEX Workspace und des Migrationsprojektes
Wir benötigen das Datenschema der ehemaligen Forms-Anwendung auf der gleichen Datenbank auf der APEX installiert ist. Dann erstellen wir einen APEX Workspace und ordnen das Datenschema diesem Workspace zu.
Daraufhin erstellen wir unser Migrationsprojekt und laden die im vorangegangenen Schritt erzeugten Metadaten in das Projekt. Zuerst geben wir unserem Projekt einen eindeutigen Namen, ordnen das Datenschema dem Projekt zu und geben die XML-Datei für die erste hochzuladende Form an.
Dabei ordnen wir das Datenschema unserem Migrationsprojekt zu. Dann laden wir nacheinander die übrigen Metadaten-Dateien hoch.
Nach dem Hochladen der übrigen Module bestätigen wir die Erstellung.

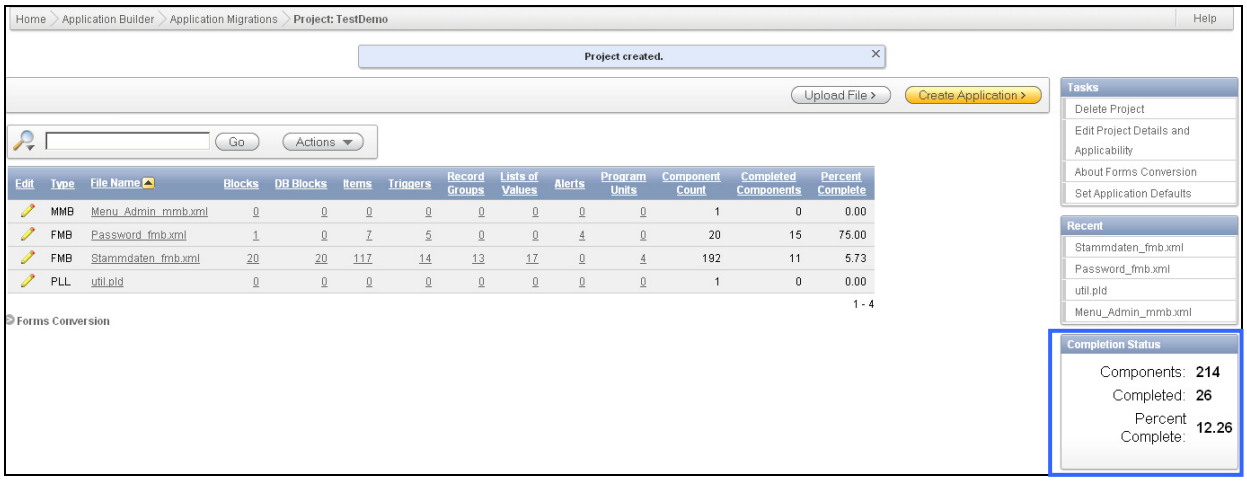

Abb. 4: Modulübersicht im Projekt und Komponentenstatus

Analyse der Oracle Forms Anwendung

Ab diesem Zeitpunkt benötigen wir die Oracle Developer Suite nicht mehr, denn alle benötigten Informationen stehen uns innerhalb des Migrationsprojektes zur Verfügung. In der Projektübersicht sehen wir wie viele Komponenten das jeweilige Modul insgesamt enthält, aber auch die Anzahl der wichtigsten Komponenten wie Blöcke, Wertelisten, Trigger, Alerts usw. Diese sind mit Links versehen, so daß direkt zur Übersicht dieser Komponenten gewechselt werden kann. Eine Spalte enthält die Anzahl der umgesetzten Komponenten. Der Umsetzungsstatus im rechten unteren Bereich zeigt uns wie weit wir in unserem Projekt fortgeschritten sind und wie viel Arbeit noch aussteht. Die 12,26% scheinen recht wenig.

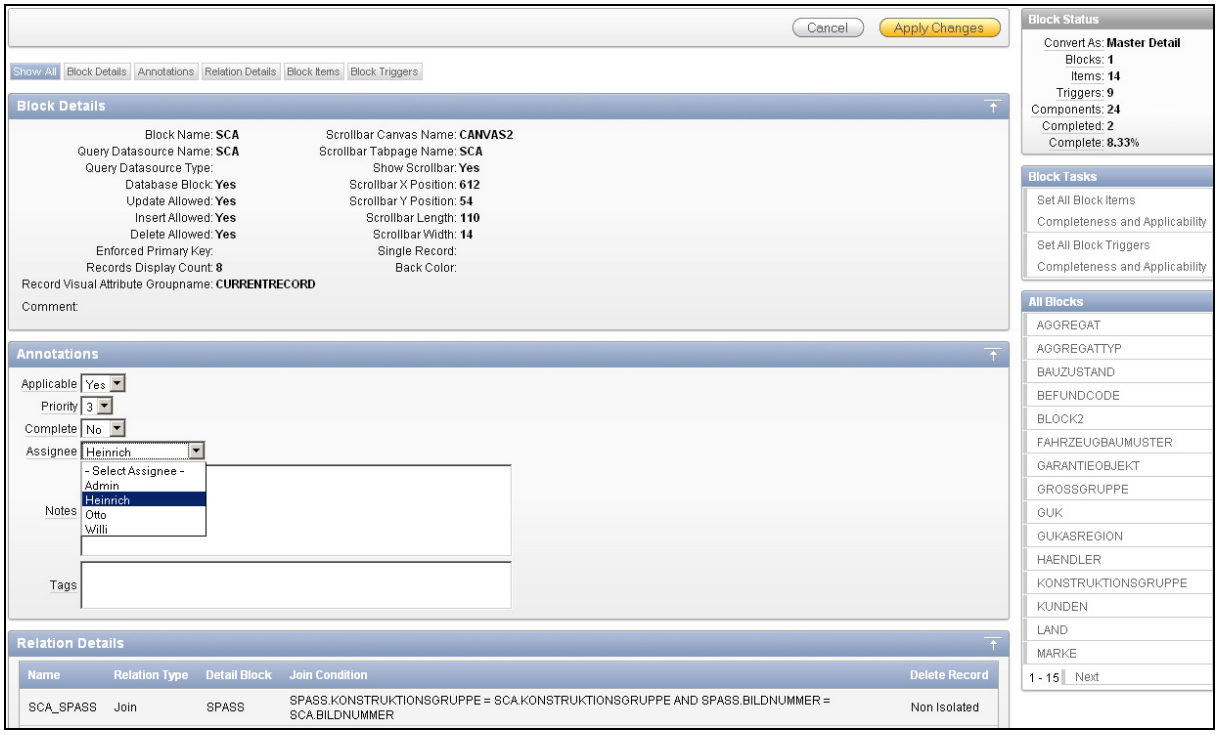

Abb. 5: Block Details mit Eigenschaften aus Forms

Jede im Projekt enthaltene Komponente kann über eine Details-Maske eingesehen werden. Hier sehen wir die objektspezifischen Eigenschaften wie sie in Forms über die Eigenschaftenpalette eingestellt wurden. Einige sind für APEX uninteressant, wie z.B. die Maße für den Rollbalken.
Im Annotations-Bereich kann für jede Komponente deren Anwendbarkeit und Status gesetzt werden. Die Komponente kann einer Person die für die Umsetzung verantwortlich ist, zugewiesen werden.
Für Datenblöcke werden anschließend auch deren Unterkomponenten wie Relationen, Elemente (z.B. Felder) und Block Trigger bzw. Item Trigger angezeigt.
Zusätzlich wird für Datenblöcke noch im rechten oberen Bereich des Fensters der Blockstatus angezeigt. Das Feld "Convert As" zeigt an in was der Block voraussichtlich in APEX umgesetzt werden soll. In unserem Block in Abb. 5 ist "Master Detail" angegeben. Hier haben wir wieder den Umsetzungsstatus der enthaltenen Komponenten, in diesem Fall der Komponenten innerhalb des Blockes. Bei der Errechnung des Umsetzungsstatus werden nicht anwendbare Komponenten als erledigt gezählt. Die beiden erledigten Komponenten in unserem Block sind zwei ON-ERROR Trigger die auf Block- und Feldebene definiert waren und als nicht anwendbar voreingestellt wurden.
Nun können wir noch Beschriftungen für Blöcke und Felder ändern, um anschließend zur Generierung der Anwendung überzugehen.

Generierung der APEX Anwendung
Wir haben die Möglichkeit, durch das Abwählen der "Include"-Auswahl einzelne Blöcke von der Übernahme in die zukünftige APEX-Anwendung auszuschließen.
Dann klicken wir auf [Create Application >] und lassen uns von dem Wizard führen.
Im ersten Wizardschritt geben wir der Anwendung einen Namen. Die ID wird automatisch vergeben und angezeigt. Das Applikationsschema wird vom Datenschema des Migrationsprojektes übernommen.
Im zweiten Wizardschritt werden die Seiten aufgelistet, welche in der zukünftigen APEX-Anwendung durch die Generierung erzeugt werden sollen. Hier können noch Seiten gelöscht oder auch neue hinzugefügt werden.
Im dritten Wizardschritt entscheiden wir über das Aussehen der zukünftigen Anwendung, indem wir eines der APEX-Themen auswählen.
Im vierten Wizardschritt sehen wir eine Zusammenfassung der Angaben über die zukünftige Anwendung. Zusätzlich zu den Angaben aus den vorangegangenen Schritten, sind hier noch die Sprache, die Angabe über die Verwendung von Tabs und das Authentifizierungsschema sichtbar.
Nun haben wir unsere fertige Anwendung und können diese starten.

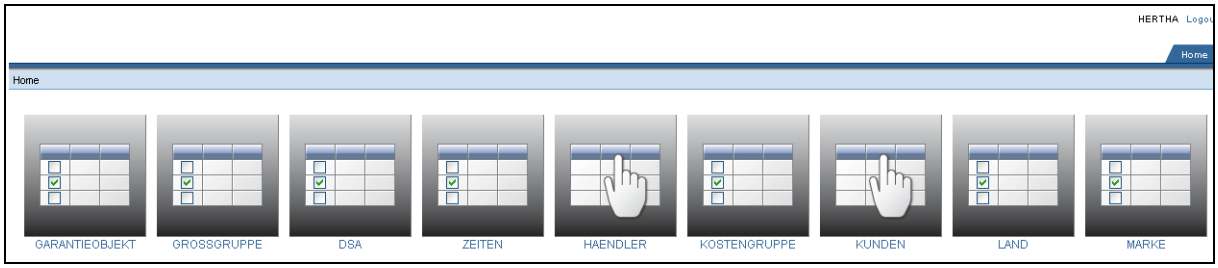

Abb. 6: Startseite der generierten APEX-Anwendung

Was ist aus unserer ursprünglichen Anwendung geworden? Wo sind unsere Reiter hin? Was ist aus unserem Menü geworden? Wo ist unsere Ausgangsmaske aus Abb.1?
Wir sehen eine horizontale Navigationsliste. Für jeden Datenbankblock aus der Formsanwendung gibt einen Eintrag. Jeder Eintrag verweist auf eine andere Seite. Da unsere Ausgangsanwendung viele Datenbankblöcke enthielt, ist die Liste der Einträge sehr lang. Wir finden die drei Blöcke aus unserer Maske aus Abb. 1 in der zweiten Hälfte der sich über mehr als zwei Bildschirmseiten erstreckenden Liste.

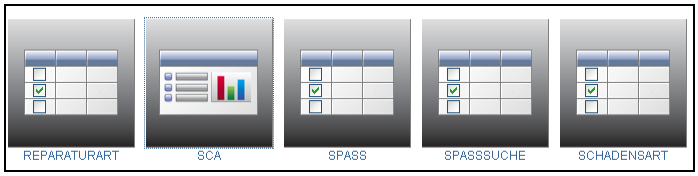

Abb. 7: Drei Blöcke aus einer Maske als einzelne Unterpunkte

Für den SCA-Block war eine "Master Detail" Maske angekündigt worden. In Wirklichkeit enthält die generierte Anwendung einen Interaktiver Bericht mit zugeordneter Pflegemaske. Die beiden anderen Blöcke der Maske wurden in eigenen Seiten zu editierbaren Berichten.
Und wo ist die Funktionalität (die PL/SQL-Blibliotheken und Trigger)?

Überarbeiten der Anwendung. Umsetzung der Funktionalität
Wir erinnern uns daran, daß der Umsetzungsstatus ursprünglich bei 12,26% lag. Das war nicht etwa der Anteil des Konverters bei der Generierung, sondern der Anteil der nicht anwendbar markierten Komponenten. Und darunter zählen die funktionalen Blöcke samt der enthaltenen Elementen (Buttons) und Triggern.
Es kommt also noch viel Arbeit auf uns zu! Die PL/SQL-Funktionalität müssen wir im Migrationsprojekt sichten und nur das nach APEX übertragen was sinnvoll ist. An anderen Stellen müssen wir neuen Code schreiben. Tabellen, die der Konverter über mehrere Seiten verteilt hat, auf eine Seite zusammenführen und unsere Buttons mit deren Funktionalität hinzufügen.
Um solche Arbeiten zu steuern, benutzen wir den bereits weiter oben erwähnten Annotations-Bereich, in dem für jede Komponente ein zuständiger Bearbeiter zuweisen und die Anwendbarkeit und der Status eingestellt werden kann. Sobald der Bearbeiter die ihm zugewiesenen Komponenten bearbeitet hat, stellt er den Status um. Dadurch kann jederzeit eingesehen werden, wie weit die Umsetzung in unserem Projekt fortgeschritten ist.

Und was kommt danach?

Wir erinnern uns an unsere eingangs geschilderte (falsche) Erwartung an den Konverter. Jetzt wird es Zeit diese zu revidieren:

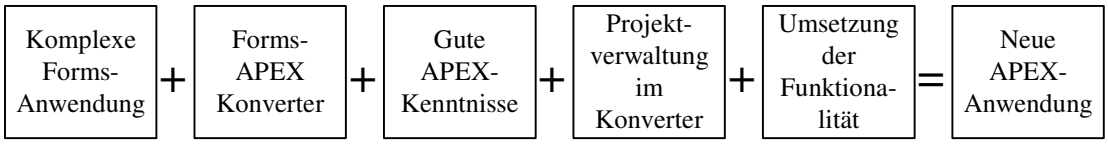

Abb. 8: Tatsächliche Umsetzung mit dem Konverter

Macht eine Neuerstellung der Anwendung in APEX nicht mehr Sinn als diese zu migrieren?
Mein Tipp: Erstellen Sie die Anwendung in APEX neu, benutzen Sie dabei aber den Konverter, denn er kann Ihnen von großer Hilfe sein:

- Nachdem einmalig die Metadaten aus Forms extrahiert wurden, brauchen Sie weiterhin weder den Forms Builder noch den Forms Quellcode.
- Nachdem Sie das Projekt erstellt und die Metadaten hochgeladen haben, ist der gesamte Code, zusammen mit allen Einstellungen der Komponenten (Farbe, Schriftart usw.) im Projekt einsehbar. Hieraus kann Code kopiert und in die Anwendung übertragen werden. Der Entwickler braucht keine Forms-Kenntnisse um Code-Abschnitte zu übertragen.

- Mit dem integrierten Projekt-Statusmanager können umzusetzende Komponenten den Entwicklern zugewiesen werden und jederzeit der Umsetzungsfortschritt eingesehen werden. Die Gefahr, daß frühere Funktionalität vergessen wird, wird verringert.
- Die den konvertierten Feldern vormals zugewiesenen Wertelisten werden in LOVs in den Shared Components abgelegt und können in der neuen Anwendung verwendet werden.
- Der Konverter nimmt Ihnen die Arbeit ab, die Wizards für die Blöcke zu klicken. Feldbeschriftungen werden dabei aus Forms weitgehend übernommen.

Was Sie beachten sollten, wenn Sie den Konverter verwenden

Spracheinstellung bei der Erzeugung der XMLs.
Wer ein deutsches Betriebssystem verwendet, in Forms aber auf englisch entwickelt hat, kennt die Umgebungsvariable DEVELOPER_NLS_LANG mit der der Forms Builder sprachlich umgestellt werden konnte. Das Forms2XML Konvertierungswerkzeug ignoriert aber diese Umgebungsvariable, so daß es im Migrationsprojekt zu seltsamer sprachlicher Vermischung kommt, wenn man die Umgebungsvariable NLS_LANG (bzw. deren NLS_LANGUAGE-Anteil) nicht auf die in APEX verwendete Sprache einstellt. Der Konverter kommt allerdings erstaunlich gut damit zurecht.

Item Type	Data Type	Database Item
Schaltfläche		Yes
Anzeigeobjekt		No
Textobjekt		No

Abb. 9: Sprachliche Vermischung

Funktionale Blöcke – undokumentiertes Feature?
Funktionale Blöcke werden laut APEX-Dokumentation bei der Konvertierung nicht berücksichtigt. Solche Blöcke sind standardmäßig abgewählt und wenn man sie dennoch (über die Include-Auswahl der Blöcke) anwählt, so wird daraus eine leere Seite generiert, so die Dokumentation. Für einen solchen Block wird auch "Convert As: Blank" im "Block Status" angezeigt. Schaut man sich die enthaltenen Items an, sind auch diese alle abgewählt und für alle im Block enthaltenen Komponenten steht die Anwendbarkeit auf "No". Das stimmt mit der Dokumentation überein. Wählt man aber einen solchen Block und einzelne Items darin an, so erlebt man nach der Konvertierung eine angenehme Überraschung:

Include	Item Name	Label	Item Prompt	Triggers	Item Type
☑	PW_ALT		Altes Paßwort	1	Text Item
☑	PW_NEU		Neues Paßwort	0	Text Item
☑	PW_NEU1		Neues Paßwort wiede	1	Text Item
☑	EXIT	Exit		1	Push Button
☐	DB_PW		PW	0	Display Item
☑	OK	OK		1	Push Button
☐	DB_USER		Datenbankbenutzer	0	Display Item

Abb. 10: Manuell angewählte Elemente eines funktionalen Blocks.

In der generierten Anwendung wird nicht eine leere Seite, sondern eine Seite vom Typ "Navigation Form" erstellt. Und darin sind die angewählten Items (hier Felder und Buttons) samt Beschriftungen enthalten!

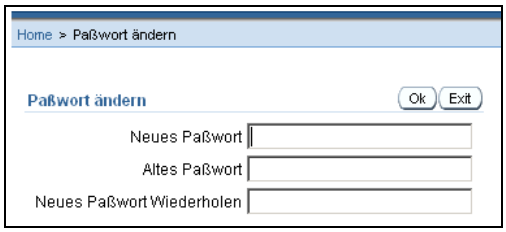

Abb. 11: Funktionaler Block nach der Anwendungsgenerierung

Mein Tipp: Wenn Ihre Forms-Anwendung funktionale Blöcke enthält, dann schalten Sie im Migrationsprojekt zunächst die Anwendbarkeit für alle darin enthaltenen Komponenten auf "Yes", auch wenn Sie diese in der Applikationsgenerierung nicht mitnehmen. Denn unter funktionalen Blöcken liegt Funktionalität die möglicherweise auch in der zukünftigen APEX-Anwendung implementiert werden soll.

Include-Auswahl. Unterschiedliches Verhalten
Die Include-Auswahl haben wir sowohl auf Block- als auch auf Item-Ebene. Darüber können wir Blöcke und Items die in der Forms-Anwendung enthalten waren, von der Anwendungsgenerierung ausnehmen. Allerdings funktioniert das nur auf Block-Ebene. Auf Item-Ebene wird die Einstellung für Datenbankblöcke ignoriert und nur bei funktionalen Blöcken berücksichtigt. Das heißt, daß Items von Datenbankblöcken auch dann in die generierte Anwendung einfließen, wenn sie abgewählt wurden.

Buttons innerhalb von Datenbankblöcken
Hat man sich bei der Forms-Entwicklung nicht an die Empfehlung gehalten, für Buttons funktionale Blöcke vorzusehen und hat diese stattdessen innerhalb von Datenbankblöcken angeordnet, so bekommt man bei der Konvertierung eine etwas seltsame Fehlermeldung:
"Database schema TESTDATEN does not contain the associated database objects for the project, DemoTest. Ensure the database schema associated with the project contains the database objects associated with the uploaded Forms Module .XML file(s)."
Interessant ist die Auflistung der angeblich invaliden Objekte:

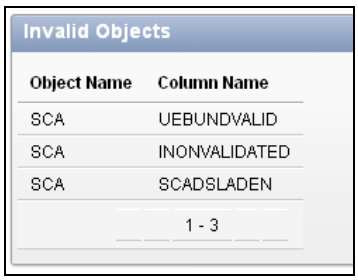

Abb. 12: Namen von Buttons als Datenbankspalten aufgelistet

Die angeblichen Spaltennamen sind eigentlich die Namen von drei Buttons die im Datenbankblock enthalten sind. Diese werden im Migrationsprojekt sogar korrekt als Buttons erkannt. Doch die Eigenschaft "Database Item" zeigt in der Itemübersicht "Yes" an. Nun müßten wir nur die Include-Auswahl für diese Items abwählen. Aber wir befinden uns in einem Datenbankblock und da wird diese Einstellung ignoriert, wie wir weiter oben gesehen haben. Also kommen wir so nicht weiter.
Dieses Verhalten ist bei Oracle als Bug registriert, und zwar Bug 9827853 - FORMS MIGRATION: BUTTONS ON DATABASE BLOCK CANNOT BE EXCLUDED FROM CONVERSION
Als Workaround empfiehlt Oracle:

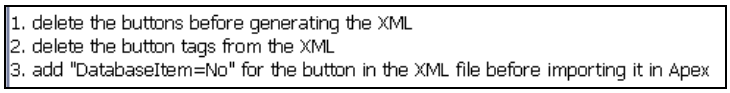

Abb. 13: Workaround für APEX Version 3.2 und 4.0

Der Workaround ist allerdings nur erforderlich, wenn Sie mit APEX Version 3.2 oder 4.0 (Basisversion) arbeiten. Der Fehler ist in der Version 4.0.1 behoben. Näheres dazu finden Sie in den Patch Set Notes zur Release 4.0.1.00.03 unter der URL
http://www.oracle.com/technetwork/developer-tools/apex/401-patch-166923.html
Mein Tipp: Egal welche Version Sie einsetzen, übertragen Sie die Buttons innerhalb von Forms in einen funktionalen Block. Dann können Sie diese, wie weiter oben beschrieben, in die Applikation übernehmen.

Upload der ersten Datei
Laut Dokumentation soll beim Anlegen eines Migrationsprojektes mit dem Hochladen der Metadaten einer Form (_fmb.XML) begonnen werden. Befolgt man diese Empfehlung, läuft auch alles richtig. Versucht man es aber zuerst mit den Metadaten eines Menüs, bekommt man eine aussagekräftige Fehlermeldung. Anders verhält es sich aber, wenn man mit einer pld-Datei beginnt, oder gar mit einer beliebigen anderen Datei. Das Hochladen wird dann zunächst akzeptiert. Erst bei Bestätigen mit [Finish] kommt die Meldung:

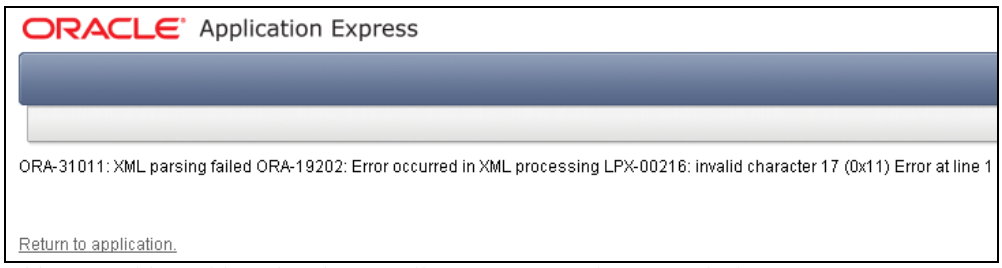

Abb. 14: Fehlermeldung bei der Erstellung eines Projektes mit falschem Dateityp.

Voraussage zu was ein Forms-Block in APEX generiert wird
Weiter oben haben wir gesehen, daß im Migrationsprojekt für die Blöcke angezeigt wird, in was diese voraussichtlich konvertiert werden. Doch diese Voraussage stimmt nicht immer.
Bei der Angabe "Master Detail" erhielten wir stattdessen einen Interaktiver Bericht mit zugeordneter Pflegemaske. Eine weitere unschöne Überraschung war, daß hier noch nicht mal berücksichtigt wurde, welche Felder aus der alten Anwendung enthalten waren. Die beiden erzeugten Seiten enthielten einfach alle Felder der Datenbanktabelle.
Bei der Angabe "Blank" für funktionale Blöcke haben wir ebenfalls gesehen, daß wir sogar mehr daraus machen können.
Die Angaben "Report and Form" und "Tabular Form" haben sich in meinen Beobachtungen allerdings bisher als richtig erwiesen. In diesen Fällen wurden auch die Feldbeschriftungen aus der alten Anwendung weitgehend richtig übernommen.

Wissen über Defaults
Um Überraschungen zu vermeiden, wenn eine mit dem Konverter generierte Maske nicht auf Anhieb funktioniert, sollte man die Dokumentation lesen, Hier nur einige Beispiele:
Datenerfassmasken werden in der Annahme generiert, daß PKs über Datenbanktrigger gefüllt werden. Ist das nicht der Fall, dann muß der Entwickler das entweder in der Datenbank nachholen oder die neue Anwendung anpassen.

Wertelisten aus Forms werden in LOVs in den Shared Components abgelegt. Das funktioniert einwandfrei bei klassischen zweispaltigen Wertelisten mit einem ID- und einem Beschriftungsfeld.
Hatte man aber in der alten Anwendung eine Werteliste die bei der Auswahl eines Eintrages gleich mehrere Felder befüllte, dann stößt man an die Grenzen von APEX wo das nicht vorgesehen ist.

Fazit

Plant man eine Migration von bestehenden Forms-Anwendungen nach APEX, dann sollte man diese wie eine Neuentwicklung der alten Anwendung sehen. Ist man sich der Möglichkeiten und Grenzen des Konverters innerhalb von APEX bewußt, dann kann dieser ein wichtiges Hilfsmittel bei der Umsetzung sein.

Kontaktadresse:

Hertha Rettinger
up to data professional services GmbH
Schornsheimer Chaussee 7
D-55286 Wörrstadt

Telefon: +49 (0) 6732-949014
Fax: +49 (0) 6732-949094
E-Mail hertha.rettinger@uptodata.de
Internet: www.uptodata.de