ADF12c Top 10 Stolpersteine und ihre Lösungen

Markus Klenke
TEAM GmbH
Paderborn

Schlüsselworte
ADF Best Practices, ADF 12c Migration

Einleitung

Die Version 12c von ADF hat viele Verbesserungen und Erweiterungen im Vergleich zu den Vorgängerversionen. Dabei wurden alle Bereiche aufgewertet, von der UI über zusätzliche Web- und Daten-Technologien, bis hin zur Architektur. Dennoch leben viele ADF-Anwendungen noch auf älteren Release-Branches der 11g oder sogar 10g FMW Familie. Die meisten Entwickler scheuen sich vor der Migration; zu groß ist die Angst vor dem Unbekannten oder möglichen Hindernissen.

Gerade der Übergang auf die neueste Version 12.2.1 enthält noch weitere Stolpersteine, da Oracle selbst hier auf eine neue Java Version (1.8) migriert hat. Durch dieses Upgrade funktionieren einige Features anders, als sie noch in 12.1.x funktioniert haben.

Um den Einstieg in die neue Entwicklungs- bzw. Laufzeitumgebung so leicht wie möglich zu machen, soll in diesem Vortrag auf einige der schwerwiegendsten Änderungen hingewiesen werden und an diesen Beispielen gezeigt werden, wie sich die Probleme bei der Migration umgehen lassen.

Die Problembeschreibungen befassen sich mit High-Level Migrationsthemen, betreffen aber gleichzeitig auch technische Komponenten wie neue Web-Technologien und Source-Code Änderungen, die sich bei der Migration auf die neue Version des Frameworks ergeben, um somit allen Teilnehmern etwas mitgeben zu können. Die folgenden Paragraphen sind Beispiele für Probleme, welche im Vortrag beschrieben und gefixt werden.

Performance Probleme in 12.2.1 bei der Neuanlage in einer Tabelle

Technisch gesehen bietet der Übergang in 12c auf Seiten der Business Components wenige Hindernisse, sofern die Daten nicht außergewöhnlich individuell geholt werden. Aber selbst auf dieser standardisierten Ebene sind einige Stolpersteine zu finden. Vor allem auf der Performance-Seite gibt es einige Elemente, welche in älteren Versionen von ADF ohne weiteres funktioniert haben, nach einer Migration hin zu 12c allerdings eine Laufzeit von mehreren Minuten haben. Insbesondere die Sortierfunktion bringt hier durch die Neu-Implementierung der Current-Row Mechanik einige Features, die bei jeder Migration überprüft werden müssen. Ein Beispiel wäre die Neuanlage eines Datensatzes in einer Tabelle mit vielen Datensätzen. Theoretisch sollte diese Aufgabe möglich sein, ohne alle Datensätzen vorerst zu laden. ADF versucht standardmäßig nun, die neu erstellte Zeile zu selektieren.

In 12c wurde die Implementierung des Key Objektes allerdings so veröndert, dass eine neue Zeile keinen „NULL“ Schlüssel erhält, sondern einen generischen Key mit NULL Werten. Grundsätzlich ist diese Änderung eine gute Idee um NullPointer-Zugriffen entgegenzuwirken. Allerdings hat die Änderung schwere Auswirkungen auf die Methode retrieveByKey im ViewObject, welche eine interne Abfrage (key==null) enthält. Diese Aussage nach dem oben beschriebenen niemals wahr, daher wird immer der Code ausgeführt, der vorsichshalber eine Query gegen das ViewObject ausführt. Diese Query ist im Normalfall nicht weiter eingeschränkt (da ja kein Key für die Einschränkung

gefunden wurde). Es wird daher ein Full Scan auf der Datenquelle des View-Objects ausgeführt. Und das nicht nur einmal, sondern für jede Zeile der aktuellen Darstellung der Oberfläche (wir suchen ja schließlich die neu erstellte Zeile). Somit haben wir bei Standardeinstellungen 25-Mal die Ausführung einer uneingeschränkten Query.

Einzige Lösung in der Version 12.2.1 ist das Überschreiben der Standardfunktionalität mit Hilfe des folgenden Codes, welcher die Null-Abfrage erweitert:

```
   @Override

   protected Row[] retrieveByKey(ViewRowSetImpl rs, String keyName, Key
key, int  maxNumOfRows, boolean skipWhere) {

        if (!key.isNull() || keyName != null) {

            return super.retrieveByKey(rs, keyName, key, maxNumOfRows,
skipWhere);

        }

        return new Row[0];

    }
```

Dies löst das beschriebene Problem, sollte bei allen Anwendungsfällen aber auch keine negativen Auswirkungen haben. Eine Alternative Lösungsmöglichkeit wäre wie so häufig eine Aktualisierung auf eine neue JDeveloper-Version, in denen das Problem behoben zu sein scheint.

Statische Listen nach Migration zu 12.1.3 nicht funktionsfähig

Für einfache und kleine Auwahllisten bietet ADF die Möglichkeit statische Listen direkt zur Nutzung anzulegen und die Liste nicht durch die klassischen Methoden (bsp. Database Query) zu akquirieren. Migriert man nun eine Anwendung, welche an manchen Stellen statische Listen verwendet, funktioniert die Migration scheinbar fehlerfrei. Wird die Anwendung gestartet, kann man allerdings sehen, dass alle statische Listen keine Werte mehr besitzen und die Fehlermeldung:

Wert XYZ kann nicht in Werteliste gefunden werden. Setze null.

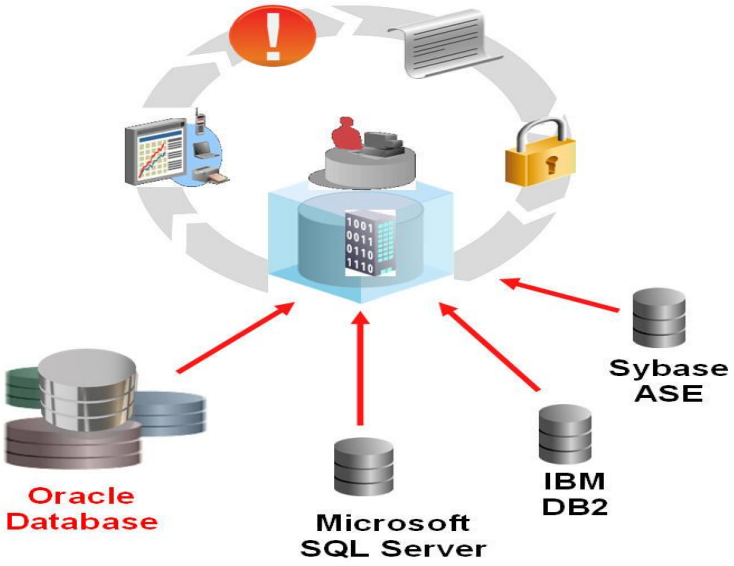

Abb. 1: Statische Liste nach Migration

Die erste Idee lässt einen Fehler in der Definition der statischen Liste vermuten. Nach kurzer Analyse kann man allerdings feststellen, dass diese Komponente fehlerfrei migriert wurde. Glücklicherweise ist ADF in Schichten konzipiert, daher ist die Fehlersuche nun einem strikten Muster zugrunde gelegt.

Wenn man sich das Binding der statischen Liste anschaut, kann man sehen, dass einzelne Parameter im Binding erneuert wurden. Insbesondere wurde das „Standard“ Binding von einer statischen Liste von einem ListOfValues-Binding auf ein abstrakteres ListBinding abgeändert. Wenn man nun per Wizard einfach ein neues ListBinding für das Feld anlegt und die statische Liste aus dem Model übernimmt, so wird automatisch das richtige Binding für die Komponente erstellt. Alternativ kann man einzelne Properties des alten Bindings anpassen (für den autmatischen „Nachmigrations“-Lauf sinnvoll) verändern, so dass die Komponente wieder Daten liefert.

Ui Verschiebungen nach Wechsel auf Alta-UI

Der Wechsel auf Alta UI ist einer der Hauptgründe für den Wechsel der ADF Umgebung. Viele neue Oberflächenkomponenten und die neue User Experience in der Denkweise lassen viele Projekte über einen Wechsel der ADF Version nachdenken.

Eine einfache direkte Migration ist im Normalfall kein Problem, auch wenn bei dieser sicherlich die User Experience nur bedingt verbessert wird. Allerdings kann man gerade bei Formulardarstellung einige verschobene Felder sehen. Daher bietet es sich an, für jede Vererbte Alta Skin einige „Standard“-Erweiterungen zu definieren. Um beispielsweise die Select One Choice Komponenten wieder korrekt in ein unverändertes PanelFormLayout einzupassen, sollte folgender Eintrag in der .css Datei gemacht werden, um alle Komponenten gleichermaßen anzupassen:

```
af|selectOneCjoice .AFPanelFormLayoutContentCell {

                padding-top : 8px;

}

af|selectOneChoice .af|panelFormLayout::label-cell {
```

```
                padding-top: 8px;
}
```

Diese Änderungen sorgen dafür, dass sowohl die Select One Choice als auch dessen label in die Standardabrückung der anderen Form-Layout Komponenten bekommt.

Und es geht weiter…
Zu den oben beschriebenen Problemfällen kommen noch einige dazu, welche in unseren Kundenprojekten und bei anderen ADF Entwicklern für Schweißperlen nach der Migration geführt haben. In dem Vortrag werden weitere (auch weniger technische) Probleme und deren Lösungen bzw. Lösungsansätze beschrieben und sollten für spannende Diskussionen während des Vortrags sorgen.

Kontaktadresse:
Markus Klenke
TEAM GmbH
Hermann Löns Str. 88
D-33104 Paderborn

Telefon:	+49 (0) 5254-8008-55
Fax:	+49 (0) 5254-8008-19
E-Mail	mke@team-pb.de
Internet:	www.team-pb.de
Blog:	http://padora.blogspot.com
Twitter:	@MarkusKlenke