Debugging in APEX

Christina Funke
Apps Associates GmbH
Dortmund

Schlüsselworte
Oracle Application Express, APEX, Debugging, Debugging Collection, Remote Debugging mit SQL-Developer, Error Handling, benutzerfreundliche Fehlermeldungen

Einleitung

Oracle Application Express ist ein Tool, mit dem relativ schnell umfangreiche Webapplikationen programmiert werden können. Leider schleicht sich dabei, wie bei allen Programmierungsarbeiten, schnell mal der ein oder andere Fehler ein. Die Fehlersuche ist dann nicht immer ganz einfach. Dieser Vortag stellt verschiedene Möglichkeiten des Debuggings und der Fehlersuche in APEX vor. Außerdem werden unterschiedliche Varianten zum Erstellen von benutzerfreundlichen und aussagekräftigen Fehlermeldungen vorgestellt, welche die Fehlersuche ebenfalls erheblich erleichtern können.

APEX Debug Mode

Ein nützliches Tool zum debuggen in APEX ist der APEX Debug Mode. Er hilft dem Developer zu verstehen, welche Aktion wann im APEX ausgeführt wird und in welcher Aktion es überhaupt zum Fehler kommt. Um den APEX Debug Mode vollständig nutzen zu können, muss dieser erst unter „Edit Application Properties“ eingeschaltet werden. Wird die Applikation nun gestartet, kann der Debug Mode geöffnet werden, indem erst der Button „Debug“ und anschließend der Button „View Debug“ in der Developer Toolbar betätigt wird. Es öffnet sich folgendes Fenster:

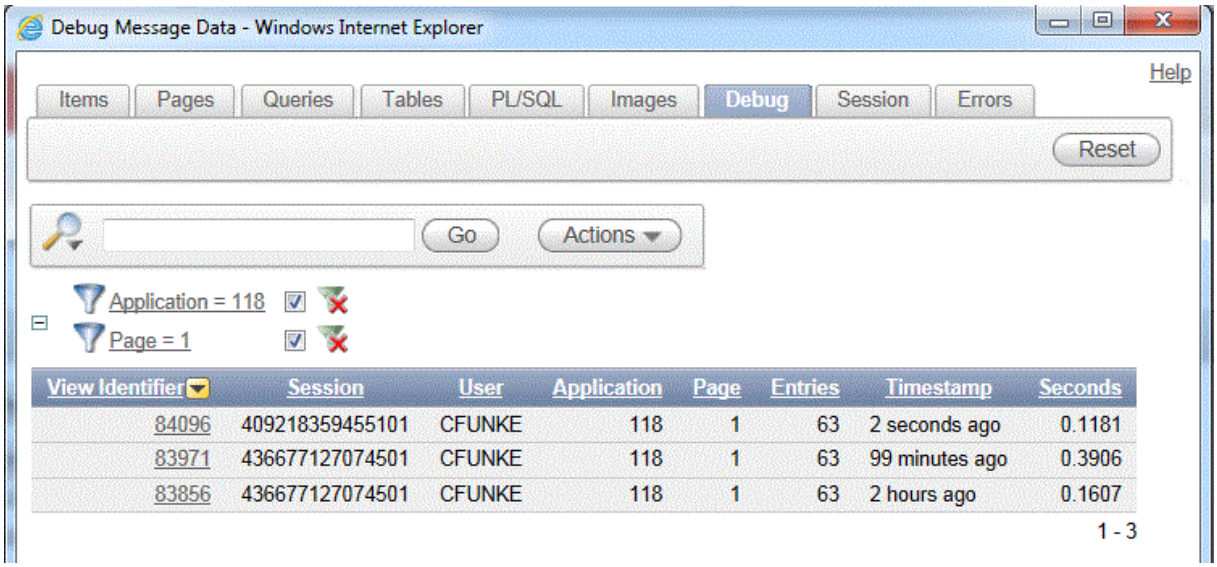

Abb. 1: Developer Toolbar

Zunächst werden einige allgemeine Informationen zu den einzelnen Debug-Durchläufen angezeigt. Hinter der Spalte „View Identifier“ verbirgt sich ein Link, der auf ein neues Fenster verweist. Hier erscheint eine Liste, in der genau steht, welche Computations, Validierungen und Prozesse in, welcher Reihenfolge mit, welchen Werten ausgeführt werden. Mit Hilfe dieser Liste können schnell Aktionen

identifiziert werden, die vielleicht gar nicht oder mit den falschen Werten ausgeführt wurden. Außerdem kann anhand dieser Liste die verbrauchte Zeit, der einzelnen Aktionen abgelesen werden. So können „Ausreißer“ identifiziert werden und ggf. die Performance verbessert werden.

Auch andere Punkte in der Developer Toolbar können bei der Fehlersuche hilfreich sein. Zum Beispiel werden unter dem Punkt Session die aktuellen Werte der einzelnen Page-Items, Apllication Items oder einige Collection-Werte angezeigt. Unter dem Punkt Error wird aufgelistet auf welcher Page welche Fehler wann aufgekommen sind.

Debug Messages
Unter umständen kann es auch sehr nützlich sein, selbst Meldungen in den Debug Mode zu schreiben, um besser nachvollziehen zu können wo es innerhalb eines PL/SQL Codes zum Fehler kommt. Mit dem Befehl *apex_debug.message* ist dies möglich. Dazu müssen die Meldungen die später im Debug Mode sichtbar sein sollen als DBMS_Output Befehl in den PL/SQL Code (z.B. einer Funktion, Prozedur oder Package) geschrieben werden. Anschließend muss nur noch der Process der den PL/SQL Code aufruft folgendermaßen angepasst werden:

```
declare
  v_zeilen dbms_output.chararr;
  v_anzahl number  := 100;
begin
  --DBMS_OUTPUT aktivieren
  dbms_output.enable;

  --eigentliche Procedure ausführen
  gehaltserhöhung;

  --Meldungszeilen aus dem Puffer abrufen
  dbms_output.get_lines(v_zeilen, v_anzahl);

  --Meldungszeilen als APEX Debugging Meldung ausgeben
  for i in 1..v_anzahl loop
    apex_debug.message(v_zeilen(i));
  end loop;
end;
```

Error Handling und benutzerfreundliche Fehlermeldungen
Aussagekräftige Fehlermeldungen können die Fehlersuche ebenfalls erheblich vereinfachen. Ein Entwickler kann, wenn es sich zum Beispiel um ein Problem mit einem Speicherprozess handelt, mit der folgenden Fehlermeldung das Problem wesentlich schneller identifizieren und reagieren, als mit einer allgemeinen Prozess Error Meldung wie „Die Daten konnten nicht gespeichert werden.“

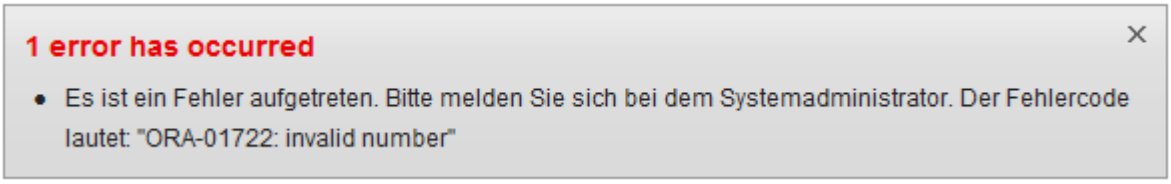

Abb. 2: Aussagekräftige Fehlermeldung

Natürlich gibt es mehrere Möglichkeiten eine solche Fehlermeldung zu erstellen. Eine Möglichkeit wäre, ein Exception Handling in den PL/SQL-Code des Prozesses zu definieren. Das könnte zum Beispiel so aussehen:

```
…
EXCEPTION WHEN OTHERS THEN
  raise_application_error(-20101,'Es ist ein Fehler aufgetreten.
                       Bitte melden Sie sich beim Systemadministrator.
                       Der Fehlercode lautet: ' || substr(SQLERRM, 1,3500));
```

Bei diesem Vorgehen darf keine Process Error Message definiert werden, da in dem Fall nur diese angezeigt wird und der User die Meldung, die im Exception Handling definiert wurde nicht sehen wird. Außerdem hat diese Variante den Nachteil, dass das gleiche oder ein sehr ähnliches Exception Handling für jeden Prozess der Applikation geschrieben werden müsste.

Diese redundante Arbeit lässt sich vermeiden, indem eine Funktion erstellt wird, in der für verschiedene Fehlertypen benutzerfreundliche und aussagekräftige Messages definiert werden. (Ein Beispiel für eine solche Funktion findet sich auf der Oracle Seite. Unter http://docs.oracle.com/cd/E23903_01/doc/doc.41/e21676/apex_error.htm). Der Name dieser Funktion wird anschließend unter Edit Application Properties → Error Handling → Error Handling Function hinterlegt.

Debugging mit APEX Collections

APEX Collections werden verwendet um mehrere Datensätze in einer Session zwischen zu speichern. In dieser spezifischen Session können die Daten angesprochen, geändert und verarbeitet werden. Aber eben nur in dieser spezifischen Session, in einer anderen Session sind die Inhalte schon nicht mehr sichtbar. Das macht das Debugging von Applikationen, die Collections verwenden so schwierig. Um sich die Inhalte einer Collection doch über SQL*Plus, dem SQL Developer oder aus einer anderen APEX-Sitzung heraus ansehen zu können, ist es hilfreich sich die Definition der View APEX_COLLECTIONS anzusehen:

```
select
  c.collection_name,
  m.seq_id, m.c001, m.c002, m.c003, m.c004, m.c005, m.c006, m.c007,
  m.c008, m.c009, m.c010, m.c011, m.c012, m.c013, m.c014, m.c015,
  m.c016, m.c017, m.c018, m.c019, m.c020, m.c021, m.c022, m.c023,
  m.c024, m.c025, m.c026, m.c027, m.c028, m.c029, m.c030, m.c031,
  m.c032, m.c033, m.c034, m.c035, m.c036, m.c037, m.c038, m.c039,
  m.c040, m.c041, m.c042, m.c043, m.c044, m.c045, m.c046, m.c047,
  m.c048, m.c049, m.c050, m.clob001, m.md5_original
from wwv_flow_collections$ c, wwv_flow_collection_members$ m
where c.id = m.collection_id
    and c.session_id = (select v('SESSION') from dual)
    and c.security_group_id = (select wwv_flow.get_sgid from dual)
    and c.flow_id = (select nv('FLOW_ID') from dual)
```

Die WHERE-Bedingung verhindert also, dass die Inhalte der Collection beispielsweise im SQL-Developer angezeigt werden. Um sich die Inhalte trotzdem anzeigen zu lassen, müssen die Werte für die Session-ID (wwv_flow.g_instance), die Workspace-ID (wwv_flow_api.set_security_group_id) und die Applikation-ID (wwv_flow.g_flow_id) richtig gesetzt werden. Die Werte der Session-ID und der Applikation-ID können aus der URL im Browser entnommen werden. Die Workspace-ID kann aus der View APEX_WORKSPACES ausgelesen werden. Anschließend können die Inhalte der Collection mit einem Select auf die View APEX_COLLECTIONS angezeigt werden.

Remote Debugging mit dem SQL-Developer

Mit Remote Debugging kann PL/SQL-Code, der in einer Application Express Anwendung ausgeführt wird im SQL Developer getestet werden. Dabei wird der Quellcode aus APEX heraus gestartet. Das Debugging findet dann im SQL Developer statt.

Damit das Remote Debugging mit dem SQL Developer auch funktioniert, muss dem Parsing Schema, in dem die APEX Applikation läuft, zunächst noch das Systemprivileg DEBUG CONNECT SESSION erteilt werden. Das Schema, unter dem die Application Express Sessions ablaufen (in der Regel APEX_PUBLIC_USER) benötigt das Privileg DEBUG ANY PROCEDURE. Das DEBUG-Privileg kann natürlich auch an nur einzelne Prozeduren vergeben werden (GRANT DEBUG ON COMMUNITY.MY_PROCEDURE to APEX_PUBLIC_USER). Außerdem ist zu beachten, dass PL/SQL-Code, der direkt als Prozesstext in Application Express hinterlegt wurde, nicht mit dem SQL Developer gedebuggt werden kann. Der Code muss dazu als PL/SQL-Funktion, -Prozedur oder -Package hinterlegt werden. Dieses Vorgehen ist sowieso zu empfehlen.

Wurden die Rechte an die entsprechenden Schemata erteilt, kann die PL/SQL-Funktion, -Prozedur oder das Package ganz normal erstellt und kompiliert werden. Anschließend muss ein Breakpoint definiert werden und die Logik noch einmal für Debug kompiliert werden.

Abb. 3: Screenshot „Für Debug kompilieren“

Nun wird der Remote Debugger eingerichtet. Dazu rechtsklick auf die Verbindung und auf „Remote Debugging…“ klicken. Es öffnet sich ein Dialogfenster. Hier wird ein freier TCP/IP-Port und die IP-Adresse des Rechners auf dem der SQL Developer läuft eingetragen. Damit APEX den SQL Developer über das Netzwerk kontaktieren kann, muss der PL/SQL-Aufruf wie folgt definiert werden:

```
begin
  dbms_debug_jdwp.connect_tcp('#IP-Adresse', #TCP/IP-Port);
  my_procedure(parameter);
  dbms_debug_jdwp.disconnect;
end;
```

Wird die APEX-Seite nun gestartet und ggf. der Prozess angestoßen wartet der Browsert. Das ist auch richtig so, da der Debugger im SQL Developer am oben definierten Breakpoint stehen geblieben ist. Nun kann der Entwickler den PL/SQL Code schrittweise durchgehen und dabei alle Variablen im Auge behalten. Die Werte der Variablen können auch geändert werden. So kann die Fehlersuche massiv erleichtert und beschleunigt werden.

Fazit

Die Fehlersuche ist bei allen Entwicklungsarbeiten eine lästige und oft auch schwierige Aufgabe. Allerdings biete APEX einige Möglichkeiten die Fehler innerhalb einer Applikation einzugrenzen und auf die Schliche zu kommen.

Sehr hilfreich ist dabei auch die Verwendung von Aussagekräftigen Fehlermeldungen. Selbst wenn nur die Beispiel Funktion von Oracle ins APEX eingebunden wird, können spezielle Fehler mit aussagekräftigen Fehlermeldungen versehen werden. Dies erleichtert die Fehlersuche erheblich.

Besonders in Verbindung mit dem SQL Developer bietet der Remote Debugger eine gute Möglichkeit genau nachzuvollziehen mit welchen Parametern der PL/SQL Code aufgerufen wird und was genau innerhalb des Codes mit den einzelnen Variablen passiert. So kann die relativ einfach herausgefunden werden, an welcher Stelle des Codes genau der Fehler steckt.

Mit diesen hier vorgestellten Werkzeugen und Vorgehensweisen fällt die lästige Aufgabe der Fehlersuche und des Debuggings zwar nicht ganz weg, aber immerhin kann sie doch sehr vereinfacht und so auch stark verkürzt werden.

Kontaktadresse:
Christina Funke
Apps Associates GmbH
Flughafenring 11
D-44319 Dortmund

Telefon:	+49 (231) 22 22 79 - 33
Fax:	+49 (231) 22 22 79 - 43
E-Mail	christina.funke@appsassociates.com
Internet:	www.appsassociates.de