Wo ist der Fehler? – Debugging in Apex

Christina Funke, Apps Associates GmbH

Oracle Application Express ist ein Tool, um relativ schnell umfangreiche Web-Applikationen zu programmieren. Leider schleicht sich dabei schnell mal der eine oder andere Fehler ein. Die Fehlersuche ist dann nicht immer ganz einfach. Dieser Artikel zeigt die verschiedenen Möglichkeiten des Debuggings und der Fehlersuche in Apex. Außerdem werden unterschiedliche Varianten zum Erstellen benutzerfreundlicher und aussagekräftiger Fehlermeldungen vorgestellt, um die Fehlersuche ebenfalls erheblich zu erleichtern.

Jeder, der schon mal eine Apex-Anwendung geschrieben hat, weiß: Die Fehlersuche kann sehr aufwändig sein. Oft weiß man gar nicht, welches Objekt gerade fehlerhaft ist. Ist es ein Prozess, eine Validierung oder eine Region? Wird die Prozedur auch mit den richtigen Werten aufgerufen und welche Daten sind eigentlich in meiner Apex-Collection gespeichert?

Apex Debug Mode

Ein nützliches Werkzeug zum Debuggen in Apex ist der Apex Debug Mode. Er hilft dem Entwickler zu verstehen, wel-

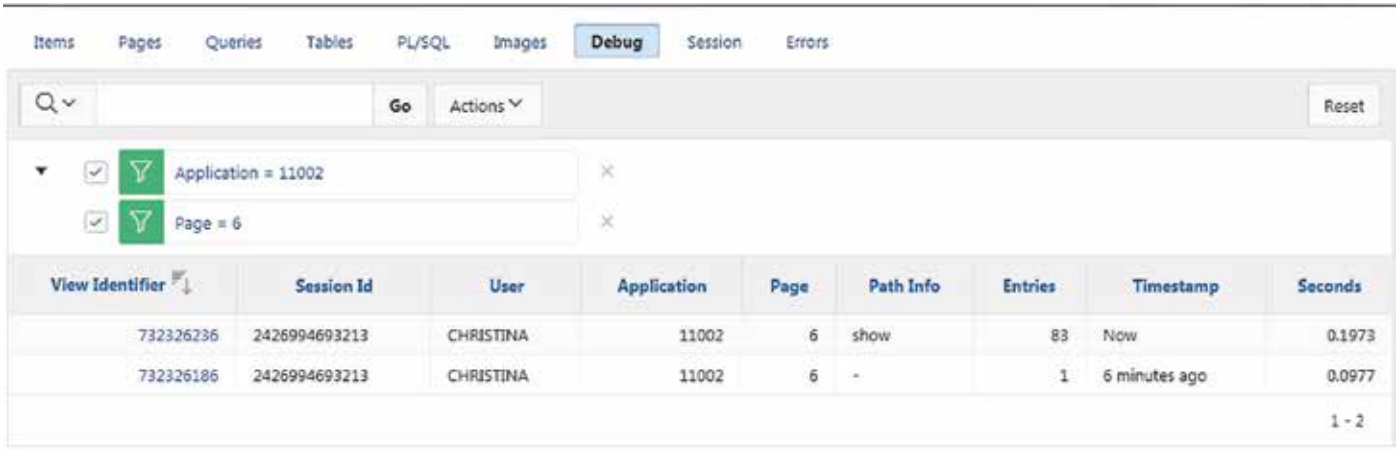

Abbildung 1: Die Developer Toolbar

che Aktion wann in Apex ausgeführt wird und bei welcher Aktion es überhaupt zum Fehler kommt. Um den Apex Debug Mode vollständig nutzen zu können, muss dieser erst unter „Edit Application Properties" aktiviert sein. Wird die Applikation nun gestartet, kann der Debug Mode geöffnet werden, indem man den Button „Debug" und anschließend den Button „View Debug" in der Developer Toolbar betätigt. *Abbildung 1* zeigt das sich öffnende Fenster.

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. Dort erscheint eine Liste, in der genau steht, welche Computations, Validierungen und Prozesse in welcher Reihenfolge mit welchen Werten ausgeführt werden. Mithilfe dieser Liste lassen sich schnell Aktionen identifizieren, die vielleicht nicht oder mit den falschen Werten ausgeführt wurden. Außerdem ist die verbrauchte Zeit der einzelnen Aktionen ablesbar. So können Ausreißer identifiziert und gegebenenfalls 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 dargestellt. Auch die Werte von Application Items (globale Variablen) können hier angezeigt und kontrolliert werden. Es werden sogar einige Collection-Werte angezeigt. Unter dem Punkt „Error" ist aufgelistet, auf welcher Page welche Fehler wann vorgekommen 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 (etwa einer Funktion, Prozedur oder Package) geschrieben werden. Anschließend muss man nur noch den Prozess anpassen, der den PL/SQL-Code aufruft. *Listing 1* zeigt ein Beispiel. Es ist natürlich auch möglich, die selbst erstellten Meldungen in einer Datenbank-Tabelle oder in einer Apex-Collection zu speichern und sie anschließend als Report in Apex sichtbar zu machen.

Error Handling und benutzerfreundliche Fehlermeldungen

Aussagekräftige Fehlermeldungen können die Fehlersuche ebenfalls erheblich vereinfachen. Die typische Prozess-Error-Message „Die Daten konnten nicht gespeichert werden" ist viel zu allgemein, als dass sie einem Entwickler bei der Fehlersuche helfen könnte. Warum die Daten nicht gespeichert werden konnten, geht aus so einer Fehlermeldung nicht hervor. Mit der Fehlermeldung „ORA-01722: invalid number" dagegen ist einem Entwickler schon sehr geholfen. Diese Fehlermeldung ist allerdings für einen Endanwender der Applikation absolut unverständlich und frustrierend. Besser wäre hier eine Fehlermeldung wie in *Abbildung 2*.

```
DECLARE
   v_zeilen dbms_output.chararr;
   v_anzahl number := 100;
BEGIN
   --DBMS_OUTPUT aktivieren
   dbms_output.enable;

   --eigentliche Procedure ausführen
   my_procedure(parameter1, parameter 2);

   --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;
```

Listing 1

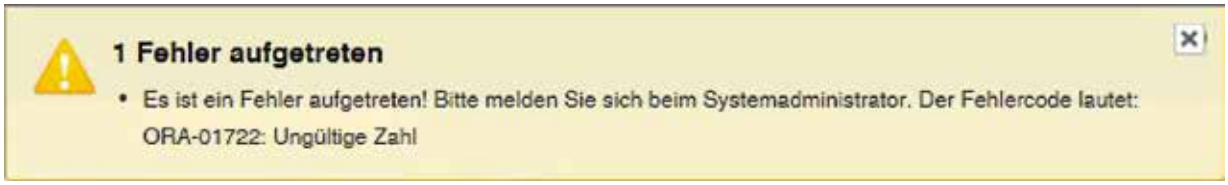

Abbildung 2: Aussagekräftige Fehlermeldung

Diese ist für den Endanwender verständlich und gibt dem Entwickler der Anwendung gleichzeitig einen Hinweis darauf, warum die Daten nicht gespeichert werden konnten. Natürlich gibt es mehrere Alternativen, eine solche Fehlermeldung zu erstellen. Eine Möglichkeit wäre, ein Exception Handling in dem PL/SQL-Code des Prozesses zu definieren. *Listing 2* zeigt auch dazu ein Beispiel.

Bei diesem Vorgehen darf keine Process Error Message definiert sein, 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 Fehler-Typen benutzerfreundliche und aussagekräftige Messages definiert sind. Ein Beispiel für eine solche Funktion findet sich auf der Oracle-Seite *„http://docs.oracle.com/cd/E23903_01/doc/doc.41/e21676/apex_error.htm"*. Der Name dieser Funktion ist anschließend unter „Edit Application Properties", „Error Handling" und „Error Handling Function" hinterlegt.

Debugging mit Apex Collections

Apex Collections werden verwendet, um mehrere Datensätze in einer Session zwischenzuspeichern. In dieser spezifischen Session lassen sich die Daten ansprechen, ändern und verarbeiten. 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, den SQL Developer oder aus einer anderen Apex-Sitzung heraus ansehen zu können, ist es hilfreich, sich die Definition der View „Apex_COLLECTIONS" anzusehen *(siehe Listing 3)*.

Die „WHERE"-Bedingung verhindert, 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 Application-ID („wwv_flow.g_flow_id") richtig gesetzt sein. Die Werte der Session-ID und der Applikation-ID können der URL im Browser entnommen und die Workspace-ID aus der View „Apex_WORKSPACES" ausgelesen werden. Anschließend lassen sich die Inhalte der Collection mit einem Select auf die View „Apex_COLLECTIONS" anzeigen.

Eine andere Möglichkeit, den Inhalt einer Apex-Collection zu betrachten, ist, eine View zu erzeugen, die nicht auf die Session-ID, die Workspace-ID und die Application-ID begrenzt ist. In diesem Fall sollte die View einen anderen Namen bekommen und vor unberechtigten Zugriffen geschützt werden.

```
…
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));
```

Listing 2

```
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)
```

Listing 3

Remote Debugging mit dem SQL Developer

Mit Remote Debugging lässt sich PL/SQL-Code, der in einer Apex-Anwendung ausgeführt wird, im SQL Developer testen. 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 System-

Privileg „DEBUG CONNECT SESSION“ erteilt werden. Das Schema, unter dem die Apex-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 Prozess-Text in Apex hinterlegt wurde, nicht mit dem SQL Developer überprüft werden kann. Der Code muss dazu als PL/SQL-Funktion, -Prozedur oder -Package hinterlegt sein. Dieses Vorgehen ist sowieso zu empfehlen.

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

Nun wird der Remote Debugger eingerichtet. Dazu braucht es einen Rechtsklick auf die Verbindung und einen Klick auf „Remote Debugging …“. Es öffnet sich ein Dialog-Fenster. Hier werden 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 definiert sein *(siehe Listing 4)*.

Wird die Apex-Seite nun gestartet und gegebenenfalls der Prozess angestoßen, wartet der Browser. 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. Das erleichtert und beschleunigt die Fehlersuche massiv.

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

Listing 4

Fazit

Die Fehlersuche ist bei allen Entwicklungsarbeiten eine lästige und oft auch schwierige Aufgabe. Allerdings bietet Apex einige Möglichkeiten, die Fehler innerhalb einer Applikation einzugrenzen und ihnen auf die Schliche zu kommen. Sehr hilfreich ist dabei auch die Verwendung von aussagekräftigen Fehlermeldungen. Selbst wenn nur die Beispiel-Funktion von Oracle in 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 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 man sie doch sehr vereinfachen und so auch stark verkürzen.

Christina Funke
christina.funke@appsassociates.com

Starker Dollar setzt Oracle weiter zu

DOAG Online

Im dritten Geschäftsquartal hat Oracle erneut Schwächen im traditionellen Geschäft mit dem Verkauf von Lizenzen und Wartung gezeigt, die auch ein Umsatzanstieg im Cloud-Geschäft nicht überwinden konnte. Wie bereits in den vorherigen Quartalen belastete der starke Dollar die Zahlen zusätzlich.

Wie Oracle zum Ende seines dritten Geschäftsquartals mitgeteilt hat, fiel der Umsatz des Unternehmens um 3,4 Prozent auf neun Milliarden US-Dollar. Auch der Nettogewinn ging um 14 Prozent auf 2,14 Milliarden US-Dollar zurück. Grund dafür sind neben des zurzeit starken Dollars vor allem die stark rückläufigen Verkäufe von Software-Lizenzen und Wartung: Die Umsätze im Kerngeschäft sanken um vier Prozent auf 6,3 Milliarden US-Dollar, während sich die Umsatzeinbußen in der Hardware- und Servicesparte auf jeweils 13 Prozent (1,1 Milliarden US-Dollar) beziehungsweise sieben Prozent (793 Millionen US-Dollar) beliefen.

Oracle versucht weiterhin, die Einbußen im traditionellen Geschäft durch Zuwächse im Cloud-Bereich wettzumachen. Trotz des allgemeinen Umsatz- und Gewinnrückgangs zeigen sich die Oracle CEOs Safra Catz und Mark Hurd zuversichtlich. Letzterer betont insbesondere den enorm gewachsenen Bruttoumsatz im Saas- und PaaS-Bereich um 96 Prozent. Außerdem hätte man im dritten Quartal 942 neue SaaS-Kunden gewonnen. Laut Larry Ellison, Oracle-Vorsitzender und CTO, wachse das Unternehmen in diesem Bereich außerdem weit schneller als die Konkurrenz.