Logging und Debugging

Gerd Volberg

OPITZ CONSULTING Deutschland GmbH

Bochum, 20. Februar 2014

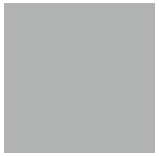
Agenda

1. Logging

- Datenmodell
  - Tabellen
  - Views

2. Debugging

- Debug-Package
  - Funktionen
  - Prozeduren
- Ein erster Test
- Best Practices



Logging

Datenmodell

Um Applikationen zu überwachen und diese Daten auf einfachste Art und Weise speichern zu können, benötigen wir eine Tabelle namens Logging und eine Sequence für den Primary Key

```
CREATE TABLE Logging (
  ID                            NUMBER(8,0) NOT NULL,
  SESSION_ID                    NUMBER(8,0),
  INSERT_DATE                   DATE NOT NULL,
  TEXT                          VARCHAR2(2000) NOT NULL);


CREATE SEQUENCE Logging_SEQ;
```

Datenmodell

Über diese Tabelle legen wir eine View, die die Daten der Tabelle 1:1 selektiert und sortiert

```
CREATE OR REPLACE VIEW V_Logging_desc
      (ID, SESSION_ID, INSERT_DATE, TEXT) AS
SELECT ID, SESSION_ID, INSERT_DATE, TEXT
  FROM Logging
 ORDER BY SESSION_ID DESC, ID DESC;
```

Die Sortierung ist wichtig, damit im SQL Developer oder TOAD die Analyse der Logging-Daten so einfach wie möglich ist.

Debugging

Debug-Package

Damit Fehlermeldungen und Debuginformationen protokolliert werden können benötigen wir ein Package pk_Debug

```
CREATE OR REPLACE PACKAGE PK_Debug IS
  FUNCTION  Debug_allowed RETURN BOOLEAN;
  FUNCTION  Next_ID       RETURN NUMBER;
  PROCEDURE Disable;
  PROCEDURE Enable;
  PROCEDURE Destroy;
  PROCEDURE Init  (P_Debug_allowed IN BOOLEAN DEFAULT TRUE);
  PROCEDURE Write (P_Text          IN VARCHAR2,
                   P_Session_ID    IN NUMBER  DEFAULT NULL);
  G_Debug_allowed BOOLEAN := TRUE;
  G_Session_ID    NUMBER;
END;
```

Funktionen

pk_Debug stellt folgende Funktionen zur Verfügung:

- **Debug_Allowed – TRUE, wenn Debugging enabled ist**
- **Next_ID – Gibt die nächste Sequence zurück**

```
CREATE OR REPLACE PACKAGE BODY PK_Debug IS

FUNCTION Debug_allowed RETURN BOOLEAN IS
BEGIN
  RETURN (G_Debug_allowed);
END;
FUNCTION Next_ID RETURN NUMBER IS
  V_ID NUMBER;
BEGIN
  SELECT Logging_SEQ.nextval INTO V_ID FROM DUAL;
  RETURN (V_ID);
END;
```

Prozeduren

Wenn man grössere Teile Sourcecodes untersucht, kann es zwischendurch interessant sein, dass man Teile, die schon untersucht wurden, vom Debugging ausschliesst. Dazu benötigen wir folgende Prozeduren

```
PROCEDURE Disable IS
BEGIN
  G_Debug_allowed := FALSE;
END;

PROCEDURE Enable IS
BEGIN
  G_Debug_allowed := TRUE;
END;
```

Prozeduren

Das Debugging kann initialisiert und destroyed werden:

```
PROCEDURE Init (
  P_Debug_allowed IN BOOLEAN DEFAULT TRUE) IS
BEGIN
  G_Debug_allowed := P_Debug_allowed;
  G_Session_ID    := Next_ID;
  Write ('--start ' || to_char (G_Session_ID)
                            || '----------------------');
END;
PROCEDURE Destroy IS
BEGIN
  Write ('----------------------stopp '
                         || to_char (G_Session_ID) || '--');
  G_Session_ID := NULL;
END;
```

Prozeduren

Die wichtigste Routine ist Write, mit der man einzelne Logging-Datensätze erzeugt

```
PROCEDURE Write (
  P_Text       IN VARCHAR2,
  P_Session_ID IN NUMBER DEFAULT NULL) IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  IF Debug_allowed THEN
    IF G_Session_ID IS NULL THEN Init; END IF;
    INSERT INTO Logging (ID, Session_ID, Insert_Date, Text)
    VALUES (Next_ID, NVL (P_Session_ID, G_Session_ID),
            Sysdate, P_Text);
    COMMIT;
  END IF;
END;
END;
/
```

Ein erster Test

In diesem Beispiel wird eine Variable V_Test untersucht und ins Logging geschrieben

```
pk_Debug.Init;
pk_Debug.Write ('Hello World: ' || V_Test);
pk_Debug.Destroy;
```

Das Ergebnis in der Loggingtabelle sieht dann so aus

```
ID Session Insert-Date     Text
=========================================
24    21   10.09.-12:38:48 ------------------------stopp 21--
23    21   10.09.-12:38:48 Hello World: 42
22    21   10.09.-12:38:48 --start 21-----------------------
```

Best Practices

Wenn man in einer Routine Fehler sucht, sollte man jeden Bereich, den man autark untersuchen möchte, mit pk_debug.Init beginnen und mit Destroy enden lassen

Wenn die Debug-Prozeduren im Sourcecode dauerhaft enthalten sein sollen, kann man das Debugging mit pk_Debug.Disable unterbrechen und mit Enable wieder fortführen.

Beim Start kann die Routine Init automatisch enabled oder disabled starten.

Just use it !

Download

Das Package inkl. der DDL-Skripte zusammen mit dieser Präsentation sind ab Freitag, 21.02.2014 unter folgender URL zu finden

http://code.google.com/p/forms-framework/

Dort dann einfach auf "Downloads" klicken

Ihr Ansprechpartner

Gerd Volberg

Solution Architect

OPITZ CONSULTING Deutschland GmbH
Kirchstr. 6, 51647 Gummersbach
Tel. +49 (2261) 60 01-0
gerd.volberg@opitz-consulting.com

talk2gerd.blogspot.com