Saubere Arbeit!
Fehlervermeidung und Debugging in APEX

Andreas Wismann
WHEN OTHERS
D-41564 Kaarst

Schlüsselworte
APEX, Oracle Application Express, Fehlervermeidung, Debugging

Einleitung
Das „Express" in Oracle Application Express verführt bisweilen dazu, schnell loszulegen, schnell fertigzustellen und schnell (und wenig) zu testen. Wohl unter der Annahme, bei APEX handele es sich um ein Entwicklungssystem der Sorte „What You See Is What You Get", wird vom optischen Eindruck auf die Gesamtqualität der Anwendung geschlossen.
In Wirklichkeit müssen bei der Programmierung mit Application Express selbstverständlich die gleichen Qualitätsmaßstäbe angelegt werden wie bei jeder anderen Entwicklungsumgebung auch. Das gilt vor allem, weil APEX einen vergleichsweise leichten Einstieg mit einer (anfangs) flachen Lernkurve und entsprechenden Erfolgserlebnissen bietet. Oft wird – ohne Rücksicht auf die spätere Wartbarkeit – der schnellstmögliche Weg bei der Programmierung eingeschlagen und die Zielerreichung durch Soll/Ist-Vergleich der Bildschirmdarstellung verifiziert. Bei solcher Vorgehensweise drohen APEX-Anwendungen zu entstehen, deren Zuverlässigkeit mit dem Qualitätssiegel „Komisch – gestern hat es noch funktioniert" beschrieben werden kann…
Dieses Manuskript und der dazugehörige Vortrag beschreiben zwei Themenbereiche, um robuste APEX-Anwendungen zu erstellen:

- Fehlervermeidung:
  Best Practices, die Sie bei der Entwicklung mit Application Express berücksichtigen können, um wartungsfreundliche und robuste Anwendungen zu entwickeln
- Debugging:
  Methoden und Tools für die Überprüfung der Arbeitsergebnisse sowie die gezielte Fehlersuche.

Diejenigen Aspekte, auf die im Vortrag nur kurz eingegangen wird, werden in diesem Manuskript ausführlicher beleuchtet – und umgekehrt. Diese Zusammenstellung erhebt – selbstredend – keinen Anspruch auf Vollständigkeit. Sie beleuchtet einige vorteilhafte Vorgehensweisen, die erfahrungsgemäß in vielen Projekten ungenutzt bleiben.

Fehlervermeidung beginnt mit der Planung
APEX besitzt keine Drei-Schichten-Architektur, sondern erlaubt dem Programmierer direkten Zugriff auf die Datenbank. Darin ähnelt es übrigens Oracle Forms, und die Erfahrung lehrt, dass dies Risiken birgt. Eine der wichtigsten Überlegungen ist, an welchen Stellen Sie Ihre Geschäftslogik unterbringen wollen.

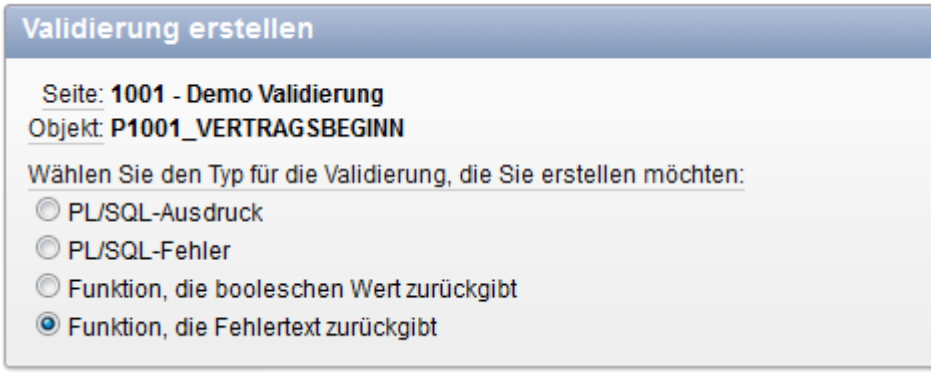
Empfehlung Nr. 1:
Validieren Sie Ihre Geschäftslogik-Regeln an zentraler Stelle.
Für die reine Format-Überprüfung von Items (auf gültiges Datum, gültiges Zahlenformat etc.) können Sie zunächst bequemerweise die standardmäßigen APEX-Validierungen („Validations") als triviale Vorab-Prüfung verwenden. Auf keinen Fall jedoch sollten diese Validierungen mit Geschäftslogik

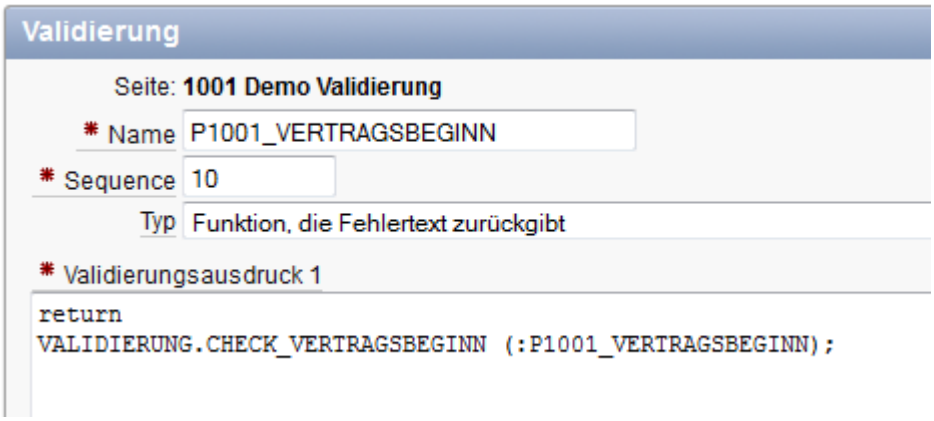
belastet werden. Ob die Maske vom Benutzer wirklich „stimmig“ im Sinne des Geschäftsprozesses ausgefüllt wurde, das prüfen Sie am besten in eigens dafür geschriebenen Funktionen, die sie in ein Validierungs-Package auslagern (dieses Package wird im Folgenden VALIDIERUNG genannt).
Am Beispiel eines Feldes namens „Vertragsbeginn“:
Zunächst schreiben Sie eine Prüfroutine im Package VALIDIERUNG (diese kann zunächst „einfach“ gehalten und im Laufe der Zeit verfeinert werden).

```
PACKAGE BODY VALIDIERUNG
IS
  FUNCTION check_vertragsbeginn (i_datum IN DATE)
  RETURN VARCHAR2 IS
    v_fehlermeldung VARCHAR2(4000);
  BEGIN
    IF i_datum < trunc(SYSDATE)
       OR extract(DAY FROM i_datum) != 1
    THEN
      v_fehlermeldung := 'Der Vertrag kann nur an einem Monatsersten '
                      || 'und nicht rückwirkend beginnen';
    END IF;
    -- weitere Prüfungen ...
    RETURN v_fehlermeldung;
  EXCEPTION
    WHEN OTHERS THEN
      RETURN 'Vertragsbeginn nicht prüfbar: ' || SQLERRM;
             -- nur als Beispiel
  END check_vertragsbeginn;
END VALIDIERUNG;
```

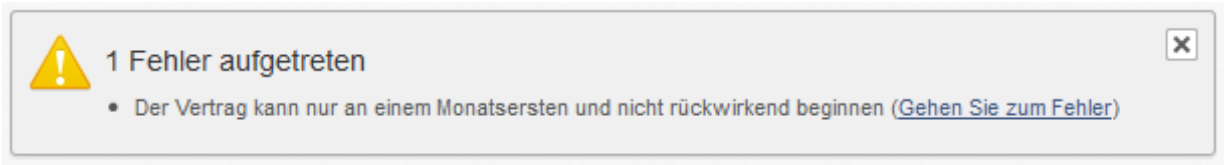
Die Package-Routine prüft, ob die geschäftlichen Rahmenbedingungen eingehalten wurden. Sie gibt im Erfolgsfall einen Leerstring (NULL) zurück und im Fehlerfall einen Hinweis für den Benutzer („Der Vertrag kann nur an einem Monatsersten und nicht rückwirkend beginnen“).

Formulieren Sie für das Vertragsbeginn-Feld eine APEX-Validation vom Typ PL/SQL, die eine Prüfroutine namens VALIDIERUNG.CHECK_VERTRAGSBEGINN aufruft.





Den Rückgabewert, wenn er nicht leer ist, zeigt APEX automatisch im Kopf der Seite an:



Worin liegt der Vorteil?
Sie haben die Geschäftslogik zentralisiert und müssen die Prüfung auf einer zweiten APEX-Seite, die ebenfalls die Eingabe des Vertragsbeginns erlaubt, nicht erneut programmieren. Dadurch vermeiden Sie Redundanz. Den „formellen“ Teil (Überprüfung auf gültiges Datumsformat) überlassen Sie APEX, indem Sie das Eingabefeld als Datumsfeld deklarieren. Sie können nun die Gültigkeit Ihrer eigenen Validierung auch ohne APEX-Maske testen, indem Sie automatisierte Tests gegen die Geschäftsfunktion schreiben (siehe dazu weiter unten).

Empfehlung Nr. 2:
Verlegen Sie so viel Code wie möglich in die Datenbank.
Die zahlreichen Stellen, an denen Ihnen APEX die Eingabe von SQL- und PL/SQL-Programmtexten ermöglicht, sollten Sie mit Bedacht nutzen. Code, den Sie hier eingeben, wird genau zwei Mal auf Korrektheit geprüft: Bei der Speicherung durch den Entwickler und unmittelbar vor der Ausführung durch den Benutzer.

Mit anderen Worten: Während der Entwicklung (die möglicherweise Wochen dauert) besitzt der APEX-interne Programmcode keinen Status (VALID, INVALID) mehr, wie Sie es vielleicht noch von Oracle Forms kennen. Wenn Sie vergessen, relevante Änderungen nachzupflegen (beispielsweise die geänderten Spaltennamen einer Tabelle), dann führt das unweigerlich zu einem Laufzeitfehler, der die Ausführung Ihrer APEX-Seite verhindert.
Als Abhilfe sollten Sie anonyme Codeblöcke grundsätzlich als benannte Objekte (Functions, Procedures, Packages) in der Datenbank speichern. Aus den APEX-Fenstern heraus rufen Sie diese Methoden mit Hilfe weniger, kurzer Befehle auf. Natürlich Entbindet Sie das nicht von einer gewissen Sorgfalt.

Worin liegt der Vorteil?
So minimieren Sie die Anzahl der Stellen, wo Änderungen an Datenbankobjekten ihren Programmcode (stillschweigend) ungültig machen können. Den Programmcode in der Datenbank können Sie im Data Dictionary jederzeit auf Gültigkeit prüfen.

Empfehlung Nr. 3:
Verwenden Sie Views anstatt umfangreicher SQL-Statements in APEX.
Wer kennt sie nicht – seitenlange SQL-Statements als Basis für APEX-Reports, in denen womöglich auch noch die Inhalte (Tabellenspalten) und die Präsentationslogik (HTML- oder CSS-Auszeichnungen) kreuz und quer vermischt werden. Aus demselben Grund wie bei Empfehlung Nr. 2 sollten Sie alles, was komplexer ist als eine flache SQL-Abfrage (angefangen bei NVL, DECODE, CASE, SUBSELECTs über JOINS, Inline Views, GROUP BY-Klauseln, CONNECT BY usw. usw), als View in der Datenbank definieren. In den APEX-Reports rufen Sie dann lediglich die Spalten „FROM View“ auf und fügen gegebenenfalls noch die passende WHERE-Bedingung anhand der entsprechenden Seitenelemente hinzu.

Worin liegt der Vorteil?
Mit der Abfrage

```
select *
  from user_objects
 where object_type = 'VIEW'
  and status = 'INVALID'
```

erfahren Sie auf einen Blick, ob Ihre APEX-Berichte ausgeführt werden können, ohne die Seite in APEX zwingend nachtesten zu müssen. Eventuelle Präsentationslogik-Bestandteile können bei Bedarf im APEX-Report hinzugefügt werden, so dass sie auf diese Weise sauber von den Daten getrennt bleiben.

Empfehlung Nr. 4:
Setzen Sie Shortcuts ein, wenn sich Inhalte wiederholen.
Bei den Gemeinsamen Komponenten („Shared Components“) finden Sie die Shortcuts, die glücklicherweise in der eingedeutschten APEX-Version nicht dem Übersetzungswahn zum Opfer gefallen sind. Wann immer Sie einen Text mehrfach verwenden müssen (beispielsweise die berühmten „rechtlichen Hinweise“ auf einer Webseite), sollten Sie ihn als Shortcut hinterlegen. Um den Shortcut aufzurufen, definieren Sie die Region als „HTML mit Shortcuts“ und benutzen den Namen des Shortcuts in einfachen doppelten Anführungszeichen ("MEIN SHORTCUT"). Sie können Shortcut-Text mit normalem Text und Variablenreferenzen (&MEINE_VARIABLE.) mischen.

Worin liegt der Vorteil?
Wird der Text im Shortcut geändert, werden alle Stellen, an denen er referenziert wird, automatisch aktualisiert. So vermeiden Sie Redundanzen und Inkonsistenzen. Shortcuts können sogar dynamische PL/SQL-Inhalte wiedergeben (zum Beispiel die aktuelle Uhrzeit zum Zeitpunkt des Seitenaufrufes oder die Inhalte beliebiger Items), ohne dafür ein Hidden Item anlegen zu müssen und ohne dass die Region, in welcher der Shortcut verwendet wird, vom Typ PL/SQL sein muss.

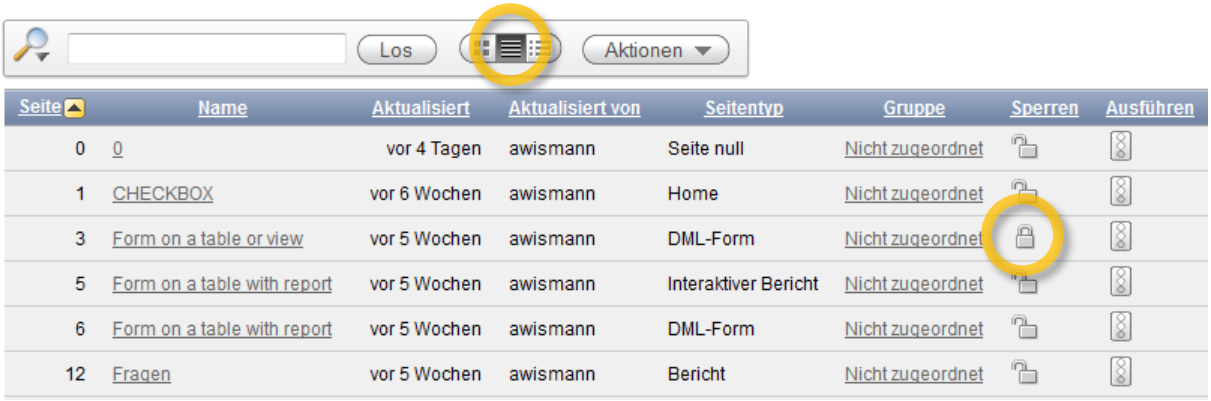
Empfehlung Nr. 5:
Löschen Sie Templates, die Sie nicht verwenden.
Besonders im Team kommt es vor, dass Entwickler Templates verwenden, die „eigentlich" nicht in das Repertoire der Anwendung gehören. Ein fabrikneues APEX-Theme beinhaltet eine Vielzahl verschiedener Templates für alle möglichen Anwendungsfälle. Einige dieser Templates ähneln sich sehr. Deshalb wird man den Unterschied anhand des Erscheinungsbildes der Seite nicht auf Anhieb bemerken. Wenn zu einem späteren Zeitpunkt beispielsweise ein Schaltflächen-Template geändert wird, bleiben womöglich die „Ausreißer" unbemerkt. Nämlich diejenigen Schaltflächen, die mit einem gleich aussehenden, aber funktional abweichenden Template programmiert wurden.
Bevor es soweit kommt, werfen Sie einen Blick unter „Gemeinsame Komponenten > Templates" und prüfen Sie anhand der zahlenmäßig angegebenen (und detailliert verlinkten!) Referenzen, ob Sie solche Ausreißer finden. Wenn Sie unsicher sind und nicht verwendete Templates lieber nicht löschen möchten, dann benennen Sie sie wenigstens um (etwa mit einem Fragezeichen als Präfix). Ihre Entwickler sehen dann auf einen Blick, woran sie sind. Um systematisch nach falsch verwendeten Templates beispielsweise für Items zu suchen, hilft Ihnen dann die Abfrage

```
SELECT *
  FROM APEX APPLICATION PAGE ITEMS
 WHERE item_label_template LIKE '?%';
```

Nicht nur Page Items, sondern auch andere Objekttypen lassen sich mit Hilfe von Views abfragen, die das verwendete Template nennen (siehe „Application Builder > Utilities > Application Express Views").

Worin liegt der Vorteil?
Sie vereinfachen die Entwicklung, indem Sie Wildwuchs bei der Erstellung von Regionen, Items, Reports etc. vermeiden. Sie gehen bei Template-Änderungen kein unnötiges Risiko ein. Ihr Nachtest-Aufwand reduziert sich auf ein Minimum. Bedenken Sie, dass nicht alle Objekte notwendigerweise zu jeder Zeit sichtbar sind.

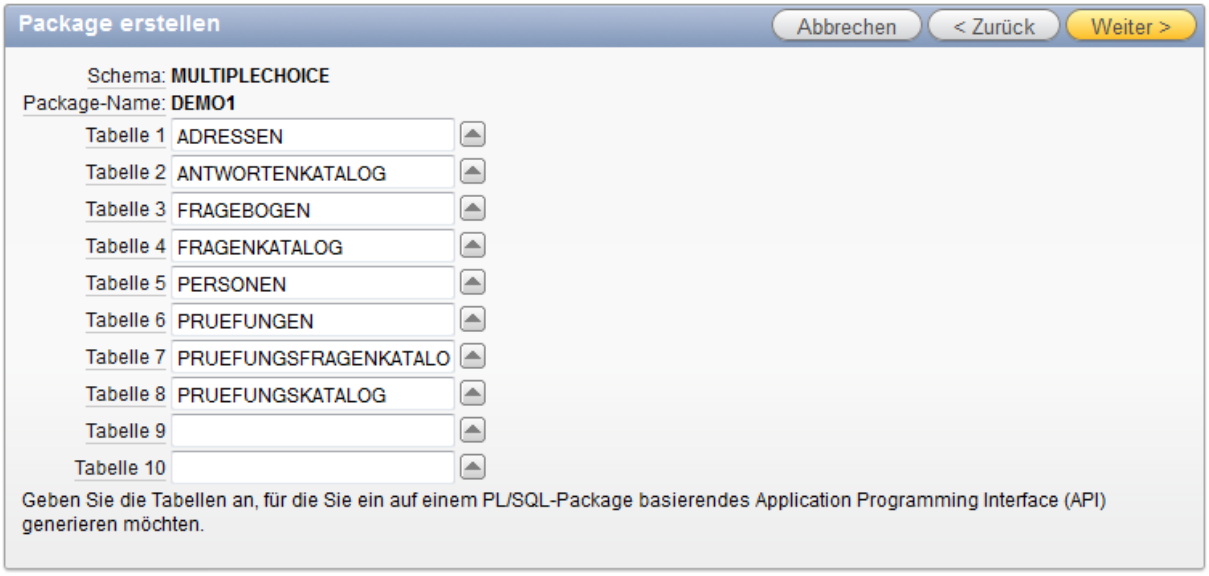
Empfehlung Nr. 6:
Sperren Sie jede APEX-Seite, die Sie bearbeiten.
Eigentlich ist dies ein trivialer Ratschlag, sollte man meinen. Doch die Erfahrung lehrt schmerzhaft, dass sich immer wieder Entwickler bei der Bearbeitung von APEX-Seiten „in die Quere kommen“ und die vorgenommen Änderungen, schlimmstenfalls unbemerkt, gegenseitig überschreiben. Dabei ist es so einfach: Wer eine Seite bearbeiten möchte, betätigt das kleine Vorhängeschloss-Symbol und trägt eine möglichst aussagekräftige Begründung als Kommentar ein. Nach getaner Arbeit wird die Seite auf dem gleichen Weg wieder freigegeben. Es empfiehlt sich für die bessere Übersicht, den Ansichtsmodus auf „Bericht anzeigen“ umzustellen (siehe farbige Markierung im Screenshot).



Falls das Entsperren einmal vergessen werden sollte, kann jeder Entwickler mit der Rolle eines Workspace-Administrators die Seitensperre wieder aufheben.

Worin liegt der Vorteil?
APEX bietet mit diesem Mechanismus ein einfaches, aber wirksames Verfahren zum Schutz der Entwicklungsarbeit, ohne gleich ein ausgewachsenes Versionsverwaltungssystem einsetzen zu müssen. Ihre Entwickler erhalten einen besseren Überblick, welche Seiten gerade in Bearbeitung sind. Leider können auf diese Weise nur *Seiten* geschützt werden – keine Gemeinsamen Komponenten, nicht die Templates, und auch nicht der Code, der in der Datenbank gespeichert ist. Aber: immerhin.

Empfehlung Nr. 7:
Nutzen Sie Table-APIs, und lassen Sie sie von APEX generieren.
APEX besitzt ein wenig bekanntes Feature: Man kann Datenbank-Packages erzeugen, die Methoden zur Datenmanipulation von Tabellen enthält. Dieses Bonbon versteckt sich unter „SQL Workshop > Utilitys > Methoden auf Tabellen“. Die Abkürzung API steht für Application Programming Interface. Sie müssen lediglich die Namen derjenigen Tabellen anklicken, für die Sie vollständige Zugriffsmethoden erstellen möchten.
Anschließend erstellt APEX das gewünschte Package quasi „maßgeschneidert“. Der Programmcode enthält Prozeduren und Funktionen zum Lesen, Einfügen, Aktualisieren und Löschen von Datensätzen inklusive der entsprechenden vollständigen Parameterlisten und (wahlweise mit und ohne) Berechnung der MD5-Hashwerte, damit Sie die standardmäßigen APEX-Sperrmechanismen (Optimistic Locking) verwenden können. Das können durchaus Tausende Zeilen Code sein!



Diese Tabellen-API können Sie bequem einsetzen, indem Sie Formulare auf Basis von Stored Procedures erstellen: Auch das lässt sich mit Hilfe von Assistenten bewerkstelligen, „Seite erstellen > Form > Form basierend auf Prozedur“.

Worin liegt der Vorteil?
Wenn Sie bereits wissen, dass Sie die volle Kontrolle über Ihre DML-Prozesse benötigen, können Sie hier massiv Zeit sparen und sich fehlerfreien Code generieren lassen, der die Default-Funktionalität implementiert und als Basis für die weitere Programmierung dient.

Empfehlung Nr. 8:
Schreiben Sie möglichst keinen Code, der indirekt auf den APEX-Sessionkontext zugreift.
Gemeint sind (zumeist lesende) Zugriffe auf APEX-Variablen unter Verwendung der v-Notation: v('REQUEST'), v('P250_NAME') usw. Wenn Sie der Empfehlung Nr. 2 gefolgt sind, haben Sie Ihren PL/SQL-Code in die Datenbank verlagert. Widerstehen Sie dort bitte der Versuchung, innerhalb des Prozedur-Rumpfes Sessionvariablen mit Hilfe der v()-Funktion auszulesen. Denn sonst lässt sich Ihr Code nur testen, wenn Sie ihn unmittelbar mit Hilfe der APEX-Anwendung ausführen. Keine dieser Variablen hat nämlich einen gültigen Wert, wenn Sie Ihre Prozedur ohne den APEX-Sessionkontext ausführen. Mit Ausnahme von spezialisierten Test-Tools, die Ihnen das automatisierte Testen von Software auch in Browseranwendungen ermöglichen, haben Sie dann keine Chance, automatisierten Softwaretest-Code für Ihre Programme zu verwenden. Um Steven Feuerstein zu zitieren, seines Zeichens PL/SQL-Guru der ersten Stunde: „Sie schreiben doch automatisierte Test, oder etwa nicht?“
Was ist dann der bessere Weg?
Parameter!
Definieren Sie alle Variablen, die Sie aus dem APEX-Sessionkontext auslesen müssen, nicht im Rumpf, sondern als Prozedur- bzw. Funktionsparameter. Sprechen Sie diese parametrisierten Methoden durch entsprechende PL/SQL-Aufrufe an, wobei Sie die Werte mit Bindevariablen übergeben (s.u.). Diese Aufrufe können Sie zum Testen mit einer ganzen Armada von automatisierten Testwerten durchführen, um Ihrem Programm auf den Zahn zu fühlen. Ein sehr hochwertiges Tool, um die Vorzüge automatisierter Tests kennenzulernen, ist der „Quest Code Tester for Oracle“, den es auch in einer funktionseingeschränkten Freeware-Version gibt.

```
-- unflexibler Ansatz, da die Prozedur an die APEX-Seite gekoppelt ist:

PROCEDURE schreibe_kundennummer
  IS
BEGIN
  UPDATE kunden
     SET kundennummer = v('P250_KUNDEN_NR')
   WHERE kunden_id = v('P250_KUNDEN_ID');
END;

-- deutlich besser, weil entkoppelt:

PROCEDURE schreibe_kundennummer
(
  i_kunden_id IN VARCHAR2
 ,i_kunden_nr IN VARCHAR2
) IS
BEGIN
  UPDATE kunden
     SET kundennummer = i_kunden_nr
   WHERE kunden_id = i_kunden_id;
END;

-- Prozedur wird in einem APEX-Prozess (oder im Test-Tool) aufgerufen:

schreibe_kundennummer(
  i_kunden_id => :P250_KUNDEN_ID,
  i_kunden_nr => :P250_KUNDEN_NR
);
```

Worin liegt der Vorteil?
Sie erhalten *lose gekoppelte*, wiederverwendbare, zuverlässige Routinen und können Ihre Software-Tests automatisieren (sprich: jederzeit wiederholen!).

Ausblick auf den Vortrag
Am Mittwoch, dem 21.11.2012, um 16:00 Uhr im Stream „Development“ gehe ich kurz auf einige der hier vorgestellten Empfehlungen ein und werde anschließend weiter gehende Beispiele präsentieren. Der Fokus wird dabei auf dem Thema „Debugging“ liegen.

Kontaktadresse:
Andreas Wismann
WHEN OTHERS
Hirschstraße 10
D-41564 Kaarst

Telefon: +49 (0) 2131-314 9966
E-Mail: wismann@when-others.com
Internet: www.when-others.com