100% PL/SQL - Web Content Management

Martin Friemel / Martin Kubitza
Enterprise Web AG
Duisburg

Schlüsselworte

Intranet, Web Content Management, Oracle RDBMS Standard Edition, Apache Webserver, PL/SQL-Softwareentwicklung, XML, Online Redaktionssystem, Intermedia Text, Suchmaschine, WebAG Automat, www.webag.com.

Zusammenfassung

Der Vortrag zeigt, wie mit dem Funktionsumfang des Oracle RDBMS Standard Edition und dem darin enthaltenen Apache Webserver leicht ein professionelles Intranet realisiert werden kann. Als Demo werden dazu praktische Beispiele und ein Web-Autorensystem gezeigt, das zu 100% in PL/SQL entwickelt wird.

Inhalt

- Einleitung
- Dynamische Webseiten mit PL/SQL entwickeln
- LDAP-Autorisierung im Intranet
- Dateien auf den Webserver hochladen und in der Datenbank speichern
- Suchmaschine für alle Dateiformate
- PL/SQL spricht mit JAVA-Programmen
- XML-Dokumente verarbeiten
- Workflows mit PL/SQL-Software steuern
- Fazit

Einleitung

Dieser DOAG-Beitrag ist ein Plädoyer für die Entwicklung von Webanwendungen in der Oracle-Programmiersprache PL/SQL. Im Lieferumfang der *Oracle Database Standard Edition* sind viele Funktionen enthalten, die zur Entwicklung komplexer Intranet-Anwendungen anregen. Bitte verstehen Sie den Vortrag als Streifzug durch die Möglichkeiten der PL/SQL-Webentwicklung, der Ihre Phantasie für neue Projekte anregen soll.

In diesem Vortrag wird am konkreten Beispiel der Architektur unseres Web-Content-Managementsystems **WebAG Automat** gezeigt, wie die einzelnen Komponenten der Oracle-Installation zur Entwicklung einer professionellen, zu 100% in PL/SQL entwickelten Web-Anwendung herangezogen werden.

Dynamische Webseiten mit PL/SQL entwickeln

Oracle liefert mit der RDBMS Standard Edition einen Apache Webserver und dazu das Modul „modPLSQL“ aus, welches den Aufruf von PL/SQL-Stored Procedures zur Erzeugung dynamischer Webseiten ermöglicht.

Die Konfiguration ist einfach: Sie müssen einen sogenannten **DAD** ("Database Access Descriptor") einrichten. Die entsprechende Einstellung wird in der Datei **`$ORACLE_HOME/Apache/modplsql/cfg/wdbsvr.app`** vorgenommen. Im folgenden Beispiel wird gezeigt, wie ein DAD namens "**`automat`**" angelegt wird:

```
[DAD_automat]
connect_string = MYDB
username = mein_schema
password = geheim
document_table = wt_blob
reuse = Yes
enablesso = No
stateful = STATELESS_RESET
custom_auth = Custom
response_array_size = 128
```

Abb. 1 – modPLSQL Konfigurationsdatei

Diese Einstellung definiert, dass sich das Modul modPLSQL an das Schema „mein_schema“ mit dem TNS-Connectstring „MYDB“ anmelden soll, um die dynamische Webseite aufzurufen. Der virtuelle Pfad zum Aufruf der Web-Anwendung im Web-Browser der Benutzer lautet demnach "`/pls/automat/<package.procedure>`".

Abb. 2 – dynamische Webseite aufrufen

Know-How zur Entwicklung von PL/SQL-Packages ist in jeder Firma, die Oracle Datenbanken einsetzt, vorhanden. HTML-Kenntnisse sind meist ebenfalls vorhanden. Es kann also losgehen!

Unsere Erfahrung zeigt, dass ein zweitägiger Workshop vollkommen genügt, um ein Entwicklerteam in die Lage zu versetzen, sein erstes PL/SQL-Webprojekt zu realisieren.

In den folgenden Kapiteln werden Lösungen für einige interessante Standardaufgaben präsentiert, die in PL/SQL-Webprojekten auftreten. Es handelt sich hierbei um Lösungswege, die sich während der Entwicklung unseres Web-Content-Managementsystems **WebAG Automat** ergaben.

Autorisierung im Intranet

Nach der Entwicklung der ersten dynamischen Webanwendung ergibt sich in der Regel sofort die Anforderung: *Diese Daten darf im Intranet aber nicht jeder sehen!*

Es muss also eine Möglichkeit geben, den Zugriff auf die PL/SQL-Webanwendung nur bestimmten, angemeldeten Benutzern zu gewähren.

Dazu sieht die modPLSQL-Konfiguration einige Varianten vor. Die bekannteste Autorisierungsvariante ist die „Database Authentication“. Dabei fragt der Webserver den Benutzer zunächst nach seinem Benutzernamen und seinem Passwort.

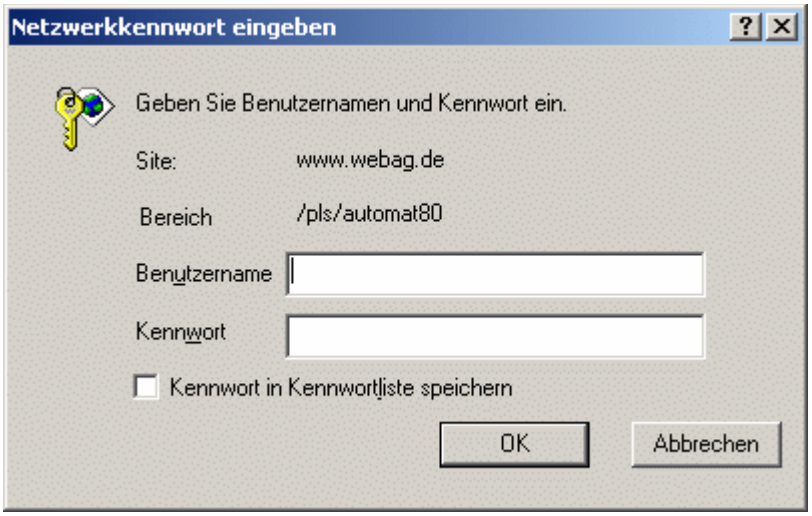

Abb. 3 – Anmeldung

Mit diesen Angaben meldet sich das Apache-Modul modPLSQL an die Datenbank an, um die geforderten PL/SQL-Prozedur aufzurufen. Sie müssen daher für jeden Ihrer Intranet-Benutzer einen Oracle-User anlegen und den Zugriff auf Ihre PL/SQL-Webanwendung mit Synonymen und EXECUTE-.Privilegien sicherstellen.

Oft ist es aber lästig, Datenbank-User für jeden Intranet-Benutzer anzulegen, zumal vielleicht sogar bereits ein zentraler LDAP-Server mit Benutzerinformationen und persönlichen Passwörtern im Unternehmen existiert. Passend zum Thema des Vortrag möchte ich deshalb eine interessante Möglichkeit vorstellen, die Autorisierung in PL/SQL zu programmieren und dabei den LDAP-Server zu nutzen. Dazu wird das PL/SQL-Package **`DBMS_LDAP`** verwendet. Wenn es

auf Ihrer Datenbank nicht vorhanden ist, können Sie es leicht mit dem Skript **$ORACLE_HOME/rdbms/admin/dbms_ldap.sql** nachinstallieren.

Zur Vorbereitung müssen Sie jedoch zunächst Ihre modPLSQL-Konfiguration dahingehend anpassen, dass vor dem Aufruf einer dynamischen PL/SQL-Webseite grundsätzlich zunächst Ihr Autorisierungs-PL/SQL-Code durchlaufen wird. Oracle sieht dafür das Package **OWA_CUSTOM** vor. Darin enthalten ist eine Function namens **AUTHORIZE**. Sie gibt *TRUE* zurück, wenn der Zugriff auf die Webseite gewährt werden kann. Mit *FALSE* signalisiert die Function AUTHORIZE, dass der angemeldete (oder anonyme) Benutzer die Seite nicht sehen darf.

In der modPLSQL-Konfiguration müssen Sie den DAD-Parameter **custom_auth** auf den Wert „**Custom**“ setzen. Damit bestimmen Sie, dass im Schema Ihrer Web-Packages das Package OWA_CUSTOM existiert und dass modPLSQL die Function **OWA_CUSTOM.AUTHORIZE** vor jeder Webseite aufrufen soll.

Nun muss die Function OWA_CUSTOM.AUTHORIZE nach Ihren Berechtigungsvorgaben angepasst und installiert werden. Ein Muster für das Package finden Sie in der Datei **$ORACLE_HOME/Apache/modplsql/owa/privcust.sql**. Nachfolgend sehen Sie einen Ausschnitt eines OWA_CUSTOM-Packages, dass einen iPlanet-LDAP-Server zur Autorisierung seiner PL/SQL-Webanwendung verwendet:

```
--------
-- LDAP-Session initialisieren
--------
l_ldap_session :=
   dbms_ldap.init (
      hostname => '<IP-Adresse des LDAP-Servers>',
      portnum  => <Port-Nummer des LDAP-Servers>
   );

--------
-- An/Abmelden an der LDAP-Server mit dem dn des Login's.
-- Wenn das klappt, ist das Web-Password OK, sonst nicht.
--------
l_ldap_return :=
   dbms_ldap.simple_bind_s (
      ld      => l_ldap_session,
      dn      => l_login_dn,
      passwd  => g_password
   );

--------
-- LDAP-Session beenden.
--------
l_ldap_return :=
   dbms_ldap.unbind_s (
      ld  => l_ldap_session
   );

IF l_ldap_return = dbms_ldap.SUCCESS THEN
   --------
   -- Autorisierung OK.
   --------
   RETURN TRUE;
```

```
ELSE
   --------
   -- Autorisierung abgelehnt.
   --------
   RETURN FALSE;
END IF;
```

Abb. 4 – OWA_CUSTOM mit LDAP

Durchforsten Sie die Package-Spec-Dokumentation des SYS-Packages OWA_CUSTOM. Ihre LDAP-Kenner werden alle benötigten LDAP-Kommandos als PL/SQL-Aufrufe erkennen!

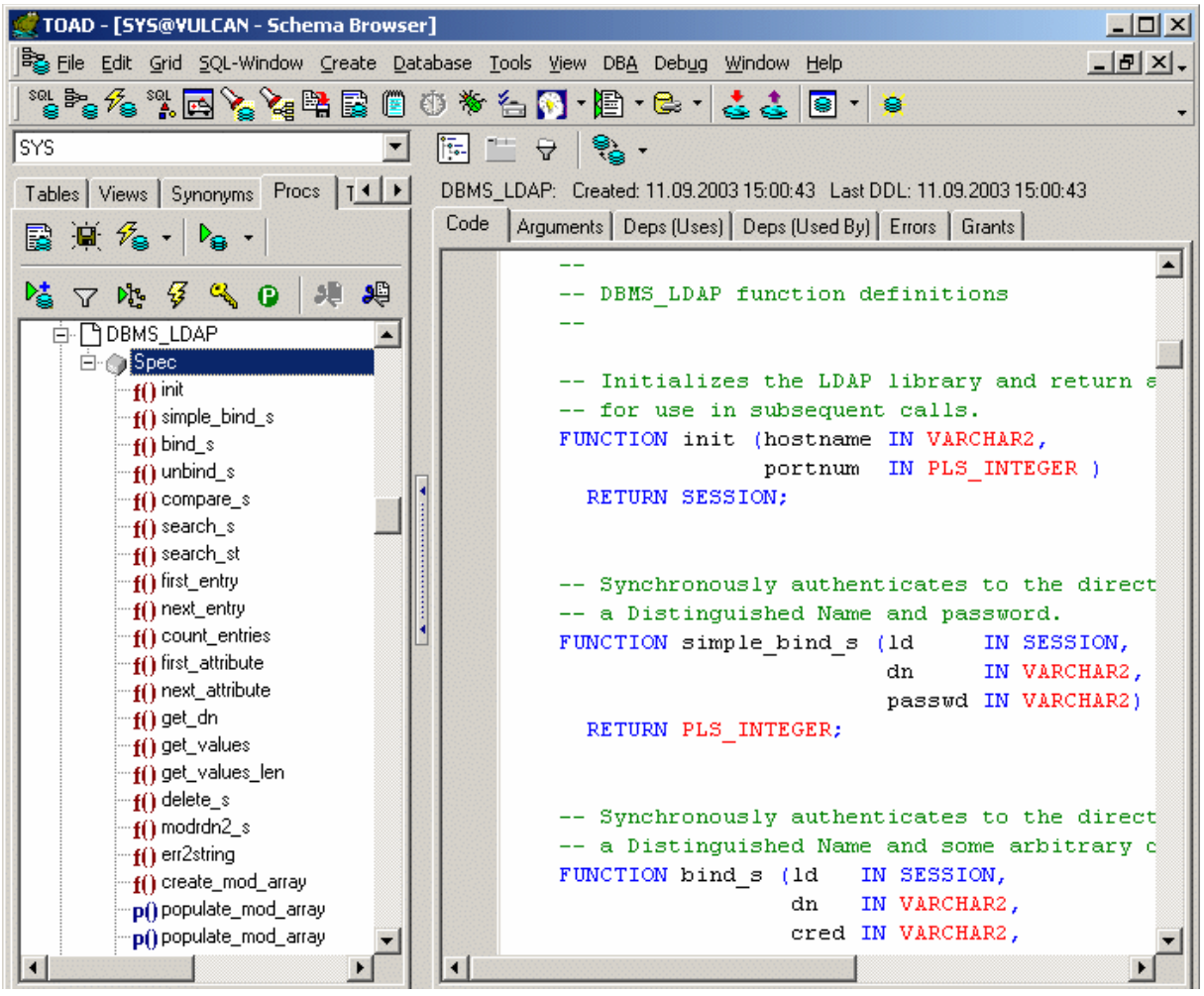

Abb. 5 – Package DBMS_LDAP

Dateien auf den Webserver hochladen und in der Datenbank speichern

Eine Standardaufgabe für Content-Managementsysteme ist die Speicherung und Verwaltung von beliebigen Dateien (wie z.B. MS-Word-Dokumente). Über die Apache / PL/SQL-Schnittstelle

modPLSQL kann man Uploads direkt in BLOB-Spalten programmieren und dazu noch eigene PL/SQL-Logik daran anknüpfen.

Die Zieltabelle für die Uploads wird in der modPLSQL-Konfiguration mit dem Parameter **`document_table = <Tabellenname>`** angegeben. Der Tabellenname ist beliebig. Die Struktur der Tabelle gibt Oracle jedoch vor:

WT_BLOB	
NAME	VARCHAR2(128)
MIME_TYPE	VARCHAR2(128)
DOC_SIZE	NUMBER
DAD_CHARSET	VARCHAR2(128)
LAST_UPDATED	DATE
CONTENT_TYPE	VARCHAR2(128)
BLOB_CONTENT	BLOB

Abb. 6 – Standard Upload-Tabelle

Die Upload-Webmaske im WebAG Automat sieht so aus:

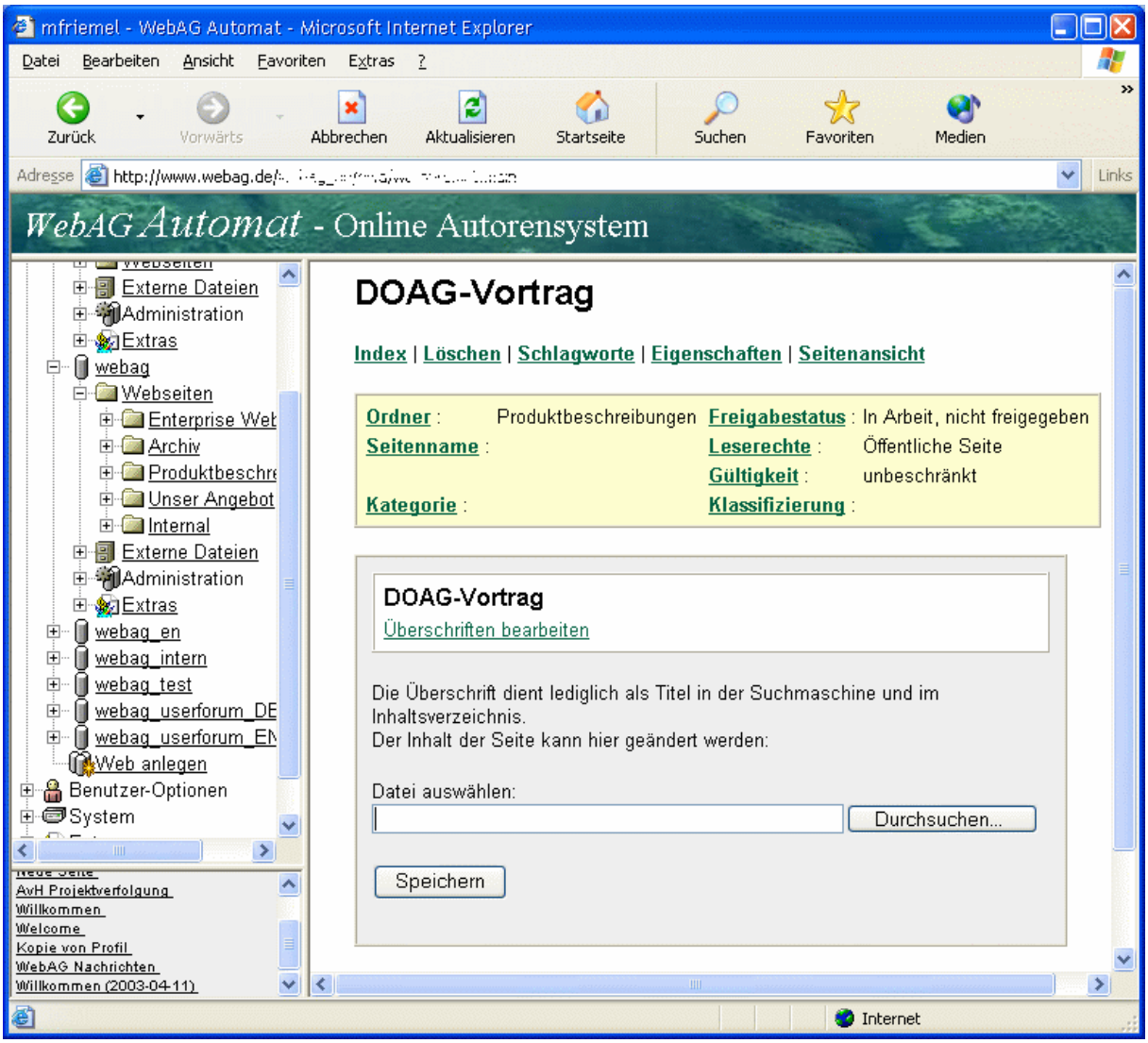

Abb. 7 – Automat Upload-Maske

Es handelt sich um ein HTML-Formular mit einem Eingabefeld vom Typ „FILE“. Auszug aus dem HTML-Code der Upload-Maske:

```
<FORM enctype="multipart/form-data"
      action="wt_au_text.upload_form_dml"
      method="POST">

<INPUT TYPE="hidden" NAME="p_text_id" VALUE="1017954">

Datei auswählen:
<INPUT TYPE="file" NAME="p_file" SIZE="50">

<INPUT TYPE="submit" NAME="p_button" VALUE="Speichern">
</FORM>
```

Abb. 8 – Upload HTML-Formular

Nachdem der Autor mit dem „Durchsuchen“-Button ein Dateiauswahlfenster geöffnet hat und die ausgewählte Datei mit dem „Speicher“-Button auf den Webserver übertragen hat, wird diese Datei von modPLSQL folgendermaßen verarbeitet:

1. Die Datei wird in die `BLOB_CONTENT`-Spalte der Standard-Uploadtabelle `WT_BLOB` gespeichert.

2. In die Spalte NAME wird der Dateiname geschrieben. Damit es aber keine Eindeutigkeitsprobleme mit Upload von Dateien gibt, die den gleichen Dateinamen haben, wird der Dateiname um einen zufällig generierten virtuellen Verzeichnisnamen erweitert. Aus „TEST.DOC“ wird z.B. „F934623/TEST.DOC“.

3. Die restlichen Spalten der Upload-Tabelle (MIME_TYPE etc.) werden ebenfalls automatisch gefüllt.

4. Danach erst wird die FORM-ACTION-Prozedur aufgerufen, die wir in unserem HTML-Formular im „`action`“-Attribut des `FORM`-Tags angegeben hatten, also im o.a. Beispiel die Prozedur `upload_form_dml` aus dem Package `wt_au_text`.

```
PROCEDURE upload_form_dml (
   p_text_id    IN NUMBER   DEFAULT NULL,
   p_file       IN VARCHAR2 DEFAULT NULL,
   p_button     IN VARCHAR2 DEFAULT NULL
);
```

Abb. 9 – Upload-Nachbearbeitungsprozedur

Dieser Standardablauf macht die Weiterverarbeitung von Upload-BLOBs einfach: Wenn unsere Prozedur `upload_form_dml` startet, kann sie bereits auf die Upload-Datei in der Tabelle

WT_BLOB zugreifen. Im Eingabeparameter p_file wird der Name der Upload-Datei übergeben. Damit kann die Prozedur den BLOB mit einer Suche in der Spalte WT_BLOB.NAME finden.

Die WebAG Automat-Weiterverarbeitung nach einem Upload beinhaltet u.a. die Indizierung der Upload-Datei in unserer Suchmaschine.

Suchmaschine für alle Dateiformate

Die Intermedia Textfilter für Fremddokumente sind Bestandteil der Oracle RDBMS Standard Edition. Mit diesen Filtern können die zuvor in die Automat-Tabelle WT_BLOB übertragenen Dateien indiziert werden. Die WebAG Automat Suchmaschine nutzt die Intermedia-Filter, um den Upload-Dateien eine Liste der jeweils darin enthaltenen Worte zu entlocken.

Die WebAG Suchmaschine ist ebenfalls eine reine PL/SQL-Software. Sie durchsucht mit einem eigenen Suchindex mit Hilfe der Intermedia-Filter alle wichtigen Dateiformate (wie Word, Excel, PDF etc.). Jede Datei, die ein Autoren auf den Webserver überträgt, kann von den Benutzern des Intranets gemäß ihrer Leserechte in der Suchmaschine gefunden werden.

Im nachfolgenden Beispiel-Screenshot wird gezeigt, wie der unscharfe Suchmodus anhand der inkorrekten Suchbegriffe „**`doak 2002 Wariohalle`**“ das Dokument zum letztjährigen WebAG-DOAG-Vortrag in Mannheim im Variosaal 4 findet:

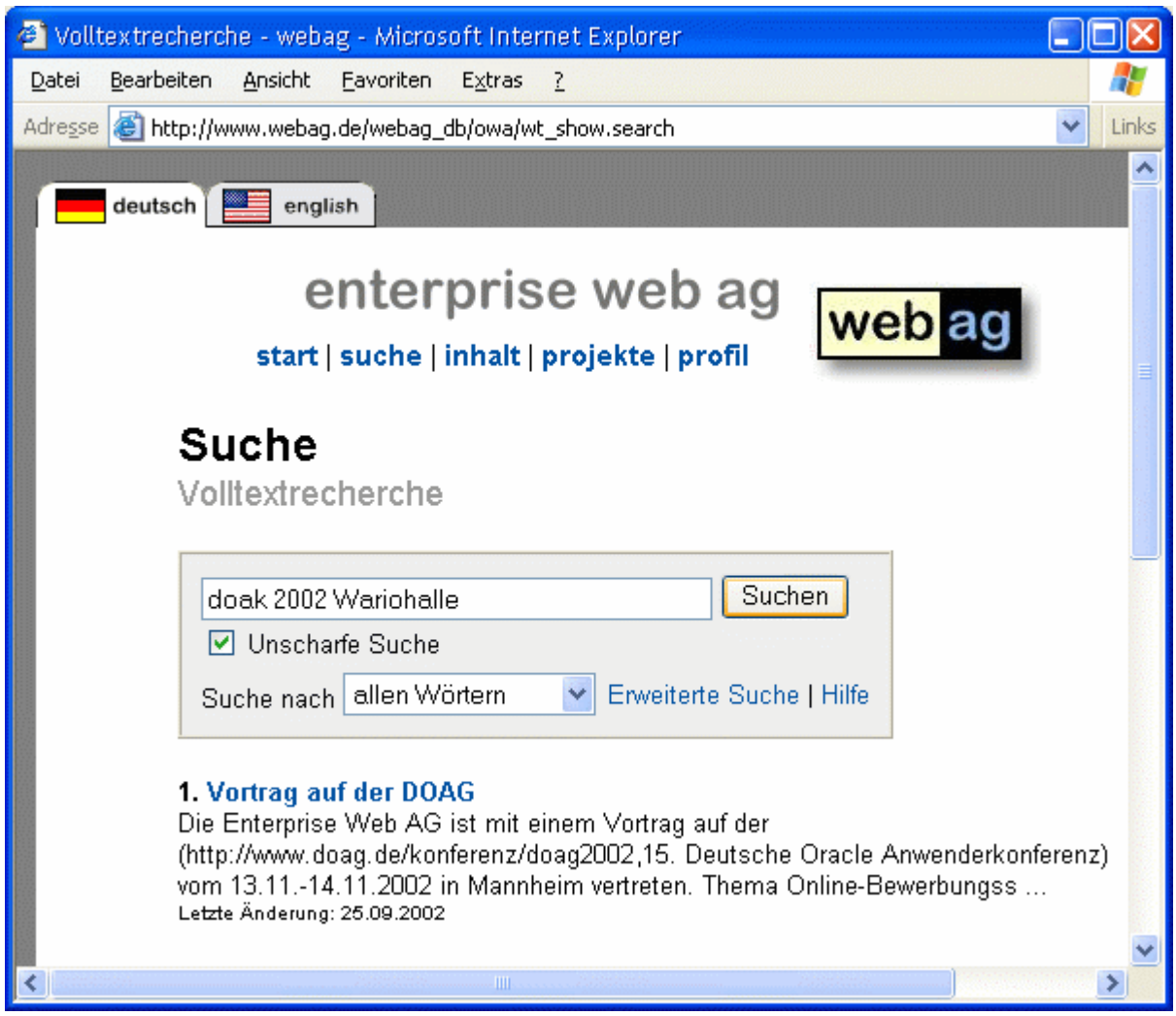

Abb. 10 – Unscharfe Suche

PL/SQL spricht mit JAVA-Programmen

Seit Oracle die Möglichkeit eröffnet hat, Stored Procedures nicht nur in der Sprache PL/SQL, sondern auch in JAVA zu programmieren, stellt sich für neue Projekte die Frage, welche der beiden Programmiersprache verwendet werden soll. Es ist dabei wichtig zu wissen, dass JAVA- und PL/SQL-Stored-Procedures sich untereinander aufrufen können. Das entschärft den „Glaubenskrieg“. Nehmen Sie für jede Teilaufgabe die Sprache, die geeigneter erscheint!

Dazu ist folgendes Beispiel aus der WebAG Automat-Entwicklung interessant: Beim Start eines Web-Content-Management-Projekts steht man immer vor der Aufgabe, bestehende Webseiten, Grafiken und Fremddokumente, die sich bereits in großer Menge in statischen Verzeichnissen des abzulösenden Intranet-Webservers befinden, in die neue Content-Datenbank zu überführen.

Leider hat PL/SQL keine Funktion, um statische Dateiverzeichnisse zu scannen. Deshalb wird diese Aufgabe von einer JAVA-Klasse übernommen.

Hier ist ein Auszug aus dem Sourcecode der JAVA-Klasse, die in die Oracle-DB installiert werden soll. Der Auszug zeigt die entscheidende Stelle, nämlich das Auslesen des Inhalts des Dateiverzeichnisses:

```
create or replace
java source named "wtXtraUpload" as

import java.io.File;

public class wtXtraUpload extends Object {

(...)

   public static File[] gFileList;

   /* ======================
    * dirInit - list filesystem directory into gFileList.
    *           Return number of files found.
    * ====================== */

   public static int dirInit (String dirName) {
      try {
         File myDir = new File (dirName);
         gFileList = myDir.listFiles();
         return gFileList.length;
      }
      catch(Exception e) {
         return 0;
      }
   } // dirInit

(...)

} // class wtXtraUpload

/
```

Abb. 11 – Dateiverzeichnisse mit JAVA auslesen

Die JAVA-Klasse kann einfach mit SQL*Plus in die Datenbank installiert werden. Dazu wurde sie mit dem Befehl „**`create or replace java source`**“ eingeleitet.

Im PL/SQL-Package wird jede JAVA-Function deklariert, damit das Interface und das skalare Return-Format im PL/SQL bekannt gemacht wird:

```
   FUNCTION dirInit (i_dirname IN VARCHAR2)
      RETURN NUMBER
      AS LANGUAGE JAVA
         NAME 'wtXtraUpload.dirInit(java.lang.String) return int';
```

Abb. 12 – JAVA-Function in PL/SQL deklarieren

Durch diesen punktuellen Einsatz einer JAVA-Stored-Procedure wurde der PL/SQL-Sprachumfang elegant um ein „DIR“ oder „ls“-Kommando ergänzt!

XML-Dokumente verarbeiten

In aktuellen Projekten werden Datenaustausch-Abläufe zwischen DV-Systemen vornehmlich als XML-Formate geplant. Als PL/SQL-Entwickler wird man also schnell mit der Anforderung konfrontiert, XML-Dokumente einzulesen und weiterzuverarbeiten. Oracle stellt dazu einige PL/SQL-Packages im Schema SYS zur Verfügung.

Die Packages **`xml_parser`**, **`xml_dom`** und **`xsl_processor`** bereichern PL/SQL um einen XML-Parser inklusive DOM-API, SAX und einen XML Schema Prozessor für XSL-Transformationen.

Im letzten Jahr befassten sich viele Vorträge mit den Oracle-XML-Features. Daher wird hier nicht ausführlich darauf eingegangen. Das Formularsystem des WebAG Automat Autorensystems speichert seine Formulardefinitionen im XML-Format und wurde mit den PL/SQL-XML-Funktionen entwickelt. Dieses Formularsystem war Thema unseres letztjährigen DOAG-Vortrags „Online-Berwerbungssystem für das Bundeskanzlerstipendiat der Alexander von Humboldt-Stiftung“, nachzulesen im DOAG 2002-Verzeichnis.

Für PL/SQL-Entwickler, die mit PL/SQL XML-Daten interpretieren und weiterverarbeiten wollen, ist folgender Hinweis sehr interessant: Das Package **`xsl_processor`** beinhaltet nicht nur das XSL-Transformations-API, sondern auch Funktionen zur Durchführung von sogenannten **XPATH**-Abfragen.

XPATH ist eine Sprache, mit der XML-Dokumente durchsucht werden können. Eine XPATH-Abfrage liefert als Suchergebnis Werte von Attributen, Textknoten-Inhalte oder aber auch komplette Teilbäume des XML-Dokuments. Der Sprachumfang wird auf dem Webserver des W3C-Konsortiums beschrieben. Das PL/SQL-Package unterstützt den kompletten Umfang der XPATH-Sprache.

Auszug aus der W3C-XPATH-Sprachbeschreibung unter der URL http://www.w3.org/TR/xpath:

> **(…)**
>
> Here are some examples of location paths using the unabbreviated syntax:
>
> - **`child::para`** selects the `para` element children of the context node
> - **`child::*`** selects all element children of the context node
> - **`child::text()`** selects all text node children of the context node
> - **`attribute::name`** selects the `name` attribute of the context node

- **child::chapter/descendant::para** selects the para element descendants of the chapter element children of the context node
- **child::*/child::para** selects all para grandchildren of the context node
- **/descendant::olist/child::item** selects all the item elements that have an olist parent and that are in the same document as the context node
- **child::para[position()=1]** selects the first para child of the context node
- **child::para[position()=last()]** selects the last para child of the context node
- **/child::doc/child::chapter[position()=5]/child::section[position()=2]** selects the second section of the fifth chapter of the doc document element
- **child::*[self::chapter or self::appendix][position()=last**()] selects the last chapt

(…)

Abb. 13 – XPATH-Patterns

Es ist sinnvoll, unter den vielen Beispielen auf der W3C-Webseite nach ähnlichen Suchmustern zu suchen, die zur eigenen Aufgabe passen. Danach rufen Sie in Ihrem PL/SQL-Code eine der drei Funktionen **selectNodes**, **selectSingeNode** oder **valueOf** auf,

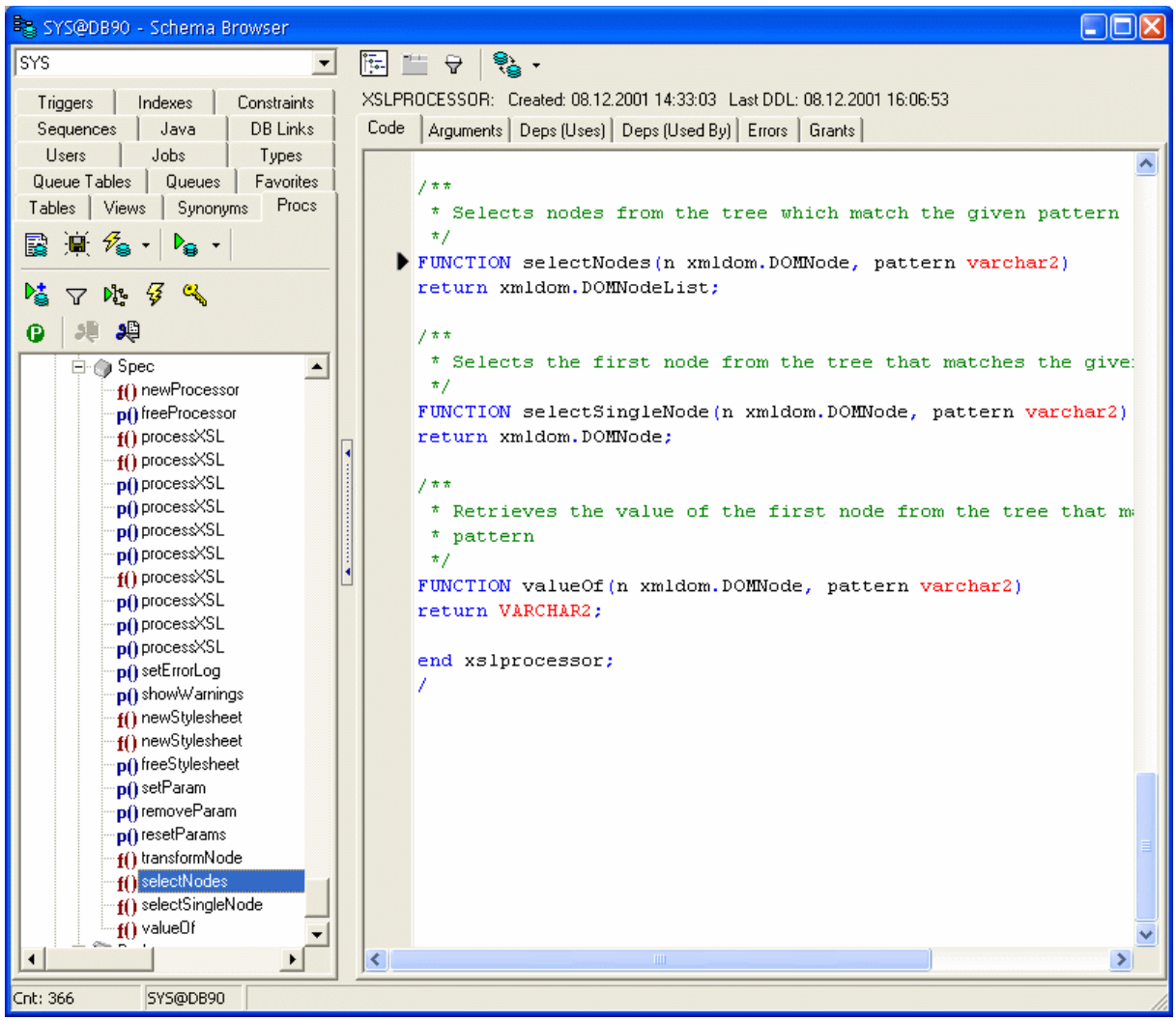

Abb. 14 – Package sys.XSLPROCESSOR

Einfaches Beispiel: Sie haben ein XML-Dokument eingelesen, welches Anweisungen für eine Datenänderungs-Transaktion enthält. Im Root-Element des XML-Codes steht die Art der Transaktion „INSERT“, „UPDATE“ oder „DELETE“:

```
<TRANSACTION type="UPDATE">
(...)
</TRANSACTION>
```

Abb. 15 – XML-Auszug

Im PL/SQL-Code ermitteln Sie mit der folgenden XPATH-Suche den Wert des Transaktionstyp-Attributes:

```
l_tx_type := XSLProcessor.valueOf (
                n => l_MyRootNode,
```

```
                  pattern => '@type'
                );
```

Abb. 16 – XPATH-Pattern in PL/SQL

Workflows mit PL/SQL-Software steuern

Nach den ersten erfolgreichen Inbetriebnahmen Ihrer PL/SQL-Webprojekte erhöht sich sicherlich die Komplexität von neuen Anforderungen, denn nach überraschend kurzen Entwicklungsphasen steigt die Lust auf schlanke Intranet-Lösungen, die mit dem Browser bedient werden.

Der Oracle Workflowserver wurde selbst in PL/SQL entwickelt. Selbstverständlich kann er mit einem PL/SQL-API von Ihren Anwendungen gesteuert werden. Die Architektur Ihrer PL/SQL-Software lässt sich einfach beschreiben:

- Alle Abläufe, Verantwortlichkeiten, Haltepunkte und Zeitlimits werden grafisch im Workflowdesigner entworfen. Das Ergebnis ist eine „Workflow-Definition“.

- Die Workflow-Definition wird in den Workflowserver installiert. Das bedeutet, die grafisch entworfenen Abläufe werden vom Workflow-Designer in ein relationales Datenmodell in die Oracle Datenbank geschrieben.

- Die Logik Ihrer PL/SQL-Webanwendungen wird einfacher, denn Sie entwickeln in einem Workflow-Projekt lediglich Masken für die einzelnen Haltepunkte in der Workflow-Definition. Die Verbindung zum nächsten Verarbeitungsschritt kennt Ihre Webmaske nicht, denn das leistet der Workflowserver.

Lesen Sie dazu bitte auch unseren Vortrag auf der DOAG 2002: „Online-Bewerbungssystem für das Bundeskanzlerstipendiat der Alexander von Humboldt-Stiftung“. In diesem Vortrag wird ein Projekt beschrieben, das auf diese Art und Weise erfolgreich realisiert worden ist.

Der Oracle Workflowserver ist die einzige in diesem Vortrag beschrieben Oracle-Komponente, die nicht in der RDBMS Standard Edition enthalten ist.

Fazit

Die einzelnen Kapitel dieses Vortrags haben anhand von interessanten Beispielen beleget, dass es sinnvoll ist, vorhandenes PL/SQL-Know-How zur Realisierung von umfangreichen Intranet- oder Web-Projekten zu nutzen. Nach unserer Erfahrung ist das eine sehr kostengünstige Art, solche Projekte in überschaubarer Zeit abzuschließen.

Die Wartbarkeit von PL/SQL-Software wurde in vielen Oracle-Projekten bewiesen, in denen PL/SQL seit vielen Jahren für den direkten Zugriff auf die Daten in der Oracle-Datenbank als Programmiersprache unumstritten ist. Wenn auf dieser sicheren Basis die Möglichkeit zur

Gestaltung der Web-Anwendungen keine Wünsche offen lässt, fällt es sehr schwer, die Kosten, die Zeit und das Risiko für den Einsatz neuer Programmiersprachen auf sich zu nehmen.

Die Enterprise Web AG nutzt sein PL/SQL-Know-How nicht nur zur Weiteentwicklung des Web-Autorensystems **WebAG Automat**, sondern auch zur Realisierung von IT-Projekten bei Firmen, die Oracle-Technologie einsetzen.

Kontaktadresse:

Enterprise Web AG
Martin Friemel
Tonhallenstraße 19
D-47051 Duisburg

Telefon: +49(0)203-2952590
Fax: +49(0)203-2952599
E-Mail mfriemel@webag.com
Internet: www.webag.com