

Entwicklung dynamischer Webseiten mit PL/SQL

Sven Tissot

pdv Technische Automation + Systeme GmbH
Hamburg

Schlüsselworte

PL/SQL, HTML, Web, XML, Session-Management, Application Server, PL/SQL Web Toolkits, dynamisches SQL, Banner Tracking, PL/SQL Gateway, PL/SQL Gateway Cache

Zusammenfassung

Der folgende Artikel beschreibt Techniken und Methoden, um dynamische, Browser-basierte Anwendungen mit PL/SQL und dem Oracle9i Application Server (IAS 1.0.2) zu erstellen:

- Dynamische Generierung von HTML,
- Session-Management mit und ohne Cookies,
- Banner Tracking,
- Erzeugung von XML Daten mit dynamischem SQL

und weitere Punkte werden exemplarisch beschrieben.

Die Anwendung

Die hier dargestellten Funktionen und Module sind Teil eines Oracle-basierten Internet-Systems mit einem automatisierten Backend zur Abwicklung einer hoch innovativen e-Lotterie.

Zur Ziehung der Gewinnzahlen wird ein zertifizierter Zufallszahlengenerator (Ziehungsgerät) eingesetzt, der die Zufallszahlen auf der Basis von Halbleiterrauschen erzeugt.

Die implementierten Zahlungswege sind neben Banküberweisung und Lastschrift die Online-Zahlung per Kreditkarte oder Geldkarte.

Das System ist in einer 3-Schichten-Architektur aufgebaut. Die Trennung in Frontend- und Backend-Datenbanken erfolgt aus Sicherheits- und Performance-Gründen. Frontend-Datenbank und Applikations-Server laufen auf unabhängigen Linux-Servern, welche wechselseitig die Aufgaben des anderen übernehmen können.

Die dynamischen Seiten werden im Frontend generiert, um einerseits die Browser-basierte Interaktion mit den Kunden zu realisieren, andererseits um den Zugriff auf Kundendaten für den Kundenservice zu ermöglichen.

Die wesentlichen Kriterien für die Entwicklung der dynamischen Seiten sind charakterisiert durch folgende Vorgaben:

- Schlanker, schneller Client
- Lauffähig auf allen verbreiteten Browsern und Browser-Versionen
- Generierung von HTML und Javascript
- Die Funktionalität der Anwendung muß auch bei ausgeschaltetem Javascript oder fehlender Cookie-Unterstützung gewährleistet sein.
- Leichte Pflege- und Erweiterung

Die Basis

Das Datenmodell und die gesamte Anwendungslogik wurden mit dem Oracle Designer spezifiziert und modularisiert. Das komplette Datenbank-Schema ist aus dem Designer mittels des Server Generators generiert; die Anwendung ist bis zur Ebene der Module entworfen worden.

Auch bei der kontinuierlichen Weiterentwicklung wird dieser Ansatz beibehalten; alle Änderungen und Erweiterungen des Systems sind im Oracle Designer Repository vollständig dokumentiert.

Tabellen

```
CREATE TABLE CD_PLZ (  
  ID NUMBER(9) NOT NULL  
  ,FK_CDBL_ID NUMBER(9) NOT NULL  
  ,PLZ VARCHAR2(5) NOT NULL  
  ,STADT VARCHAR2(100) NOT NULL  
  ,KREIS VARCHAR2(100)  
  ,BK_LAND NUMBER(2)  
  ,KGS VARCHAR2(8)  
  ,GEAENDERT_AM DATE DEFAULT sysdate NOT NULL  
  ,GEAENDERT_VON VARCHAR2(30) NOT NULL  
)  
COMMENT ON COLUMN CD_PLZ.FK_CDBL_ID IS 'ID Bundesland'  
COMMENT ON COLUMN CD_PLZ.PLZ IS 'Postleitzahl'  
COMMENT ON COLUMN CD_PLZ.STADT IS 'Name der Stadt'  
COMMENT ON COLUMN CD_PLZ.KREIS IS 'Name des Kreis'  
...  
INSERT INTO CG_REF_CODES  
  (RV_DOMAIN, RV_LOW_VALUE, RV_HIGH_VALUE, RV_ABBREVIATION, RV_MEANING)  
VALUES  
  ('BUNDESLAND', '1', '99', ',')  
...  
ALTER TABLE CD_PLZ  
  ADD (CONSTRAINT AVCON_1182368_BK_LA_000 CHECK (BK_LAND BETWEEN 1 AND 99))  
...  
CREATE OR REPLACE TRIGGER TRG_BIU_CDPL  
BEFORE INSERT OR UPDATE  
ON CD_PLZ  
FOR EACH ROW  
BEGIN  
  if INSERTING then  
    select CDPL_SEQ.NEXTVAL into :new.id from dual;  
  end if;  
  :new.geaendert_am := sysdate;  
  :new.geaendert_von := user;  
END TRG_BIU_CDPL;
```

Abb. 1 Tabellen

Sämtliche Tabellen im Schema haben einen automatisch generierten Primärschlüssel. Für jede Tabelle wird **grundsätzlich** ein Basis-Modul mit Standardfunktionen zum Zugriff auf die wesentlichen Kennfelder als Datenbank-Package realisiert.

```
CREATE OR REPLACE
PACKAGE pcd_plz IS
...

FUNCTION get_id(i_plz IN cd_plz.plz%TYPE)
RETURN cd_plz.id%TYPE ;
PRAGMA RESTRICT_REFERENCES(get_id, WNDS, WNPS);
...
FUNCTION get_plz(i_id IN cd_plz.id%TYPE)
RETURN cd_plz.plz%TYPE ;
PRAGMA RESTRICT_REFERENCES(get_plz, WNDS, WNPS);

FUNCTION get_stadt(i_id IN cd_plz.id%TYPE)
RETURN cd_plz.stadt%TYPE;
PRAGMA RESTRICT_REFERENCES(get_stadt, WNDS, WNPS);

...
END pcd_plz;
```

Abb. 2 Basis-Funktionen

Web Baukasten

Für die Module zur Darstellung der dynamischen Seiten gelten folgende Bedingungen:

- Funktionale Trennung von Layout und Anwendungslogik
- Zugriff auf Daten aus der Datenbank ausschließlich über Funktionen oder Prozeduren

Um die Implementierung produktiv und wartungsfreundlich zu ermöglichen, ist ein umfangreicher „Baukasten“ aufsetzend auf die Oracle-Basis-Pakete des PL/SQL-Web-Toolkits mit folgenden Modulen programmiert worden:

- Layout-Modul mit Basisfunktionen wie z.B.
 - *Paint_Button* zum Erzeugen von Buttons
 - *Menuepunkt, Menue* zum dynamischen Erzeugen der Menüs
 - *Header, Footer* zum Erzeugen der entsprechenden Seitenkomponenten
 - *font, color ...* - HTML Basis-Tags
 - *Table, Text, Print ...* - eigene zusammengesetzte Tags
- Forms-Modul für die dynamischen Masken
Für jede dynamische Maske gibt es eine Prozedur, welche die Maske erzeugt. Diese Prozeduren verknüpfen die Layout-Funktionen mit den Daten- und Verarbeitungsfunktionen und dem Session-Management.

- Javascript-Modul für die Generierung aller Javascript-Funktionen
- Funktionsorientierte Module für die Interaktion und den Datenbankzugriff
 - Kundenanmeldung und Verwaltung
 - Bestellung und Bezahlung
 - Kunden-Konten
 - Darstellung der Ziehungsergebnisse
- Session-Management
- Fehlerbehandlung und Logging

Die Layout-orientierten Oracle-Pakete (Package HTTP und HTF) werden ausschließlich in den Layout-Modulen aufgerufen.

Externe Schnittstellen

Die Integration der externen C-Funktionen

- zur Kommunikation mit dem Ziehungsgerät über Sockets und
- zur Integration von Verschlüsselungsroutinen

erfolgt über PL/SQL-Libraries und den *External Procedure Call Listener*.

Online-Validierung

Die Online-Validierung der Kreditkartenzahlungen ist über eine direkte Sub-Server-Server-Kommunikation realisiert. Die Daten werden über eine PL/SQL-Library-Prozedur verschlüsselt und dann via GET-Methode an den Server des kooperierenden Finanzdienstleisters übertragen.

```

...
ret := MyEncryptString( Transaktionsdaten, encrypted_string);
IF ( ret <> 0 ) THEN raise encryption_error; END IF;

-- URL zur Validierung
url := cc_url || encrypted_string ; /* etwas verkürzt ... */

-- call online validator
data := utl_http.request( url , pc_systemparameter.get_value('web_proxy') ) ;
...

```

Abb. 3 Server-Server-Kommunikation

Session-Management

Das Session-Management wird durch Cookies oder IP-Tracking ermöglicht. Lässt der Anwender die Verwendung von Cookies in seinem Browser zu, haben diese Vorrang. In ihnen wird die zufällig erzeugte Session-ID gespeichert. Falls dies nicht möglich ist, erfolgt eine Identifizierung über die IP-Adresse. Jede Maske – bzw. die sie erzeugende Prozedur – prüft als erstes die Gültigkeit der Session-ID, welche aus dem Cookie gelesen bzw. als Parameter übergeben wird.

```

...
v_akt_session_id := get_session_id;

-- Aktuelle IP-Adresse des Kunden
v_akt_ip_nummer := owa_util.get_cgi_env('REMOTE_ADDR');
-- Ist dieses Login gültig ?
... pf_session.check_session(v_akt_session_id, v_akt_ip_nummer);

```

Abb. 4 Session-Management

Die Session-Daten werden in einer Tabelle im Frontend gespeichert und mit jedem Maskenaufruf aktualisiert. Veralterte Einträge (ungültige Sessions) werden durch eine Garbage-Collection durch die Batchverwaltung (Redwood Cronacle) des Backends angestoßen.

Session-Management mit Java Servlets und PL/SQL

Alternativ kann für das Session-Management eine Kombination von Java Servlets und PL/SQL aufgesetzt werden. Dazu werden die Methoden des Java Servlet Development Kits (JSDK) im Apache Jserv Modul verwendet.

Innerhalb einer HTML-Form werden Benutzername und Kennwort entgegen genommen und anschließend ein Java Servlet für das Session-Handling gestartet (SessionHandler.class). Dieses Servlet öffnet bei der Anmeldung eine neue Session und baut über den JDBC-Treiber eine Verbindung zur Oracle8i-Datenbank auf. Der Benutzer wird verifiziert und ein entsprechender Datensatz mit der vom Apache Jserv übergebenene Session-ID in einer Verwaltungstabelle eingetragen.

Nach der erfolgreichen Authentifizierung ruft das Java Servlet die eigentliche URL zum Aufbau einer dynamischen, mit PL/SQL entwickelten HTML-Seite auf, wobei die soeben ermittelte Session-ID als Parameter weitergereicht wird.

```

...
public class SessionHandler extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {

        HttpSession session = request.getSession(true);
        if (session.isNew()) {

            session_id = session.getId();
            ip_adresse = request.getRemoteAddr();
        }
    }
}
...

```

Abb. 5 Session-Handler als Java Servlet

Für die Abmeldung von Benutzern wird analog das Java Servlet *SessionLogout.class* aufgerufen, welches die aktive Session beendet.

Anbindung von Flash-Objekten an die Datenbank

Die animierte Darstellung der Ziehungsergebnisse wird in einem extern erstellten Flash-Objekt angezeigt. Dieses ruft (ähnlich der Online-Validierung bei Kreditkartenzahlungen) direkt über HTTP eine Funktion auf, um bei jedem „Start“ aktuell die Ziehungsdaten aus der Datenbank zu holen.

XML

Zu Beginn der Programmierung standen die Oracle XML-Funktionen in der Datenbankversion 8.1.5 noch nicht zur Verfügung. Inzwischen wird das im Oracle XML Developer Kit (XDK) enthaltene XML SQL Utility (XSU) verwendet. Zur Bereitstellung von XML-Daten an externe Call Center wurde eine eigene prozedurale Schnittstelle geschaffen. Exemplarische XSL-Stylesheets werden statisch bereitgestellt.

Die Prozedur `xml_query(sql_string,...)` erzeugt aus beliebigen SQL-Anweisungen den entsprechenden XML-Stream.

Das Parsen und Ausführen des SQL-Statements erfolgt über das Oracle-Package `dbms_sql` :

```
cur_hdl := dbms_sql.open_cursor;
dbms_sql.parse(cur_hdl, i_stmt_str, dbms_sql.native);
-- describe defines
FOR i IN 1 .. i_col_anz LOOP
  dbms_sql.define_column(cur_hdl, i, col_type,max_col_length);
END LOOP;
-- execute
rows_processed := dbms_sql.execute(cur_hdl);

LOOP
  IF dbms_sql.fetch_rows(cur_hdl) > 0 THEN
    -- for each row, fetch columns from the row
    -- and write to array
    FOR i IN 1 .. i_col_anz LOOP
      j:=j+1;
      dbms_sql.column_value(cur_hdl, i, o_column_values(j));
    END LOOP;
    o_rows:=DBMS_SQL.LAST_ROW_COUNT;
  END LOOP;
```

```

xml_query(,SELECT col1,col2,col3 FROM mytable' ...)

<mytable>
  <row>
    <col1>Wert 1</col1>
    <col2>Wert 2</col2>
    <col3>Wert 3</col3>
  </row>
  ...
</mytable>

```

Abb. 6 Beispiel XML Daten

Web-Statistiken und Banner-Tracking

Um eine von Cookies und Log-Files unabhängige Werbeerfolgskontrolle für das Schalten von Bannern zu ermöglichen, werden Banner-Links auf eine Prozedur verwiesen, welche die notwendigen Protokoll-Einträge in entsprechenden Statistik-Tabellen vornimmt und dann via *redirect* auf die eigentliche Zielseite verweist:

owa_util.redirect_url(ZIEL_URL)

Die Auswertung der Click-Through-Werte kann dann mit beliebigen Werkzeugen (z.B. SQL*Plus, MS Excel) erfolgen.

Entwicklungsumgebung

Für Entwicklung und Produktion wurden 4 Datenbanken aufgesetzt: Entwicklung, Test, Qualitätssicherung und Produktion. Das initiale Aufsetzen des Schemas erfolgt über die aus dem Oracle Designer generierten SQL*Plus-Skripte.

Charakteristische Systemparameter wie z.B. URL-Daten, Portnummern, Adressen der System-Komponenten (Ziehungsgeräte, u.a.), Pfad-Angaben, Mapping der PL/SQL Routinen und weitere Informationen werden in eine Tabelle gespeichert. Dadurch ist eine einfache Anpassung des Systems an die verschiedenen Arbeitsumgebungen (Test, QS, Produktion) möglich.

Folgende Entwicklungswerkzeuge haben sich bewährt:

- Oracle Designer *Entwurf und Datenmodell*
- Oracle Developer *Backoffice Masken und Reports*
- TOAD *PL/SQL , Entwicklung, Test*
- UltraEdit *Editor*
- Dreamweaver *HTML*
- Keystone Tracker *Fehlerverfolgung und Change Management*
- CVS *Versionsmanagement*
- XML Spy *XML, XSLT, Schema*

Erfahrungen bei der Installation und dem Betrieb des Oracle 9i Application Servers

Die ersten Entwicklungsarbeiten wurden unter SuSE Linux 6.x als Plattform mit dem Oracle Application Server Rel. 4.0.8 (OAS 4.0.8) begonnen, das sich jedoch als sehr instabil erwiesen hat. Dies hatte eine Migration auf Release 1.0.2 (IAS 1.0.2) zur Folge mit der Konsequenz, dass die Konfiguration des Spyglass Web Servers mit PL/SQL Cartridges für den Oracle HTTP Server (genauer: Apache HTTP Server) und mod_plsql (dem PL/SQL Gateway) komplett überarbeitet werden mußte. Alle Einstellungen mußten von Hand für die neue Zielkonfiguration angepaßt werden. Dies gilt insbesondere auch für die SSL-Zertifikate des HTTPS Protokoll, da der IAS 1.0.2 nun mit OpenSSL und mod_ssl arbeitet.

Mit Beginn der Produktion wurde der IAS 1.0.2 ohne Web- oder Datenbank-Cache betrieben. Die Performance und die Stabilität sind auch in Spitzenzeiten dank des Oracle HTTP Servers sehr gut.

Eine einfache Form der Performancesteigerung für dynamische Seiten stellt das **PL/SQL Gateway Cache** dar. Die Einstellungen dafür werden Brower-basiert über das PL/SQL Gateway Configuration Menu vorgenommen.

Innerhalb des Source Codes macht sich das Caching unter Verwendung der Expires Technique im Zusammenspiel mit dem System-Level-Cache im Layout Module wie folgt bemerkbar:

```
-- Anzahl Minuten, die der Content gültig ist
v_minuten NUMBER := 0;
--
BEGIN
  -- Gültigkeitszeitraum setzen
  v_minuten := pb_systemparameter.plsql_cache_expires;
  owa_cache.set_expires(v_minuten, 'SYSTEM');
END;
```

Abb. 7 Beispiel PL/SQL Gateway Cache

Die Gültigkeitsdauer von Cache-Inhalten ist sehr sorgfältig zu definieren, um auch kunden-spezifische Daten im Web immer aktuell zu halten. Desweiteren ist das Zusammenspiel der Proxy-Caches bei den Internet Service Providern (ISP) gesondert zu betrachten und intensiv zu testen.

Mit dem Oracle9i Application Server Rel. 9.0.2 hat sich der Installationsaufwand wiederum erhöht, da nun zuerst eine Oracle9i Infrastructure Server mit Repository auf einer separaten Maschine aufzusetzen ist, bevor man mit der eigentlichen Installation des Oracle9i Application Servers einschließlich Oracle-HTTP-Server und PL/SQL-Gateway beginnen kann. Ganz zu schweigen vom Umfang und Plattenbedarf des Programmpaketes. Eine schlanke und einfache Installation bzw. Konfiguration der hier erwähnten Arbeitsumgebung ist so leider zukünftig immer aufwendiger aufzusetzen.

Fazit

Die Realisierung komplexer interaktiver Anwendungen im Web ist auch in Zeiten von J2EE mit PL/SQL produktiv und sicher möglich. Die Performance ist hervorragend, da gerade der datenintensive Teil der Anwendung in der Datenbank abläuft!

Der Nachteil der teilweise notwendigen engen Verknüpfung von Layout und Funktion wird wettgemacht durch die performante und transparente Anbindung an die Datenbank. Eine ideale Lösung für zukünftige Entwicklungen ist die Kombination der hier vorgestellten Techniken mit Methoden aus J2EE – wie beispielsweise die Integration des Session-Managements. Dies wird in der aktuellen Version des IAS 9.0.2 durch die Integration von J2EE und PL/SQL unterstützt.

Anhang

Die Powerpoint-Präsentation des Vortrages finden sie unter <http://www.pdv-tas.de/downloads/doag2002/>

Kontaktadresse:

pdv Technische Automation + Systeme GmbH
Sven Tissot
Dorotheenstraße 64
D-22301 Hamburg

Telefon: +49 (0)40 / 69213 266
Fax: +49 (0)40 / 69213 278
E-Mail tissot@pdv-tas.de
Internet: <http://www.pdv-tas.de>