IOUW97 Paper No. 111R

Building Your First Oracle WebServer Application

A Tutorial

Bill Pribyl/DataCraft, Inc.   Houston Texas
http://www.datacraft.com/

Most recent Update: 5 May 1997

"Do I Want to Sit Through This?"

Major topics covered:

- *PL/SQL live generation of web pages*
- *Bare essentials about installation*

Does not cover:

- *Comparison of WebServer to other tools*
- *Java, NCA, SSL, LiveHTML*
- *Custom cartridge development*

Skills assumed: PL/SQL V2, some HTML

Experience Synopsis

Designer 1.2, WebServer 2.1, Solaris 2.5

Three-month project started with Designer/2000

Built system for entry and searching of a database of graphic images

Demo on the web

- *Viewable at http://oracle.raid.com/imagebase*
- *Sorry, no 30-day free trial*

And now a very short quasi-demo...

<Netscape screen shot>

Getting Set Up

DBA Installation Tasks

1. Specify admin userid, password, and TCP/IP port via "orainst"
2. Run root.sh, which starts a special administrative web listener
3. Point browser at http://host:port/ows-abin/boot
4. Follow the detailed instructions

...more quasi-demo...

<Netscape screen shot>

Some Installation Tips

If you need to restart the installation listener, run ~/ows2/install/fmods.sh as root

After setup, to control a listener named admin (run as Oracle):

- *wlctl21 start admin [configFile]*
- *wlctl21 stop admin*

Forget the admin password? Look in ~/ows21/admin/svadmin.cfg

Agent connection data is in ...owa.cfg

What Happens When You Create an Agent

Oracle userid created if it doesn’t exist

Userid, password, and other configuration stored in file accessible to WebServer

Optionally, PL/SQL packages are installed in the Oracle user’s account

- *If you have a lot of agents, create packages once and issue synonyms and grants*

WebServer will respond to the right URL by running PL/SQL in the Agent’s Oracle account

To Save You Some Agony...

See the paper for a step-by-step description of:

- *Creating your own PL/SQL web agent/DCD*
- *...and configuring it to use the Web Request Broker rather than CGI*

Oracle documentation expects some clairvoyance in this area

A "Hello, Web" Program

```
CREATE OR REPLACE PROCEDURE myhello AS
BEGIN
   htp.print('<HTML>
   <HEAD>
      <TITLE>My First Page</TITLE>
   </HEAD>
   <BODY BGCOLOR="white">
      <H1>Hello, Web!</H1>
   </BODY>
</HTML>');
END;
```

Viewing the Generated Page

http://xyz/myapp/owa/myhello

xyz	Host name (port 80)
myapp	Name of the WebServer PL/SQL Agent
owa	A tag which by convention is set up to imply either CGI or WRB; a virtual directory is mapped to myapp/owa
myhello	Name of PL/SQL procedure

and it looks like...

<Netscape screen shot>

Using Oracle's PL/SQL HyperText Built-Ins

An alternate myhello implementation

```
CREATE OR REPLACE PROCEDURE myhello AS
BEGIN
   htp.htmlOpen;
   htp.headOpen;
   htp.title('My First Page');
   htp.headClose;
   htp.bodyOpen( cattributes => 'BGCOLOR="white"');
   htp.header(1, 'Hello, Web!');
   htp.bodyClose;
   htp.htmlClose;
END;
```

Use the Built-ins or not?

(Either way, you'll wind up learning HTML)

Yes	No
+ Can reduce total lines of code	– If you know HTML, the built-ins may get in the way
+ You may need to learn slightly less HTML	– Lengthened learning curve

But you'll want to build your own reusable utilities...

A Generic Page Open/Close Utility

Package Spec

```
CREATE OR REPLACE PACKAGE hutil AS
   PROCEDURE hOpen( p_title   IN VARCHAR2 DEFAULT NULL
                   ,p_bgcolor IN VARCHAR2 DEFAULT '"white"' );
   PROCEDURE hClose;
END;
```

A Generic Page Open/Close Utility

Package Body

```
CREATE OR REPLACE PACKAGE BODY hutil AS
   PROCEDURE hOpen( p_title IN VARCHAR2
                   ,p_bgcolor IN VARCHAR2 ) IS
   BEGIN
      http.htmlOpen;
      http.headOpen;
      http.title( p_title );
      http.headClose;
      http.bodyOpen( cattributes=> 'BGCOLOR=' || p_bgcolor );
    END;
   PROCEDURE hClose IS
   BEGIN
      http.bodyClose;
      http.htmlClose;
   END;
END;
```

Using "hutil" Package in myhello

```
CREATE OR REPLACE PROCEDURE myhello AS
BEGIN
   hutil.hOpen('My First Page');
   htp.header(1, 'Hello, Web!');
   hutil.hClose;
END;
```

Yes, putting table data on a page is what you would expect...

Printing Data from an Oracle Table

```
CREATE OR REPLACE PROCEDURE show_max_sal AS
   maxSal  scott.emp.sal%TYPE;
   cursor salCur IS SELECT MAX(sal) FROM scott.emp;
   noEmp EXCEPTION;
BEGIN
   hutil.hOpen( 'Salary Max');
   OPEN salCur;  FETCH salCur INTO maxSal;
   IF salCur%NOTFOUND THEN
      CLOSE salCur;  RAISE noEmp;
   END IF;
   CLOSE salCur;
   htp.print('The highest salary is: ' || maxSal);
   hutil.hClose;
EXCEPTION
   WHEN noEmp THEN
      htp.print('Warning: There are no employees.');
      hutil.hClose;
END;
```

<Netscape screen shot>

Printing Tables as Tables

```
CREATE OR REPLACE PROCEDURE empPrint AS
BEGIN
   hutil.hOpen('Employees');
   htp.tableOpen('BORDER');              -- <TABLE BORDER>
   htp.tableRowOpen;                     -- <TR>
   htp.tableHeader('Emp#');              -- <TH>Emp#</TH>
   htp.tableHeader('Name');              -- <TH>Name</TH>
   htp.tableHeader('Sal');               -- <TH>Sal</TH>
   htp.tableRowClose;                    -- </TR>
   FOR theEmp IN (SELECT empno, ename, sal
                  FROM scott.emp ORDER BY empno) LOOP
      htp.tableRowOpen;                  -- <TR>
      htp.tableData( theEmp.empno );     -- <TD>(employee # from table)</T
      htp.tableData( theEmp.ename );     -- ditto for ename
      htp.tableData( theEmp.sal );       -- ditto for sal
      htp.tableRowClose;                 -- </TR>
   END LOOP;
   htp.tableClose;                       -- </TABLE>
   hutil.hClose;
END;
```

Accepting Arguments

```
CREATE OR REPLACE PROCEDURE empPrint (p_deptno IN VARCHAR2 DEFAULT '%')
BEGIN
   hutil.hOpen('Employees');
   htp.tableOpen('BORDER');
   -- the table header stuff goes here…
   FOR theEmp IN (SELECT empno, ename, sal FROM scott.emp
                  WHERE deptno like p_deptno
                  ORDER BY empno ) LOOP
      htp.tableRowOpen;
      htp.tableData( theEmp.empno );
      htp.tableData(theEmp.ename );
      htp.tableData(theEmp.sal );
      htp.tableRowClose;
   END LOOP;
   htp.tableClose;
   hutil.hClose;
END;
```

Passing Arguments via URL

http://xyz/myapp/owa/empPrint?p_deptno=20

empPrint	Name of PL/SQL procedure (as before)
p_deptno	Name of argument (must match name of PL/SQL formal parameter!)
20	Value of argument

All arguments are passed as VARCHAR2

<Netscape screen shot>

Passing Arguments via HTML "Form"

```
htp.formOpen('empPrint');
htp.print('Department Number to show');
htp.formText('p_deptno',20);
htp.formSubmit;
htp.formClose;
```

...which generates:

```
<FORM ACTION="empPrint" METHOD="POST">
Department Number to show
<INPUT TYPE="text" NAME="p_deptno" SIZE="20">
<INPUT TYPE="submit" VALUE="Submit">
</FORM>
```

Passing Multiple Parameters with the Same Name?

There may be multiple items on an HTML form which have the same name

The corresponding PL/SQL input argument should be of type owa_util.ident_arr

- *PL/SQL "table" data type*
- *See example in WebServer's online documentation at http://myhost/ows-adoc/plaex5.htm*

Turns out to be common

Query Screen Design

- Do you want to tell your user how many records matched their criteria?
- How do you want to deal with queries that return...

  No values?

  A single value?

  More than one value?

  More than a thousand?

Query Screen Design

The Designer/2000-Generated Solution

Show a fixed (small) number of records per page; if only one is retrieved, you can go directly to a detail screen

In the PL/SQL display routine:

- *Keep track of last record shown (for example, in hidden form field)*
- *Put "next" and "previous" buttons on as appropriate*
- *Check what action (button) the user requested; open cursor; fetch and skip appropriate number of records; finally fetch and show the desired records*

(Cursors are not kept open between pages!)

Evaluating Search Criteria

If your search screen provides query by example (like Oracle Forms)...

...and the user may either put a value in each search field or leave it blank...

then you should use dbms_sql to dynamically evaluate the query

Dynamic Evaluation

1 of 3

```
CREATE OR REPLACE PROCEDURE empPrint ( p_deptno IN VARCHAR2 default
NULL) AS
   l_dcurs         INTEGER;        -- cursor id for dynamic SQL
   l_theQuery      VARCHAR2(512);  -- variable to hold the query
   l_undefined     INTEGER;        -- execute will return undefined value
   l_rows_fetched  INTEGER;        -- result of fetch_rows function
   l_empno         NUMBER;         -- local variable to hold table data
   l_ename         VARCHAR2(10);
   l_sal           NUMBER;
BEGIN
   IF p_deptno IS NULL THEN
      hutil.hOpen('All Employees');
      htp.header(1, 'All Employees');
   ELSE
      hutil.hOpen('Employees in Department ' || p_deptno );
      htp.header(1, 'Employees in Department ' || p_deptno);
   END IF;
   htp.tableOpen('BORDER');
   -- the table header stuff goes here…
```

Dynamic Evaluation

```
IF p_deptno IS NULL THEN
   l_theQuery := 'select empno, ename, sal from scott.emp order by
empno';
ELSE
   l_theQuery := 'select empno, ename, sal from scott.emp '
                || 'where deptno=' || p_deptno
                || ' order by empno';
END IF;
-- These calls are standard fare for dynamic SQL.  For more info,
-- look at $ORACLE_HOME/rdbms/admin/dbmssql.sql
l_dcurs := dbms_sql.open_cursor;
dbms_sql.parse( l_dcurs, l_theQuery, dbms_sql.v7);
dbms_sql.define_column( l_dcurs, 1, l_empno );
dbms_sql.define_column( l_dcurs, 2, l_ename, 10 );
dbms_sql.define_column( l_dcurs, 3, l_sal );
l_undefined := dbms_sql.execute( l_dcurs );
l_rows_fetched := dbms_sql.fetch_rows ( l_dcurs );
```

Dynamic Evaluation

3 of 3

```
  WHILE l_rows_fetched != 0
  LOOP
     dbms_sql.column_value( l_dcurs, 1, l_empno);
     dbms_sql.column_value( l_dcurs, 2, l_ename);
     dbms_sql.column_value( l_dcurs, 3, l_sal);
     htp.tableRowOpen;
     htp.tableData( l_empno );
     htp.tableData( l_ename );
     htp.tableData( l_sal );
     htp.tableRowClose;
     l_rows_fetched := dbms_sql.fetch_rows ( l_dcurs );
  END LOOP;
  dbms_sql.close_cursor ( l_dcurs );
  htp.tableClose;
  hutil.hClose;
END;
```

Adding a Drop-Down List

owa_util.listprint (p_thequery, p_cname, p_nsize, p_multiple)

p_thequery	Select statement. Retrieve three fields: (1) value; (2) name; (3) null/non-null for HTML "SELECTED" tag
p_cname	Name of HTML text field
p_nsize	"Size" of HTML drop-down; that is, the number of items displayed at one time. Values greater than one will create scroll buttons or bars rather than a drop-down.
p_multiple	TRUE or FALSE to indicate whether multiple selections allowed

Adding a Drop-Down List

Example

```
owa_util.listprint( 'select deptno, dname,
          decode(deptno,20,''SELECTED'',null) from scott.dept
          union select to_number(null), '' '', null from dual
          order by 2'
         ,'P_DEPTNO'
         ,1
         ,FALSE);
```

...which generates:

```
<SELECT NAME="P_DEPTNO" SIZE="1">
<OPTION value="">
<OPTION value="10">ACCOUNTING
<OPTION value="40">OPERATIONS
<OPTION SELECTED value="20">RESEARCH
<OPTION value="30">SALES
</SELECT>
```

<Netscape screen shot>

Exception Handling

The Designer/2000 Approach

Each Des2K-generated program unit includes something like the following:

```
EXCEPTION
   WHEN OTHERS THEN
      ShowError(SQLCODE, SQLERRM, 'thisModuleName');
      htuil.hClose;
END;
```

Designer/2000’s showerror Procedure

```
PROCEDURE showerror(p_errno   IN VARCHAR2,   p_errm IN VARCHAR2,
                    p_context IN VARCHAR2, p_action IN VARCHAR2) IS
   BEGIN
      hutil.hopen('Error');
      htp.para;
      IF p_context IS NOT NULL THEN
         htp.p(p_context);
         htp.para;
      END IF;
      htp.p(p_errm);
      IF p_action IS NOT NULL THEN
        htp.para;
        htp.p(p_action);
     END IF;
     hutil.hclose;
   END;
END;
```

Common Errors You'll See During Development

- "Request Failed. We were unable to process your request at this time. Please try again later."

Usually an argument mismatch. Look at the end of the error log file, which is something like $ORACLE_HOME/ows2/log/myapp.err

```
Sun Feb 16 20:24:50 1997
OWS-05111: Agent : no stored procedure matches this call
with the arguments passed
OWA SERVICE: MYAPP
PROCEDURE: empprint0
PARAMETERS:
===========
P_DEPTNO:
20
```

Common Errors (Cont'd)

- The requested URL was not found.

  Make sure the URL is typed correctly and that the requested virtual directory and Agent exists.

- (No response)

  Make sure the listener process is up and running

owa_util.showpage

Development and Debugging Tool

Convenient utility for quick check of generated HTML

Shows the output of the most recent PL/SQL routine

Requires SERVEROUTPUT ON

owa_util.showpage Example

```
SQL> set serveroutput on size 100000
SQL> execute empdetail(7788)
PL/SQL procedure successfully completed.

SQL> execute owa_util.showpage
<HTML>
<HEAD>
<TITLE>Employee Detail for 7788</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<H1>Employee 7788</H1>
<P>
<B>Name:</B>SCOTT<BR>
<B>Dept:</B>20<BR>
</BODY>
</HTML>
PL/SQL procedure successfully completed.
```

What’s All This About Transactions?

Challenge of building web database applications is the "statelessness" of HTTP

In WebServer V1.0 and 2.x, all DML operations commit immediately

- *Design applications accordingly*
- *Session variables do not persist between calls*

Version 3 allows XA-compliant transaction management

The Big Leap into Read/Write Applications

It’s not just a problem with transactions; you’ll want to give some thought to the overall user interface model

Some issues

- *Will your entry form look like your query form?*
- *How and when should validation failures occur?*
- *What screen (and choices) will you show the user when their insert/update/delete operation succeeds?*
- *...when it fails for a reason other than failing validation?*

One Approach

Use single PL/SQL module to generate more than one HTML page

Include optional PL/SQL argument(s) to receive value of a unique identifier

If key argument is null, display Query and Create buttons; otherwise display Update and Delete buttons

Code Fragments

HTML Form for QBE, Insert, Update, Delete

```
CREATE OR REPLACE PROCEDURE empForm (p_empno IN VARCHAR2 DEFAULT NULL) AS
BEGIN                                  -- obligatory page header stuff omitted
   IF p_empno IS NULL THEN
      htp.head(1, 'Query or Create Employee');
   ELSE
      htp.head(1, 'Update or Delete Employee');
   END IF:
   htp.formOpen('empProcess');    -- displaying of fields omitted
   IF p_empno IS NULL THEN
      htp.formSubmit('ACTION', 'Create New');
      htp.formSubmit('ACTION', 'Query');
   ELSE
      htp.formSubmit('ACTION', 'Save Changes');
      htp.formSubmit('ACTION', 'Delete Employee');
   END IF;
   htp.formClose;                 -- obligatory page footer stuff omitted
END;
```

More Code Fragments

Companion Code to Process Form

```
CREATE OR REPLACE PROCEDURE empProcess (p_empno IN VARCHAR2 DEFAULT NULL
                                       ,p_ename IN VARCHAR2 DEFAULT NULL
                                       ,etc...
                                       ,action IN VARCHAR2 DEFAULT NULL) AS
   l_empno NUMBER;
BEGIN
   IF action = 'Query' THEN
      -- check for exactly one hit (omitted); if so, call empForm
      empForm(p_empno);
      RETURN;
       -- otherwise display all of the queried records (omitted)
  END IF;
   IF action = 'Create New' THEN
      l_empno := manage_emp.create_one(p_ename, etc.);
   ELSIF action = 'Save Changes' THEN
      manage_emp.update_one(p_ename, etc.);
   ELSIF action = 'Delete Employee' THEN
      manage_emp.delete_one(p_empno);
   ELSE                                 -- print an error message
   END IF;                              -- page footer stuff
END;
```

The User Identification Problem

Without the concept of a login session, how can your PL/SQL confirm that a request comes from an authorized user?

Our Solution

A Custom Scheme

Obtain user name and password via HTML form

Compare against Oracle table of names and passwords

If valid, generate a "cookie" and save on client side and in Oracle table

Every PL/SQL module first looks for client cookie; if not found or invalid, jumps to login screen

Requesting the User Name and Password

```
CREATE OR REPLACE PROCEDURE login AS
   BEGIN
      hutil.hOpen('Login');
      htp.header(1,'Please Enter Your Login Name and Password');
      htp.formOpen('validate_login');
      htp.tableOpen;
      htp.tableRowOpen;
      htp.tableData(htf.bold('Login Name:'),'right');
      htp.tableData(htf.formText('p_name',18,18));
      htp.tableRowClose;
      htp.tableRowOpen;
      htp.tableData(htf.bold('Password:'),'right');
      htp.tableData(htf.formPassword('p_pw',18,18));
      htp.tableRowClose;
      htp.tableRowOpen;
      htp.tableData;
      htp.tableData(htf.para || htf.formSubmit(null,'Login'));
      htp.tableRowClose;
      htp.tableClose;
      hutil.hClose;
   END;
```

Validating the Name and Password

```
CREATE OR REPLACE PROCEDURE validate_login( p_name IN VARCHAR2, p_pw IN VARCHAR2) IS
      l_individual_id NUMBER;
      l_session_id    NUMBER;
      l_web_password  t_users.web_password%TYPE;
      CURSOR indCur IS SELECT individual_id, web_password FROM t_users
                              WHERE web_username = UPPER(p_name);
   BEGIN
         OPEN indCur; FETCH indCur INTO l_individual_id, l_web_password;
         IF indCur%FOUND THEN
            CLOSE indCur;
            IF l_web_password = upper(p_pw) THEN
               l_session_id := get_session_id;
               -- delete any old session records, insert a record for the individual
               t_user.create_user_session( l_individual_id, l_session_id );
               -- this procedure allows the user into the opening page
               home( l_individual_id );
            ELSE login;
               RETURN;
            END IF;
         ELSE CLOSE indCur;
            login;
            RETURN;
         END IF;
   END;
```

Getting the Cookie

```
CREATE OR REPLACE FUNCTION get_session_id RETURN NUMBER IS
      l_session_id NUMBER;
      l_session_id_cookie owa_cookie.cookie;
   BEGIN
      l_session_id_cookie := owa_cookie.get('SESSION_ID');
      owa_util.mime_header('text/html', FALSE);
      IF l_session_id_cookie.num_vals > 0 THEN
         owa_util.http_header_close;
         RETURN l_session_id_cookie.vals(1);
      ELSE
         SELECT session_seq.nextval INTO l_session_id FROM dual;
         owa_cookie.send('SESSION_ID', TO_CHAR(l_session_id));
         owa_util.http_header_close;
         RETURN l_session_id;
      END IF;
   END;
```

Validating the Cookie

```
FUNCTION get_individual_id RETURN NUMBER IS
  l_session_id NUMBER;
  l_individual_id NUMBER;
BEGIN
  l_session_id := get_session_id;
  DECLARE
     CURSOR indCur IS SELECT individual_id FROM t_sessions
                      WHERE session_id = l_session_id;
  BEGIN
     OPEN indCur;
     FETCH indCur INTO l_individual_id;
     IF indCur%NOTFOUND THEN CLOSE indCur;
        login;
     ELSE CLOSE indCur;
     END IF;
  END;
  RETURN l_individual_id;
END;
```

The Reusable Call to Validate the Cookie

```
CREATE OR REPLACE whatever AS
   skipit          EXCEPTION;
   l_individual_id NUMBER;
BEGIN
      l_individual_id := get_individual_id;
      IF l_individual_id IS NULL THEN
         RAISE SKIPIT;
      END IF;
...
EXCEPTION
   WHEN skipit THEN NULL;
END;
```

Other Issues in WebServer Application Development

Applications seem code-heavy

Latest Designer/2000 generates read/write applications

Other environments will be more intuitive to non-Oracle programmers

What about connecting to non-Oracle databases?

Further Resources

http://www.datacraft.com	Home of this presentation and paper
http://govt.us.oracle.com	Some nifty downloadable utilities
http://www.olab.com/beta/	Oracle’s own site for Internet server products including V3.0 beta

Further Resources (Cont'd)

http://www.onwe.co.za/frank/faqweb.htm	Frank Naude & Steve Kilbane's WebServer FAQ
http://merlin9.npac.syr.edu/cgi-bin/news/hypermail_home	ORAWEB-L archives