

Banner Self-Service Web Programming Training Workbook

Release 8.0 - April 2008

Updated 8/1/2008



SUNGARD HIGHER EDUCATION

What can we help you achieve?

SunGard Higher Education
4 Country View Road
Malvern, Pennsylvania 19355
United States of America
(800) 522 - 4827

Customer Support Center website
<http://connect.sungardhe.com>

Distribution Services e-mail address
distserv@sungardhe.com

Other services

In preparing and providing this publication, SunGard Higher Education is not rendering legal, accounting, or other similar professional services. SunGard Higher Education makes no claims that an institution's use of this publication or the software for which it is provided will insure compliance with applicable federal or state laws, rules, or regulations. Each organization should seek legal, accounting and other similar professional services from competent providers of the organization's own choosing.

Trademark

Without limitation, SunGard, the SunGard logo, Banner, Campus Pipeline, Luminis, PowerCAMPUS, Matrix, and Plus are trademarks or registered trademarks of SunGard Data Systems Inc. or its subsidiaries in the U.S. and other countries. Third-party names and marks referenced herein are trademarks or registered trademarks of their respective owners.

Revision History Log

Publication Date	Summary
Original Date	New version that supports x.x.x software.
Revision Date	Revised to support x.x.x. Include a brief description of major changes to the document.

Notice of rights

Copyright © SunGard Higher Education 2005-8. This document is proprietary and confidential information of SunGard Higher Education Inc. and is not to be copied, reproduced, lent, displayed or distributed, nor used for any purpose other than that for which it is specifically provided without the express written permission of SunGard Higher Education Inc.



Think before you print.

Table of Contents

Introduction	6
Agenda	7
Web Architecture and Components	8
What is Banner Self-Service?.....	9
Architecture	10
Oracle Application Server	12
How It All Ties Together	15
Building Blocks: HTML	18
Introduction	19
HTML	20
HTML Exercise #1	22
HTML Tags	23
HTML Exercise #2	28
HTML (Advanced).....	29
HTML Form Tags.....	31
HTML Exercise #3	33
HTML References	34
Building Blocks: PL/SQL	35
Introduction	36
PL/SQL Modularity	39
PL/SQL Procedures	41
PL/SQL Functions	42
Cursors	43
Package Specification	44
Package Body	45
PL/SQL Exercise #1	46
Information Hiding & Data Encapsulation	47
PL/SQL Overloading.....	48
PL/SQL Exception Handling	49
PL/SQL Exercise #2	50
PL/SQL References	51

Building Blocks: PL/SQL Toolkit	52
Introduction	53
PL/SQL Documentation	55
Viewing Packages	56
HTP Package	57
HTF Package	58
Using PL/SQL Toolkit to Create HTML	59
PL/SQL Toolkit Exercise #1	61
Using PL/SQL Toolkit to Create HTML: Tags and Formatting	62
PL/SQL Toolkit Exercise #2	65
Using PL/SQL Toolkit to Create HTML: Display Tags and Lists	66
PL/SQL Toolkit HTML Tables	69
PL/SQL Toolkit Exercise #3	72
PL/SQL Toolkit HTML Forms	73
PL/SQL Toolkit Exercise #4	75
PL/SQL Toolkit References	76
Banner Self-Service	77
Introduction	78
Source Code Framework	79
Naming Conventions	80
Banner Self-Service Page Design	82
New UI & CSS	85
CSS Statements	86
SunGard Higher Education-Delivered Style Sheets	87
CSS References	88
Internationalization Enhancement	89
Banner 7.x+ API	90
Web Tailor	91
Introduction	92
Security	93
User Roles	94
Web Tailor Usage	96
Web Tailor Menu	97
Menu Option - Customize a Web Menu or Procedure	98
Menu Option - Customize a Graphic Element	101
Menu Option - Customize Information Text	104
Menu Option - Customize Menu Items	107
Menu Option - Update User Roles	109
Menu Option - Customize a Web Module	111
Menu Option - Customize Web Rules	114
Menu Option - Customize Web Tailor Parameters	115
Menu Option - Customize a Login Return Location	116
Menu Option - Customize Web Tailor Overrides	118
Menu Option - Customize GUI Settings	119
Web Tailor Summary	120

Additional Topics	121
Banner Self-Service Source (Packages)	122
Web Tailor Packages	125
Viewing Packages	126
Web Tailor Tables	127
Banner Self-Service Coding Methodology	128
twbkwbis.F_ValidUser	129
twbkwbis.P_OpenDoc	130
twbkwbis.p(f)_DispInfo	131
twbkwbis.p_CloseDoc	132
Banner Self-Service Coding Example	133
SSB Exercise #1	134
Banner Formatting Procedures	135
Common Code for Text	136
Banner Standards for Tables	137
Using Tables to Display Data	138
SSB Exercise #2	139
Linking to Another Page	140
Adding Packages to SSB	141
Modifying Banner Self-Service	142
Security	145
Introduction	146
Internet Traffic Security	147
Security References	149
Banner Self-Service Security	150
Oracle Database Security	151
User ID and PIN	152
Banner WEBID Security	153
TWGBWSES	154
Banner Self-Service Roles	155
Product-Specific Features	156
Additional Information	157
Exercise Answers	158
HTML Exercise #1	159
HTML Exercise #2	160
HTML Exercise #3	162
PL/SQL Exercise #1	164
PL/SQL Exercise #2	165
PL/SQL Toolkit Exercise #1	167
PL/SQL Toolkit Exercise #2	169
PL/SQL Toolkit Exercise #3	171
PL/SQL Toolkit Exercise #4	174
Banner Self-Service Exercise #1	178
Banner Self-Service Exercise #2	180

Introduction



Course goal

This course is designed to provide instruction in Web Programming in a Banner context.

Agenda

Agenda

- Web Architecture & Components
- Building blocks
 - HTML, PL/SQL, PL/SQL Toolkit
- Banner Self-Service (SSB)
- WebTailor, SSB Web coding
- Security
- SSL, Firewalls, ANO... oh my!

Web Architecture and Components



Introduction

This section describes the architecture and components that make up Self-Service Banner.

What is Banner Self-Service?

Introduction

- Add-on to Banner baseline products
- Advantages of Banner Web Design
 - reduced network processing of data servers
 - move to thin client on desktop
- No middle tier forms, logic maintained on database server
- Oracle PL/SQL Packages

Architecture

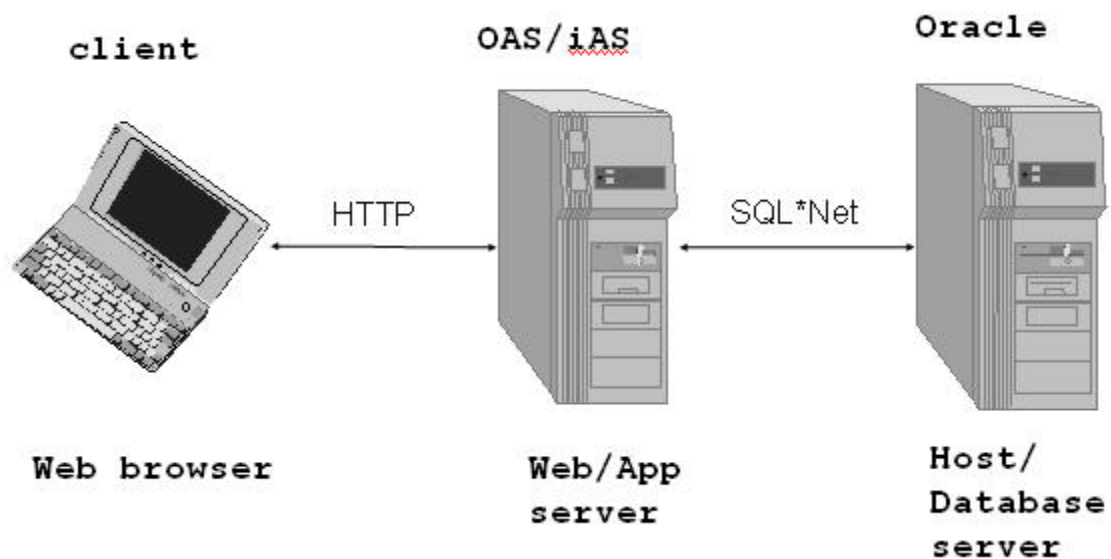
Tiers

Top Tier: User interfaces with browser

Middle Tier: Browser communicates with Oracle9i application server (web server)

Database Tier: Web requests are processed by stored procedures

Web Architecture



Web client

- Any browser that supports cookies
- Mozilla Firefox
- Internet Explorer
- Safari
- Others... Opera?

HTTP/Web Server

- Oracle Application Server (OAS) or Internet Application Server (iAS)
 - connects to Oracle via SQL*Net
- Components:
 - Listener
 - DAD - Database Access Descriptor
 - PL/SQL Agent/Gateway

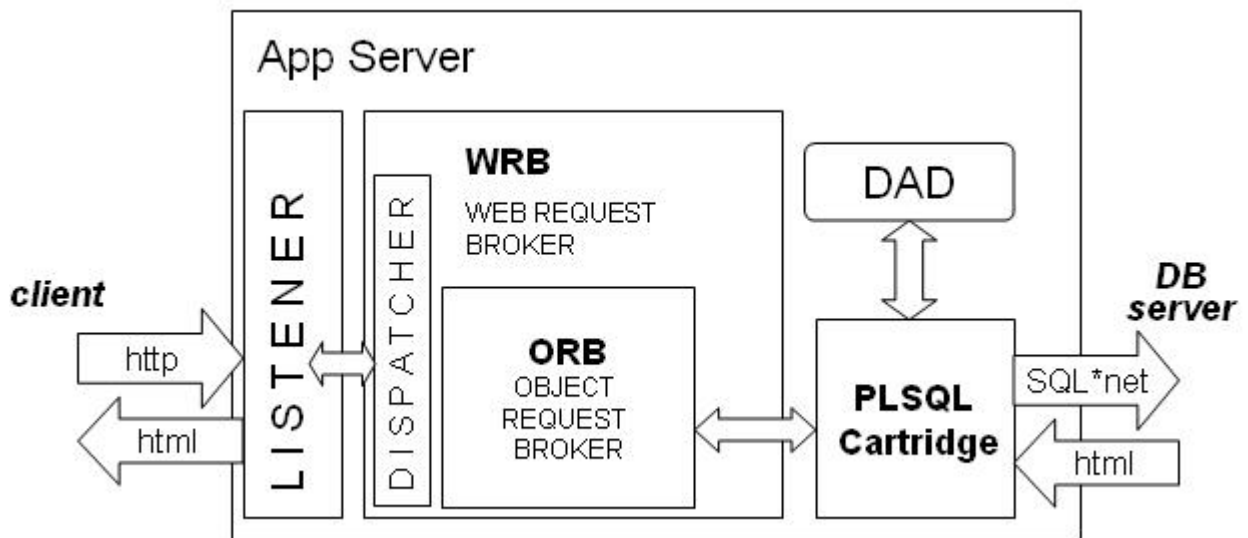
Oracle Application Server

Description

Oracle Application Server puts into place an architecture which will allow you to build applications that tap into the Oracle database. All the while using the deployable and versatile medium, the World Wide Web.

- Transaction management
- Multithreaded processing of requests
- Thin client, server-side processing
- Platform and browser independent
- Pre-disposed to growth factors

Diagram



Listener

- Software daemon
 - listens for requests on pre-defined ports (use any higher than 1000)
- Responsible for opening a port on the web server to receive, secure, and interpret HTTP requests
- Best to have separate listeners for each database
 - Allows systems to be brought up and down independently
 - Allows SSL to be used for web

Database Access Descriptor

- Specifies;
 - host machine & port where Oracle Database resides
 - Establish SQL*Net connection
 - Oracle user to connect to database
 - Execute requested package.procedure
- uses TNSNAMES.ORA file

PL/SQL Agent

- Cartridge or software module
 - programming plug-ins to OAS/iAS
 - Types; PL/SQL, C, Perl, Java,...
- invokes PL/SQL procedures stored in Oracle databases, which will return data in dynamic HTML pages

Oracle PL/SQL Toolkit

- A series of packages and procedures delivered by Oracle which create html tags
- Web displays using HTP & HTF packages
- Delivered as part of 9ias installation and created in the SYS database schema
`$ORACLE_HOME/Apache/modplsql/owa/owaload.sql`

Generating HTML in PL/SQL

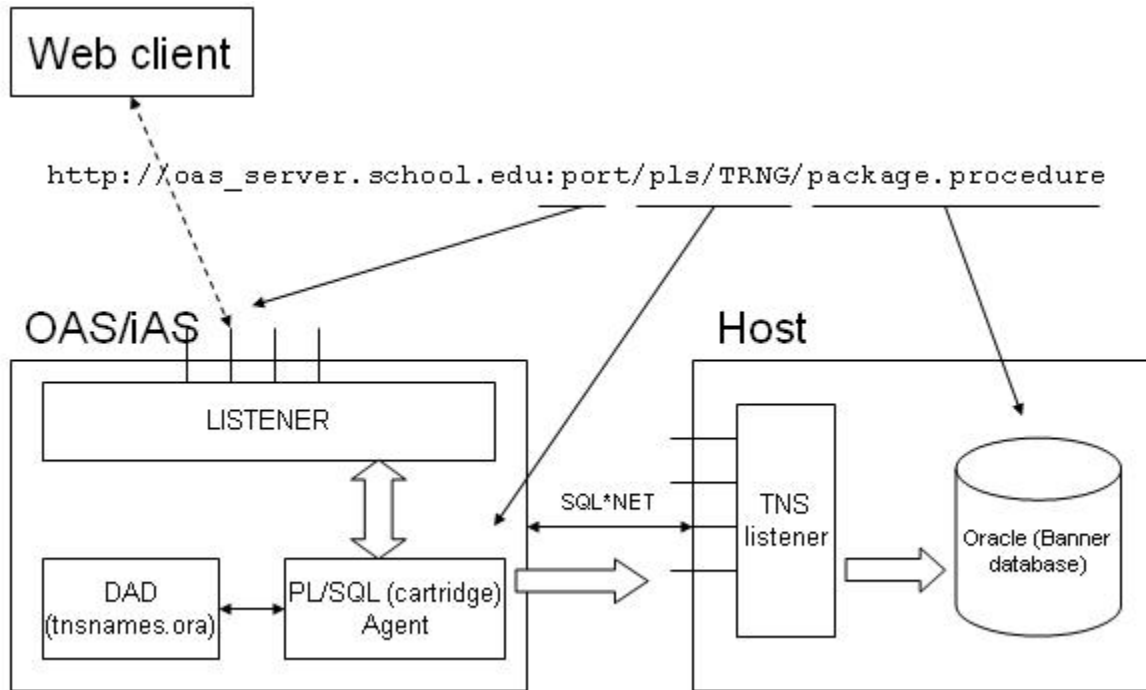
- Use packages and logical units
- Must follow HTML standards
- DAD user must have execute privileges to the packages created by the developer
 - public synonyms and grants
- Owner of package.procedure needs execute privilege on PLSQL toolkit packages

Host/Database Server

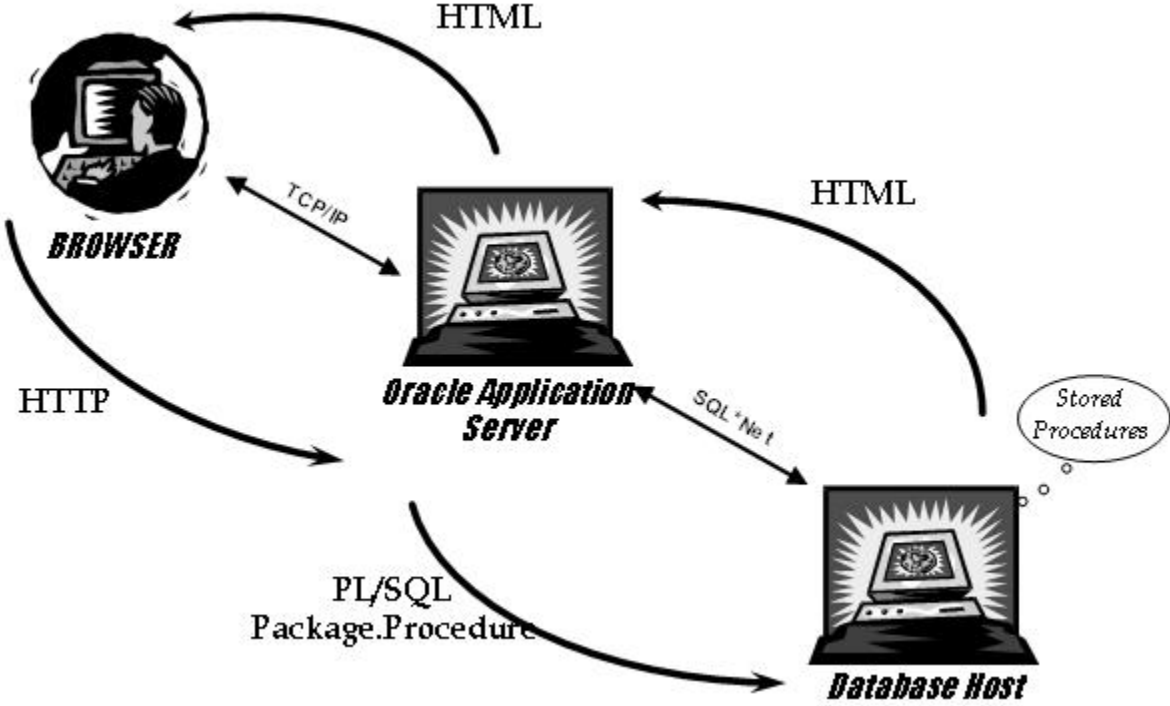
- Oracle database
 - RDBMS 8i or 9i
 - shared_pool >=100 MB
- Oracle listener
- Banner Products
 - Student, General, etc...
 - Self-Service products

How It All Ties Together

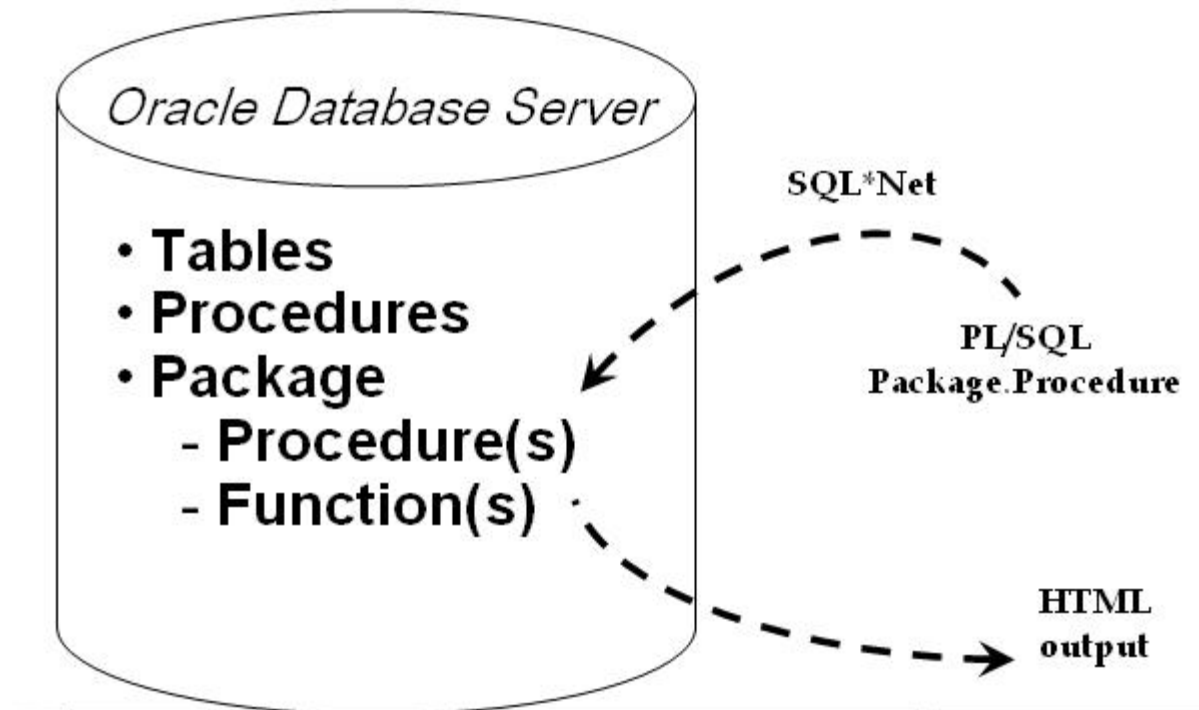
Diagram



Diagram



Diagram



Building Blocks: HTML



Introduction

This section discusses HTML code and commonly used HTML tags.

Introduction

Components

- HTML
- PL/SQL
- PL/SQL Toolkit
- HTP, HTF & OWA%

HTML

What is HTML?

- Hyper Text Markup Language
 - language of the web
 - series of tags in an ASCII text document
 - instructs browser what to do
- HTML commands are set off by < >
 - travel in pairs & affect what's in between tags
 - <html> is a start, and </html> is a stop
 - you can always 'view source'

HTML Syntax

- Most HTML tags have attributes

```
<FONT color=blue face="Arial" size=6> </FONT>
<A HREF="homepage.htm"> click here </A>
<P ALIGN=direction>
```
- Tags need to be closed </BODY>
- Some tags need attributes <HREF..>

Sample

```
<html>
  <!-- this is how to do comments -->
  <head>
    <title> Your Title appears at the very top of window bar
  </title>
  </head>

  <body>
    <h1> Hello World </h1>

    <p> This is our first web document!

  </body>
</html>
```

Think about the blocks!

HTML Exercise #1

Exercise

- Do the HTML exercise "Hello World" from the sample on the previous page.
- Create the file on your desktop.
- Put in HTML commands.
- Show the results in your browser.

HTML Tags

Basic Tags

Commands	Purpose
<code><html> </html></code>	creates an html document
<code><head> </head></code>	info portion about the html document
<code><title> </title></code>	displays info in window title bar
<code><body> </body></code> <code><body background=? bgcolor=? text=? link=? vlink=? alink=?></code> <u>example:</u> <code><body></code> <code><body bgcolor=cyan text="#FFFFFF"></code>	visible portion of document just open body (nothing else) white on cyan background
<code><h1> </h1></code> <code><h2> </h2></code> <code><h3> </h3></code> <code><h4> </h4></code> <code><h5> </h5></code> <code><h6> </h6></code> <u>example:</u> <code><h1> header line 1 </h1></code> <code><h2> header line 2 </h2></code>	header or headline

Singleton tags

- Beginning tag with no required ending tag
- Note: These tags SHOULD use ending tags as good coding practice, but it is not required yet

Commands	Purpose
<pre><p> </p></pre> <pre><p/></pre> <p><u>example:</u></p> <pre><p> paragraph 1...</pre> <pre><p> paragraph 2...</pre>	paragraph
<pre>
</pre> <pre>
</pre> <p><u>example:</u> if this is on the first line
 this will appear on the next line</p>	inserts a line break
<pre><hr size=? width=? noshade></pre> <pre><hr/></pre>	horizontal rule line (Line drawn horizontal across page)

Additional formatting tags

Commands	Purpose
<code> </code> <u>ex:</u> <code> this is going to be bold! </code>	Make text bold
<code><i> </i></code> <u>ex:</u> <code><i> this will be italics </i></code> <code><i> this will be bold & italics </code> <code></i></code>	change text to italics
<code> </code> <u>ex:</u> <code> Don't supersize, emphasize!</code> <code></code>	emphasize
<code><pre> </pre></code> <u>example:</u> <code><pre></code> The computer was first invented ... <code></pre></code>	preformatted text

Commands	Purpose
<p> </p> <p></p> <p><u>example:</u></p> <p> This is the biggest! </p> <p> Slightly smaller. </p> <p> The smallest you can imagine. </p> <p> This should stand out. </p> <p> This will really stand out. </p>	<p>set font attributes</p>
<p><dl> </dl></p> <p><dt></p> <p><dd></p> <p><u>ex:</u> Investing</p> <p> <dl></p> <p> <dt> IRA</p> <p> <dd> Individual Retirement Account</p> <p> </dl></p>	<p>definition list</p> <p>precedes each definition term</p> <p>precedes each definition</p>

Commands	Purpose
<p><code> </code></p> <p><code> </code></p> <p><u>ex:</u> Grocery List</p> <pre> bread milk </pre>	<p>unordered/bulleted list</p> <p>precedes each list item</p>
<p><code><ol type="?"> </code></p> <p>The TYPE attribute indicates style:</p> <p>"1" = Arabic numerals (the default)</p> <p>"a" = lowercase alpha</p> <p>"A" = uppercase alpha</p> <p>"i" = lowercase Roman</p> <p>"I" = uppercase Roman</p>	<p>numbered/ordered list</p>
<p><code> </code></p> <p><u>ex:</u> Grocery List</p> <pre><ol type="A"> bread milk </pre> <p>you'd see;</p> <p>A. bread</p> <p>B. milk</p>	<p>precedes each list item</p>

HTML Exercise #2

Exercise

- Use a text editor and create an HTML file
- Play with different tags we've seen
 - create a basic html file
 - use bold, italics & preformatted text
 - change the body background color
 - use the font tag to change text display
- View HTML file with a Web browser

HTML (Advanced)

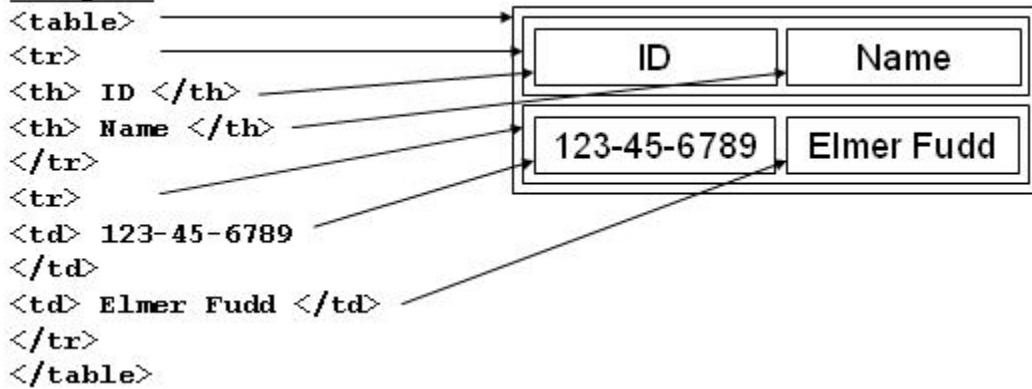
Advanced tags

Commands	Purpose
<pre> example: </pre>	image (picture) source
<pre><a href> something here email address example: Click to Yahoo! da Pres </pre>	hyperlink reference

HTML Tables

Commands	Purpose
<code><table> </table></code>	creates a table of data
<code><th> </th></code>	specifies the table header
<code><td> </td></code>	specifies the table data
<code><tr> </tr></code>	specifies table row (where does it end)

example;



HTML Form Tags

Form tags

Commands	Purpose
<code><form action="URL" method=POST></code>	creates a form to call a CGI (process)
<code></form></code>	closes the form

To create a pulldown box of values;

```
<select [multiple] name="?" [size="?"]>
<option value="?"> descriptive text </option>
</select>
```

Radio Selection boxes (pick one value);

```
<input type="radio" name="?" value="?">
```

Check all that apply with a checkbox;

```
<input type="checkbox" name="?" value="?">
```

Pass an unseen variable with the ACTION_URL;

```
<input type="hidden" name="?" value="?">
```

Different text boxes for input;

```
<input type="text" name="?" size="?">
<input type="password" name="?" size="?">
<textarea name="?" cols=? rows=?> </textarea>
```

To submit the form for processing;

```
<input type="submit" value=" different text for button ">
```

To clear or set data back to original state;

```
<input type="reset">
```

Example

```
<form action="process_to_execute.cgi">

<br> Name: <input type="text" name="name_in" size=25>
<br> Gender:
<select name="gender">
<option value="N"> unknown </option>
<option value="F"> female </option>
<option value="M"> male </option>
</select>

<br> Do you have a sweet tooth?
<input type="radio" name="sweettooth" value="yes"> Yes
<input type="radio" name="sweettooth" value="no"> No

<input type="submit" value="Press Me">
<input type="reset">
</form>
```

Form parameters

Upon submitting a form, a URL is composed of the name/value pairs and appended to the ACTION URL

Format:

```
ACTION_URL ? var1=123 & var2=abc [&...]
```

How it would look from above example:

```
process_to_execute.cgi?name_in=Smith&gender=F&sweettooth=no
```


HTML Exercise #3

Exercise

- Use a text editor and add to your HTML file
- Play with advanced tags (for example)
 - display an image file in your page
 - create a link to another site
 - create a link for an email address
 - create an HTML table
 - create an HTML form
- View HTML file with a Web browser

HTML References

References

Books

- HTML for dummies
 - any reference book will do
- Classes
 - Introduction to HTML
- Web sites
 - search Yahoo for HTML
 - <http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>
 - www.webmonkey.com
 - www.htmlgoodies.com

Building Blocks: PL/SQL



Section goal

This section provides an introduction to PL/SQL.

Introduction

Definition

Procedural Language/Standard Query Language

Basic

- Variables
- Control Structures
- if-then-else, for-loop, while-loop, exit-when and goto
- Cursor & cursor FOR loops
- Exception Handling

Advanced

- Modularity; subprograms & packages
- Information Hiding & Data Encapsulation
- Overloading

Block Structured

- logical unit of work (a module)
- basic units; named (procedures, functions and packages) and anonymous blocks
- supports the idea of *stepwise refinement*

Three Parts

- Declaration (optional); make variables known
- Execution
- Exception (optional) handler for warnings or errors

Nest

Sub-blocks allowed in Executable and Exception parts and can define subprograms in Declaration

Example of Blocking

```
Procedure myproc
is
  procedure subproc is
  begin
    dbms_output.put_line ('subproc running...');
  end;
begin
  begin
    subproc; /* call to procedure defined in
declaration section */
  exception
    when others then null;
  end;
exception
  when others then dbms_output.put_line('uh oh,
something happened!');
end myproc;
```

PL/SQL Example

```
DECLARE
    qty_on_hand NUMBER(5);
BEGIN
    SELECT quantity INTO qty_on_hand FROM inventory
        WHERE product = 'TENNIS RACKET'
        FOR UPDATE OF quantity;

    IF qty_on_hand > 0 THEN -- check quantity
        UPDATE inventory SET quantity = quantity - 1
            WHERE product = 'TENNIS RACKET';
        INSERT INTO purchase_record
            VALUES ('Tennis racket purchased', SYSDATE);
    ELSE
        INSERT INTO purchase_record
            VALUES ('Out of tennis rackets', SYSDATE);
    END IF;
    COMMIT;
END;
```

PL/SQL Modularity

Modularity

- Break a program down into manageable modules aiding in top-down design
 - reusability
 - maintainability
 - abstraction

Subprograms

- Procedures
 - takes parameters (optional) and performs an action
 - verb-noun; procedure CALCULATE_TOTALS
- Functions
 - takes parameters (optional) and returns computed value
 - preposition for boolean functions;
function IS_VALID_TYPE return boolean
 - noun for other functions;
function TOTAL_SALES return number

Packages

- database object which is a collection of logically common variables, cursors, procedures & functions
- entire package loaded into memory or can 'Pin' into memory
- DBMS_SHARED_POOL.[un]keep(<package_name>)
- status in Database must be 'VALID' in order to run

Two Parts

- Package Specification (interface)
 - public declarations that are visible to your application
- Package Body
 - implementation details (actual code) and private declarations

PL/SQL Procedures

Syntax

```
create or replace
PROCEDURE name [ (parameter list) ]
IS
<declaration section>
BEGIN
<execution section>
[EXCEPTION
<exception section>]
END name;
```

Note: [] indicate item is optional

Example

```
create or replace
PROCEDURE P_Display_Hello (parm1 varchar2)
IS
    var1 char(5) := 'there';
BEGIN
    dbms_output.put_line('Hello ' || var1 || ' ' || parm1);
EXCEPTION
    when others then
dbms_output.put_line('Problem occurred...');
END P_Display_Hello;
```

Note: need 'set serveroutput on size 10000' for put_line output.

PL/SQL Functions

Syntax

```
create or replace
FUNCTION name [ (parameter list) ]
RETURN return_datatype
IS
<declaration section>
BEGIN
<execution section>
[EXCEPTION
<exception section>]
END name;
```

Example

```
create or replace
FUNCTION get_balance (acct_id integer)
RETURN real
IS
acct_balance real := 0;
BEGIN
select bal into acct_balance from accts
    where acctno = acct_id;
return acct_balance;
END get_balance;
```

Cursors

Syntax

```
CURSOR name [ (parameter list) ]  
RETURN return_specs  
IS  
SELECT <statement>;
```

Note: return_specs is a record or rowtype in a database table, not a single field datatype (i.e. number or varchar).

Cursor FOR Loop

```
DECLARE  
    CURSOR emp_cursor (dnum NUMBER) IS  
        SELECT sal, comm FROM emp WHERE deptno = dnum;  
    total_wages    NUMBER(11,2) := 0;  
    higher_comm    NUMBER(4) := 0;  
BEGIN  
    /* The number of iterations will equal the number of rows *  
    * returned by emp_cursor.                                     */  
    FOR emp_record IN emp_cursor(20) LOOP  
        emp_record.comm := NVL(emp_record.comm, 0);  
        total_wages := total_wages + emp_record.sal +  
emp_record.comm;  
        IF emp_record.comm > emp_record.sal THEN  
            higher_comm := higher_comm + 1;  
        END IF;  
    END LOOP;  
    INSERT INTO temp VALUES (higher_comm,  
        'Total Wages: ' || TO_CHAR(total_wages));  
    COMMIT;  
END;  
/
```

Package Specification

Syntax

```
Create [or Replace] PACKAGE package_name
AS
<declaration of public variables>

<declaration of public cursors>

<declaration of public functions & procedures>

END [package_name];
      ←----- (optional)
```

Example

```
Create or Replace PACKAGE hello_world
IS
MinPerHr constant number := 60; -- public variable
/* 2nd kind of comment */
Procedure p_display_hello;
END hello_world;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_world;
whenever sqlerror exit rollback;
create public synonym hello_world for hello_world;
rem grant execute on hello_world to public;
start gurgnth hello_world    ←-- Security Patch
```

Package Body

Syntax

```
Create [or replace] package BODY package_name
IS
<declaration of private variables>
<declaration of cursors - with SELECT>
<declaration of functions & procedures - BODY>
BEGIN
...
[EXCEPTION
... ]
END package_name;
```

Example

```
Create or Replace PACKAGE BODY hello_world
IS
    curr_release constant varchar2(10) := '1.0'; -- private variable

    cursor dept_salary (dnum integer) is
select salary from emp where deptno = dnum;

    procedure p_display_hello is
begin
for sal_rec in dept_salary (20) loop
    dbms_output.put_line(sal_rec.salary);
end loop;
    end p_display_hello;
END hello_world;
/
```

PL/SQL Exercise #1

Exercise

- Create PL/SQL package specification and body
- Display Hello World using `dbms_output.put_line` function

Information Hiding & Data Encapsulation

Description

- General Implementation of Object Orientation
- More work initially but saves time and effort later on
- Information Hiding (Oracle level)
 - hide internals of a software application or database
 - separation of package specification & body and separate recompilation
- Data Encapsulation (Application level)
 - no data should be viewed or modified except through defined routines
 - put all DML into a package and require applications to use package

PL/SQL Overloading

Object Polymorphism

Given operation behaves the same even when applied to different datatypes

Give two or more subprograms the same name!

Number or scope of parameters must be different

Benefits

- convenience (black box); allows others to quickly share & use
- transfers the details from user to overloaded program

Examples

to_date & to_char

```
...
381 -- Output Procedures
382 procedure print (cbuf in varchar2 DEFAULT NULL);
383 procedure print (dbuf in date);
384 procedure print (nbuf in number);
385
386 -- Output without the newline
387 procedure prn (cbuf in varchar2 DEFAULT NULL);
388 procedure prn (dbuf in date);
389 procedure prn (nbuf in number);
390
391 -- Abbrev call to print()
392 procedure p (cbuf in varchar2 DEFAULT NULL);
393 procedure p (dbuf in date);
394 procedure p (nbuf in number);
...
```


PL/SQL Exception Handling

Event driven handling of errors

- pre-defined exceptions
 - NO_DATA_FOUND, ZERO_DIVIDE, ...
- user defined exceptions

Different section for error processing code

- transfer out of normal execution sequence

Reliable error handling

- handled & un-handled exceptions will always stop normal processing
- only catches what is defined!!!

Example

```
DECLARE
  out_of_stock  EXCEPTION;
  number_on_hand NUMBER(4);
BEGIN
  ...
  IF number_on_hand < 1 THEN
    RAISE out_of_stock;
  END IF;
  ...
EXCEPTION
  WHEN out_of_stock THEN
    -- handle the error
END;
```

PL/SQL Exercise #2

Exercise

- Use a text editor to create a PL/SQL package & body
- Use a Cursor FOR loop to find a person in SPRIDEN
- Use DBMS_OUTPUT.PUT_LINE for output
- Include an Exception Handler
- Extra credit for Overloading & Data Encapsulation!
- To install your package
`sqlplus baninst1/u_pick_it @<filename> of package.procedure>`
- Run your procedure
`exec[ute] <package.procedure>`

PL/SQL References

Books

- Oracle PL/SQL User's Guide & Reference
- Oracle PL/SQL Programming (Ant book)
 - Steven Feuerstein, O'Reilly & Associates Inc.

Tools

- TOAD (Tool for Oracle Appl Developer)
<http://www.toadsoft.com> for freeware
- SQL Navigator
http://www.quest.com/sql_navigator

Web sites

- www.revealnet.com or search Yahoo for PL/SQL
- <http://www.ezsql.net> for info and good links

Building Blocks: PL/SQL Toolkit



Introduction

This section provides an introduction to the PL/SQL Toolkit.

Introduction

Purpose

The purpose of the toolkit is to allow you to more easily generate web content from the information contained in the Banner (Oracle) Database.

The packages provide datatypes, procedures, and functions to be used by the Oracle Application Server (OAS) and Banner Self-Service.

- Installed into each Database (owned by SYS)
- Comprised of Packages;
 - HTP - HyperText Procedures
 - package of procedures; one for each HTML tag available
 - HTF - HyperText Functions
 - package of functions; one for each HTML tag available and used to *NEST* several tags together
 - OWA - Oracle Web Agent Packages
 - OWA, OWA_SEC, OWA_TEXT, OWA_COOKIE, OWA_UTIL

Oracle Web Agent Packages

OWA

functions & procedure called internally by PL/SQL cartridge

OWA_SEC

setup and return of security information

OWA_PATTERN

perform pattern matching against strings using regular expressions

OWA_TEXT

functions used internally by OWA_PATTERN

OWA_COOKIE

set and retrieve cookies from client browsers

OWA_UTIL - utility procedures

- `showsource`
print out the source of a PL/SQL stored object
- `showpage`
dumps the HTML buffer to a SQL*Plus or Server Manager screen.
- `print_cgi_env`
prints out all of the HTTP environment variables for the session.
- `redirect_url`
redirects output to another URL.
- `tableprint`
prints out an Oracle table.

PL/SQL Documentation

Documentation

Documentation for each HTP, HTF and OWA% tag is available on Oracle's technet web site

Let's review Oracle documentation -- direct your browser to

http://download-west.oracle.com/docs/cd/A97335_01/index.htm

- login, click on "Run Web Sites and Applications", scroll to Oracle PL/SQL and click on the "Using PL/SQL Gateway" link

Viewing Packages

Dumping a package

To dump a package from Oracle database;

```
unix> more dbasrc.sql
set hea off  term 999 linesize 122
col line for 9999
col text for a110 word_wrap

select line, text from DBA_SOURCE
  where name like upper('%&1%')
     and type in ('PACKAGE','PACKAGE BODY')
  order by type,line
/
exit;
```


HTP Package

Code

```
> sqlplus baninst1/u_pick_it @dbasrc htp | more
  1 package body htp as
  2
  3 /* The broken line below is intentional */
  4 NL_CHAR    constant  varchar2(1) := '
  5 ';
  6 NLNL_CHAR constant  varchar2(2) := '
  7
  8 ' ;
<skip>
 35
 36 /* STRUCTURE tags */
 37 procedure htmlOpen is
 38 begin p(htf.htmlOpen); end;
 39
 40 procedure htmlClose is
 41 begin p(htf.htmlClose); end;
Standard input ..
```

Using the HTP package

```
CREATE OR REPLACE PACKAGE Hello_World
IS
    Procedure P_DisplayHello;
END Hello_World;
/
CREATE OR REPLACE PACKAGE BODY Hello_World
IS
    Procedure P_DisplayHello IS
    BEGIN
    HTP.P('Hello World');
    END P_DisplayHello;
END Hello_World;
/
```

To invoke... Hello_World.P_DisplayHello

HTE Package

Code

```
> sqlplus baninst1/u_pick_it @dbasrc htf | more
  1 package htf as
  2
  3 /* STRUCTURE tags */
  4 /*function*/ htmlOpen          constant varchar2(7) :=
  '<HTML>';
  5 /* No attributes in HTML 3.0 spec as of 6/7/95 */
  6 /*function*/ htmlClose        constant varchar2(7) :=
  '</HTML>';
  7 /* No attributes in HTML 3.0 spec as of 6/7/95 */
  8 /*function*/ headOpen         constant varchar2(7) :=
  '<HEAD>';
  9 /* No attributes in HTML 3.0 spec as of 6/7/95 */
 10 /*function*/ headClose        constant varchar2(7) :=
  '</HEAD>';
 11 /* No attributes in HTML 3.0 spec as of 6/7/95 */
 12 function      bodyOpen (cbackground in varchar2 DEFAULT NULL,
 13 cattributes in varchar2 DEFAULT NULL) return varchar2;
 14 /*function*/ bodyClose        constant varchar2(7) :=
  '</BODY>';
```

Using the HTE Package

```
CREATE OR REPLACE PACKAGE Hello_World
IS
  Procedure P_DisplayHello;
END Hello_World;
/
CREATE OR REPLACE PACKAGE BODY Hello_World
IS
  Procedure P_DisplayHello IS
  BEGIN
    HTP.P(HTF.Bold('Hello World'));
  END P_DisplayHello;
END Hello_World;
/
```

Using PL/SQL Toolkit to Create HTML

Example

If the HTML should look like this;

```
<IMG SRC="/images/mylogo.gif" ALIGN="CENTER" ALT="Logo">
```

then the PL/SQL would look like this;

```
HTP.IMG('/images/mylogo.gif','CENTER','Logo');
```

Structure tags

Tag	Purpose
HTP.htmlOpen	creates <html>
HTP.htmlClose	creates </html>
HTP.headOpen	creates <head>
HTP.headClose	creates </head>
HTP.BodyOpen ('background', 'attributes')	creates <body background='graphic file' attributes>
HTP.BodyClose	creates </body>

Example

```
BEGIN

HTP.htmlOpen;
htp.headOpen;
htp.title('Hello World title');
htp.comment('This is my html heading section. ');
HTP.headClose;

HTP.bodyOpen(null, 'text="blue"');
htp.p('Yeah, you guessed it... Hello World. ');

HTP.bodyClose;
HTP.htmlClose;

END;
```

Question: Why is NULL in BodyOpen?

Above example would look like this;

```
<html>
<head>
<title>Hello World title </title>
<!-- This is my html heading section. -->
</head>
<body text="blue">
Yeah, you guessed it... Hello World.
</body>
</html>
```

Question: Where are the blank lines from above?

PL/SQL Toolkit Exercise #1

Exercise

- Login into database server as your training user and create a directory called "local" under \$HOME
- cd to the new directory and use your favorite editor to create a hello_world_your_name package.procedure which displays your name. (Hint: use the "create or replace command")
- Create your package by running the script as a user who has "create procedure" and "create public synonym" privileges
- Grant execute on your package to web user and create a public synonym
- Execute your hello world via the browser:
- Hint:
`http://servname.domain.edu/pls/dad/hello_world_name.p_hello`

Using PL/SQL Toolkit to Create HTML: Tags and Formatting

HEAD related tags

- `HTP.title ('title text');`
`htp.title('Banner Web Starting Page (PPRD)');`
`<title> Banner Web Starting Page (PPRD) </title>`
- `HTP.meta ('http-equiv', 'name', 'content');`
`htp.meta('refresh', null, '30');`
`<meta http-equiv="Refresh" content="30">`
- `HTP.script ('script text', 'language');`
`htp.script('script source', 'Javascript');`
`<script language=Javascript> script source </script>`
- `HTP.style ('style_text');`
`<style> style_text </style>`
- `HTP.comment ('ctext');`
`htp.comment('This is the start of P_OpenDoc.');`
`<!-- This is the start of P_OpenDoc -->`

BODY related tags

- `HTP.print('text');` `print`
- `HTP.p('text');` short version of `print`
`htp.p('Hello World!');`
`Hello World!`
- `HTP.prn;` same as `htp.print` but without the newline `'\n'`
- `HTP.prints & htp.ps;` same as `htp.print` but substitutes `<` for `'<'`,
 `>` for `'>'`, `"` for `"` and `&` for `'&'`
- `HTP.hr;` `horiz rule line`
`<hr>`
- `HTP.br;` `line break`
`
`
- `HTP.para`
`HTP.paragraph (align, nowrap, clear, attributes)`
- `htp.para;`
`<p>`
- `htp.paragraph('left','1');`
`<p align="left" NOWRAP>`

Formatting

- `HTP.bold ('text','attributes');`
`htp.bold('Important Stuff');`
` Important Stuff `
- `htp.p('text from ' || htf.bold('somewhere'));`
`text from somewhere `
- `HTP.italic ('text','attributes');`
`htp.italic('Presidential Term');`
`<i> Presidential Term </i>`
- `HTP.underline ('text','attributes');`
`htp.underline ('Document Root directory');`
`<u> Document Root directory </u>`
- `HTP.header ('size','text',...'attributes');`
`htp.header (1,'Overview');`
`<h1> Overview </h1>`
- `HTP.center ('text');`
`htp.center ('Proposal');`
`<center> Proposal </center>`
- `HTP.fontOpen ('color', 'face', 'size','attributes');`
`htp.fontOpen ('cyan', 'Courier,Courier New', '7');`
``
- `HTP.fontClose;`
``

PL/SQL Toolkit Exercise #2


Exercise

- Use a text editor to create a PL/SQL package
 - use PL/SQL toolkit to create HTML output
 - necessary html, head & body tags
 - headers, formatting, comments & print
- Put package into the Oracle DB

```
sqlplus baninst1/u_pick_it @<filename> of package>
```
- Use a Web browser to run your package
 - `http://<websvr:port>/<db>/owa/<package.procedure>`

Using PL/SQL Toolkit to Create HTML: Display Tags and Lists

Other common display tags

- HTP.img ('url', 'align', 'alt...');
`htp.img ('/school_logo.gif', 'center', 'School Logo');`
``
- HTP.anchor ('url', 'text', 'name', 'attributes');  Careful!
`htp.anchor ('http://oracle.com', 'Oracle's Homepage');`
` Oracle's Homepage `

Lists

- `HTP.dlistOpen ('clear', 'attributes');`
`htp.dlistopen;`
`<dl>`
- `HTP.dlistClose;`
`htp.dlistclose;`
`</dl>`
- `HTP.dlistTerm ('text', 'clear', 'attributes');`
`htp.dlistterm('IRA');`
`<dt> IRA`
- `HTP.dlistDef ('text', 'clear', 'attributes');`
`htp.dlistdef('IRA');`
`<dd> Individual Retirement Account`
- `HTP.ulistOpen ('clear', 'wrap', 'bullet');`
`htp.ulistopen;`
``
- `HTP.ulistClose;`
`htp.ulistclose;`
``
- `HTP.olistOpen ('clear', 'wrap', 'attributes');`
`htp.olistopen;`
``
- `HTP.olistClose;`
`htp.olistclose;`
``
- `HTP.listHeader ('text', 'attributes');`
`htp.listheader('Name');`
`<lh> Name </lh>`
- `HTP.listItem ('text', 'clear', 'bullet');`
`htp.listitem('Paul Bunyan');`
` Paul Bunyan `

List Example

```
BEGIN
  ...
  HTP.ulistOpen;
  HTP.listHeader('Registration Checklist');
  FOR my_rec IN my_cursor LOOP
    HTP.listItem(my_rec.checklist_desc);
  END LOOP;
  HTP.ulistClose;
  ...
```

List Example HTML output;

```
...
<ul>
<lh> Registration Checklist </lh>
<li> Biology 101
<li> English Refresher
<li> Physical Education
</ul>
...
```

PL/SQL Toolkit HTML Tables

Procedure calls

These toolkit package.procedure calls are used to create and populate an HTML Table:

- HTP.tableOpen ('border', 'align', 'wrap', 'clear', 'attributes')
`htp.tableopen;`
`<table>`
- HTP.tableClose;
`htp.tableclose;`
`</table>`
- HTP.tableRowOpen ('align', 'vertical', 'dp', 'wrap', 'attributes');
`htp.tablerowopen;`
`<tr>`
- HTP.tableRowClose;
`htp.tablerowclose;`
`</tr>`
- HTP.tableHeader ('text', 'align', 'dp', 'wrap', 'rowspan', 'colspan', 'attributes');
`htp.tableheader('Account');`
`<th> Account </th>`
- HTP.tableData('text','align','dp','wrap','rowspan','colspan',
'attributes');
`htp.tabledata('3820');`
`<td> 3820 </td>`

Tables Example 1

```
...
HTP.tableOpen('border=1','center',null,null,'BGCOLOR=gray');

HTP.tableRowOpen('left',null, null, null,'BGCOLOR=yellow');
HTP.tableHeader('Account');
HTP.tableHeader('Balance');
HTP.tableRowClose;

HTP.tableRowOpen;
HTP.tableData(htf.italic(acct_num));
HTP.tableData(htf.italic(acct_bal));
HTP.tableRowClose;

HTP.tableClose;
...
```

Tables Example 1 HTML output

```
...
<table "border=1" align="center" BGCOLOR=gray>
<tr ALIGN="left" BGCOLOR=yellow>
<th> Account </th>
<th> Balance </th>
</tr>
<tr>
<td> <I> 302076 </I> </td>
<td> <I> 8,204.78 </I> </td>
</tr>
</table>
...
```

Tables Example 2

```
HTP.tableOpen; -- simple table open

HTP.tableRowOpen;
HTP.tableHeader('Student ID');
HTP.tableHeader('Grade in Class');
HTP.tableRowClose;

FOR my_rec IN my_cursor LOOP
HTP.tableRowOpen;
HTP.tableData(my_rec.stu_id);
HTP.tableData(my_rec.class_grd);
HTP.tableRowClose;
END LOOP;

HTP.tableClose;
...
```

Tables Example 2 HTML output

```
<table>
<tr>
<th> Student ID </th>
<th> Grade in Class </th>
</tr>
<tr>
<td> 382482771 </td>
<td> B+ </td>
</tr>
<tr>
<td> 983278820 </td>
<td> C- </td>
</tr>
</table>
```

PL/SQL Toolkit Exercise #3

Exercise

- Show ID, Name, Activity-Date from the SPRIDEN table
- Use your existing package
- Define a cursor that selects from the SPRIDEN table
- Use an implicit cursor to loop through each record and display the items
- Use an HTML table to display results

PL/SQL Toolkit HTML Forms

Procedure calls

These toolkit package.procedure calls are used to create and populate an HTML form:

- HTP.formOpen ('url', 'method', 'target', 'enctype', 'attributes');
 http.formopen('package.procedure');
 <form action="package.procedure" method="POST">
- HTP.formClose;
 http.formclose;
 </form>
- HTP.formCheckbox ('name', 'value', 'checked', 'attributes');
 http.formcheckbox('checkbox_variable');
 <input type="checkbox" name="checkbox_variable">
- HTP.formRadio ('name','value', 'checked','attributes')
 http.formradio('insured', 'YES');
 <input type="radio" name="insured" value="YES">
- HTP.formText ('name', 'size', 'maxlength', 'value', 'attributes');
 http.formtext('textbox_variable');
 <input type="text" name="textbox_variable">
- HTP.formPassword ('name', 'size', 'maxlength', 'value', 'attributes');
 http.formpassword('pin', 6, 6);
 <input type="password" name="pin" size="6" maxlength="6">
- HTP.formHidden ('name', 'value', 'attributes');
 http.formhidden('pidm', '39282');
 <input type="hidden" name="pidm" value="39282">
- HTP.formSelectOpen ('name', 'prompt', 'size', 'attributes');
 http.formselectopen('state', 'Pick a State:');
 Pick a State:
 <select name="state">
- HTP.formSelectOption ('value','selected','attributes');
 http.formselectoption('Vermont');
 <option>Vermont
- HTP.formSelectClose;
 http.formselectclose;
 </select>

- `HTP.formReset ('value', 'attributes');`
`htp.formreset('Back to original');`
`<input type="reset" value="Back to original">`
- `HTP.formSubmit ('name', 'value', 'attributes');`
`htp.formsubmit;`
`<input type="submit">`

`htp.formsubmit('proc_pmt', 'Process Payment');`
`<input type="submit" name="proc_pmt" value="Process Payment">`

Forms Example

```
HTP.formOpen ('bwzkwpay.p_calc_refund');

HTP.formHidden('pidm', '482911');

HTP.formSelectOpen('refund_class', 'Pick class to request a
refund:');
HTP.formSelectOption('English 410');
HTP.formSelectOption('Biology 305');
HTP.formSelectOption('Physics 400');
HTP.formSelectOption('Mathematics 320');
HTP.formSelectClose;

HTP.formSubmit (NULL, ' Press here ');
HTP.formClose;
```

Forms Example generates this

```
<form action="bwzkwpay.p_calc_refund" method="POST">

<input type="hidden" name="pidm" value="482911">

Pick class to request a refund:
<select name="refund_class">
  <Option> English 410
  <Option> Biology 305
  <Option> Physics 400
  <Option> Mathematics 320
</select>

<input type="submit" value=" Press here ">
</form>
```

PL/SQL Toolkit Exercise #4

Exercise

- Prompt for Last name and show results from SPRIDEN
- Use your existing package
- Add New procedure to setup form, prompt for last-name and call existing procedure
- Change existing procedure to accept parameter which is then passed to cursor to search SPRIDEN

PL/SQL Toolkit References

Books

- Oracle Application Server
- PL/SQL Web Toolkit Reference
- Using PL/SQL Gateway

Tools

- WebAlchemy (HTML to PL/SQL converter)
<http://www.users.bigpond.com/ahobbs/>

Banner Self-Service



Introduction

This section discusses Banner Self-Service in more detail.

Introduction

Description

- Add-on to Banner baseline products
- Three-tier architecture
 - reduce network processing of data servers
 - thin client on desktop
- Uses Oracle PL/SQL Packages
- Static html & image files on Web Server
- Product Standards
 - Source code Framework
 - Naming Conventions
 - Web Page Design
 - New User Interface (CSS)
- WebTailor
- Database Source
- Web Coding Methodology

Source Code Framework

Web Interface Layer (WIL)

- process flow and application logic
- web page displays
- error handling

Reusable Object Layer (ROL)

- core business processes;
GPA, Leave Accrual, Enrollment counts

Upgrades to Self-Service products will affect the Web Interface Layer only.

- The Reusable Object Layer is used throughout BANNER and will therefore be updated through upgrades to the parent product
- An upgrade applied to Student module may affect a ROL package and therefore affect Student Self-Service

Naming Conventions

Web Interface Layer (WIL)

- First two positions - 'BW' or 'TW'
 - BW = Banner Self-Service, TW = WebTailor
- Third position is product identifier
 - G = General, S = Student, etc.
- Fourth position is object type
 - K = Package, H = HTML, G = GIF, etc.
- Remaining positions identify object

Examples of Web Interface Layer objects

- BWSKFREG (old name-HWSKFREG)
Student Self-Service registration processing package
- TWBKLIBS (TWGKLIBS)
WebTailor library package
- BWGGINFO (HWGGINFO)
General Self-Service Info GIF image
- BWGHMAIN (HWGHMAIN)
General Self-Service Help HTML file for Main Menu

Web Interface Layer source files for packages

`$BANNER_HOME/web_product/dbprocs`

- *package_name*.sql is Package Specification
- drop the last character and replace with a 1 for Package Body

bwskfreg.sql = source for package specification

bwskfre1.sql = package body

twbklibs.sql = source for package specification

twbklib1.sql = package body

Reusable Object Layer (ROL)

Follow same conventions as parent products (7 characters)

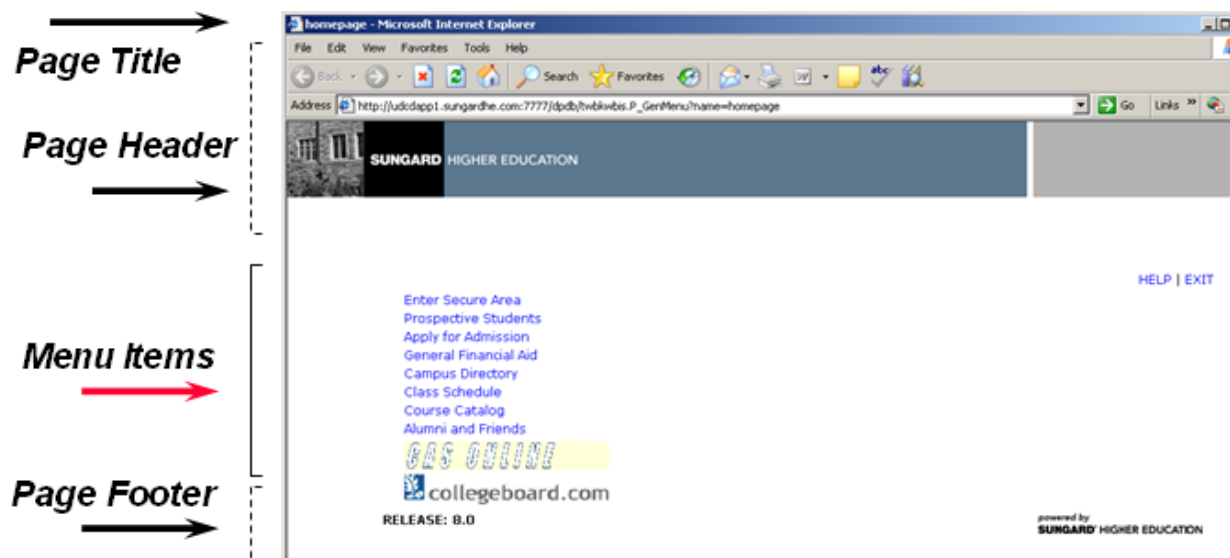
- Position 1 is Banner product
 - G = General, S = Student, etc.
- Position 2 is module identifier
 - F = Registration, A = Admissions, etc.
- Position 3 is object type
 - K = Package, F = function, etc.
- Positions 4 - 7 identify unique process

Banner Self-Service Page Design

Characteristics

- Each page generated by BANNER Self-Service has the same regions defined
- Web Interface Layer generates these regions
- WebTailor allows for customization of these regions

Web Page Layout - Homepage



Web Page Layout – Menu Page

The image shows a screenshot of a web browser displaying a menu page. The browser window title is "WebTailor Menu - Microsoft Internet Explorer". The address bar shows the URL: http://udcdapp1.sungardhe.com:7777/dpdb/twblvbls_P_GetMenu?name=bnenu_P_WebTailorMenu. The page content includes a header with the SUNGARD HIGHER EDUCATION logo, a navigation menu with items like "Personal Information", "Alumni and Friends", "Advancement Officers", "Student and Financial Aid", "Faculty Services", and "WebTailor Administration", a search box, and a list of menu items under the heading "WebTailor Menu". The footer contains the text "RELEASE: 0.0" and "powered by SUNGARD HIGHER EDUCATION".

Page Title →

Page Header Navigation

Desc →

Menu Items →

Page Footer

Web Page Layout – Application Page

The screenshot shows a web browser window displaying a page titled "Web Tailor Administration". The page has a header with navigation tabs: "Personal Information", "Alumni and Friends", "Advancement Officers", "Student and Financial Aid", "Faculty Services", and "Web Tailor Administration". Below the header is a search bar and a main content area titled "Customize a Web Menu or Procedure". The main content area contains a "Create" button, a search instruction, and a table of available procedures. The table has columns for "Procedure Name", "Procedure Description", and "Enabled Source".

Annotations on the left side of the screenshot identify the following components:

- Page Title:** Points to the browser title bar.
- Page Header Navigation:** Points to the navigation tabs.
- Page Desc & Info Text:** Points to the main content area.
- Data Elements:** Points to the table of available procedures.
- Page Footer:** Points to the bottom of the browser window.

Procedure Name	Procedure Description	Enabled Source
bwrksach_P_DispsAP	Academic Progress	Y L
bwskotm_P_ViewTran	Academic Transcript	Y L
bwskotm_P_ViewTermTran	Academic Transcript	Y L
bwrkapt_P_DispsAcptAwdAdYear	Accept Award Offers	Y B
bwk0lb_P_DispsFuncNotAllowed	Access Not Available	Y B
bmenu_P_AdvancementResources	Access to resources for Advancement Professionals	Y L
twbksite_P_DispsAccessibility	Accessibility Information	Y B

New UI & CSS

UI – User Interface

New as of Banner 6

CSS – Cascading Style Sheets

- Control the way documents are presented via browsers and are printed
- Provide more consistency and structure (ie. Uniformity) into web pages
- Improved usability for all self-service constituents of the Banner Self-Service products, including the visually-impaired
- A more flexible user interface with greater ability to customize the "look and feel" to meet institutional needs
- Greater extensibility for local modifications due to increased use of industry standards, including World Wide Web Consortium (W3C) guidelines
- An improved foundation for future changes in Web technology

CSS Statements

CSS statements

```
selector{property:value}
```

- Selector – item you want to control; e.g. a paragraph
- Property – characteristic of item; e.g. font text
- Value – what you want property to be; e.g. sans serif
- `p{font-family:"sans serif"}`

Style Class example

```
p.right {text-align: right}
p.left {text-align: left}
.center {text-align: center}    -- omit the selector clause to allow
the style to be called by any tag
```

```
<p class="right"> This paragraph will be right-aligned. </p>
<p class="left"> This paragraph will be left-aligned. </p>
<h1 class="center"> This heading will be center-aligned </h1>
```

CSS Levels

Using WebTailor, you can specify a style sheet file location at the following levels:

- System level - updated via Customize Global User Interface Settings
- Module level - updated via Customize a Web Module
- Web page level - updated via Customize a Web Menu or Procedure

These file locations are expressed as URLs; for example, `/css/`

SunGard Higher Education-Delivered Style Sheets

Style sheets

- web_defaulthome.css
settings for new Banner Self-Service UI home page.
- web_defaultmenu.css
settings for new menu pages.
- web_defaultapp.css
settings for new application pages.

- Designed to be used together
- SunGard Higher Education recommends that your institution use them

CSS References

CSS References

- HTML Validator –
<http://validator.w3.org/file-upload.html>
- Browser Support –
<http://css.nu/pointers/bugs.html>
- Browser Support - **<http://www.richinstyle.com/bugs/table.html>**
- CSS Tutorial - <http://www.w3schools.com/css/default.asp>
- CSS Validator –
<http://jigsaw.w3.org/css-validator/validator.html.en>
- CSS Guru - <http://www.meyerweb.com/eric/css/>
- CSS List Server - <http://two.pairlist.net/mailman/listinfo/css-discuss>

Internationalization Enhancement

Features

- Enable translation of text to targeted languages, and support other date and number formats
- Appropriate hard-coded strings in the web packages have been replaced by calls to a new package called G\$_NLS
- A new schema owner, TRANMGR, was created for this release
- It owns the package G\$_NLS, and the version table TMURVERS. A record has been added to the GENERAL.GTVSYSI table, with GTVSYSI_CODE=TM and GTVSYSI_DESC=Translation Manager

Banner 7.x+ API

API - Application Programming Interface

DB package controlling CRUD (create, retrieve, update and delete).

Before APIs

```
UPDATE GOBT PAC SET gobtpac_pin = v_GOBT PAC_PIN  
where gobtpac_pidm = v_GOBT PAC_PIDM;  
commit;
```

With APIs

```
GB_THIRD_PARTY_ACCESS.P_UPDATE(  
p_pidm           => v_GOBT PAC_PIDM,  
p_pin           => v_GOBT PAC_PIN,  
p_rowid         => lv_rowid);  
gb_common.p_commit;
```

Web Tailor



Section goal

This section provides an introduction to Web Tailor and its functions.

Introduction

Overview of Web Tailor

- Security
- Menu Options
- Demonstration

Purpose

- Set up global Web rules
- Define procedures, menus, menu items, role access, Info graphics and Info text
- Customize the look of web pages
- Enhanced with CSS
- Create text messages to appear on Employees Self-Service Pages and add links to other web info
- Link to online help pages
- Two main components
 - Security
 - Customization
- Refer to General Self-Service with WebTailor Implementation Guide

Security

Banner Self-Service Security

- User Name and PIN required for access
- Checks for PIN expiration date
- Cookie to ensure user is valid
- Temporary variety
- Browser must be set to accept cookies
- Application Level Security
- IP Addresses

- **User roles are identified with corresponding Banner identification**
- **Web Tailor role must be assigned**
- TWADMINU.sql must be run to assign initial WEB TAILOR ADMINISTRATOR role to an existing User ID
- This role can then assign future Web Tailor assignments, Development Officer assignments, Executive assignments, etc.

User Roles

Roles

Web roles are automatically assigned to users based upon specific records that exist in the Banner database:

- If assigned on PEAEMPL (employee form), they will be assigned as an EMPLOYEE role
- If assigned on SGASTDN (student form), they will be assigned as a STUDENT role
- If assigned on FOMPROF (finance form) they will be assigned as a FINANCE role
- If assigned on SOAROLE (faculty/advisor) they will be assigned as a FACULTY/ADVISOR role
- If assigned on APACONS (many levels of alumni) they will be assigned as a ALUMNI role

Creates additional assignments outside of baseline Banner:

- Advancement Data Tailor
- Advancement Moves Manager
- Development Officer
- Executive
- Finance Data Tailor
- Web Tailor Administrator
- Executives Self-Service Administrator

Search

Update User Roles

 Please select the roles you would like to give the user, then Submit Changes.

You have selected: Gail George

- Advancement Data Tailor**
- Advancement Moves Manager**
- Banner Channel Administration**
- Development Officer**
- EPAF Administrator**
- Executive**
- Faculty Compensation Administrator**
- Finance Data Tailor**
- HR Manager**
- Master Salary Planner**
- Web Tailor Administrator**
- Web for Executives Administrator**

[Select another User to update](#)

Web Tailor Usage

Usage

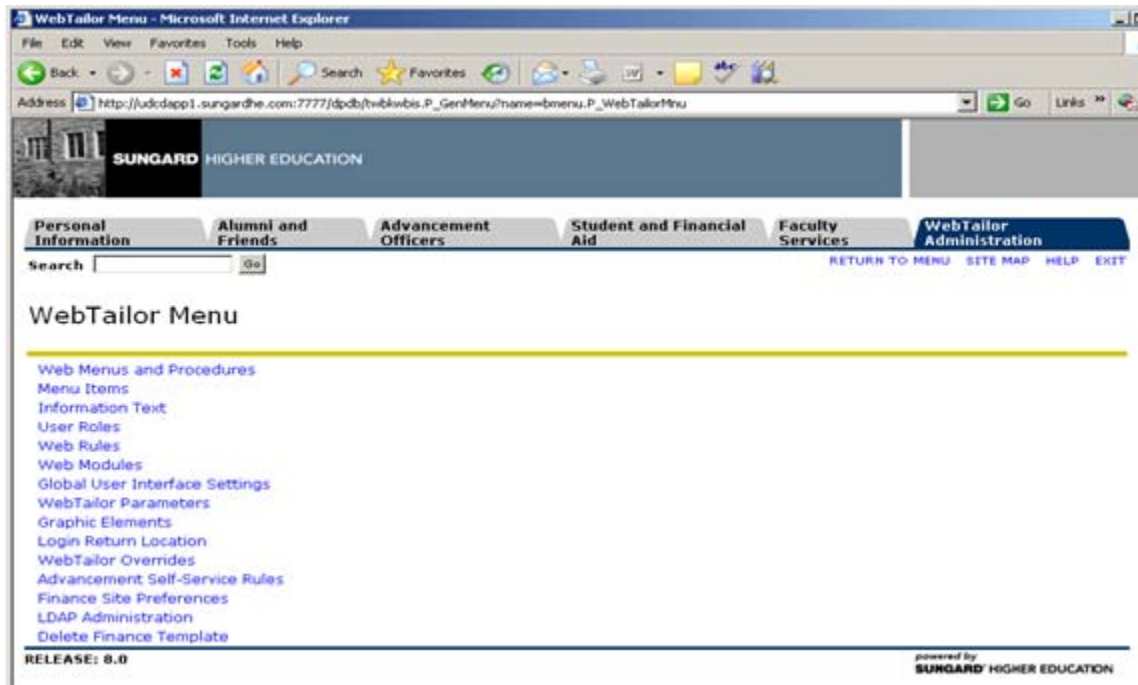
- Customize currently existing Banner Self-Service web pages
- Web Tailor Page Change = Oracle Table Change
- Fields in the web pages hold values within Oracle Tables similar to Banner data

New vs. Old

- Past releases, one version of Web Tailor pages
- Any upgrades would undo customizations
- The SS environment is now installed with baseline Web Tailor pages
- Once altered, a local copy is created
- Allows for upgrades while maintaining customization

Web Tailor Menu

Menu



Menu Option - Customize a Web Menu or Procedure

Customizing attributes

Attributes can be customized such as title, header, graphics, background color, etc.

Menus

Two types

- Bottom Links - appear at the bottom of the page, associated with one application or with an entire module
- Full Page Menus - links appear in bullet format on a full web page

Procedures (within packages)

PL/SQL or C code executes to carry out a specific function on the web

The screenshot shows a web browser window titled "Web Tailor Administration - Microsoft Internet Explorer". The address bar shows the URL: http://udcdapp1.sungardhe.com:7777/dpb/twtkwmenu_P_OptionPgWebMan. The page header includes the SunGard Higher Education logo and navigation tabs: Personal Information, Alumni and Friends, Advancement Officers, Student and Financial Aid, Faculty Services, and Web Tailor Administration. Below the header is a search bar and a "RETURN TO THE WEB TAILOR MENU SITE MAP HELP EXIT" link. The main content area is titled "Customize a Web Menu or Procedure" and contains a "Create" button and instructions for searching for a Web Menu or Procedure to customize. Below the instructions is a search form with "Search by Name:" and "OR Search by Description:" fields, and a "Search" button. At the bottom, there is a table titled "Select one of the available procedures" with columns for Procedure Name, Procedure Description, and Enabled Source.

Procedure Name	Procedure Description	Enabled	Source
bwrksaph_P_DispsAP	Academic Progress	Y	L
bwrksotrn_P_ViewTran	Academic Transcript	Y	L
bwrksotrn_P_ViewTermTran	Academic Transcript	Y	L
bwrksacct_P_DispsAcptAwdAidYear	Accept Award Offers	Y	B
bwrksolb_P_DispsFundNotAllowed	Access Not Available	Y	B
twtkwmenu_P_AdvancementResources	Access to resources for Advancement Professionals	Y	L
twtkwsite_P_DispsAccessibility	Accessibility Information	Y	B

Create a Web Menu or Procedure

Create a Web Menu or Procedure

Please update the information and Submit Changes.

* - indicates a required field.

Local	
Page Name: *	<input type="text"/>
Description: *	<input type="text"/>
Module: *	<input type="text" value="Select"/>
Comments:	<input type="text"/>
Enabled Indicator:	<input checked="" type="checkbox"/>
Non Secured Access Allowed:	<input type="checkbox"/>
Web Page Caching Override:	<input type="text" value="Use System Setting"/>
Page Title:	<input type="text"/>
Header Text:	<input type="text"/>
Header Graphic:	<input type="text" value="Select"/> Preview Image
Page CSS URL:	<input type="text"/>
Map Title:	<input type="text"/>
Help Link URL:	<input type="text"/>
Help CSS URL:	<input type="text"/>
Back Link URL:	<input type="text"/>
Back Link Text:	<input type="text"/>
Back Link Image:	<input type="text" value="Select"/> Preview Image
Back Link Menu Indicator:	<input type="checkbox"/>
Admin Secured:	<input type="checkbox"/>

Associated Roles

- Local Role**
- Advancement Data Tailor
 - Advancement Moves Manager
 - All Web Users
 - Alumni
 - Banner Channel Administration
 - Development Officer
 - EPAF Administrator
 - Employee
 - Executive
 - Faculty
 - Faculty Compensation Administrator
 - Finance Data Tailor
 - Finance user
 - Friend user
 - HR Manager
 - Master Salary Planner
 - Student
 - Web Tailor Administrator
 - Web for Executives Administrator

Customize Menu Items

Customize Information Text

Submit Changes

Reset All Fields

[Select another Web Menu or Procedure to customize](#)

Customize a Web Menu or Procedure

Customize a Web Menu or Procedure

 Please update the information and Submit Changes.

* - indicates a required field.

Baseline	
Page Name: *	bwakpldg.P_Add_A_Pledge
Description: *	Add a Pledge
Module: *	Alumni and Friends
Comments:	
Enabled Indicator:	Yes
Non Secured Access Allowed:	No
Web Page Caching Override:	Use System Setting
Page Title:	Add a Pledge
Header Text:	Add a Pledge
Header Graphic:	
Page CSS URL:	
Map Title:	
Help Link URL:	
Help CSS URL:	
Back Link URL:	bmenu.P_GivingMnu
Back Link Text:	Return to Menu
Back Link Image:	
Back Link Menu Indicator:	Yes
Admin Secured:	No

Associated Roles

Baseline Role

No	Advancement Data Tailor
No	Advancement Moves Manager
No	All Web Users
Yes	Alumni
No	Banner Channel Administration
No	Development Officer
No	EPAF Administrator
No	Employee
No	Executive
No	Faculty
No	Faculty Compensation Administrator
No	Finance Data Tailor
No	Finance user
Yes	Friend user
No	HR Manager
No	Master Salary Planner
No	Student
No	Web Tailor Administrator
No	Web for Executives Administrator

Customize Menu Items

Customize Information Text

Copy Baseline to Local

[Select another Web Menu or Procedure to customize](#)

Menu Option - Customize a Graphic Element

Define a graphic's attributes


- Location (URL)
- Description
- Size
- Alignment
- Spacing
- Special Effects (mouseover effects)
- Alternative text

Personal Information Alumni and Friends Advancement Officers Student and Financial Aid Faculty Services WebTailor Administration

Search RETURN TO THE WEB TAILOR MENU SITE MAP HELP EXIT



Customize a Graphic Element

Select Create to add a new Graphic Element.

 Or search for a Graphic element to customize.
1) Search text is not case sensitive.
2) If text is entered in 'Name' then the text in 'URL' is ignored.
3) You may use wildcards for searching. i.e. %=match any number of characters _=match 1 character.
4) If no % is entered then a match will be found if the text is located anywhere in the field. i.e. if you enter car, matches will be found on Carolina, macaroni, and boxcar

Search by Name: OR Search by URL:

Select one of the available images

Image Name	Image Description	Image URL	Image
AccessAmerica-Blue		/stugifs/hwsqaamb.jpg	
AccessAmerica-White		/stugifs/hwsqaamw.jpg	

Create a Graphic Element

Create a new Graphic Element

 Update the Graphic Element Information and then Submit Changes.

* - indicates a required field.

Graphic Element Name: *	<input type="text"/>	
Image URL: *	<input type="text"/>	Preview Image
Description:	<input type="text"/>	
Comments:	<input type="text"/>	
Image Width in pixels: *	<input type="text"/>	
Image Height in pixels: *	<input type="text"/>	
Highlighted Image URL:	<input type="text"/>	
Alternative Text:	<input type="text"/>	
Status Bar Text:	<input type="text"/>	
Image Align:	<input type="text" value="None"/>	
Image Border in pixels:	<input type="text"/>	
Vertical Spacing in pixels:	<input type="text"/>	
Horizontal Spacing in pixels:	<input type="text"/>	

[Select another Graphic Element to customize](#)

Customize a Selected Graphic Element

Customize the selected Graphic Element

 Update the Graphic Element Information and then Submit Changes.

* - indicates a required field.

Graphic Element Name: *	<input type="text" value="AccessAmerica-Blue"/>
Image URL: *	<input type="text" value="/stugifs/hwsgaamb.jpg"/> Preview Image
Description:	<input type="text"/>
Comments:	<input type="text" value="Access America icon with blue background"/>
Image Width in pixels: *	<input type="text" value="100"/>
Image Height in pixels: *	<input type="text" value="42"/>
Highlighted Image URL:	<input type="text"/>
Alternative Text:	<input type="text" value="Access America for Students"/>
Status Bar Text:	<input type="text"/>
Image Align:	<input type="text" value="middle"/>
Image Border in pixels:	<input type="text" value="0"/>
Vertical Spacing in pixels:	<input type="text"/>
Horizontal Spacing in pixels:	<input type="text"/>

[Select another Graphic Element to customize](#)

Menu Option - Customize Information Text

Customize information text

- Appears under the header
- Simple instructions
- Brief overview
 - error messages
- Change the order of the information text
- Add new information Text
- Customize existing information text

Information Text (Modify Local)

- Appears under header
- Add new or change existing information text
- Simple instructions
- Brief overview
- Error messages
- CAN ONLY MODIFY LOCAL COPY
- ONE LOCAL COPY PER BASELINE COPY

Select Information Text to Customize



Search for a web menu or procedure for which you wish to customize an information text entry..

1) Search text is not case sensitive.

2) If text is entered in 'Name' then the text in 'Description' is ignored.

3) You may use wildcards for searching. i.e. %=match any number of characters _=match 1 character.

4) If no % is entered then a match will be found if the text is located anywhere in the field. i.e. if you enter car, matches will be found on **Carolina**, **macaroni**, and **boxcar**

Search by Name: OR Search by Description:


Search

Select one of the available procedures

Procedure Name	Procedure Description	Enabled Source	
bwrksaph.P_DispsAP	Academic Progress	Y	L
bwskotrn.P_ViewTran	Academic Transcript	Y	L
bwskotrn.P_ViewTermTran	Academic Transcript	Y	L
bwrkacpt.P_DispAcptAwdAidYear	Accept Award Offers	Y	B
bwpkolib.P_DispFuncNotAllowed	Access Not Available	Y	B
bmenu.P_Advancement Resources	Access to resources for Advancement Professionals	Y	L
twbksite.P_DispAccessibility	Accessibility Information	Y	B
bwskoacc.P_ViewAcctTerm	Account Detail for Term	Y	B
bwpsdst.P_DispAcctDist	Account Distribution	Y	B
bwpsfdt.P_ListOfValuesLd	Account Distribution Lookup	Y	B
bwpktelc.P_lbd	Account Distribution Web Page	Y	B
bwskoacc.P_ViewAcctTotal	Account Summary	Y	L
bwskoacc.P_ViewAcct	Account Summary by Term	Y	L
bwfkbwsh.P_Acct_Prog_Code_Lookup	Account/Program Code Lookup	Y	B
bwkgens.p_disp_active_regs	Active Registration	Y	B
bwksreg.p_fac_active_regs	Active Registrations	Y	B
bwksreg.p_active_regs	Active Registrations	Y	B
bwksaact.P_DisAppActivities	Activities	Y	B

Reorder/Customize Info Text

Select Information Text to Customize

-  Search for a web menu or procedure for which you wish to customize an information text entry..
- 1) Search text is not case sensitive.
 - 2) If text is entered in 'Name' then the text in 'Description' is ignored.
 - 3) You may use wildcards for searching, i.e. %=match any number of characters _=match 1 character.
 - 4) If no % is entered then a match will be found if the text is located anywhere in the field. i.e. if you enter car, matches will be found on **Carolina**, **macaroni**, and **boxcar**

Search by Name: OR Search by Description:

Select one of the available procedures


Procedure Name	Procedure Description	Enabled Source	
bwrksaph.P_DispsAP	Academic Progress	Y	L
bwskotrn.P_ViewTran	Academic Transcript	Y	L
bwskotrn.P_ViewTermTran	Academic Transcript	Y	L
bwrkacpt.P_DispsAcptAwdAidYear	Accept Award Offers	Y	B
bwpkolib.P_DispsFuncNotAllowed	Access Not Available	Y	B
bmenu.P_Advancement_Resources	Access to resources for Advancement Professionals	Y	L
twbksite.P_DispsAccessibility	Accessibility Information	Y	B
bwskoacc.P_ViewAcctTerm	Account Detail for Term	Y	B
bwpkdst.P_DispsAcctDist	Account Distribution	Y	B
bwpkfdt.P_ListOfValuesLd	Account Distribution Lookup	Y	B
bwpktelc.P_Jlbd	Account Distribution Web Page	Y	B
bwskoacc.P_ViewAcctTotal	Account Summary	Y	L
bwskoacc.P_ViewAcct	Account Summary by Term	Y	L
bwfkbwsh.P_Acct_Prog_Code_Lookup	Account/Program Code Lookup	Y	B
bwckgens.p_disps_active_regs	Active Registration	Y	B
bwksreg.p_fac_active_regs	Active Registrations	Y	B
bwksreg.p_active_regs	Active Registrations	Y	B
bwksaact.P_DispsAppActivities	Activities	Y	B

Menu Option - Customize Menu Items

Characteristics

- Both bottom links and full page menus
- Change the order of the menu items
- Add new menu items
- Customize existing menu items
- Remove existing menu items
- Change link text within menu items

Select Menu Items to Customize

 Search for a web menu or procedure for which you wish to customize a menu item..
1) Search text is not case sensitive.
2) If text is entered in 'Name' then the text in 'Description' is ignored.
3) You may use wildcards for searching. i.e. %=match any number of characters _=match 1 character.
4) If no % is entered then a match will be found if the text is located anywhere in the field. i.e. if you enter car, matches will be found on Carolina, macaroni, and boxcar

Search by Name: OR Search by Description:

Select one of the available procedures

Procedure Name	Procedure Description	Enabled	Source
bwrksaph.P_DispSAP	Academic Progress	Y	L
bwskotrn.P_ViewTran	Academic Transcript	Y	L
bwskotrn.P_ViewTermTran	Academic Transcript	Y	L
bwrkacpt.P_DispAcptAwdAidYear	Accept Award Offers	Y	B
bwpkolib.P_DispFuncNotAllowed	Access Not Available	Y	B
bmenu.P_AdvancementResources	Access to resources for Advancement Professionals	Y	L
twbksite.P_DispAccessibility	Accessibility Information	Y	B
bwskoacc.P_ViewAcctTerm	Account Detail for Term	Y	B
bwpksdst.P_DispAcctDist	Account Distribution	Y	B
bwpksfdt.P_ListOfValuesLd	Account Distribution Lookup	Y	B
bwpktelec.P_Jlbd	Account Distribution Web Page	Y	B
bwskoacc.P_ViewAcctTotal	Account Summary	Y	L

Reorder or Customize Menu Items

Reorder or Customize Menu Items

 To update an individual menu item, select the associated URL.

Menu Items for: Academic Transcript

Sequence Number	Source	Link Text	URL
1	Baseline Overall Financial Aid Status		bwrksumm.P_Dispsumm
2	Baseline Financial Aid Eligibility Menu		bmenu.P_FACostMnu
3	Baseline Request Printed Transcript		bwskwtrr.p_disp_transcript_address
4	Baseline Transcript Request Status		bwskwtrr.p_disp_order_requests

Add a New Menu Item

Copy Baseline to Local

Customize the Associated Web Menu or Procedure

[Select another set of Web Menu Items to Customize](#)


Menu Option - Update User Roles

Web roles

Web roles are automatically assigned to users based upon specific records that exist in the Banner database.

- For instance, a user with a PEAEMPL record will be assigned the EMPLOYEE Role
- Use this menu option to assign additional roles if needed
- However, a user that does not have a SGASTDN cannot be assigned the STUDENT role here
- An ID can have multiple roles

Update User Role

 Enter User ID, then select Submit.

User ID:


RELEASE: 7.3

Update Roles for Selected User

Personal Information Alumni and Friends Advancement Officers Student and Financial Aid Faculty Services WebTailor Administration

Search Go [SITE MAP](#) [HELP](#) [EXIT](#)

Update User Roles

 Please select the roles you would like to give the user, then Submit Changes.

You have selected: Gail George

Advancement Data Tailor	<input checked="" type="checkbox"/>
Advancement Moves Manager	<input checked="" type="checkbox"/>
Banner Channel Administration	<input type="checkbox"/>
Development Officer	<input checked="" type="checkbox"/>
EPAF Administrator	<input type="checkbox"/>
Executive	<input checked="" type="checkbox"/>
Faculty Compensation Administrator	<input type="checkbox"/>
Finance Data Tailor	<input checked="" type="checkbox"/>
HR Manager	<input type="checkbox"/>
Master Salary Planner	<input type="checkbox"/>
Web Tailor Administrator	<input checked="" type="checkbox"/>
Web for Executives Administrator	<input checked="" type="checkbox"/>

[Select another User to update](#)

Menu Option - Customize a Web Module

Modules

Modules distinguish different sections of the product

You can create a new module specific to your site needs

You can modify an existing one

Customize a Module

Select Create to add a new Module.

Create

Choose a Module from the list and select Customize Module.

Select

Customize Module

RELEASE: 5.2

Create a new module

Create a new Module

 Please update the information and Submit Changes.

* - indicates a required field.

Module Code: *	<input type="text"/>
Module Description: *	<input type="text"/>
Module CSS URL:	<input type="text"/>
Module Help URL:	<input type="text"/>
Help CSS URL:	<input type="text"/>
Global Menu Bottom Links:	<input type="text" value="Select"/>
Current Release Number:	<input type="text"/>
Display Exit Link:	<input type="checkbox"/>
Back Image:	<input type="text" value="Select"/> Preview Image

[Select another Module to customize](#)

Customize the selected module

Customize the selected Module

 Please update the information and Submit Changes.

* - indicates a required field.

Module Code:	DEV
Module Description: *	<input type="text" value="Advancement Officers"/>
Module CSS URL:	<input type="text"/>
Module Help URL:	<input type="text"/>
Help CSS URL:	<input type="text"/>
Global Menu Bottom Links:	<input type="text" value="Select"/>
Current Release Number:	<input type="text" value="8.0"/>
Display Exit Link:	<input checked="" type="checkbox"/>
Back Image:	<input type="text" value="web_back"/> Preview Image

[Select another Module to customize](#)

Menu Option - Customize Web Rules

Characteristics

- Set the number of idle minutes before a Time-out
- Set maximum number of login attempts
- Set the PIN expiration period
- Set the *Terms of Usage* page display on/off
- Define error/warning/exit graphics
- Use descriptive text in Web Tailor pages instead of package names

Customize Web Rules

Enter the Web Rules information you wish to change and Submit Changes.

* - indicates a required field.

CGI-BIN Directory: *	<input type="text" value="/malls6_smpi"/>
CGI-BIN Admin Directory Suffix: *	<input type="text" value="Admin DIR here"/>
Web Timeout in minutes: *	<input type="text" value="90"/>
Maximum Number of Login Attempts: *	<input type="text" value="5"/>
Date Display Format Mask: *	<input type="text" value="Mon DD, YYYY"/>
Date Input Format Mask: *	<input type="text" value="MM/DD/YYYY"/>
Time Format Mask: *	<input type="text" value="HH:MI am"/>
Start Page: *	<input type="text" value="Main Menu"/>
Start Page is a Dynamic Menu:	<input checked="" type="checkbox"/>
PIN Expiration in days:	<input type="text" value="9999"/>
Display Usage Page:	<input type="checkbox"/>
Display Descriptive Names:	<input checked="" type="checkbox"/>
Enable Web Page Caching:	<input type="checkbox"/>
Use HTTP Redirection After Login:	<input type="checkbox"/>
Java Classpath:	<input type="text"/>
Document Type Definition:	<input type="text" value="HTML PUBLIC"/>
Document Type Definition FPI:	<input type="text" value="-//W3C//DTD HTML 4.0"/>
Document Type Definition URL:	<input type="text"/>

Menu Option - Customize Web Tailor Parameters

Characteristics

- Used to specify settings that can vary from site to site
- Parameter names will be referenced in stored database procedures that generate Web pages
- Identifies cookie paths
- Identifies help links
- Cascading Style Sheet definition

Customize a Web Tailor parameter

Customize a WebTailor Parameter

Select the parameter name you wish to update or delete.

Parameters

Parameter Name	Parameter Value
CCMAXPIPESIZE	8192
CCREADPIPETIME	10
CCRESPONSEPIPETIME	90
CCUSEADDRESS	Y
CCWRITEPIPETIME	5
CPBASEURL	http://udcdlum1.sungardhe.com
CPCOOKIEDOMAIN	.sungardhe.com
CPCOOKIEName	CPSESSID
CPCOOKIEPATH	/
CPINUSE	N
CPPASSWDEXP	N
CPTIMEOUTURL	/cp/ip/timeout?sys=sghessb&api=
CSSURL	/css/old_new.css
HEADERDISP	Y
HELPURL	/wtlhelp/twghhelp.htm
IDMCOOKIE	IDMSESSID
IDMCOOKIEDOMAIN	sungardhe.com
IDMCOOKIEPATH	/
IDMHEADER	IDMHEADER
IDMLOGINURI	Update Me
IDMSSO	N
IDMTIMEOUT	0
LDAPFUNCTION	twbklogn.f_idap_cpsearch
LDAPMAPUSER	DEFAULT
LDAPPWDLENGTH	10
MAXSEARCHRESULTS	100

Menu Option - Customize a Login Return Location

Characteristics

Customize the location a user is sent to after a successful login and before they reach the bmenu.P_MainMnu page (Main Page)

- Ideas?
- Seasonal Greetings
- Daily News
- Basic Information
- Site specific terms of usage
- Etc.

Customize a Login Return Location

Select Create to specify the page where the user is sent when they log back in after timing out.

Create

Choose a Login Return Location then select Customize Return Location. No Return Locations are defined

RELEASE: 8.0

powered by
SUNGARD HIGHER EDUCATION

Create a new return location

Create a New Return Location

 Update the Return Location information and Submit Changes.

* - indicates a required field.

Return Code: *

Description: *

Return Location: *

Location is: **Web Procedure** **Web Menu (generated by twbkwbis.P_GenMenu)**

Associated Packages

Add New Package Associations

1: 2: 3:

Associated Modules

Add New Module Associations

1:

2:

3:

Select another Return Location to customize

Menu Option - Customize Web Tailor Overrides

Characteristics

- Used to specify replacement values used under certain conditions within a stored database procedure
- Be cautious deleting an override
- Ensure that the value is not referenced by a stored database procedure

Customize a WebTailor Override

No WebTailor override conditions have been defined.

Create a WebTailor Override Condition

RELEASE: 5.2

Create/Customize a Web Tailor Override

Customize a WebTailor Override

 Customize your WebTailor override information and Submit Changes.

* - indicates a required field.

Override Condition: *

Replacement Value: *

Submit Changes

Reset All Fields

[Customize an additional WebTailor Override Condition](#)

RELEASE: 5.2

Menu Option - Customize GUI Settings

Characteristics

- Utilizes CSS (Cascading Style Sheets) technology
- Customize the look and feel of dynamic Web pages
- Customize global color, font and image settings
- First place to change color-Customize GUI Settings Menu Option
- Second place is CSS file - TWGHSTYL.css in the /css directory

Customize Global User Interface Settings

Update your Global User Interface Settings and Submit Changes.

* - indicates a required field.

System or Institution Name: *	<input type="text"/>
Header Image:	<input type="text" value="Select"/> Preview Image
Name of Main Menu:	<input type="text" value="My Services"/>
CSS URL:	<input type="text" value="/css/web_defaultapp.c"/>
Help URL:	<input type="text" value="/wthelp/twbhhelp.htm"/>
Help CSS URL:	<input type="text" value="/css/web_defaulthelp.c"/>
Error Image:	<input type="text" value="web_stop"/> Preview Image
Warning Image:	<input type="text" value="web_caution"/> Preview Image
Required Image:	<input type="text" value="web_required"/> Preview Image
Back Image:	<input type="text" value="Select"/> Preview Image
Submenu Image:	<input type="text" value="Select"/> Preview Image
Application Page Image:	<input type="text" value="Select"/> Preview Image
SCT Homepage URL:	<input type="text" value="http://www.sungardhe."/>

Submit Changes

Reset All Fields

RELEASE: 6.0

Web Tailor Summary

Summary

- Web Tailor allows a certain amount of customization for the Web pages according to each site's preferences
- Somewhat technical
- Access should be limited
- The *Help* icon at the top of each Web page provides some valuable information
- For more detailed information see:
General Self-Service with Web Tailor
- Questions?

Additional Topics



Introduction

This section contains additional topics related to Banner Self-Service.

Banner Self-Service Source (Packages)

Diagram

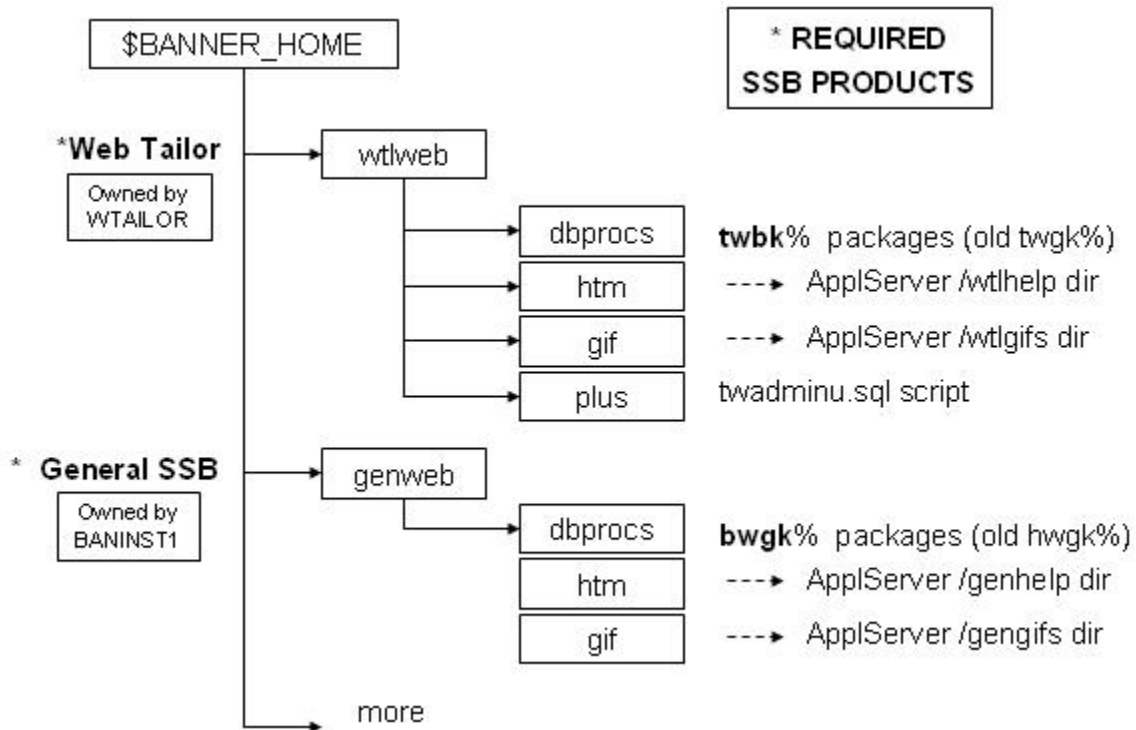


Diagram 2

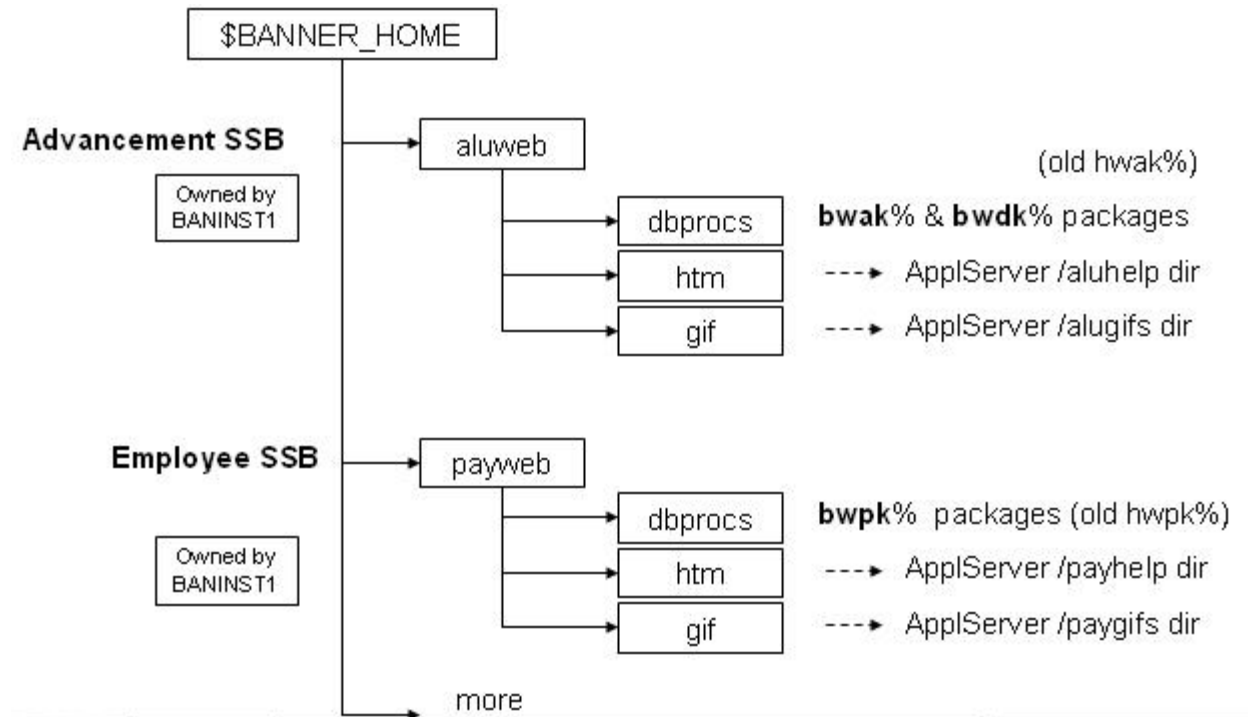
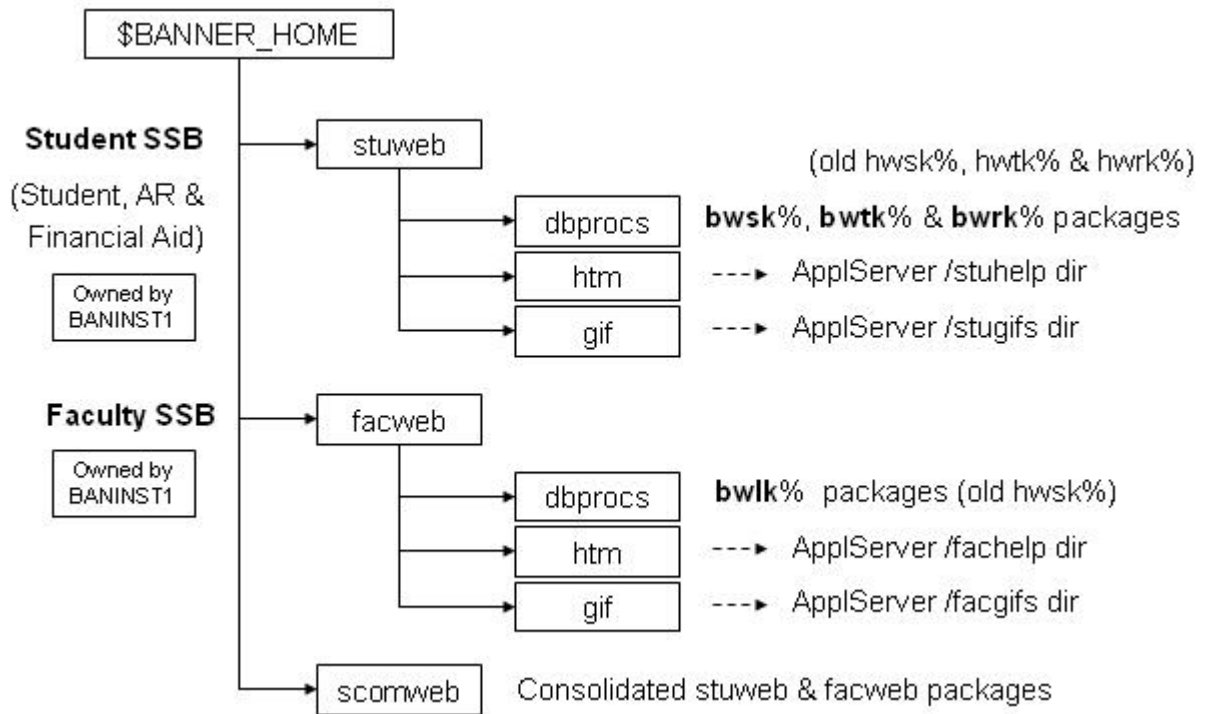


Diagram 3



Web Tailor Packages

Packages

OWNER	OBJECT_NAME	OBJECT_TYPE	STATUS
-----	-----	-----	-----
WTAILO	CMPRINFO	PACKAGE	VALID
WTAILO	CMPRMENU	PACKAGE	VALID
WTAILO	CMPRWMNU	PACKAGE	VALID
WTAILO	TWBKAUTH	PACKAGE	VALID
WTAILO	TWBKBSSF	PACKAGE	VALID
WTAILO	TWBKFRMT	PACKAGE	VALID
WTAILO	TWBKFUNC	PACKAGE	VALID
WTAILO	TWBKGLIB	PACKAGE	VALID
WTAILO	TWBKGLUI	PACKAGE	VALID
WTAILO	TWBKIMAG	PACKAGE	VALID
WTAILO	TWBKINTE	PACKAGE	VALID
WTAILO	TWBKJSCR	PACKAGE	VALID
WTAILO	TWBKLIBS	PACKAGE	VALID
WTAILO	TWBKMENU	PACKAGE	VALID
WTAILO	TWBKMODS	PACKAGE	VALID
WTAILO	TWBKMODU	PACKAGE	VALID
WTAILO	TWBKOVRR	PACKAGE	VALID
WTAILO	TWBKPARM	PACKAGE	VALID
WTAILO	TWBKRETC	PACKAGE	VALID
WTAILO	TWBKSELS	PACKAGE	VALID
WTAILO	TWBKSITE	PACKAGE	VALID
WTAILO	TWBKSLIB	PACKAGE	VALID
WTAILO	TWBKSRCH	PACKAGE	VALID
WTAILO	TWBKTABS	PACKAGE	VALID
WTAILO	TWBKTLBR	PACKAGE	VALID
WTAILO	TWBKUROL	PACKAGE	VALID
WTAILO	TWBKVALD	PACKAGE	VALID
WTAILO	TWBKWBIS	PACKAGE	VALID
WTAILO	TWBKWINF	PACKAGE	VALID
WTAILO	TWBKWMNU	PACKAGE	VALID
WTAILO	TWBKWRUL	PACKAGE	VALID
WTAILO	TWBKXNAV	PACKAGE	VALID
WTAILO	VIEWINFO	PACKAGE	VALID
WTAILO	VIEWMENU	PACKAGE	VALID
WTAILO	VIEWWMNU	PACKAGE	VALID

Viewing Packages

SQL scripts

Use SQL Script to dump package from Oracle database

- SQL Scripts
 - Source files delivered from SunGard Higher Education
 - \$BANNER_HOME/
 - wtlweb/dbprocs
 - genweb/dbprocs
 - stuweb/dbprocs
 - facweb/dbprocs
 - aluweb/dbprocs
 - payweb/dbprocs

Web Tailor Tables

Tables

TABLE_NAME	COMMENTS
TWGBFRAM	Web Tailor HTML Frameset Information table
TWGBGLUI	Web Tailor - Global Web User Interface settings table
TWGBIMAG	Web Tailor Image Information table
TWGBOVRR	Web Tailor - Web Overrides table
TWGBPARM	Web Tailor - Web Parameters table
TWGBRETC	Web Tailor Login Return Code base table
TWGBTLBR	Web Tailor Navigation Bar Information table
TWGBWMNU	Web Tailor Main Web Page Settings table
TWGBWRUL	Web Tailor - Web Rules table
TWGBWSES	Web Tailor Web SessionID table
TWGRFRRL	Web Tailor Frameset Roles table
TWGRINFO	Web Tailor Repeating Information Text table
TWGRMENU	Web Tailor Repeating Menu Item table
TWGRRETC	Web Tailor Login Return Code repeating table
TWGRROLE	Web Tailor User Roles repeating table
TWGRTLBR	Web Tailor Navigation Bar Elements table
TWGRVERS	Web Tailor Version Tracking Table
TWGRWMRL	Web Tailor Menu Roles table
TWGRWPRM	Web Tailor User Parameters table
TWTVMODU	Web Tailor Module validation table
TWTVROLE	Web Tailor User Role validation table

Banner Self-Service Coding Methodology

Elements

Four essential elements (from TWBKWBIS package)

- F_ValidUser (pidm)
- P_OpenDoc('<procedure-name>');
- P_DisplInfo('<procedure-name>');

<body of procedure>

- P_CloseDoc;

Refer to 'UI Conversion Methodology with Banner Web / April 2002'

twbkwbis.F_ValidUser

Characteristics

- Security procedure is called in EVERY Banner package which displays data to the end user
- To integrate your custom package into Banner Security, simply add the procedure call
- Procedure has PIDM parameter:
internal ID number for user logged into web
 - The PIDM is actually an output variable
 - Determined by value in TWBKWSES
- twbkwbis.F_ValidUser(pidm number)

Characteristics

- A procedure call creates the header of your HTML document
- Web Tailor allows you to define items that display at the top of document:
 - A header image
 - Top links
 - Page title
 - A page header and header links
- twbkwbis.p_opendoc('procedure defined in web tailor')
 - Creates the <HTML> <HEAD> <BODY> tags

twbkwbis.p(f)_DispInfo

Characteristics

- A procedure call creates the information text associated with a procedure in Web Tailor
- Displays information text defined in TWGRINFO table
`twbkwbis.p_dispinfo('top level procedure', 'label name of text - DEFAULT is chosen if blank')`

Print Info Text

- (text that describes the web page or displays errors or status on the page)

twbkwbis.p(f)_dispinfo (label=> ?)

- Print a caution or error message that may contain client-specific information or instructions (e.g., "Invalid Time Ticket, see the registrar for a valid time ticket.")
- This is done when redisplaying a page after an error was WebTailor

twbkwbis.p_CloseDoc

Characteristics

- A procedure call creates the footer of your HTML document
- Displays:
 - Return Link
 - Powered by SunGard Higher Education logo
 - Release Version
- Creates `</BODY>` `</HTML>` tags
- `twbkwbis.p_closedoc(<release number>)`

Banner Self-Service Coding Example

Example code

```
CREATE OR REPLACE PACKAGE BODY Hello_World
IS
  Procedure P_DisplayHello IS
    pidm number; -- DEFINE FOR OUTPUT
  BEGIN
    if not twbkwbis.F_ValidUser(pidm) then return; end if;
    twbkwbis.P_OpenDoc('Hello_World.P_DisplayHello');
    twbkwbis.P_DispInfo('Hello_World.P_DisplayHello');
    --
    -- application code goes here
    --
    twbkwbis.P_CloseDoc('5.1');
  END;
  . . .
```

SSB Exercise #1

Exercise

- Create a package to display 'Hello World' in the Banner Self-Service (SSB)
- Use the methodology on the previous slides
- You will need to register your package in WebTailor
- Create a menu item for your application page

Banner Formatting Procedures

Characteristics

- Used instead of Oracle HTP and HTF procedures
- Advantages of Banner formatting procedures
- Save development time
- Create a uniform look and feel
- Simplify code
- Only core application display needs to be developed

Common Code for Text

`twbkfmt.p(f)_printhead`

Print section headers

`twbkfmt.p(f)_printtext`

Print normal text

`twbkfmt.p(f)_printtext (class_in=> ?)`

Print bold, big, small, centered, italicized, underlined, font, colors, etc...

`twbkfmt.p_printmessage`

Print a note, warning or error message that does not contain client-specific information.

`twbkfmt.f_printrequired`

Include the mandatory indicator * as part of a form label

`twbkfmt.p_printrequiredmsg`

Print the message that a field is required

Banner Standards for Tables

Standard formats

Banner HTML tables have three standard formats for opening a html table:

Data Display: Format to display uniform data
`twbkfrmt.P(F)_TableOpen('DATADISPLAY');`

Data Entry: Format to display input for forms
`twbkfrmt.P(F)_TableOpen('DATAENTRY');`

Plan Display: Format to display items, links or images
`twbkfrmt.P(F)_TableOpen('PLAIN');`

Using Tables to Display Data

Table tags

Before you start creating your first table, please reference the standards.doc for a list of table tags.

- twbkfrmt.P(F)_TableDataOpen(datatype => 'nontabular'); to format non-number data with <TD some attributes> tag
- twbkfrmt.P(F)_TableDataLabel('Label of data'); to generate a label next to your data
- twbkfrmt.P(F)_TableRowOpen; to start a new row
- twbkfrmt.p_tableopen (ccaption => 'Display's Address Types');

Table example

```
twbkfrmt.P_TableOpen('DATADISPLAY',  
    ccaption => 'Your E-mail Address');  
twbkfrmt.P_TableRowOpen;  
twbkfrmt.p_tabledataheader ('E-mail Address');  
twbkfrmt.P_TableData ('flastnam@sungardhe.com');  
twbkfrmt.p_tablerowclose;  
twbkfrmt.p_tableclose;
```

SSB Exercise #2

Exercise

- Create SSB application page to prompt for a last name in a form
- Show SPRIDEN results in a table using SSB formatting calls.
- Hint: Use the 'DATADISPLAY' option to make your work easy

Linking to Another Page

Procedure

- Any Page that displays in the browser URL box needs to be registered with Web Tailor
- Use Banner function to find "/pls/dad" parameter
 - Allows you to connect to another procedure
 - twbkwbis.F_CgiBin
- Use Banner procedure call
 - twbkrfmt.P_PrintAnchor('link', ctext >= 'link text');

Example of Banner link

```
twbkrfmt.P_PrintAnchor(twbkwbis.f_cgibin  || 'bwlkoids.P_FacEnterID'  
||  
'?term=' || hold_term ||  
'&calling_proc_name=' || calling_proc_name ||  
'&calling_proc_name2=' || calling_proc_name2,  
ctext=>'Enter Student ID Directly');
```

-- creates

```
<A HREF="/axp7t50/owa/bwlkoids.P_FacEnterID  
?term=199910  
&calling_proc_name=bwlkfrad.P_FacAddDropCrse  
&calling_proc_name2=bwlkfrad.P_FacAddDropCrse">Enter Student ID  
Directly</A>
```

Adding Packages to SSB

Procedure

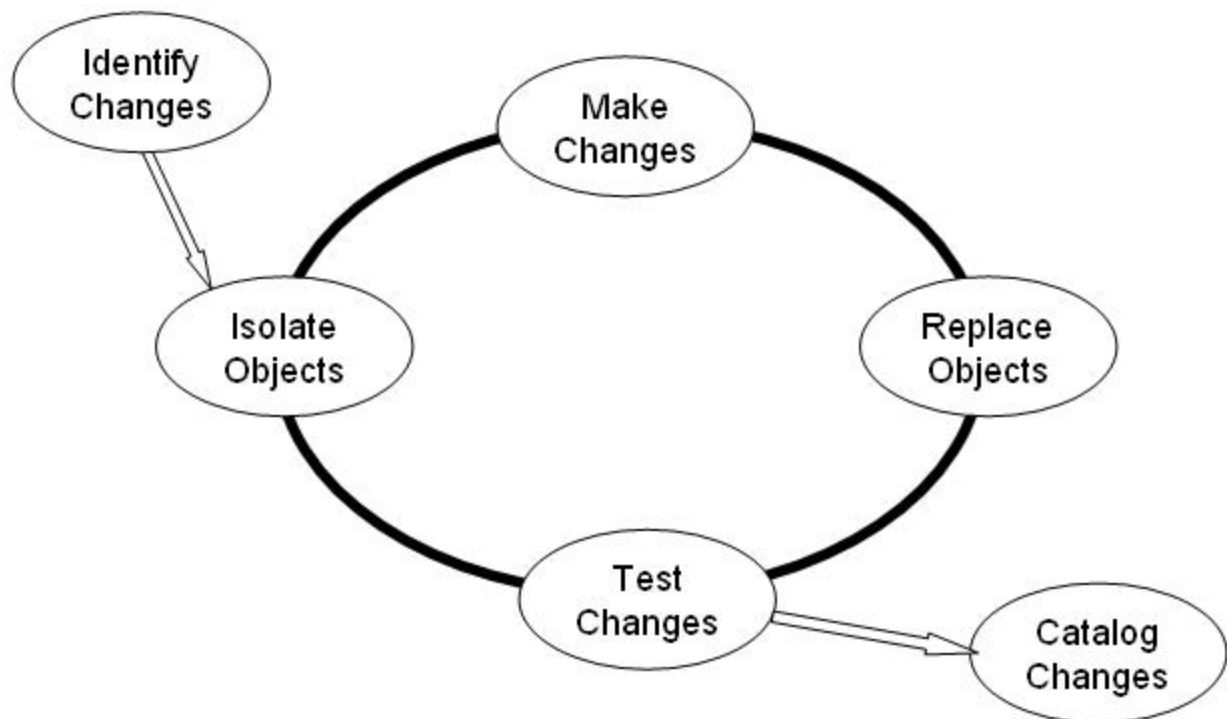
- Create the package under the BANINST1 user and grant execute on the package to PUBLIC (security patch; use WWW_USER).
- Register the necessary procedures in new package in Web Tailor:
 - Customize a Web Menu or Procedure
 - Create new procedure(s). Use an existing procedure as a guideline of what to enter in the fields.
- Add link to new procedure
 - Customize a Web Menu or Procedure:
 - Create new menu, and under "Customize Menu Items", "Add a New Menu Item" for each procedure. Click "Database Procedure", but not "Submenu Indicator" for the procedure menu item(s).
- Add link from the new menu to a previously existing menu
 - choose the previously existing menu under "Customize a Set of Menu Items" (e.g. choose amenu.P_MainMnu for the Banner Web Main Menu), add the new menu as a new menu item, and click both "Database Procedure" and "Submenu Indicator".
- If you have a help text file for the menu/procedure(s), place the file in the appropriate help directory on your web server, and enter the URL (e.g. /genhelp/textfile.htm) in the "Help Link (URL)" field for your menu/procedure.
- If you want Info text displayed for the menu/procedure(s), there is a "Customize Information Text" button at the bottom of the "Customize a Web Menu or Procedure" form, as well as a separate WebTailor menu item "Customize a Set of Information Text". (Your procedure must have a call to the "twbkwbis.P_DisplInfo" procedure in order for the info text to be displayed.)

Modifying Banner Self-Service

Steps

- Identify the Changes
- Isolate the Objects
- Make Changes
- Replace or Create Objects
- Test/Review Changes
- Catalog Changes

Cycle



Identify changes

- Is the required change clearly defined?
- Locate where in Banner Self-Service the change would be done
- Note the Banner Self-Service package and procedure in the URL

Isolate objects

- As delivered all Banner objects are in;
\$BANNER_HOME
- Within this directory, where is the files we need to modify?
- Copy the packages or other files to another location
- NEVER modify in \$BANNER_HOME!
- If using Source Control software, catalog the initial version

Make changes

- Using a Text editor, edit the package
 - Unix: vi or emacs
 - Windows: Notepad (transport via ftp or samba)
- Start small
 - put in a simple display command
- Change may require new package or may need to create new objects like tables
- Maintain Naming standards for new objects like BWZKxxxx.sql & BWZKxxx1.sql

Replace/create objects

- Change doesn't go into effect until object has been created or replaced in the Oracle database
- Use SQL*Plus to source the file and overwrite the existing/baseline object in Oracle
`sqlplus baninst1/u_pick_it @<filename>`
- Find and fix any errors
- Remember to make small modifications at a time

Test/review changes

- Go back into Banner Self-Service and note the outcome
- if simple change, can just click on the browser REFRESH button
- Did you get the desired results?
- Repeat the cycle

Catalog changes

- Once change has been completed and accepted, the new and/or changed objects need to be moved to the Production database
 - copy file(s) to local modified directory
 - source files with SQL*Plus to put into database
- Remember to analyze and re-apply your changes when a new release of the package is delivered

Example

Let's add the SPRADDR_ACTIVITY_DATE to the Banner Self-Service ViewAddress web page.

Security



Introduction

This section discusses security considerations with respect to Banner Self-Service.

Introduction

Introduction

- How much can you afford?
 - To lose?
 - To spend?
- No security is 100%!
- Internet Traffic Security
 - hackers to app and/or host server
 - sniffers or eavesdropping
- Banner Self-Service Security

Internet Traffic Security

Firewalls

- prevent unauthorized network access and attacks by protecting the points of network entry

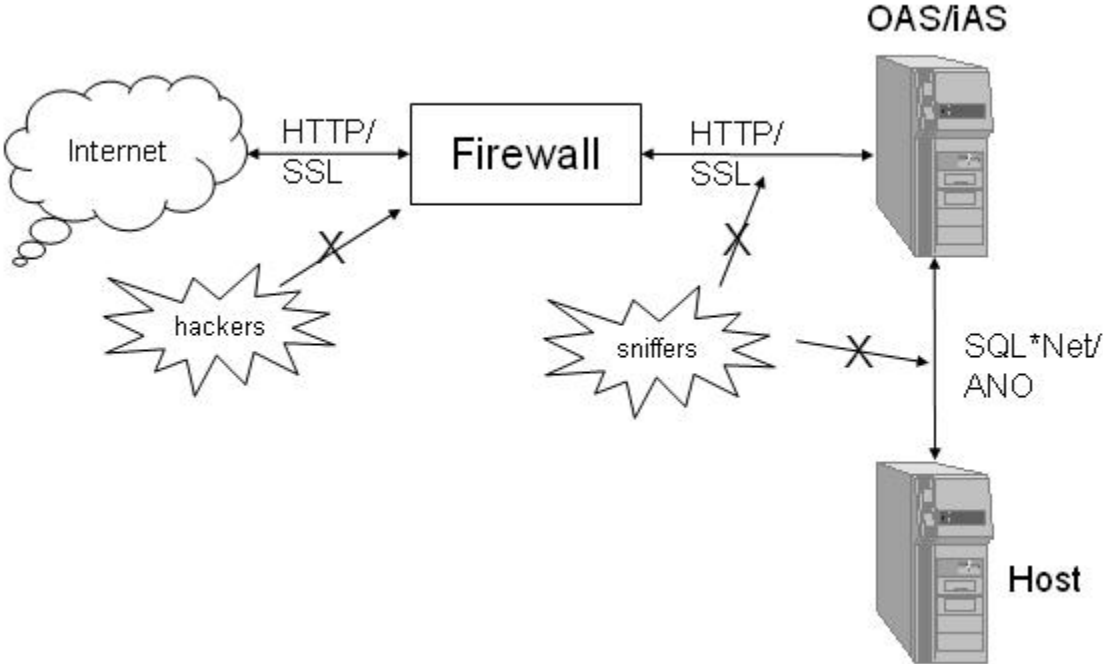
SSL - Secured Socket Layer

- encrypts packets between client & server
- Industry leading security for Web servers.
- Requires VERISIGN certificate.

ANO - Advanced Networking Option

Oracle Add-on to encrypt SQL*Net communication

Diagram



Security References

Books

- Hacking Exposed, Network Security Secrets & Solutions. McClure, Scambray & Kurtz. Osborne.
- Check bookstore

Classes

- Oracle, SunGard Higher Education, etc...

Web sites

- http://www.oracle.com/database/documents/secure_app_deployment.pdf
- <http://www.hackingexposed.com>

Banner Self-Service Security

Characteristics

- Oracle DB user
- Banner User ID & PIN
- WEBID
- Web roles
- Product-specific security features

Oracle Database Security

Characteristics

- WWW_USER Oracle acct in Database
 - created as part of DAD configuration
 - connect, resource, create session, execute any procedure
- No direct table privileges
- Security patch
 - direct grants to PUBLIC changed to WWW_USER

User ID and PIN

Characteristics

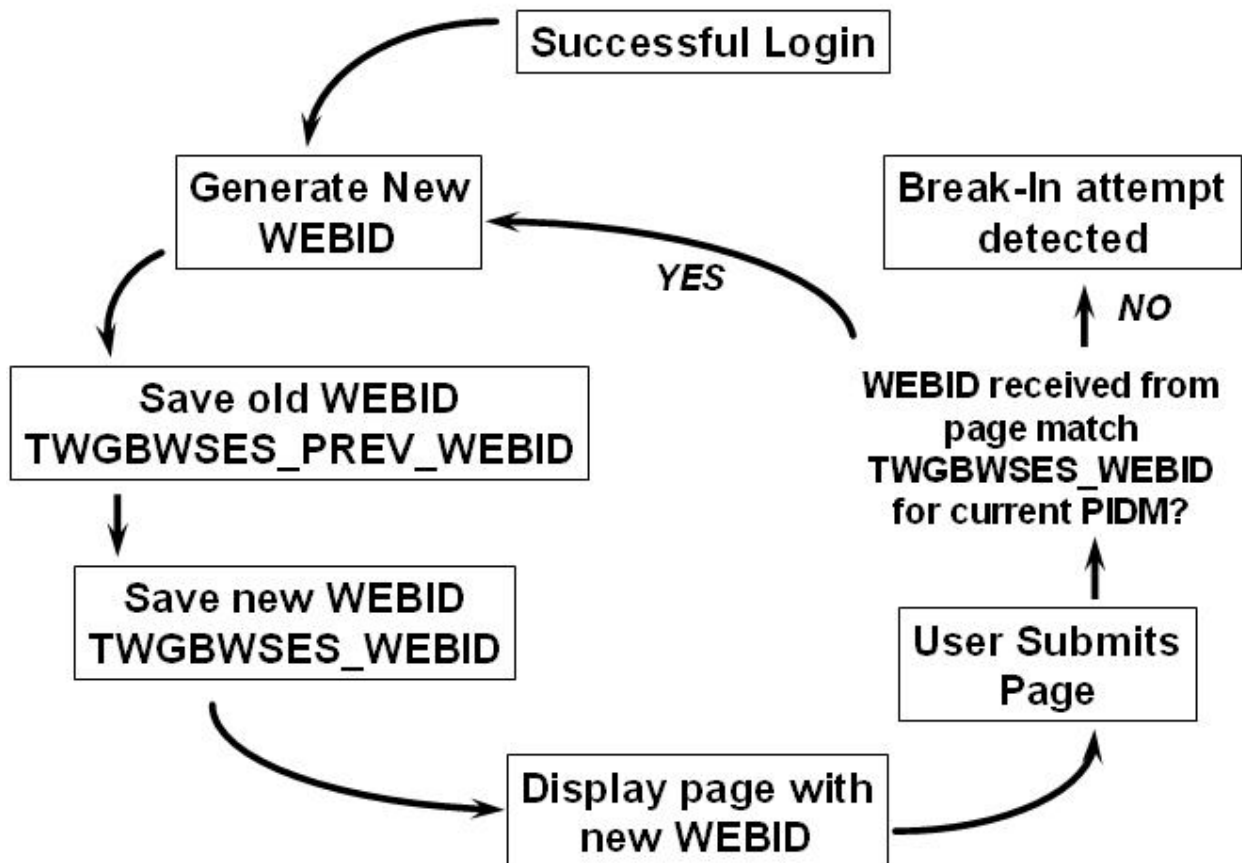
- Access to any module except Admissions is preceded by userid & PIN entry
- GOATPAC/GOATPAD establishes third party access (Banner table; GOBTPAC)
- Access to Admissions also require userid and PIN entry
 - no prior setup required as it is public access
- WebTailor Web Rules page
 - pin expirations, session timeouts, login attempts

Banner WEBID Security

Characteristics

- Unique WEBID is generated for each transaction sent to the database
- Six alphanumeric characters (i.e. 7GU68L)
- TWGBWSES is ordered by PIDM to ensure that the ID logged on is accessing only records associated with that ID
- WEBID is checked each time a new page is opened
- PIDM & WEBID are passed back & forth via Web Cookies

Cycle



TWGBWSES

Code

```
select twgbwses_pidm,  
       substr(f_format_name(twgbwses_pidm, 'LF30'),1,25) Name,  
       twgbwses_pidm tw_pidm,  
       substr(TO_CHAR(twgbwses_last_access, 'DD-MON-YY  
HH24:MI:SS'),1,20) Last_Access,  
       twgbwses_login_attempts login_attempts  
  from twgbwses  
 where twgbwses_prev_webid is not null  
       and twgbwses_webid is not null  
       and sysdate < (twgbwses_last_access +  
([minutes_in]/[minutesPerDay]))  
--[minutes_in] is the value of twgbwrul_time_out from twgbwrul  
table.  
--[minutesPerDay] is a constant equal to 1440  
 order by 2;
```

Banner Self-Service Roles

Roles

Used on Banner WebTailor Roles page to determine which groups of web users can access web pages

- STUDENT role = if ID has a SGBSTDN record
- FACULTY role = if ID has a SIBINST record
- EMPLOYEE role = if ID has a PEBEMPL record
- ALUMNI role = if ID has an APBCONS record
- WTAILORADMIN role = assigned

Product-Specific Features

General Self-Service

uses Web Roles to determine which group of web users can update which address types

Faculty Self-Service

will only allow faculty to update grades for classes they are assigned to

Student Self-Service

will only register a student for a class after all other BANNER registration checks are applied

Additional Information

Sources

- Banner Self-Service Documentation
- List Servers; BINFO
 - Weekly Known Issues
- Customer Support Center
 - FAQs & Known Issues
- Action-Line
 - AMBANWEB@sungardhe.com
- Consulting Services

Exercise Answers



Section goal

This section contains the answers to the exercises found in this workbook.

HTML Exercise #1

HTML Exercise #1

- Create new file on desktop called html1.htm (or other appropriate name you choose).
- Put HTML commands below in the file.
- Save the file.
- Open file using an internet browser.

```
<html>
<!-- this is how to do comments -->
<head>
<title> Your Title appears at the very top of window bar </title>
</head>

<body>
<h1> Hello World </h1>

<p> This is our first web document!

</body>
</html>
```

HTML Exercise #2

HTML Exercise #2

- Create new file on desktop called html2.htm (or other appropriate name you choose).
- Put HTML commands below in the file.
- Save the file.
- Open file using an internet browser.

```
<html>
<!-- this is how to do comments -->
<head>
<title> Your Title appears at the very top of window bar </title>
</head>

<body bgcolor=cyan text="#FFFFFF">

<p> This is our second web document!

<hr> Horizontal rule line

<h1> Header 1 </h1>
<h2> Header 1 </h2>
<h3> Header 1 </h3>
<h4> Header 1 </h4>
<h5> Header 1 </h5>
<h6> Header 1 </h6>

<br> Line break

<b> this is going to be bold! </b>

<I> this will be <b> bold & italics </b> </I>

<pre>
Some code sample here ...
</pre>
```



```
<font size=7> This is the biggest! </font>
<font size=-1> Slightly smaller. </font>
<font size=1> The smallest you can imagine. </font>

<font color=blue> This should stand out. </font>
<font color=red> This will really <font size=1> stand </font>
  out. </font>

</body>

</html>
```

HTML Exercise #3

HTML Exercise #3

- Create new file on desktop called html3.htm (or other appropriate name you choose).
- Put HTML commands below in the file.
- Save the file.
- Open file using an internet browser.

```
<html>
<!-- this is how to do comments -->
<head>
<title> Your Title appears at the very top of window bar </title>
</head>

<body bgcolor=cyan text="#FFFFFF">

<p> This is our third web document!

<table>
<tr>
<th> ID </th>
<th> Name </th>
</tr>
<tr>
<td> 123-45-6789
</td>
<td> Elmer Fudd </td>
</tr>
</table>

<form action="process_to_execute.cgi">

<br> Name: <input type="text" name="name_in" size=25>
<br> Gender:
<select name="gender">
<option value="N"> unknown </option>
<option value="F"> female </option>
<option value="M"> male </option>
</select>
```

```
<br> Do you have a sweet tooth?  
<input type="radio" name="sweettooth" value="yes"> Yes  
<input type="radio" name="sweettooth" value="no"> No  
  
<input type="submit" value="Press Me">  
<input type="reset">  
</form>  
  
</body>  
</html>
```

PL/SQL Exercise #1

PL/SQL Exercise #1

- Create new file on the database server called plsql1.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @plsql1
- execute hello_world.p_display_hello;

Remember, 'set serveroutput on size 50000'.

```
Create or Replace PACKAGE hello_world
IS
    Procedure p_display_hello;
END hello_world;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_world;
whenever sqlerror exit rollback;
create public synonym hello_world for hello_world;
start gurgnth hello_world;

Create or Replace PACKAGE BODY hello_world
IS

    procedure p_display_hello is
    begin
        dbms_output.put_line('Hello PL/SQL World.');
```

```
end p_display_hello;
END hello_world;
/
```

PL/SQL Exercise #2

PL/SQL Exercise #2

- Create new file on the database server called plsql2.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @plsql2
- execute hello_world.p_display_spriden;

Remember, 'set serveroutput on size 50000'.

```
Create or Replace PACKAGE hello_world
IS
    Procedure p_disp_spriden;
END hello_world;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_world;
whenever sqlerror exit rollback;
create public synonym hello_world for hello_world;
start gurgnth hello_world;
```

```

Create or Replace PACKAGE BODY hello_world
IS
    Cursor c_spriden is
        Select SPRIDEN_pidm
            ,SPRIDEN_id
            ,SPRIDEN_last_name
            ,SPRIDEN_first_name
            ,SPRIDEN_mi
            ,SPRIDEN_entity_ind
            ,SPRIDEN_change_ind
            ,SPRIDEN_activity_date
        from SPRIDEN
        where SPRIDEN_last_name = 'Abbe';

    procedure p_disp_spriden is
    begin
        dbms_output.put_line('Hello_world.p_disp_spriden is
starting...');
        for rec in c_spriden loop
            dbms_output.put_line(rec.spriden_pidm || ' ' ||
                rec.spriden_id || ' ' ||
                rec.spriden_last_name || ', ' ||
                rec.spriden_first_name || ' ' ||
                rec.spriden_mi || ' ' ||
                rec.spriden_entity_ind || ' ' ||
                rec.spriden_change_ind || ' ' ||
                rec.spriden_activity_date);

            end loop;
        end p_disp_spriden;
    END hello_world;
/

```

PL/SQL Toolkit Exercise #1

PL/SQL Toolkit Exercise #1

- Create new file on the database server called web1.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @web1
- Display results

http://<your environment>/hello_web1.p_display_hello;

```
Create or Replace PACKAGE hello_web1
IS
    Procedure p_display_hello;
END hello_web1;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_web1;
whenever sqlerror exit rollback;
create public synonym hello_web1 for hello_web1;
start gurgnth hello_web1;
```

```
Create or Replace PACKAGE BODY hello_web1  
IS
```

```
    procedure p_display_hello is  
    begin  
        HTP.htmlOpen;  
        http.headOpen;  
        http.title('Hello World title');  
        http.comment('This is my html heading section.');
```

```
        HTP.headClose;
```

```
        HTP.bodyOpen;  
        http.p('Yeah, you guessed it... Hello World.');
```

```
        HTP.bodyClose;  
        HTP.htmlClose;  
    end p_display_hello;
```

```
END hello_web1;  
/
```


PL/SQL Toolkit Exercise #2

PL/SQL Toolkit Exercise #2

- Create new file on the database server called web2.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @web2
- Display results

http://<your environment>/hello_web2.p_display_hello;

```
Create or Replace PACKAGE hello_web2
IS
    Procedure p_display_hello;
END hello_web2;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_web2;
whenever sqlerror exit rollback;
create public synonym hello_web2 for hello_web2;
start gurgnth hello_web2;
```

```
Create or Replace PACKAGE BODY hello_web2
IS

    procedure p_display_hello is
    begin
        HTP.htmlOpen;
        htp.headOpen;
        htp.title('Hello Web #2 title');
        htp.comment('This is my html heading section. ');
        HTP.headClose;
```

```

HTP.bodyOpen(null,'text="blue"');
htp.p('Hello World.');
```

htp.hr; -- horizontal line rule

htp.para; -- paragraph

```

htp.bold('Important Stuff');
htp.p('text from ' || htf.bold('somewhere'));
htp.italic('Presidential Term');
```

htp.br; -- line break

```

htp.underline ('Need to underline something?);
```

```

htp.header (1,'Header 1');
htp.header (2,'Header 2');
htp.header (3,'Header 3');
htp.header (4,'Header 4');
htp.header (5,'Header 5');
htp.header (6,'Header 6');
```

```

htp.center ('Only this is centered.');
```

htp.centerOpen;

```

htp.p ('But this is centered.');
```

htp.p ('and this...');

```

htp.p ('and this... you get the idea.');
```

htp.centerClose;

```

htp.fontOpen ('cyan', 'Courier,Courier New', '7');
```

htp.p('We can do some poorly designed stuff here.');

```

HTP.fontClose;
```

```

HTP.bodyClose;
HTP.htmlClose;
end p_display_hello;
```

```

END hello_web2;
/
```

PL/SQL Toolkit Exercise #3

PL/SQL Toolkit Exercise #3

- Create new file on the database server called web3.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @web3
- Display results

http://<your environment>/hello_web3.p_disp_spriden;

```
Create or Replace PACKAGE hello_web3
IS
    Procedure p_disp_spriden;
END hello_web3;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_web3;
whenever sqlerror exit rollback;
create public synonym hello_web3 for hello_web3;
start gurgnth hello_web3;
```

```
Create or Replace PACKAGE BODY hello_web3
IS
```

```
Cursor c_spriden is
  Select SPRIDEN_pidm
         ,SPRIDEN_id
         ,SPRIDEN_last_name
         ,SPRIDEN_first_name
         ,SPRIDEN_mi
         ,SPRIDEN_entity_ind
         ,SPRIDEN_change_ind
         ,SPRIDEN_activity_date
  from SPRIDEN
  where SPRIDEN_last_name = 'Abbe';

procedure p_disp_spriden is
begin
  HTP.htmlOpen;
  http.headOpen;
  http.title('Hello World title');
  http.comment('This is my html heading section.');
```

```
  HTP.headClose;

  HTP.bodyOpen;
  http.p('Yeah, you guessed it... Hello World.');
```

```
  HTP.tableOpen; -- simple table open

  HTP.tableRowOpen;
  HTP.tableHeader('PIDM');
  HTP.tableHeader('Banner ID');
  HTP.tableHeader('Name');
  HTP.tableHeader('Entity Ind');
  HTP.tableHeader('Change Ind');
  HTP.tableHeader('Activity Date');
```

```
  HTP.tableRowClose;
```

```
FOR rec IN c_spriden LOOP
    HTP.tableRowOpen;
    HTP.tableData(rec.spriden_pidm);
    HTP.tableData(rec.spriden_pidm);
    HTP.tableData(rec.spriden_last_name || ', ' ||
rec.spriden_first_name || ' ' || rec.spriden_mi);
    HTP.tableData(rec.spriden_entity_ind);
    HTP.tableData(rec.spriden_change_ind);
    HTP.tableData(rec.spriden_activity_date);
    HTP.tableRowClose;
END LOOP;

HTP.tableClose;

HTP.bodyClose;
HTP.htmlClose;
end p_display_hello;

END hello_web3;
/
```

PL/SQL Toolkit Exercise #4

PL/SQL Toolkit Exercise #4

- Create new file on the database server called web4.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @web4
- Display results

http://<your environment>/hello_web4.p_create_form;

```
Create or Replace PACKAGE hello_web4
IS
Procedure p_disp_spriden (lname_in varchar2);
Procedure p_create_form;
END hello_web4;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_web4;
whenever sqlerror exit rollback;
create public synonym hello_web4 for hello_web4;
start gurgnth hello_web4;
```

```
Create or Replace PACKAGE BODY hello_web4
IS
```

```
Cursor c_spriden (lname_in varchar2) is
  Select SPRIDEN_pidm
         ,SPRIDEN_id
         ,SPRIDEN_last_name
         ,SPRIDEN_first_name
         ,SPRIDEN_mi
         ,SPRIDEN_entity_ind
         ,SPRIDEN_change_ind
         ,SPRIDEN_activity_date
  from SPRIDEN
  where upper(SPRIDEN_last_name) = upper(lname_in);
```

```
procedure p_disp_spriden (lname_in varchar2) is
begin
  HTP.htmlOpen;
  http.headOpen;
  http.title('Hello World title');
  http.comment('This is my html heading section.');
```

```
  HTP.headClose;

  HTP.bodyOpen;
  http.p('Yeah, you guessed it... Hello World.');
```

```
  HTP.tableOpen; -- simple table open

  HTP.tableRowOpen;
  HTP.tableHeader('PIDM');
  HTP.tableHeader('Banner ID');
  HTP.tableHeader('Name');
  HTP.tableHeader('Entity Ind');
  HTP.tableHeader('Change Ind');
  HTP.tableHeader('Activity Date');
  HTP.tableRowClose;
```

```

-- parameter we get for procedure, we simply pass to cursor

FOR rec IN c_spriden (lname_in) LOOP
    HTP.tableRowOpen;
    HTP.tableData(rec.spriden_pidm);
    HTP.tableData(rec.spriden_pidm);
    HTP.tableData(rec.spriden_last_name || ', ' ||
rec.spriden_first_name || ' ' || rec.spriden_mi);
    HTP.tableData(rec.spriden_entity_ind);
    HTP.tableData(rec.spriden_change_ind);
    HTP.tableData(rec.spriden_activity_date);
    HTP.tableRowClose;
END LOOP;

HTP.tableClose;

HTP.bodyClose;
HTP.htmlClose;
end p_display_hello;

procedure p_create_form is
begin
    HTP.htmlOpen;
    http.headOpen;
    http.title('Hello World title');
    http.comment('This is my html heading section. ');
    HTP.headClose;

    HTP.bodyOpen;
    http.p('Yeah, you guessed it... Hello World. ');

```



```
        HTP.formOpen ('hello_web4.p_disp_spriden');
        Htp.header (1, 'Please enter a last-name to search for:');
        http.formtext('lname_in'); -- must match the input
parameter
        http.br;
        HTP.formSubmit (NULL, ' Search Now! ');
        HTP.formClose;

        HTP.bodyClose;
        HTP.htmlClose;
end p_create_form;

END hello_web4;
/
```

Banner Self-Service Exercise #1

Banner Self-Service Exercise #1

- Create new file on the database server called ssb1.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @ssb1
- Using WebTailor, register your package.procedure to make is accessible.
- Create menu item to link your application page to a menu.
- Click on your menu link to try out your page.

```
Create or Replace PACKAGE hello_ssb1
IS
    Procedure p_display_hello;
END hello_ssb1;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_ssb1;
whenever sqlerror exit rollback;
create public synonym hello_ssb1 for hello_ssb1;
start gurgnth hello_ssb1;
```

```
Create or Replace PACKAGE BODY hello_ssb1
IS
    curr_release CONSTANT varchar2(10) := '1.0';

    procedure p_display_hello is
        pidm number; -- DEFINE FOR OUTPUT
    begin
        if not twbkwbis.F_ValidUser(pidm) then return; end if;
        twbkwbis.P_OpenDoc('hello_ssb1.p_display_hello');
        twbkwbis.P_DispInfo('hello_ssb1.p_display_hello');
        --
        -- application code goes here
        --
        http.p('Yeah, another Hello World but for SSB this time.');
```

```
        twbkwbis.P_CloseDoc(curr_release);

    end p_display_hello;

END hello_ssb1;
/
```

Banner Self-Service Exercise #2

Banner Self-Service Exercise #2

- Create new file on the database server called ssb2.htm (or other appropriate name you choose).
- Put PL/SQL commands below in the file to create a database package. **Note, make sure that your package name is unique.**
- Save the file.
- Source your file with PL/SQL package into the Oracle database.
sqlplus baninst1/password @ssb2
- Using WebTailor, register your package.procedure to make is accessible.
- Create menu item to link your application page to a menu.
- Click on your menu link to try out your page.

```
Create or Replace PACKAGE hello_ssb2
IS
    Procedure p_disp_ssb_spriden (lname_in varchar2 default null);
END hello_ssb1;
/
show errors;
set scan on;
whenever sqlerror continue;
drop public synonym hello_ssb2;
whenever sqlerror exit rollback;
create public synonym hello_ssb2 for hello_ssb2;
start gurgnth hello_ssb2;
```

```

Create or Replace PACKAGE BODY hello_ssb2
IS
  curr_release CONSTANT varchar2(10) := '1.0';
  pidm number; -- DEFINE FOR OUTPUT

  Cursor c_spriden (lname_in varchar2) is
    Select SPRIDEN_pidm
           ,SPRIDEN_id
           ,SPRIDEN_last_name
           ,SPRIDEN_first_name
           ,SPRIDEN_mi
           ,SPRIDEN_entity_ind
           ,SPRIDEN_change_ind
           ,SPRIDEN_activity_date
    from SPRIDEN
    where upper(SPRIDEN_last_name) = upper(lname_in);

  procedure p_disp_ssb_spriden (lname_in varchar2 default null) is
  begin
    if not twbkwbis.F_ValidUser(pidm) then return; end if;
    twbkwbis.P_OpenDoc('hello_ssb2. p_disp_ssb_spriden');
    twbkwbis.P_DispatchInfo('hello_ssb2. p_disp_ssb_spriden');
    --
    -- application code goes here
    --
    twbkfrmt.p_printText ('');

    HTP.formOpen ('hello_ssb2.p_disp_ssb_spriden');
    Htp.header (1, 'Please enter a last-name to search for:');
    htp.formtext('lname_in'); -- must match the input
parameter
    htp.br;
    HTP.formSubmit (NULL, ' Search Now! ');
    HTP.formClose;

```

```

twbkfrmt.P_TableOpen('DATADISPLAY');

twbkfrmt.p_tableRowOpen;
twbkfrmt.p_tableHeader('PIDM');
twbkfrmt.p_tableHeader('Banner ID');
twbkfrmt.p_tableHeader('Name');
twbkfrmt.p_tableHeader('Entity Ind');
twbkfrmt.p_tableHeader('Change Ind');
twbkfrmt.p_tableHeader('Activity Date');
twbkfrmt.p_tableRowClose;

-- parameter we get for procedure, we simply pass to cursor

FOR rec IN c_spriden (lname_in) LOOP
    twbkfrmt.p_tableRowOpen;
    twbkfrmt.p_tableData(rec.spriden_pidm);
    twbkfrmt.p_tableData(rec.spriden_pidm);
    twbkfrmt.p_tableData(rec.spriden_last_name || ', ' ||
rec.spriden_first_name || ' ' || rec.spriden_mi);
    twbkfrmt.p_tableData(rec.spriden_entity_ind);
    twbkfrmt.p_tableData(rec.spriden_change_ind);
    twbkfrmt.p_tableData(rec.spriden_activity_date);
    twbkfrmt.p_tableRowClose;
END LOOP;

twbkfrmt.p_tableClose;

twbkwbis.P_CloseDoc(curr_release);

end p_disp_ssb_spriden;

END hello_ssb2;
/

```