Saving Our Sanity with an Error Logging System

Brett Morrill

University Computing

11/20/2013

Session Etiquette

- Set your cell phone to stun
- If your brain starts to turn to JELL-O and run out of your ears please leave the session as discreetly as possible
  - Indiscreet exits never go well
- If you have a side conversation during the session please speak loud enough so we can all participate
  - Even if we don't know what you're talking about, we all have an opinion on what you should do.

Presentation Roadmap

- Genesis of Error Tacking
- Anatomy of an Error Tracking System
- Making it Work in Custom Code
- The Next Steps As We See It

Genesis of an Error Tracking System

Where Did The Idea Come From?

- Implemented Banner Baseline
- Most Custom Code is for Add-Ons
- A Conversation with the Security Guy
- Fueling the Imagination Engine

Implemented Banner Baseline

- WSU Implemented Banner with very few Custom Modifications
  - Only if Absolutely Required
  - Needed to be Approved by Change Committee
- Banner Programs Handle Errors Reasonably Well
  - Error Messages In .lis and .log files.

Most Custom Code is for Add-Ons

- Examples
  - State Payroll, Personnel, and Accounting Interfaces
  - TIAA-CREF, ING, and Other Interfaces
- Originally Written as Anonymous Block PL/SQL
- Generate .lis and .log Files
- If program died ORA-Errors spilled onto .log files
- Required direct user access to tables

A Conversation with the Security Guy

- “Would you be willing to turn your anonymous block PL/SQL programs into procedures or packages?”
- If I can’t do it with the payroll interface all bets are off!
- More secure way of coding
- UC4

Fueling the Imagination Engine

- *PL/SQL Training*
  - Functions, Procedures, and Packages Oh My!
- *Oracle PL/SQL Best Practices: Optimizing Oracle Code*
  - Steven Feurestein
    - Discusses having an exception handling architecture in Chapter 1
    - Chapter 5 – Exception Handling
- The Idea was Born!

Anatomy of an Error Tracking System

Accessed by All so it must be General

- Located in WSU's Custom General Schema
  - WSUGEN
- Three Tables with Generated APIs
- An Oracle Sequence
- An Error Processing API
  - Oracle Package

GWVERRT – Error Type Validation Table

- Currently Contains Three Types of “Events”
  - E – Error
  - W – Warning
  - D – Debug

GWBOBJD – Object Base Table

- One row per custom object
- Contains a description of the custom object
- Controls whether debug messages are logged

GWRERRL – Error Log Table

- Errors are written to this table
- Information collected:
  - Object owner (Schema)
  - Object name
  - Code module where error occurred
  - Error type from GWVERRT
  - ORA error code
  - ORA error text
  - Custom error message
  - Other technical info

All Three Tables have Business Entity APIs

- WSU_GB_<table name>
- WSU_GB_<table name>_RULES
- WSU_GB_<table name>_STRINGS
- DML_<table name>
- Generated using Ellucian's automated API code generation scripts

Keeping Order with an Oracle Sequence

- Originally didn't use an Oracle Sequence
- Became hard to tell what order debugging messages were being logged in
- The sequence helps keep the event messages in order
- WSU_ERROR_LOG_SEQ
  - Basically a one up number

The Error Processing API

- WSU_GP_ERROR_PROCESSING
  - What else would we name it?
- f_pkg_version
  - Returns version number of the package
- f_print_debug
  - Returns debug indicator from GWBOBJD
- p_log_error
  - Handles the logging of the error
  - PRAGMA AUTONOMOUS_TRANSACTION

Making Error Logging Work in Custom Code

Exception Handling

- PL/SQL Block Structure
- Will need to use the Exception Handling Section
- Yes it's optional
- No…You must us it
- Really

- Sections:
  - Header
  - Declarative Section
  - Executable Section
  - Exception Handling Section

You Made It – You Name It

- Added two constants to the package spec
  - M_ENTITY_OWNER
  - M_ENTITY_NAME
- M_ENTITY_OWNER
  - Database schema where package is installed
- M_ENTITY_NAME
  - Name of package
- These should match the entry made in GWBOBJD

You Made It – You Name It

- Example
  - Defined in the Package Specification

```
/** * Business Entity Owner */
M_ENTITY_OWNER CONSTANT VARCHAR2(7) := 'WSUPAY';
--
/** * Business Entity Name */
M_ENTITY_NAME CONSTANT VARCHAR2(30) := 'WSU_PP_PZPPWEI';
--
```

Error Handling Variables

- Add error handling variables to package body
  - Standardized
  - Add to the global declarative section of a package body so they can be referenced by all functions and procedures within the package body

```
/* ------------------------------------------------------------
    Error Handling Variables
------------------------------------------------------------ */
gv_code_module  gwrerrl.gwrerrl_code_module%TYPE  := NULL;
gv_error_type   gwrerrl.gwrerrl_errt_code%TYPE    := NULL;
gv_error_code   gwrerrl.gwrerrl_error_code%TYPE   := NULL;
gv_error_msg    gwrerrl.gwrerrl_error_text%TYPE   := NULL;
gv_custom_msg   gwrerrl.gwrerrl_custom_text%TYPE  := NULL;
```

Using Error Handling Variables

- gv_code_module
  - Function/Procedure name being executed
- gv_error_type
  - GWVERRT Code describing type of event
- gv_error_code
  - ORA-????? Code from system
- gv_error_msg
  - ORA-????? Message from system
- gv_custom_msg
  - Anything we want that will help debugging

Using Error Handling Variables

- Example

```
PROCEDURE p_get_curr_emp_status IS
/* ---------------------------------------------------------------------------
  Procedure Cursor to pull Home Address Information
--------------------------------------------------------------------------- */
CURSOR pc_home_addr( pcv_pidm NUMBER ) IS
  SELECT
~~~~~~~~~~~~~~
BEGIN
gv_code_module := 'p_get_curr_emp_status';
gv_custom_msg := 'Processing pidm = '||gr_emp_data.emp_pidm||'::Opening pc_home_addr';

gr_new_pwrpwei_rec := NULL;
OPEN pc_home_addr(gr_emp_data.emp_pidm);
FETCH pc_home_addr INTO pr_home_addr;
IF pc_home_addr%NOTFOUND THEN pr_home_addr := NULL;
END IF;
CLOSE pc_home_addr;

gv_custom_msg := gv_custom_msg||'::Setting gr_new_pwrpwei_rec Variables';
```

Exception Handling Section

- Need to trap and process errors
- Execute error handling package here
- Raise custom exception so program dies

```
/*---------------------------------
   Global Exception Handling
---------------------------------*/
EXCEPTION

WHEN OTHERS THEN
gv_error_code := SQLCODE;
gv_error_msg := SQLERRM;
gv_error_type := 'E';

wsu_gp_error_processing.p_log_error (
m_entity_owner
,m_entity_name
,gv_code_module
,gv_error_type
,gv_error_code
,gv_error_msg
,gv_custom_msg );

raise_application_error(-20999,
'Error Occurred and Logged in GWRERRL');
```

Data Logged in GWRERRL

GWRERRL SEQ NO	GWRERRL OWNER	GWRERRL OBJECT	GWRERRL CODE MODULE	GWRERRL ERRT CODE	GWRERRL ERROR CODE	GWRERRL ERROR TEXT	GWRERRL CUSTOM TEXT
16	WSUFIN	WSU_FP_FZPRMSD	p_update_twtrmsd	E	-1013	ORA-01013: user requested cancel of current operation	Ready to update twtrmsd misc fields
17	WSUFIN	WSU_FP_FZPRMSD	p_flag_bad_recs_twtrmsd	E	-1013	ORA-01013: user requested cancel of current operation	Ready to update twtrmsd record
18	WSUFIN	WSU_FP_FZPRMSD	p_save_twtrmsd	E	-54	ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired	Ready to update twtrmsd record
19	WSUFIN	WSU_FP_FZPRMSD	p_update_twtrmsd	E	-1013	ORA-01013: user requested cancel of current operation	Ready to update twtrmsd misc fields
20	WSUFIN	WSU_FP_FZPRMSD	p_save_twtrmsd	E	-8103	ORA-08103: object no longer exists	Just closed pidm cursor

The Next Steps
As We See It

Future Enhancements

- Define Common Error Handling Variables in Package Specification
- Event Sequence Number
  - Group multiple debug entries together
  - Currently relying on activity date and time stamp
- Utilize our Custom Notification System
  - Email when error is logged
  - Another Presentation for Another Day!
- Web Front End to work errors
  - History table
  - Track resolutions

Questions and Contact Information

- Contact Information

Brett Morrill
Senior System Analyst
Wichita State University
brett.morrill@wichita.edu
316-978-3172