all·APEX

Instrumentation for Oracle Database Development

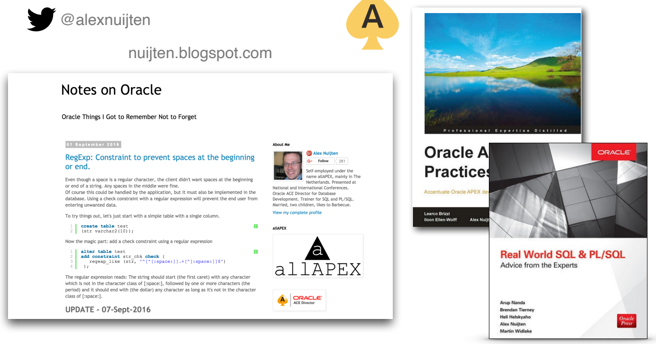
Alex Nuijten

---

all·APEX

@alexnuijten

nuijten.blogspot.com



ORACLE
ACE Program

500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical contributions in the Oracle community

3 membership tiers

ORACLE ACE Director | ORACLE ACE | ORACLE ACE Associate

For more details on Oracle ACE Program:
bit.ly/OracleACEProgram

Oracle Groundbreakers

Nominate
yourself or someone you know:
acenomination.oracle.com

Connect: **oracle-ace_ww@oracle.com** Facebook.com/oracleaces @oracleace

---

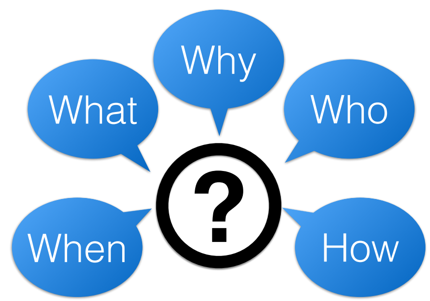


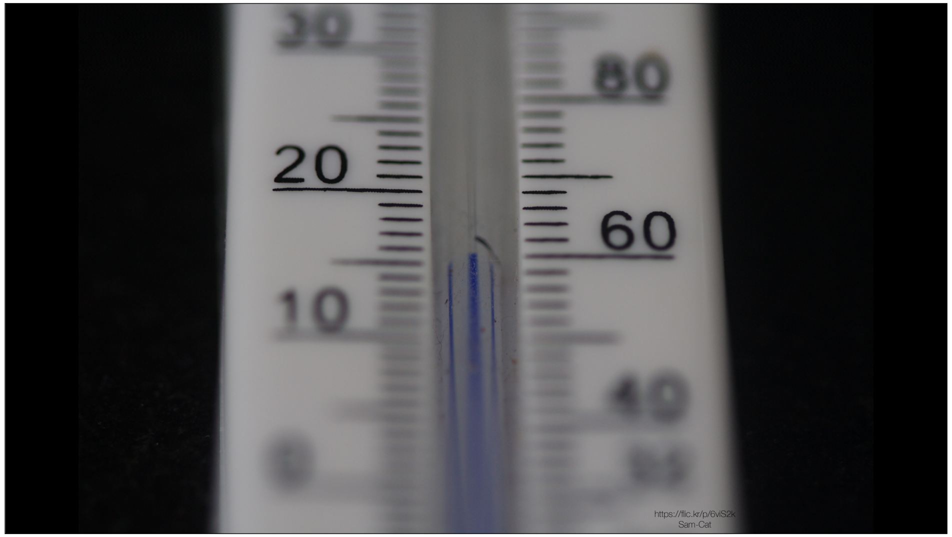
Photo by Dex Ezekiel on Unsplash

What is Instrumentation?

"An instrument is a device that measures a physical quantity such as flow, temperature, level, distance, angle, or pressure.

Instruments may be as simple as direct reading thermometers or may be complex multi-variable process analyzers."

Source: Wikipedia



https://flic.kr/p/6viS2k
Sam-Cat

"If you can't measure it, you can't manage it."

David Garvin

Why?

```
SQL> select empno
           ,ename
       from emp
      order by empno
     /

     EMPNO ENAME
---------- ----------
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER
      7876 ADAMS
      7900 JAMES
      7902 FORD
      7934 MILLER

14 rows selected.
```

Note the range of EMPNO

```
function fullname (p_empno in number)
   return varchar2
is
   l_retval varchar2(25);
begin
   select e.ename
     into l_retval
     from emp e
    where e.empno = p_empno;
   return l_retval;
end fullname;
```

```
SQL> begin
       dbms_output.put_line (
          fullname (p_empno => 1234)
       );
     end;
     /
```

```
SQL> begin
      dbms_output.put_line (
         fullname (p_empno => :bind)
      );
   end;
   /
```


```
SQL> begin
      dbms_output.put_line (
         fullname (p_empno => :bind)
      );
   end;
   /
```


Guess

```
function fullname (p_empno in number)
   return varchar2
is
   l_retval varchar2(25);
begin
   select e.ename
     into l_retval
     from emp e
    where e.empno = p_empno;
   return l_retval;
end fullname;
```

~~Guess~~

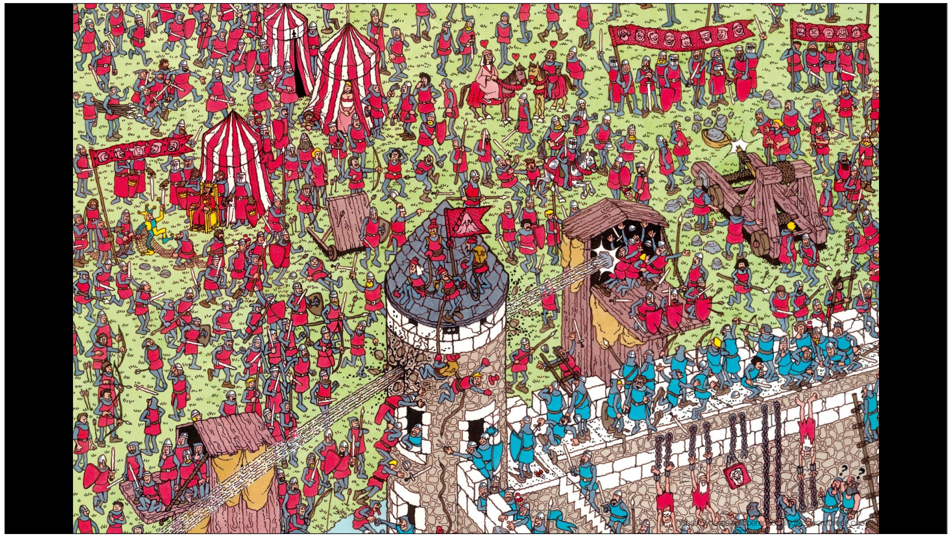
Know

```
SQL> set timing on
SQL> begin
  2     long_running_process;
  3  end;
  4  /

PL/SQL procedure successfully completed.

Elapsed: 00:02:15.42
```

```
SQL> set timing on
SQL> begin
  2     long_running_proce
  3  end;
  4  /

PL/SQL procedure succe          eted.

Elapsed: 03:17:33.42
```

`long_running_process`

1. Startup
2. Gather Data
3. Pre-process
4. Process
5. Cleanup



`long_running_process`

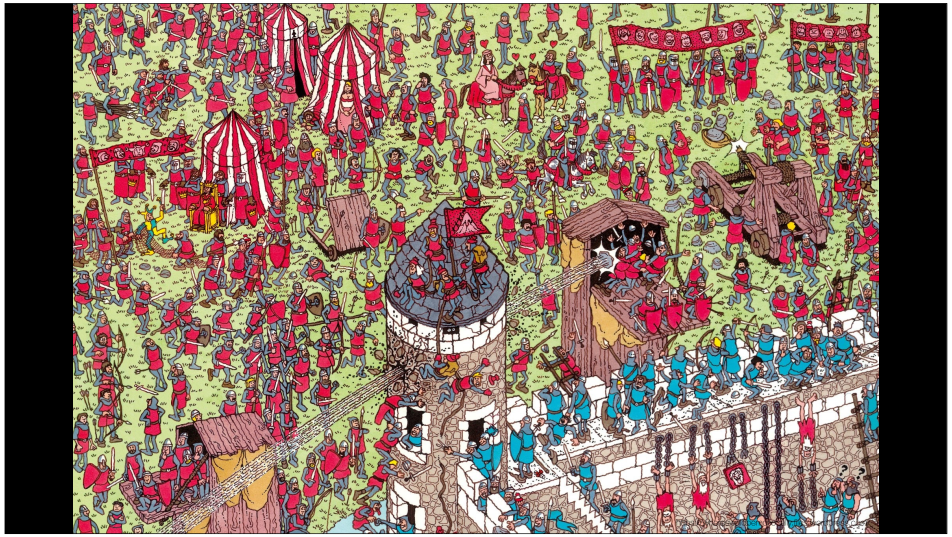
1. Startup	00:00:00.22
2. Gather Data	00:53:36.54
3. Pre-process	00:06:09.16
4. Process	02:34:16.42
5. Cleanup	00:00:01.20

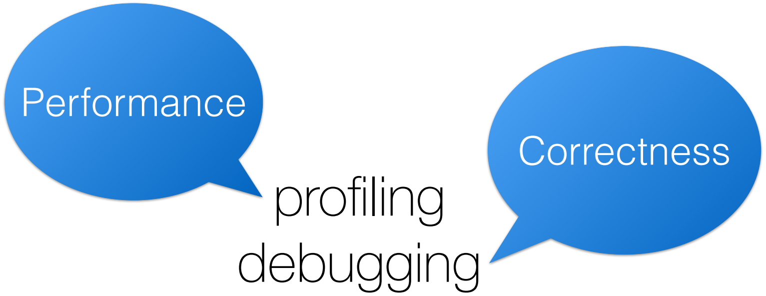
`long_running_process`

1. Startup	00:00:00.22
2. Gather Data	00:53:36.54
3. Pre-process	00:06:09.16
4. Process	**02:34:16.42**
5. Cleanup	00:00:01.20

`long_running_process`

1. Startup	00:00:00.22
2. Gather Data	**00:53:36.54**
3. Pre-process	00:06:09.16
4. Process	**02:34:16.42**
5. Cleanup	00:00:01.20



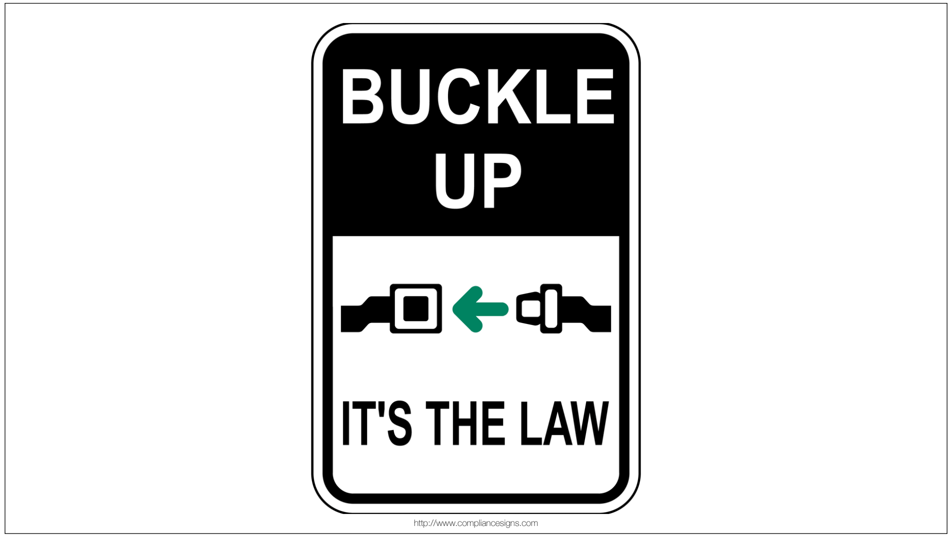


Performance
Correctness
profiling
debugging

Make Sense of the Spaghetti
https://flic.kr/p/mUYoMc
jeffreyw

Why Not?

SPEED
LIMIT
30
https://flic.kr/p/7KL1fx
Ben Grey


BUCKLE
UP
IT'S THE LAW
http://www.compliancesigns.com


Pfizer-BioNTech COVID-19 Vaccine
After dilution, vial contains 5 doses of 0.3 mL
For intramuscular use. Contains no preservative.
For use under Emergency Use Authorization.
DILUTE BEFORE USE.
Discard 6 hours after dilution when
stored at 2 to 25°C (35 to 77°F).
Dilution date and time:
By Arne Müseler / www.arne-mueseler.com, CC BY-SA 3.0 de, https://commons.wikimedia.org/w/index.php?curid=98215942

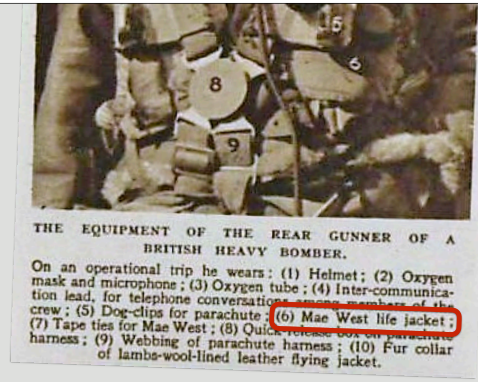

Nicole Nicky


ADULTS
http://bit.ly/1O6utS8

THE EQUIPMENT OF THE REAR GUNNER OF A BRITISH HEAVY BOMBER.

On an operational trip he wears: (1) Helmet; (2) Oxygen mask and microphone; (3) Oxygen tube; (4) Inter-communication lead, for telephone conversations among members of the crew; (5) Dog-clips for parachute; (6) Mae West life jacket; (7) Tape ties for Mae West; (8) Quick-release box on parachute harness; (9) Webbing of parachute harness; (10) Fur collar of lambs-wool-lined leather flying jacket.

Illustrated London News of Saturday 21 November 1942



THE EQUIPMENT OF THE REAR GUNNER OF A BRITISH HEAVY BOMBER.

On an operational trip he wears: (1) Helmet; (2) Oxygen mask and microphone; (3) Oxygen tube; (4) Inter-communication lead, for telephone conversations among members of the crew; (5) Dog-clips for parachute; (6) Mae West life jacket; (7) Tape ties for Mae West; (8) Quick-release box on parachute harness; (9) Webbing of parachute harness; (10) Fur collar of lambs-wool-lined leather flying jacket.

Illustrated London News of Saturday 21 November 1942

Mae West

http://bit.ly/1Q6Wri4

Instrumentation is ***NOT*** optional

Who?

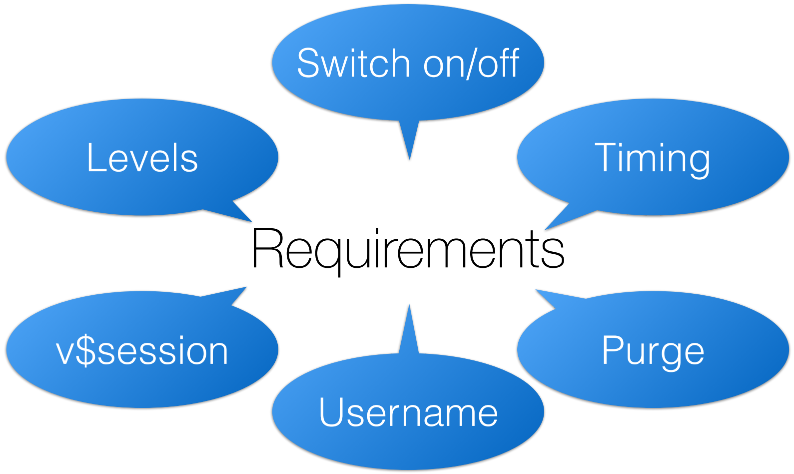
When?

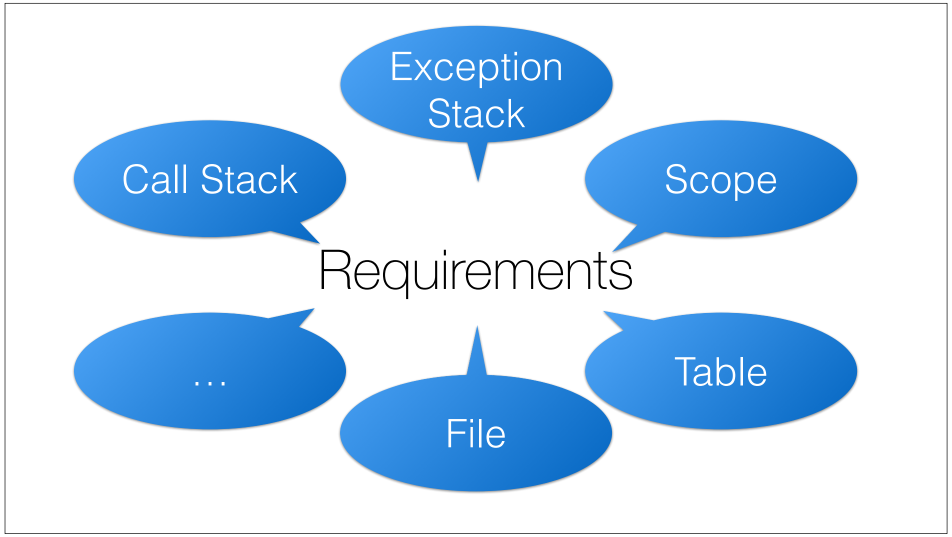
from day one

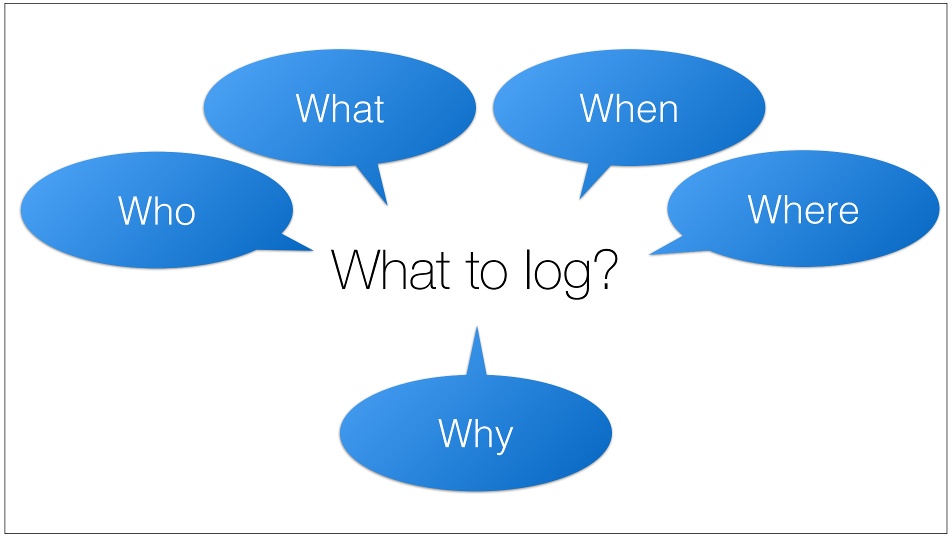
…or day two

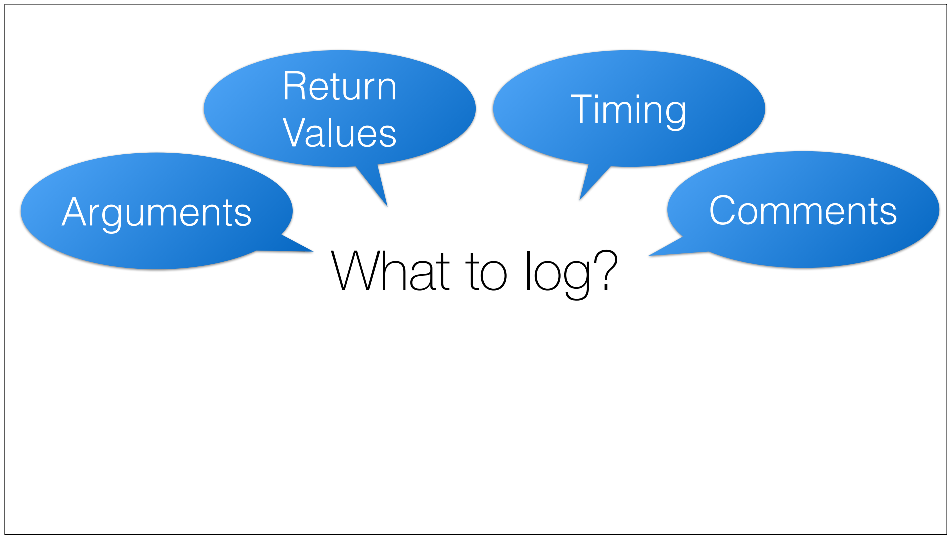
…ASAP

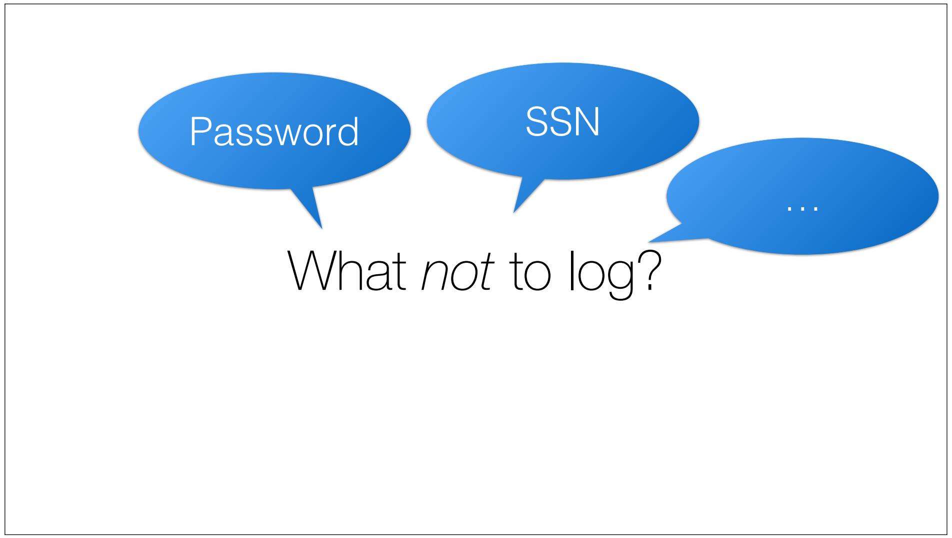
How?




Exception
Stack
Call Stack
Scope
Requirements
...
File
Table


What
When
Who
Where
What to log?
Why


Return
Values
Timing
Arguments
Comments
What to log?


Password
SSN
...
What *not* to log?

```
function fullname (p_empno in number)
   return varchar2
is
   l_retval varchar2(25);
begin
   -- Get the name from the EMP table
   -- based on the incoming argument
   select e.ename
     into l_retval
     from emp e
    where e.empno = p_empno;
   return l_retval;
end fullname;
```

```
function fullname (p_empno in number)
   return varchar2
is
   l_retval varchar2(25);
begin
   write_log ('Get the name from the EMP table
               based on the incoming argument');
   select e.ename
     into l_retval
     from emp e
    where e.empno = p_empno;
   return l_retval;
end fullname;
```

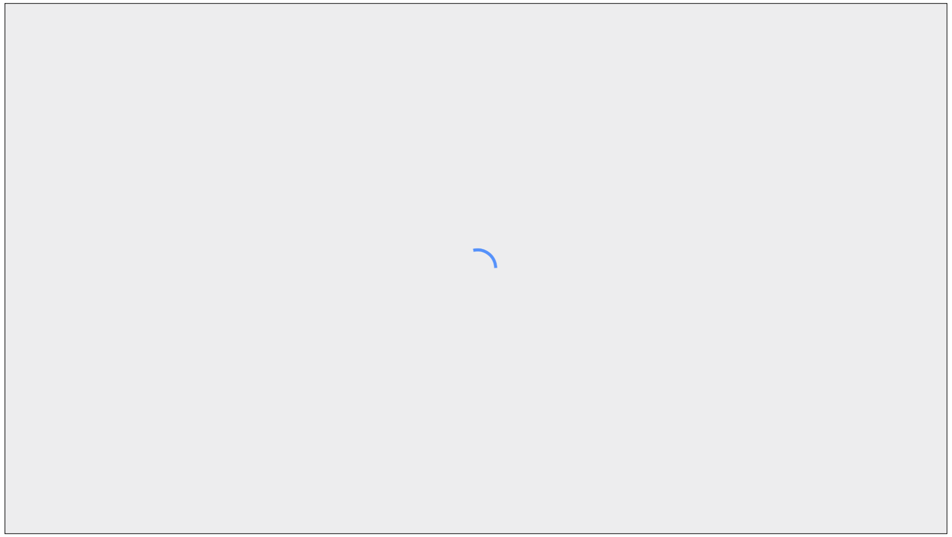
```
function fullname (p_empno in number)
   return varchar2
is
   l_retval varchar2(25);
begin
   write_log ('Get the name from the EMP table
               based on the incoming argument');
   write_log ('incoming argument p_empno: '||p_empno);
   select e.ename
     into l_retval
     from emp e
    where e.empno = p_empno;
   write_log ('return value: '||l_retval);
   return l_retval;
end fullname;
```

Method of Logging

DBMS_OUTPUT

no output until done

```
SQL> begin
  2      long_running_process;
  3  end;
  4  /
```



```
SQL> begin
  2     long_running_process;
  3  end;
  4  /
```

```
SQL> begin
  2     long_running_process;
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

```
set serveroutput on
```

```
SQL> begin
  2     long_running_process;
  3  end;
  4  /
```

```
SQL> begin
  2     long_running_process;
  3  end;
  4  /
begin
*
ERROR at line 1:
ORA-20000: ORU-10027: buffer overflow, limit of 2000 bytes
ORA-06512: at "SYS.DBMS_OUTPUT", line 32
ORA-06512: at "SYS.DBMS_OUTPUT", line 97
ORA-06512: at "SYS.DBMS_OUTPUT", line 112
ORA-06512: at line 3
```

```
set serveroutput on size unlimited
```

UTL_FILE

..but why?

Custom Tables

pragma autonomous_transaction

DBMS_APPLICATION_INFO

```
SQL> select username
  2        ,module
  3        ,action
  4        ,client_info
  5    from v$session
  6   where username = 'ALEX'
  7* /

USERNAME        MODULE          ACTION          CLIENT_INFO
--------------- --------------- --------------- ---------------
ALEX            SQL Developer
ALEX            SQLcl
ALEX            SQLcl
```

```
create or replace
procedure set_info
is
begin
   sys.dbms_application_info.set_module
       (module_name => 'Set Info'
       ,action_name => 'testing'
       );
   sys.dbms_application_info.set_client_info
       (client_info => user);
end set_info;
/
```

```
create or replace
procedure set_info
is
begin
   sys.dbms_application_info.set_module
       (module_name => 'Set Info'
       ,action_name => 'testing'
       );
   sys.dbms_application_info.set_client_info
       (client_info => user);
end set_info;
/
```

```
SQL> begin
  2     set_info;
  3  end;
  4* /

PL/SQL procedure successfully completed.
```

```
SQL> select username
  2        ,module
  3        ,action
  4        ,client_info
  5    from v$session
  6   where username = 'ALEX'
  7* /

USERNAME        MODULE          ACTION          CLIENT_INFO
--------------- --------------- --------------- ---------------
ALEX            SQL Developer
ALEX            SQLcl
ALEX            Set Info        testing         ALEX
```

```
SQL> select username
  2        ,module
  3        ,action
  4        ,client_info
  5    from v$session
  6   where username = 'ALEX'
  7* /

USERNAME        MODULE          ACTION          CLIENT_INFO
--------------- --------------- --------------- ---------------
ALEX            SQL Developer
ALEX            SQLcl
ALEX            Set Info        testing         ALEX
```

```
create or replace
procedure reset_info
is
begin
   sys.dbms_application_info.set_module
       (module_name => null
       ,action_name => null
       );
   sys.dbms_application_info.set_client_info
       (client_info => null);
end reset_info;
/
```

```
create or replace
procedure reset_info
is
begin
   sys.dbms_application_info.set_module
       (module_name => null
       ,action_name => null
       );
   sys.dbms_application_info.set_client_info
       (client_info => null);
end reset_info;
/
```

```
SQL> begin
  2     reset_info;
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

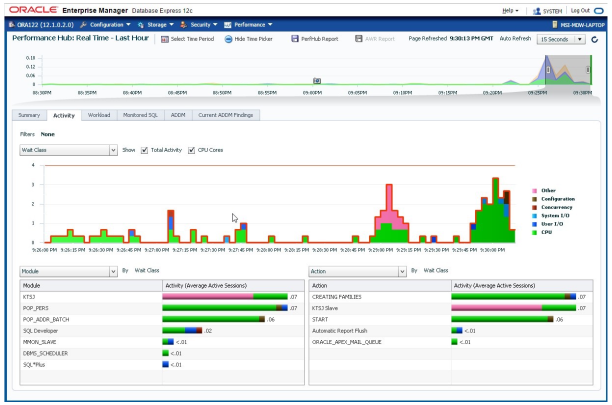
```
SQL> select username
  2        ,module
  3        ,action
  4        ,client_info
  5    from v$session
  6   where username = 'ALEX'
  7* /

USERNAME        MODULE          ACTION          CLIENT_INFO
--------------- --------------- --------------- ---------------
ALEX            SQL Developer
ALEX            SQLcl
ALEX
```

```
create or replace
procedure proc
is
begin
   sys.dbms_application_info.set_module
      (module_name => 'PROC'
      ,action_name => 'testing'
      );
   sys.dbms_application_info.set_client_info
      (client_info => user);
   -- Do some interesting PL/SQL Stuff here
   sys.dbms_application_info.set_module
      (module_name => null
      ,action_name => null
      );
   sys.dbms_application_info.set_client_info
      (client_info => null);
end proc;
```

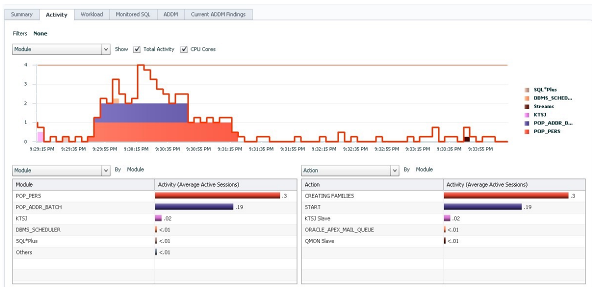
```
create or replace
procedure proc
is
begin
   sys.dbms_application_info.set_module
      (module_name => 'PROC'
      ,action_name => 'testing'
      );
   sys.dbms_application_info.set_client_info
      (client_info => user);
   -- Do some interesting PL/SQL Stuff here
   sys.dbms_application_info.set_module
      (module_name => null
      ,action_name => null
      );
   sys.dbms_application_info.set_client_info
      (client_info => null);
end proc;
```

```
create or replace
procedure proc
is
begin
   sys.dbms_application_info.set_module
      (module_name => 'PROC'
      ,action_name => 'testing'
      );
   sys.dbms_application_info.set_client_info
      (client_info => user);
   -- Do some interesting PL/SQL Stuff here
   sys.dbms_application_info.set_module
      (module_name => null
      ,action_name => null
      );
   sys.dbms_application_info.set_client_info
      (client_info => null);
end proc;
```

```
create or replace
procedure proc
is
begin
   sys.dbms_application_info.set_module
      (module_name => 'PROC'
      ,action_name => 'testing'
      );
   sys.dbms_application_info.set_client_info
      (client_info => user);
   -- Do some interesting PL/SQL Stuff here
   sys.dbms_application_info.set_module
      (module_name => null
      ,action_name => null
      );
   sys.dbms_application_info.set_client_info
      (client_info => null);
end proc;
```



Screenshot by Martin Widlake



Screenshot by Martin Widlake

Let's
not
Reinvent
the
Wheel

Log 4 pl/sql

Oracle Database Logging tools

General information

LOG4PLSQL is a PLSQL framework for logging in all PLSQL code

- Package,
- Procedure,
- Function,
- Trigger,
- PL/SQL Web application,
- Oracle Tools (Form, report,...).

Purpose:

- Easy of implemented and use
- Possibility to log apart from transaction
- Possibility to adapt the level of logging depending on user requirements
- Ability to use all LOG4J feature.
- Provide a complet set of tools PL/SQL developpement (bebug variable, SQLERRM, assert, call stack, ...).

Search

◇ METHOD R™

Software and services to make you and your systems faster

Menu

Instrumentation Library for Oracle (ILO)

ILO, the Method R instrumentation library for Oracle, is an open source project directed by Method R Corporation developers. It contains PL/SQL procedures for measuring tasks using Oracle extended SQL trace. The procedures are installed and compiled in the target Oracle instance and can be called by code written in any language that has access to the Oracle database tier.

Over 7,200 downloads at sourceforge.net.

DOWNLOAD

http://www.oraopensource.com/

Requirements

session

- ✔ Levels
- ✔ Toggle On/Off
- ✔ Timing
- ✔ Purge
- ✔ Username
- ✔ v$session
- ✔ Call Stack
- ✔ Exception Stack
- ✔ Scoping
- ✔ Table
- ✘ File
- ✔ APEX Session state

LOGGER_LOGS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partit

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	(null)	1	(null)
2	LOGGER_LEVEL	NUMBER	No	(null)	2	(null)
3	TEXT	VARCHAR2(4000 BYTE)	Yes	(null)	3	(null)
4	TIME_STAMP	TIMESTAMP(6)	No	(null)	4	(null)
5	SCOPE	VARCHAR2(1000 BYTE)	Yes	(null)	5	(null)
6	MODULE	VARCHAR2(100 BYTE)	Yes	(null)	6	(null)
7	ACTION	VARCHAR2(100 BYTE)	Yes	(null)	7	(null)
8	USER_NAME	VARCHAR2(255 BYTE)	Yes	(null)	8	(null)
9	CLIENT_IDENTIFIER	VARCHAR2(255 BYTE)	Yes	(null)	9	(null)
10	CALL_STACK	VARCHAR2(4000 BYTE)	Yes	(null)	10	(null)
11	UNIT_NAME	VARCHAR2(255 BYTE)	Yes	(null)	11	(null)
12	LINE_NO	VARCHAR2(100 BYTE)	Yes	(null)	12	(null)
13	SCN	NUMBER	Yes	(null)	13	(null)
14	EXTRA	CLOB	Yes	(null)	14	(null)
15	SID	NUMBER	Yes	(null)	15	(null)
16	CLIENT_INFO	VARCHAR2(64 BYTE)	Yes	(null)	16	(null)

```
off
permanent
error
warning
information
debug
```

Debug

```
log
log_permanent
log_error
log_warn[ing]
log_info[rmation]
```

```
procedure log_information
    (p_text    in varchar2
    ,p_scope   in varchar2  default null
    ,p_extra   in clob      default null
    ,p_params  in tab_param default logger.gc_empty_tab_param
    );
```

```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

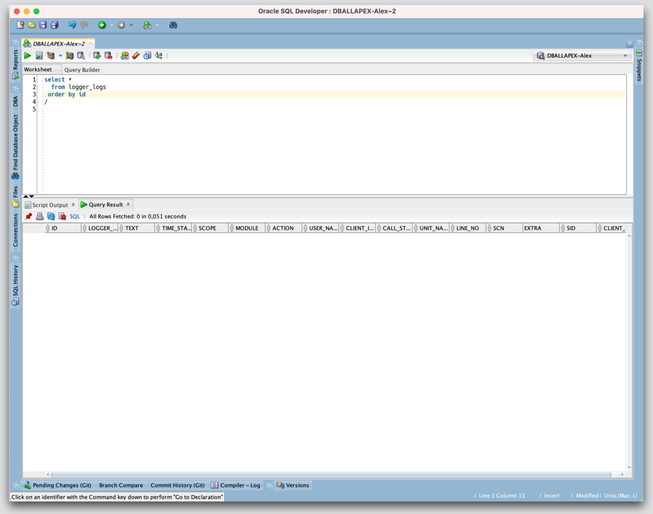
```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

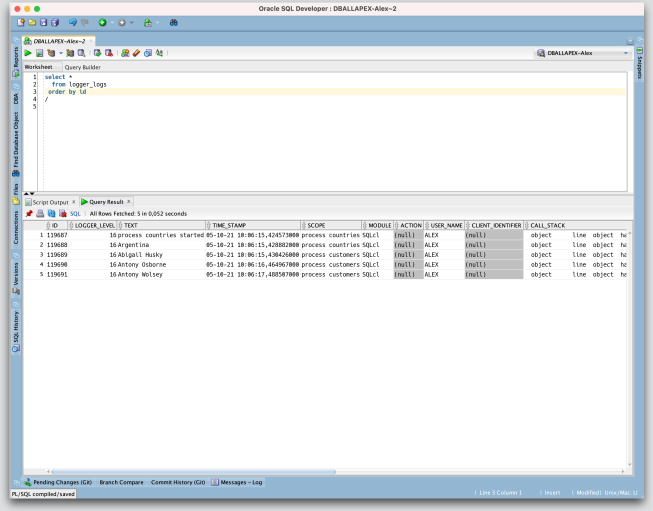
```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

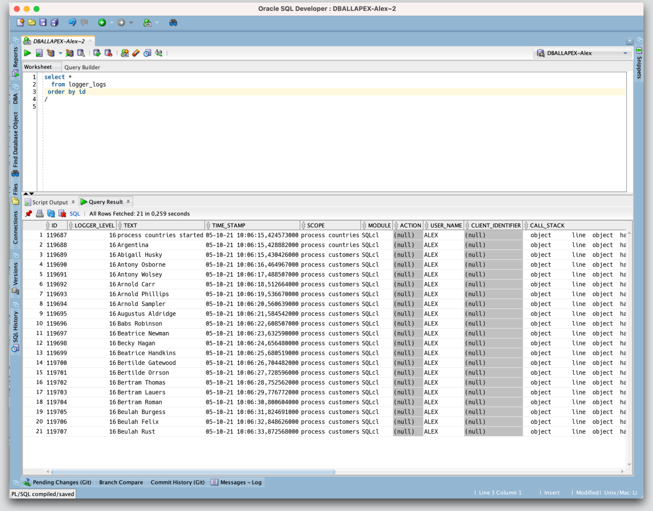
```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

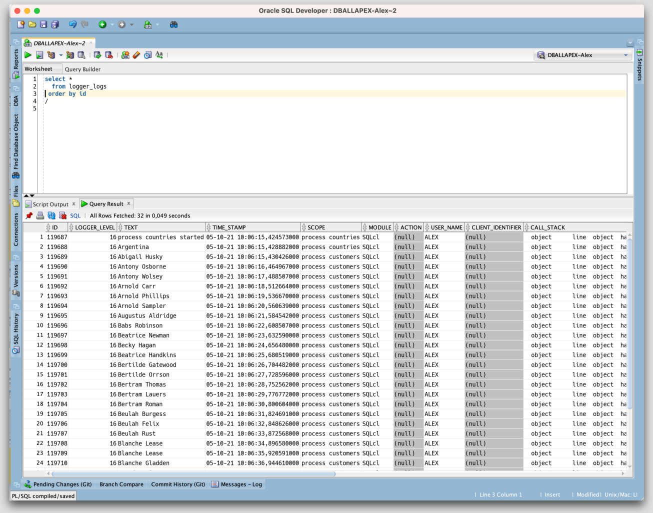
```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

```
procedure process_countries
is
   l_country_scope  varchar2 (25) := 'process countries';
   l_customer_scope varchar2 (25) := 'process customers';
begin
   logger.log (p_text  => l_country_scope ||' started'
              ,p_scope => l_country_scope
              );
   for r_cty in (...)
   loop
      logger.log (p_text  => r_cty.country_name
                 ,p_scope => l_country_scope
                 );
      ...
      for r_ctr in (...)
      loop
         logger.log (p_text  => r_ctr.customer_name
                    ,p_scope => l_customer_scope
                    );
         ...
      end loop;
   end loop;
   logger.log (p_text  => l_country_scope ||' finished'
              ,p_scope => l_country_scope
              );
end process_countries;
```

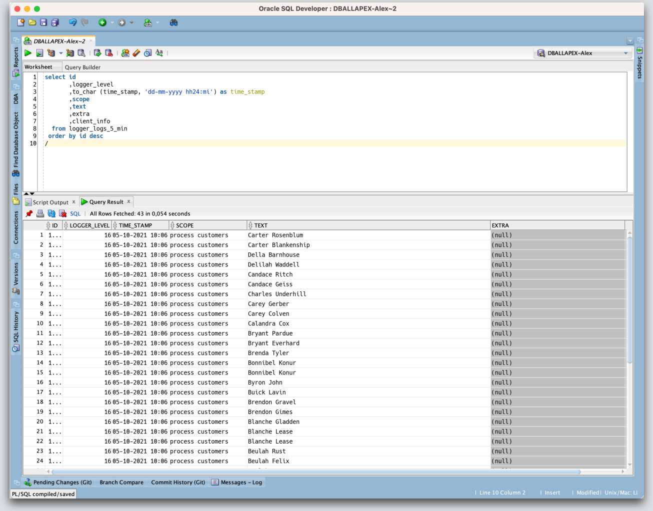








```
select id
      ,logger_level
      ,time_stamp
      ,scope
      ,text
      ,extra
      ,client_info
  from logger_logs_5_min
 order by id desc
```

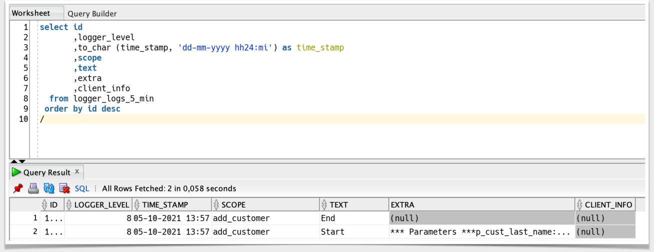


Append_param

```
procedure append_param(
  p_params in out nocopy logger.tab_param,
  p_name   in varchar2,
  p_val    in varchar2);
```

```
logger.append_param (p_params => l_params
                    ,p_name   => 'p_employee_id'
                    ,p_val    => p_employee_id
                    );
```

```
begin
   add_customer
   (p_cust_last_name         => 'Nuijten'
   ,p_cust_first_name        => 'Alex'
   ,p_cust_gender            => 'M'
   ,p_cust_year_of_birth     => ''
   ,p_cust_marital_status    => 'M'
   ,p_cust_street_address    => 'Mainstreet'
   ,p_cust_postal_code       => '4321'
   ,p_cust_city              => 'Oosterhout'
   ,p_cust_city_id           => null
   ,p_cust_state_province    => 'NB'
   ,p_cust_state_province_id => null
   ,p_country_id             => 52784
   ,p_cust_main_phone_number => '+31610395654'
   ,p_cust_income_level      => 1000
   ,p_cust_credit_limit      => 1234
   ,p_cust_email             => 'alex@allapex.nl'
   ,p_cust_total             => ''
   ,p_cust_total_id          => null
   ,p_cust_src_id            => null
   ,p_cust_eff_from          => sysdate
   ,p_cust_eff_to            => sysdate + interval '2' year
   ,p_cust_valid             => 'Y'
   );
end;
/
```



$$NO_OP

```
procedure log_information(
    p_text in varchar2,
    p_scope in varchar2 default null,
    p_extra in clob default null,
    p_params in tab_param default logger.gc_empty_tab_param)
  is
  begin
    $if $$no_op $then
      null;
    $else
      if ok_to_log(logger.g_information) then
        log_internal(
          p_text => p_text,
          p_log_level => logger.g_information,
          p_scope => p_scope,
          p_extra => p_extra,
          p_callstack => dbms_utility.format_call_stack,
          p_params => p_params);
      end if;
    $end
  end log_information;
```


```
procedure log_information(
    p_text in varchar2,
    p_scope in varchar2 default null,
    p_extra in clob default null,
    p_params in tab_param default logger.gc_empty_tab_param)
  is
  begin
    $if $$no_op $then
      null;
    $else
      if ok_to_log(logger.g_information) then
        log_internal(
          p_text => p_text,
          p_log_level => logger.g_information,
          p_scope => p_scope,
          p_extra => p_extra,
          p_callstack => dbms_utility.format_call_stack,
          p_params => p_params);
      end if;
    $end
  end log_information;
```


Level	Actionable	Target Audience	~%Reference
Debug	No	Developers	90%
Information	No	Developers/Business	1%
Warning	Yes	Developers/Business	1%
Error	Yes	Developers/IT/DBA	5%
Permanent	No	Developers/Business	0.5%

https://github.com/alexnuijten

```
procedure add_customer
   (p_cust_last_name         in sh.customers.cust_last_name%type
   ,p_cust_first_name        in sh.customers.cust_first_name%type
   ,p_cust_gender            in sh.customers.cust_gender%type
   ,p_cust_year_of_birth     in sh.customers.cust_year_of_birth%type
   ,p_cust_marital_status    in sh.customers.cust_marital_status%type
   ,p_cust_street_address    in sh.customers.cust_street_address%type
   ,p_cust_postal_code       in sh.customers.cust_postal_code%type
   ,p_cust_city              in sh.customers.cust_city%type
   ,p_cust_city_id           in sh.customers.cust_city_id%type
   ,p_cust_state_province    in sh.customers.cust_state_province%type
   ,p_cust_state_province_id in sh.customers.cust_state_province_id%type
   ,p_country_id             in sh.customers.country_id%type
   ,p_cust_main_phone_number in sh.customers.cust_main_phone_number%type
   ,p_cust_income_level      in sh.customers.cust_income_level%type
   ,p_cust_credit_limit      in sh.customers.cust_credit_limit%type
   ,p_cust_email             in sh.customers.cust_email%type
   ,p_cust_total             in sh.customers.cust_total%type
   ,p_cust_total_id          in sh.customers.cust_total_id%type
   ,p_cust_src_id            in sh.customers.cust_src_id%type
   ,p_cust_eff_from          in sh.customers.cust_eff_from%type
   ,p_cust_eff_to            in sh.customers.cust_eff_to%type
   ,p_cust_valid             in sh.customers.cust_valid%type
   )
```

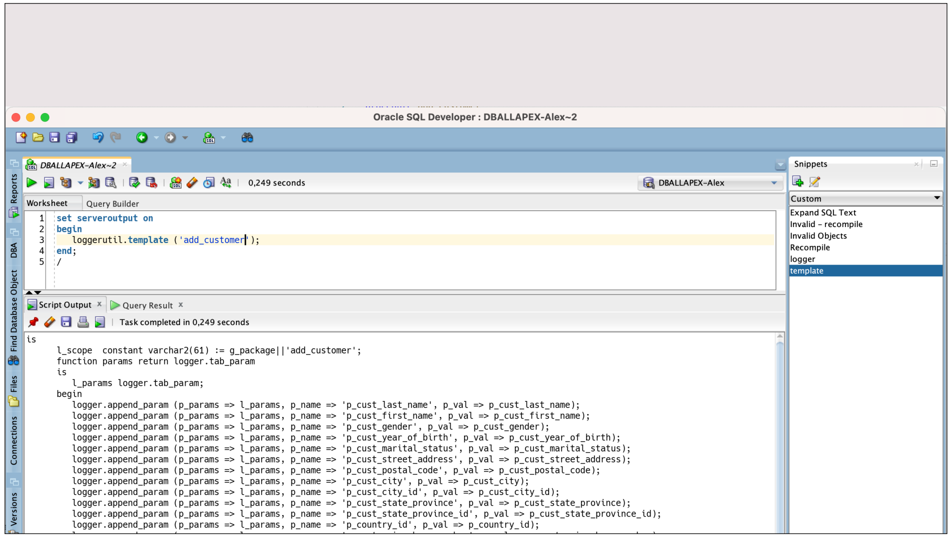
```
logger.append_param (p_params => l_params, p_name => 'p_cust_last_name', p_val => p_cust_last_name);
```

```
logger.append_param (p_params => l_params, p_name => 'p_cust_last_name', p_val => p_cust_last_name);
logger.append_param (p_params => l_params, p_name => 'p_cust_first_name', p_val => p_cust_first_name);
```

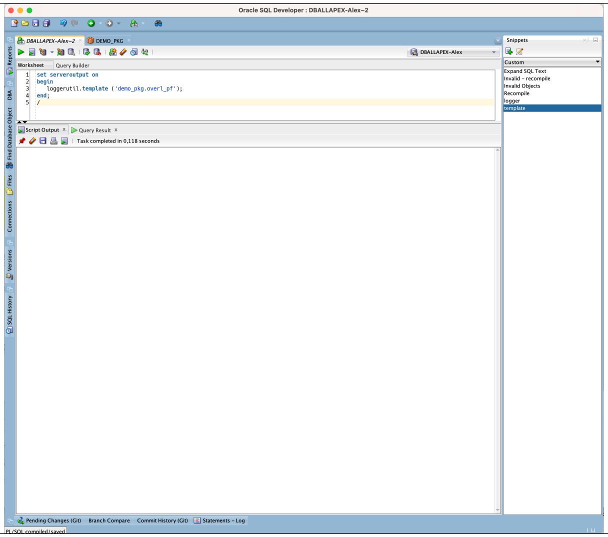
```
logger.append_param (p_params => l_params, p_name => 'p_cust_last_name', p_val => p_cust_last_name);
logger.append_param (p_params => l_params, p_name => 'p_cust_first_name', p_val => p_cust_first_name);
logger.append_param (p_params => l_params, p_name => 'p_cust_gender', p_val => p_cust_gender);
logger.append_param (p_params => l_params, p_name => 'p_cust_year_of_birth', p_val => p_cust_year_of_birth);
logger.append_param (p_params => l_params, p_name => 'p_cust_marital_status', p_val => p_cust_marital_status);
logger.append_param (p_params => l_params, p_name => 'p_cust_street_address', p_val => p_cust_street_address);
logger.append_param (p_params => l_params, p_name => 'p_cust_postal_code', p_val => p_cust_postal_code);
logger.append_param (p_params => l_params, p_name => 'p_cust_city', p_val => p_cust_city);
logger.append_param (p_params => l_params, p_name => 'p_cust_city_id', p_val => p_cust_city_id);
```

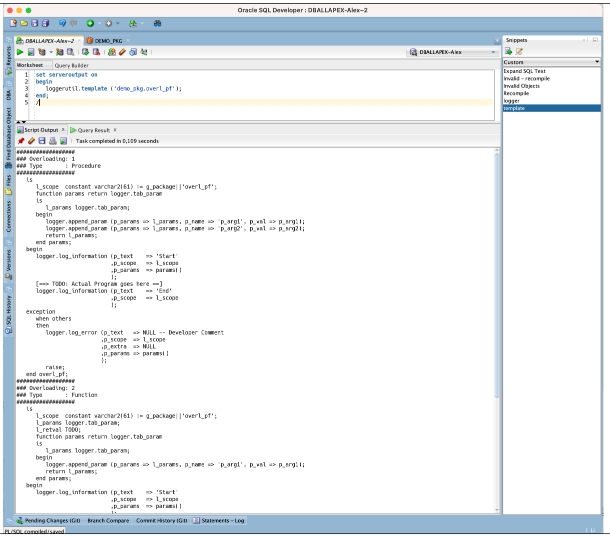
```
logger.append_param (p_params => l_params, p_name => 'p_cust_last_name', p_val => p_cust_last_name);
logger.append_param (p_params => l_params, p_name => 'p_cust_first_name', p_val => p_cust_first_name);
logger.append_param (p_params => l_params, p_name => 'p_cust_gender', p_val => p_cust_gender);
logger.append_param (p_params => l_params, p_name => 'p_cust_year_of_birth', p_val => p_cust_year_of_birth);
logger.append_param (p_params => l_params, p_name => 'p_cust_marital_status', p_val => p_cust_marital_status);
logger.append_param (p_params => l_params, p_name => 'p_cust_street_address', p_val => p_cust_street_address);
logger.append_param (p_params => l_params, p_name => 'p_cust_postal_code', p_val => p_cust_postal_code);
logger.append_param (p_params => l_params, p_name => 'p_cust_city', p_val => p_cust_city);
logger.append_param (p_params => l_params, p_name => 'p_cust_city_id', p_val => p_cust_city_id);
logger.append_param (p_params => l_params, p_name => 'p_cust_state_province', p_val => p_cust_state_province);
logger.append_param (p_params => l_params, p_name => 'p_cust_state_province_id', p_val => p_cust_state_province_id);
logger.append_param (p_params => l_params, p_name => 'p_country_id', p_val => p_country_id);
logger.append_param (p_params => l_params, p_name => 'p_cust_main_phone_number', p_val => p_cust_main_phone_number);
logger.append_param (p_params => l_params, p_name => 'p_cust_income_level', p_val => p_cust_income_level);
logger.append_param (p_params => l_params, p_name => 'p_cust_credit_limit', p_val => p_cust_credit_limit);
logger.append_param (p_params => l_params, p_name => 'p_cust_email', p_val => p_cust_email);
logger.append_param (p_params => l_params, p_name => 'p_cust_total', p_val => p_cust_total);
logger.append_param (p_params => l_params, p_name => 'p_cust_total_id', p_val => p_cust_total_id);
logger.append_param (p_params => l_params, p_name => 'p_cust_src_id', p_val => p_cust_src_id);
logger.append_param (p_params => l_params, p_name => 'p_cust_eff_from', p_val => p_cust_eff_from);
logger.append_param (p_params => l_params, p_name => 'p_cust_eff_to', p_val => p_cust_eff_to);
logger.append_param (p_params => l_params, p_name => 'p_cust_valid', p_val => p_cust_valid);
```

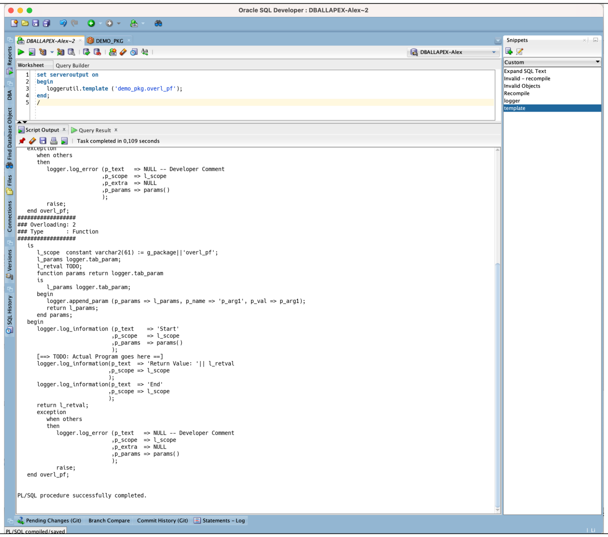
generates procedure or function body



```
create or replace package demo_pkg
as
  procedure overl_pf (p_arg1 in      varchar2
                     ,p_arg2 in out varchar2
                     ,p_arg3 out    varchar2
                     );
  function overl_pf  (p_arg1 in varchar2)
     return number;

end demo_pkg;
```







```
begin
   loggerutil.set_custom_template
      (p_type => 'P'
      ,p_template =>
q'{-- ... Existing declarations go here
   -- Error handling helpers
   C_METHOD_NAME CONSTANT logger_logs.scope%TYPE := '#procname#';
   FUNCTION params RETURN logger.tab_param IS
      v_tab_param logger.tab_param;
   BEGIN
      #logarguments#
   END params;
   -- /Error handling helpers
BEGIN
   NULL; -- ///// @TODO: write code..
EXCEPTION
   WHEN OTHERS THEN
      logger.log_error(
         p_text   => NULL -- Developer Comment
        ,p_scope  => C_METHOD_NAME
        ,p_extra  => NULL
        ,p_params => params()
      );
      RAISE;
END #procname#;
}'
      );
end;
/
```

```
begin
   loggerutil.set_custom_template
      (p_type => 'P'
      ,p_template =>
q'{-- ... Existing declarations go here
   -- Error handling helpers
   C_METHOD_NAME CONSTANT logger_logs.scope%TYPE := '#procname#';
   FUNCTION params RETURN logger.tab_param IS
      v_tab_param logger.tab_param;
   BEGIN
      #logarguments#
   END params;
   -- /Error handling helpers
BEGIN
   NULL; -- ///// @TODO: write code..
EXCEPTION
   WHEN OTHERS THEN
      logger.log_error(
         p_text   => NULL -- Developer Comment
        ,p_scope  => C_METHOD_NAME
        ,p_extra  => NULL
        ,p_params => params()
      );
      RAISE;
END #procname#;
}'
      );
end;
/
```

```
begin
   loggerutil.set_custom_template
      (p_type => 'P'
      ,p_template =>
q'{-- ... Existing declarations go here
   -- Error handling helpers
   C_METHOD_NAME CONSTANT logger_logs.scope%TYPE := '#procname#';
   FUNCTION params RETURN logger.tab_param IS
      v_tab_param logger.tab_param;
   BEGIN
      #logarguments#
   END params;
   -- /Error handling helpers
BEGIN
   NULL; -- ///// @TODO: write code..
EXCEPTION
   WHEN OTHERS THEN
      logger.log_error(
         p_text   => NULL -- Developer Comment
        ,p_scope  => C_METHOD_NAME
        ,p_extra  => NULL
        ,p_params => params()
      );
      RAISE;
END #procname#;
}'
      );
end;
/
```

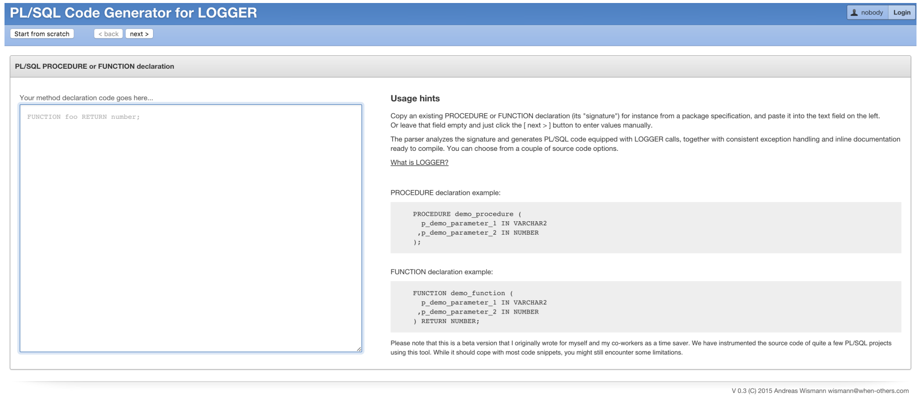
```
-- ... Existing declarations go here
   -- Error handling helpers
   C_METHOD_NAME CONSTANT logger_logs.scope%TYPE := 'overl_pf';
   FUNCTION params RETURN logger.tab_param IS
      v_tab_param logger.tab_param;
   BEGIN
      logger.append_param (p_params => l_params, p_name => 'p_arg1', p_val => p_arg1);
      logger.append_param (p_params => l_params, p_name => 'p_arg2', p_val => p_arg2);
   END params;
   -- /Error handling helpers
BEGIN
   NULL; -- ///// @TODO: write code..
EXCEPTION
   WHEN OTHERS THEN
      logger.log_error(
         p_text   => NULL -- Developer Comment
        ,p_scope  => C_METHOD_NAME
        ,p_extra  => NULL
        ,p_params => params()
      );
      RAISE;
END overl_pf;
```

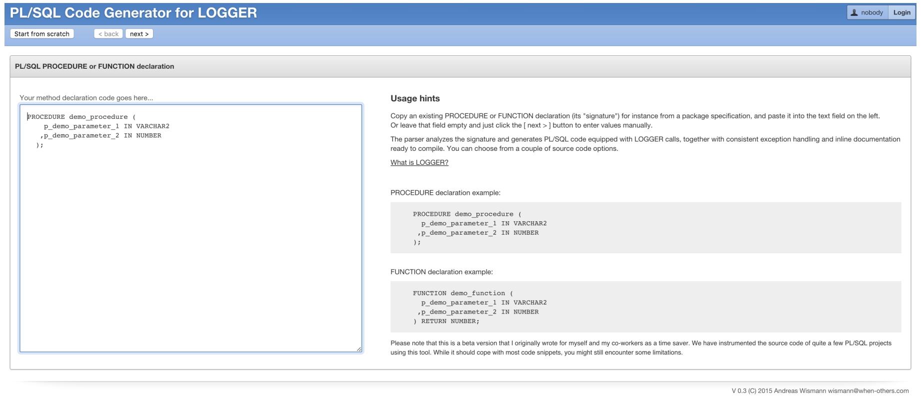
```
begin
   loggerutil.reset_default_templates;
end;
/
```

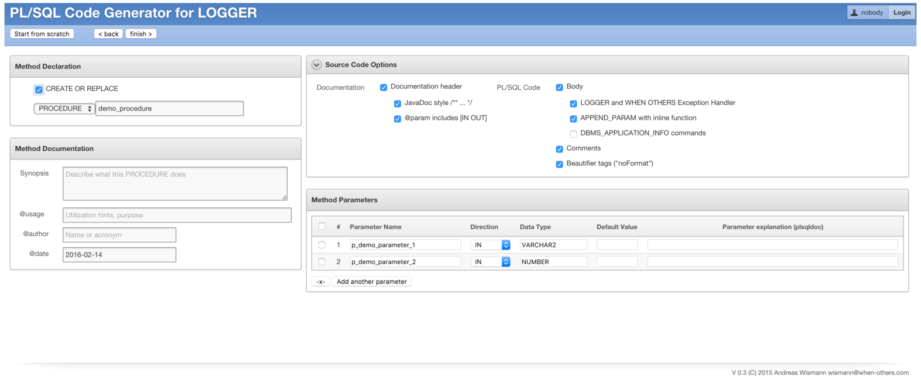
There is
an APP for that!

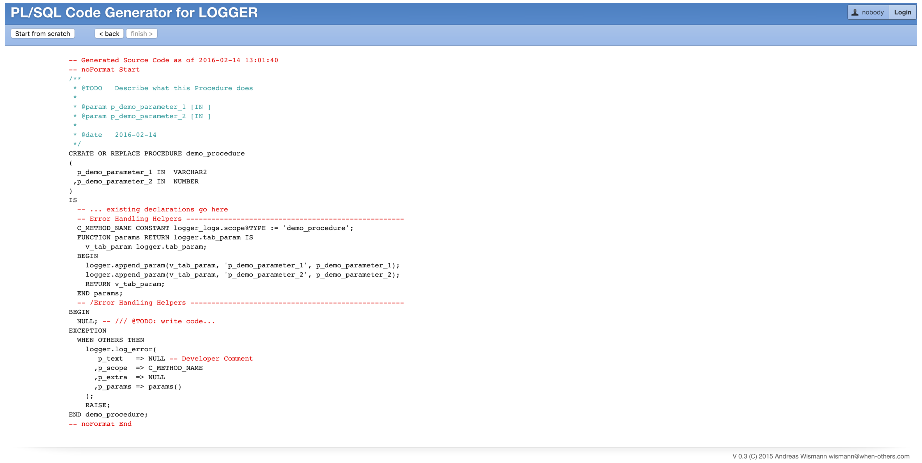
APEX

https://apex.oracle.com/pls/apex/f?p=58849:10





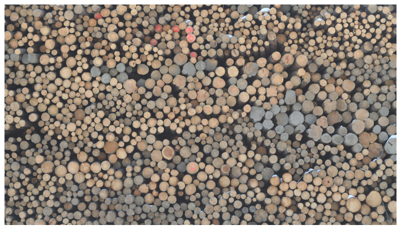





```
-- Generated Source Code as of 2016-02-14 13:01:40
-- noFormat Start
/**
 * @TODO   Describe what this Procedure does
 *
 * @param p_demo_parameter_1 [IN ]
 * @param p_demo_parameter_2 [IN ]
 *
 * @date   2016-02-14
 */
CREATE OR REPLACE PROCEDURE demo_procedure
(
  p_demo_parameter_1 IN  VARCHAR2
 ,p_demo_parameter_2 IN  NUMBER
)
IS
  -- ... existing declarations go here
  -- Error Handling Helpers ---------------------------------------------------
  C_METHOD_NAME CONSTANT logger_logs.scope%TYPE := 'demo_procedure';
  FUNCTION params RETURN logger.tab_param IS
    v_tab_param logger.tab_param;
  BEGIN
    logger.append_param(v_tab_param, 'p_demo_parameter_1', p_demo_parameter_1);
    logger.append_param(v_tab_param, 'p_demo_parameter_2', p_demo_parameter_2);
    RETURN v_tab_param;
  END params;
  -- /Error Handling Helpers --------------------------------------------------
BEGIN
  NULL; -- /// @TODO: write code...
EXCEPTION
  WHEN OTHERS THEN
    logger.log_error(
       p_text   => NULL -- Developer Comment
      ,p_scope  => C_METHOD_NAME
      ,p_extra  => NULL
      ,p_params => params()
    );
    RAISE;
END demo_procedure;
-- noFormat End
```

https://flic.kr/p/7Sb2i
Werner Wattenbergh

http://bit.ly/1O6utS8



https://flic.kr/p/vSwhv
iangbl

https://flic.kr/p/rHzuWM
Alexandre Dulaunoy

http://www.oraopensource.com/
https://github.com/alexnuijten

all·APEX

Alex Nuijten

all·APEX

What Questions do You Have?

Alex Nuijten

www.allAPEX.nl alex@allAPEX.nl @alexnuijten nuijten.blogspot.com