

# SwingBench 2.2

## Reference and User Guide

Author :

Dominic Giles

**Oracle Corporation UK Ltd**

Oracle Parkway,  
Thames Valley Park,  
Reading,  
Berkshire RG6 1RA

# Document Control

## Authors

Name	Company
Dominic Giles	Oracle

## Change Record

Version	Date	Description
Draft 1	2 <sup>nd</sup> January 2002	Initial Version
Draft 2	8 <sup>th</sup> April 2003	Update to reflect code improvements
Draft 3	8 <sup>th</sup> July 2003	New functionality detailed
Draft 4	3 <sup>rd</sup> November 2003	New functionality detailed
Draft 5	1 <sup>st</sup> December 2003	Information on default benchmarks and wizards included.
Draft 6	5 <sup>th</sup> January 2004	Updated to reflect new functionality for 2.1f
Draft 7	8 <sup>th</sup> August 2005	Update to reflect 2.2

## Reviewers

Name	Position
John Nangle	Oracle

## Purpose Of Document

The purpose of this document is to detail the functionality and operation of the *swingbench* test harness.

## Introduction

The UK based Oracle Database Solutions group have developed “*swingbench*”, a Java based test harness, to demonstrate the stress testing of Oracle Databases. *Swingbench* enables developers to define their own classes by implementing a simple interface. These classes are then loaded and run by *swingbench* according to the parameters defined by the user.

This tool is not intended in anyway as a replacement for commercial load generators, it should be viewed as a free alternative to demo certain features of the Oracle database. The code is continually under development and certain features described in this document are likely to change.

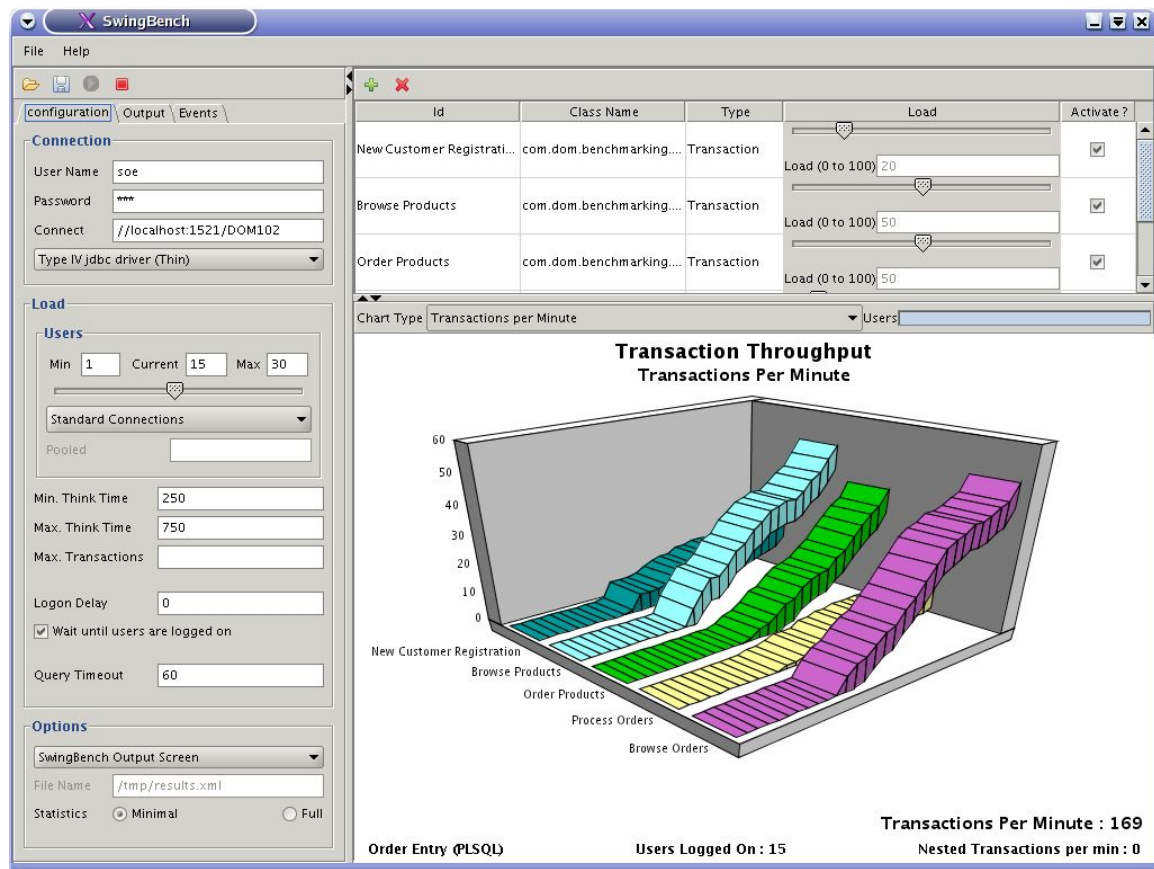
# Table of Contents

.....	2
<b>DOCUMENT CONTROL.....</b>	<b>3</b>
<b>PURPOSE OF DOCUMENT.....</b>	<b>4</b>
<b>INTRODUCTION.....</b>	<b>4</b>
<b>SWINGBENCH.....</b>	<b>6</b>
OVERVIEW.....	6
NEW IN 2.2.....	7
DIRECTORY STRUCTURE.....	8
INSTALLATION.....	9
RUNNING SWINGBENCH.....	10
CONFIGURATION.....	12
<i>User Interface (Swingbench only).....</i>	<i>12</i>
Configuration Tab.....	12
Output Tab.....	13
Events Tab.....	14
Transaction Panel.....	14
Graphing Panel.....	15
<i>XML Configuration File.....</i>	<i>17</i>
ConnectionInformation.....	18
TransactionList.....	18
CoordinatorInformation.....	19
Charts.....	19
ConnectionInitializationCommands.....	19
AllowedErrorCodes.....	19
Statistics.....	19
EnvironmentVariables.....	20
<i>Command Line Options.....</i>	<i>21</i>
Examples.....	22
DEVELOPING TRANSACTIONS.....	23
PL/SQL stubs.....	23
Developing Java Transactions.....	25
SUPPLIED BENCHMARKS.....	28
WIZARDS.....	29
APPENDIX A (SIMPLE BENCHMARK WALK THROUGH).....	31
Step 1 (Set Up the Environment).....	31
Step 2 (Create the Calling Circle Schema).....	31
Step 3 (Generate Data for a Benchmark Run).....	33
Optional Step (Start CPU Monitor on target system).....	34
Alternative Step 4a (Run Swinbench with command line options).....	35
Alternative Step 4b (Modify the swingconfig.xml File).....	36
Step 5 (Run the Swingbench Load Generator).....	36

# SwingBench

## Overview

Swingbench is designed to stress test a database by simulating a workload using user-defined transactions. The user can control the number of users (threads) that attach to a database and the amount and type of work they perform. Users can dynamically monitor the response times and load which is displayed in a series of graphs.



*Swingbench Load Generator*

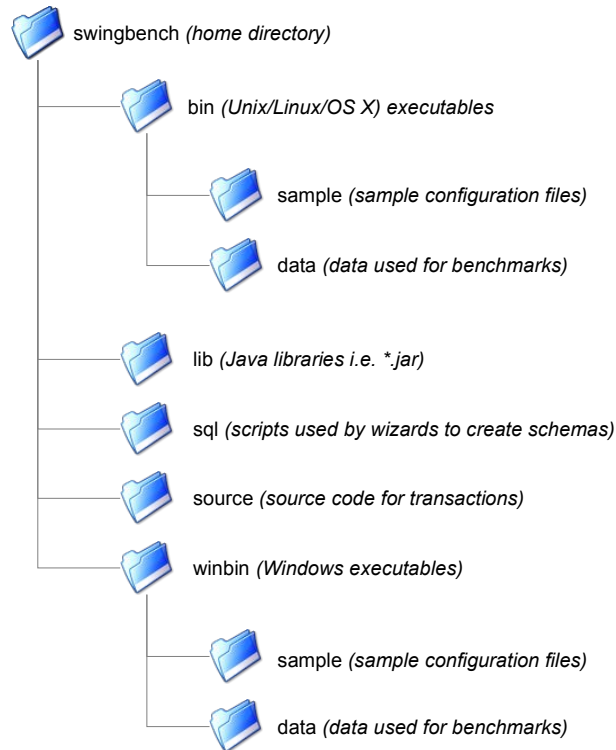
## ***New in 2.2***

The latest release of swingbench 2.2 introduces a number of new features these include.

- A new lightweight graphical load generator called "minibench"
- The coordinator now has a number of command line options (start, stop, status)
- The coordinator can now be run in either graphical or character mode
- The charting engine now uses Oracle's BI-Beans graphing technology
- Better exception handling and error messages in both swingbench and clusteroverview
- Users can now turn off jumping to the events panel in swingbench
- The user chart in clusteroverview now allows users to specify monitored users
- Swingbench can logon/logoff users between transactions (experimental)
- Minor structural changes to swingconfig.xml
- New command line options for coordinator/swingbench/minibench/charbench
- CPU monitor output in charbench
- Simpler configuration for multiple load generators
- Fixes to clusteroverview
- Fixes to wizards
- The order entry benchmark can now be scaled to 100GB
- CPU monitor for database nodes in clusteroverview/swingbench

## Directory Structure

By default the directory structure for the swingbench framework is as follows.



Windows users should use the winbin directory when running swingbench scripts. The scripts used inside of the bin directory use the bash shell, these will need to be modified if the OS does not support this shell (i.e HP UX)

The top level directory (home directory) is referred to as \$SWINGHOME in this document.



## Installation

To run swingbench a Java Virtual Machine (JVM) must be installed on the client platform. The author currently recommends using the latest available 1.4 JVM for the platform. An Oracle client must also be available, this can either be in the form of a full blown Oracle database install or the Oracle instant client downloadable from the Oracle technology network

<http://www.oracle.com/technology/software/tech/oci/instantclient/index.html>

Swingbench is supplied in a single zip file. To uncompress this file issue the command (Unix/Linux)

```
[oracle@dgiles-uk swingbench]$ unzip swingbench
```

On Windows use a tool such as WinZip to perform this operation.

The default installation of swingbench is performed by modifying the values in the \$SWINGHOME/swingbench.env file under Linux/Unix and in \$SWINGHOME/swingbenchenv.bat file under Windows. The contents of an example swingbench.env are shown below.

```
#!/bin/bash
export ORACLE_HOME=/home/oracle/orabase/product/10g
export JAVAHOME=/usr/java/j2sdk1.4.2_08
export SWINGHOME=/home/oracle/swingbench
export ANTHOME=$SWINGHOME/lib
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ORACLE_HOME/lib

export CLASSPATH=$JAVAHOME/lib/rt.jar:$JAVAHOME/lib/tools.jar:
$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$SWINGHOME/lib/mytransactions.jar:$
{SWINGHOME}/lib/swingbench.jar:$ANTHOME/ant.jar
```

The values shown in red need to be modified to reflect the file structure into which the software has been installed.

## Running Swingbench

The core kernel of swingbench has three front ends

- Swingbench : A rich graphical front end ideal for demonstrations.
- Minibench : A lighter weight graphical front end useful for testing over remote connections
- Charbench : A character mode front end useful for data collection (importing into spreadsheets)



*Minibench Load Generator*

All of them are capable of running the same benchmarks and use the same infrastructure.

Swingbench can be invoked on Unix/Linux using the commands

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./swingbench
```

Or on Windows using the commands

```
C:\> cd winbin
C:\> swingbench
```

swingbench, minibench and charbench have a number of command line options, these can be displayed using the “-h” option i.e.

```

[oracle@dgiles-uk bin]$ ./swingbench -h
usage: parameters:
-D <variable=value>    use value for given environment variable
-a                    run automatically
-c <filename>          specify config file
-co <coordinator>      specify/override coordinator in configuration
                        file. i.e. "///<hostname>/CoordinatorServer"
-cpuloc <CPUmonitor > specify/override location of the cpu monitor.
                        Value is in the form "///<hostname>/CPUMonitor"
-cs <connectstring>    override connect string in configuration file
-dt <drivertype>       override driver type in configuration file. Value
                        is either "thin" or "oci"
-h,--help             print this message
-i                    run interactively (default)
-max <milliseconds>   override maximum think time in configuration file
-min <milliseconds>   override minimum think time in configuration file
-p <password>          override password in configuration file
-r <filename>          specify results file
-u <username>          override username in configuration file
-uc <number>          override user count in configuration file.

```

## Configuration

Swing Bench is initialized in one of three ways, via an XML configuration file, manually entering new parameters into the user front end or by using command line options at startup. Because of the complexities of building a fully functional user interface the most complete method of initializing the test harness is via the XML file. All three methods will be discussed in the following sections.

### User Interface (Swingbench only)

The user interface is composed of four sections

- The configuration, results and events panel (left hand side), this is responsible for the entry and updating of parameters such as user populations, connect strings, think times etc.
- The transaction panel (top right hand), maintains a list of the current active transactions
- The graphing panel (bottom right hand), displays results such as transactions per minute and average response time.
- The menu, this provides similar functionality to the toolbar with the addition of more save options

Users can navigate to each of these panels via the mouse. Changes to the current configuration by loading new versions from the menu.

### Configuration Tab

Users can enter data relating to a benchmark run within the “Configuration” tab

The screenshot shows the 'configuration' tab selected in the top navigation bar. The interface is divided into three main sections: 'Connection', 'Load', and 'Options'.  
The 'Connection' section includes fields for 'User Name' (soe), 'Password' (masked with asterisks), 'Connect' (//localhost:1521/DM102), and a dropdown for 'Type IV jdbc driver (Thin)'.  
The 'Load' section includes a 'Users' subsection with 'Min' (1), 'Current' (15), and 'Max' (30) values, a slider, and a 'Standard Connections' dropdown. Below this are fields for 'Min. Think Time' (250), 'Max. Think Time' (750), 'Max. Transactions', 'Logon Delay' (0), a checked checkbox for 'Wait until users are logged on', and 'Query Timeout' (60).  
The 'Options' section includes a dropdown for 'SwingBench Output Screen', a 'File Name' field (/tmp/results.xml), and radio buttons for 'Statistics' (Minimal selected, Full unselected).

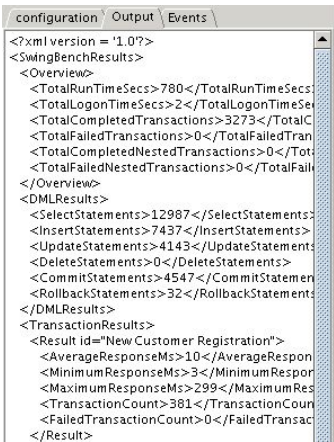
*Configuration Tab*

A description of each of the fields is described in the following table

Field	Description
User Name	The username to which each of the users (threads) will connect
Password	The password for the user
Connect String	The jdbc connect string. This will have the following format if the thin drivers are used <<Host>>:<<Port>>:<<SID>> or //<<Host>>:<<Port>>/<<Service>> or a simple TNS names entry if OCI drivers are used. User may also use the new easy connect format when running against an Oracle10g database.
Driver Type	Used to tell <i>swingbench</i> which driver to use. The OCI driver requires a Oracle client to be installed. The thin driver emulates Oracle Net packets
No. of Users	The total number of users that Swing Bench will attempt to connect to the database (can be changed during benchmark run)
Connection option	This drop down list defines whether users will all have their own connection or use a shared pool of connections (implemented using the Oracle10g datasource implementation)
Pooled	The number of physical connections used (only valid if pooled connections are used)
Min. Think Time	The minimum think time (delay) between transactions (ms)
Max Think Time	The maximum think time (delay) between transactions (ms)
Max Trans	The maximum number of transactions to be run (approximate)
Logon Delay	The time between session connecting to the database (ms)
Wait until users logged on	Indicates whether transactions should be started before all of the user population is logged on.
Query Timeout	The time before swingbench considers a transaction to have failed.
Output Option	Indicated whether the results should be send to the output tab, a user defined file or the OS's standard output.
File Name	The output file if relevant.
Statistics	Indicates whether full or minimal statistics are collected. A full collection requires more runtime memory.

Output Tab

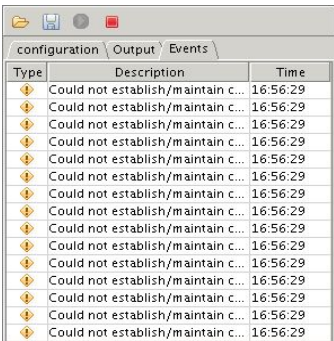
The “Output” tab is used to display the results of a completed benchmark run when the user selects the “output” tab in the configuration panel. The results are displayed in simple XML document.



Results Tab

Events Tab

The “Events” tab displays information generated by swingbench indicating the status or errors within a benchmark run. Typical information might include the failure of a session due to connection issues or lack of data. By default swingbench will automatically jump to this screen whenever a new event occurs, this can be disabled by modifying the configuration file (described in a later section)



Events Tab



- Transaction Throughput Graph displays the number of transactions that have completed per minute. These transactions may have been explicitly declared or be implicit in the processes that are running.
- Nested Transaction Throughput Graph displays the number of nested transactions that have completed per minute. Nested Transactions are those that are dynamically registered by a parent transaction.
- Transaction Maximum, Minimum and Average Graph dynamically displays the response times of transactions
- Nested Transaction Maximum, Minimum and Average Graph displays the response times of nested transactions.
- DML Throughput Graph displays the selects, inserts, updates, deletes, commits and rollbacks.
- CPU History (likely to be deprecated) displays the current CPU load on the target system since the current swingbench session has started.
- CPU and Transaction Overview Graph provides an overview of the target cpuload overlaid against the total transaction load (requires a cpu monitor running on the target system)



## XML Configuration File

By default swingbench reads its configuration properties from a file called swingconfig.xml located in the \$SWINGHOME/bin directory. Users can edit this file to persist changes. There are also sample files located in the \$SWINGHOME/bin/sample directory which can be copied and edited. Users can specify different configuration files by using the “-c” option at startup of swingbench/minibench/charbench.

```
<?xml version = '1.0'?>
<SwingBenchConfiguration Name="Order Entry (PLSQL)" StartMode="Manual"
Output="true" JumpToEvents="false">
  <ConnectionInformation>
    <UserName>soe</UserName>
    <Password>soe</Password>
    <ConnectionString>//localhost:1521/DOM102</ConnectionString>
    <DriverType>thin</DriverType>
    <NumberOfUsers>15</NumberOfUsers>
    <MinNumberOfUsers>1</MinNumberOfUsers>
    <MaxNumberOfUsers>30</MaxNumberOfUsers>
    <Pooled>-1</Pooled>
    <LogonDelay>0</LogonDelay>
    <LogOutPostTransaction>>false</LogOutPostTransaction>
  </ConnectionInformation>
  <TransactionList WaitTillAllLogon="true" MinDelay="250" MaxDelay="750"
MaxTransactions="1" QueryTimeout="60">
    <Transaction Id="New Customer Registration" ShortName="NCR"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess
" Weight="20" Enabled="true"/>
    <Transaction Id="Browse Products" ShortName="BP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts"
Weight="50" Enabled="true"/>
    <Transaction Id="Order Products" ShortName="OP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewOrderProcess"
Weight="50" Enabled="true"/>
    <Transaction Id="Process Orders" ShortName="PO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrders"
Weight="10" Enabled="true"/>
    <Transaction Id="Browse Orders" ShortName="BO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseAndUpdateOrd
ers" Weight="50" Enabled="true"/>
  </TransactionList>
  <ProcessList/>
  <CoordinatorInformation/>
  <CPUMonitor/>
  <Charts>
    <Chart Name="Transactions per Minute" Autoscale="true"
MaximumValue="-1.0"/>
    <Chart Name="DML Operations per Minute" Autoscale="true"
MaximumValue="-1.0"/>
    <Chart Name="Transactions Maximum, Minimum and Average" Autoscale="true"
MaximumValue="-1.0"/>
  </Charts>
  <ConnectionInitializationCommands>
    <Command Type="Connection Property">BatchUpdates=1</Command>
    <Command Type="Connection Property">FetchSize=1</Command>
    <Command Type="Connection Property">StatementCaching=50</Command>
    <Command Type="Connection Property">TcpNoDelay=true</Command>
    <Command Type="SQL Command">alter session set sql_trace = false</Command>
    <Command Type="SQL Command">alter session set optimizer_mode =
first_rows</Command>
  </ConnectionInitializationCommands>
  <AllowedErrorCodes/>
  <EnvironmentVariables>
    <Variable Key="SOE_PRODUCTSDATA_LOC" Value="data/productids.txt"/>
    <Variable Key="SOE_NAMESDATA_LOC" Value="data/names.txt"/>
    <Variable Key="SOE_NLSDATA_LOC" Value="data/nls.txt"/>
  </EnvironmentVariables>
  <Statistics CollectionType="Minimal"/>
</SwingBenchConfiguration>
```

The use of an XML file offers more functionality to the user. It has eight main sections

- ConnectionInformation
- TransactionList
- CoordinatorInformation
- Charts
- ConnectionInitializationCommands
- AllowedErrorCodes
- Statistics
- EnvironmentVariables

The following sections describes their use and attributes

### *ConnectionInformation*

```
<ConnectionInformation>
  <UserName>soe</UserName>
  <Password>soe</Password>
  <ConnectionString>//localhost:1521/DOM102</ConnectionString>
  <DriverType>thin</DriverType>
  <NumberOfUsers>15</NumberOfUsers>
  <MinNumberOfUsers>1</MinNumberOfUsers>
  <MaxNumberOfUsers>30</MaxNumberOfUsers>
  <Pooled>-1</Pooled>
  <LogonDelay>0</LogonDelay>
  <LogOutPostTransaction>>false</LogOutPostTransaction>
</ConnectionInformation>
```

As with the user interface the “ConnectionInformation” node enables users to define the connection and nature of the connections to the database. All of the node’s attributes and child nodes should be considered mandatory. The “ConnectionString” element can be either be a valid oci connectring or a java thin connectring of the form <<hostname>>:<<port>>:<<SID>>. Users can also use Oracle10g's easy connect of the form // <<hostname>>/<<service>>. The “LogonDelay” element, specified in milliseconds, determines how long swingbench should wait between logging users on to the system. A value of -1 for MaxTransactions and Pooled indicates that this functionality is not to be used.

### *TransactionList*

```
<TransactionList WaitTillAllLogon="true" MinDelay="250" MaxDelay="750"
MaxTransactions="-1" QueryTimeout="60">
  <Transaction Id="New Customer Registration" ShortName="NCR"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess
" Weight="20" Enabled="true"/>
  <Transaction Id="Browse Products" ShortName="BP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts"
Weight="50" Enabled="true"/>
  <Transaction Id="Order Products" ShortName="OP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewOrderProcess"
Weight="50" Enabled="true"/>
  <Transaction Id="Process Orders" ShortName="PO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrders"
Weight="10" Enabled="true"/>
  <Transaction Id="Browse Orders" ShortName="BO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseAndUpdateOrd
ers" Weight="50" Enabled="true"/>
</TransactionList>
```

The “TransactionList” describes which transactions are to be executed, where they are located and how often they are to be run. A transaction may be based on the same source file as other processes in the “TransactionList” but must have a unique “Id”. As with the user Interface the “Weight” should be between 0 and 100. The “ShortName” attribute of “Transaction” is used by charbench when displaying the transactions in a table like format.

## CoordinatorInformation

```
<CoordinatorInformation>
  <Location>//kgyl1034.uk.oracle.com/CoordinatorServer</Location>
</CoordinatorInformation>
```

The CoordinatorInformation node describes the location of a Java RMI process responsible for the coordination of multiple swingbench load generators. This functionality allows a much larger load to be run against a single database or clustered databases.

## Charts

```
<Charts>
  <Chart Name="Transactions per Minute" Autoscale="true" MaximumValue="-1.0"/>
  <Chart Name="DML Operations per Minute" Autoscale="true"
    MaximumValue="-1.0"/>
  <Chart Name="Transactions Maximum, Minimum and Average" Autoscale="true"
    MaximumValue="-1.0"/>
</Charts>
```

The Charts node describes the way data is charted in the various graphs. Users can set maximum values and disable autoscaling if they know the profile of the benchmark they are running. This can make it easier for side by side comparisons.

## ConnectionInitializationCommands

```
<ConnectionInitializationCommands>
  <Command Type="Connection Property">BatchUpdates=1</Command>
  <Command Type="Connection Property">FetchSize=1</Command>
  <Command Type="Connection Property">StatementCaching=50</Command>
  <Command Type="Connection Property">TcpNoDelay=true</Command>
  <Command Type="SQL Command">alter session set sql_trace = false</Command>
  <Command Type="SQL Command">alter session set optimizer_mode =
    first_rows</Command>
</ConnectionInitializationCommands>
```

The ConnectionInitializationCommands node allows users to initialize a connection before any transactions are executed against it. This is useful for the debugging of performance problems and to take advantage of vendor specific optimizations. Two ConnectionInitializationCommands child node types exist, Connection Property and SQL Command. The first sets properties specific to the connection itself and is usually vendor specific, only two properties are available in this release BatchUpdates and FetchSize. The latter enables users to set dynamic database properties such as sql\_trace, the full SQL command should be included.

## AllowedErrorCodes

```
<AllowedErrorCodes>
  <ErrorCode Id="1401" />
  <ErrorCode Id="2291" />
  <ErrorCode Id="1" />
</AllowedErrorCodes>
```

The AllowedErrorCodes node simply allows users to indicate that certain database errors should be ignored and not displayed the console. This is useful if the transactions are allowed to insert bogus values into the database to simulate operator error.

## Statistics

```
<Statistics CollectionType="Minimal"/>
```

The Statistics node specifies the type of timings that are recorded. It can currently have two values “Minimal” or “Full”, when “Minimal” is specified only minimum, maximum and averages are collected for each type of transaction or process

## EnvironmentVariables

```
<EnvironmentVariables>
  <Variable Key="SOE_PRODUCTSDATA_LOC" Value="data/productids.txt" />
  <Variable Key="SOE_NAMESDATA_LOC" Value="data/names.txt" />
  <Variable Key="SOE_NLSDATA_LOC" Value="data/nls.txt" />
</EnvironmentVariables>
```

The EnvironmentVariables section supports the runtime specification of variables used by a given benchmark. Each variable comprises of a key (lookup) and a value. Typically these contain locations of seed data or configuration files. Users who have defined their own transactions may specify their own environmental variables here.

The following environmental variables used by swingbench for the two supplied benchmarks are described here.

Environmental Variable	Description
SOE_PRODUCTSDATA_LOC	Location of the product data for the soe benchmark
SOE_NAMESDATA_LOC	Location of the names data for the soe benchmark
SOE_NLSDATA_LOC	Location of the nls data for the soe benchmark
CC_NEWPROCESS_FILE_LOC	Location of the new customer data for the cc benchmark
CC_QUERYPROCESS_FILE_LOC	Location of the query data for the cc benchmark
CC_UPDATEPROCESS_FILE_LOC	Location of the updater data for cc benchmark
CC_DATA_DIR_LOC	Location of the directory containing all of the data files for the cc benchmark (can be used instead of the three individual variables above)

## Command Line Options

Most of the parameters inside of swingbench can be overwritten from the command line. Command line options can be listed by invoking swingbench/minibench/charbench with the “-h” option.

```
[oracle@dgiles-uk bin]$ ./swingbench -h
usage: parameters:
-D <variable=value>      use value for given environment variable
-a                        run automatically
-c <filename>            specify config file
-co <coordinator>        specify/override coordinator in configuration
                        file. i.e. "///<hostname>/CoordinatorServer"
-cpuloc <CPUmonitor >   specify/override location of the cpu monitor.
                        Value is in the form "///<hostname>/CPUMonitor"
-cs <connectstring>      override connect string in configuration file
-dt <drivertype>         override driver type in configuration file. Value
                        is either "thin" or "oci"
-h,--help               print this message
-i                       run interactively (default)
-max <milliseconds>     override maximum think time in configuration file
-min <milliseconds>     override minimum think time in configuration file
-p <password>           override password in configuration file
-r <filename>            specify results file
-u <username>            override username in configuration file
-uc <number>            override user count in configuration file.
```

The character version of swingbench has a few additional parameters

```
usage: parameters:
-D <variable=value>      use value for given environment variable
-a                        run automatically
-c <filename>            specify config file
-co <coordinator>        specify/override coordinator in configuration
                        file. i.e. "///<hostname>/CoordinatorServer"
-cpuloc <CPUmonitor >   specify/override location of the cpu monitor.
                        Value is in the form "///<hostname>/CPUMonitor"
-cs <connectstring>      override connect string in configuration file
-d <seconds>            delay between transaction samples in seconds
-dt <drivertype>         override driver type in configuration file. Value
                        is either "thin" or "oci"
-h,--help               print this message
-i                       run interactively (default)
-max <milliseconds>     override maximum think time in configuration file
-min <milliseconds>     override minimum think time in configuration file
-p <password>           override password in configuration file
-r <filename>            specify results file
-s                       run silent
-u <username>            override username in configuration file
-uc <number>            override user count in configuration file.
-v                       display run statistics (vmstat/sar like output)
-vc                     display run statistics including cpu load
                        (requires cpu monitor running on server)
-vd                     display run statistics including DML values
                        (vmstat/sar like output)
-vt                     display run statistics including transaction
                        values (vmstat/sar like output)
```

## Examples

The following example launches swingbench changing its connect string to use the easy connect string “//kgyl1034/db10g2”. The “-dt” option instructs swingbench to use the oci driver to establish the connections. The “-min” and “-max” options tell swingbench to use a minimum think time of 200 milliseconds and a maximum of 5000 milliseconds. The “-cpuloc” tells swingbench the location of a cpu monitor. The “-c” option tells swingbench to use the sample config file “soeconfig.xml”.

```
[oracle@dgiles-uk:bin] $ ./swingbench -cs //kgyl1034/db10g2 -dt oci -min 200  
-max 5000 -uc 200 -cpuloc //kgyl1034/CPUMonitor -c sample/soeconfig.xml
```

The next example launches charbench with the options discussed in the previous example with the following addition of the “-r” option which tells swingbench to put the results of the run in a file called “doms.txt”. The “-a” option starts charbench running automatically without any user intervention. The “-vc” option displays both the transaction profile as well as the cpu load.

```
$ >./charbench -c sample/soeconfig.xml -cs //kgyl1034/db10g2 -dt oci -min 200  
-max 5000 -uc 200 -cpuloc //kgyl1034/CPUMonitor -r doms.txt -a -vc  
Author : Dominic Giles  
Version : 2.2
```

Results will be written to doms.txt.

Time	Users	TPM	Nested TPM	User	System	Wait	Idle
10:23:46	0	0	0	0	0	0	0
10:23:48	44	0	0	28	19	1	52
10:23:50	134	0	0	60	40	0	0
10:23:52	200	0	0	35	24	1	40
10:23:54	200	7	0	15	8	5	71
10:23:56	200	27	0	32	13	10	45
10:23:58	200	41	0	25	7	4	64
10:24:00	200	67	0	44	9	2	44
10:24:03	200	84	0	46	6	2	46
10:24:05	200	99	0	38	3	4	55
10:24:07	200	121	0	51	4	4	42
10:24:09	200	149	0	38	3	2	57
10:24:11	200	166	0	23	4	7	67
10:24:13	200	187	0	23	3	6	68
10:24:15	200	200	0	16	4	4	75
10:24:17	200	224	0	22	3	5	71
10:24:19	200	244	0	13	6	5	77
10:24:21	200	261	0	9	2	6	83
10:24:23	200	277	0	30	4	4	62
10:24:25	200	296	0	10	7	6	78
10:24:27	200	312	0	15	3	7	75
10:24:29	200	340	0	16	5	4	75
10:24:31	200	361	0	21	3	8	68
10:24:33	200	389	0	27	4	3	66
10:24:35	200	414	0	26	4	4	67
10:24:37	200	436	0	30	5	3	62

## Developing Transactions

There are two approaches to developing transactions with the swingbench framework, developers can either use a series of PL/SQL stubs to add their own code or create java transactions from scratch. Whilst the first approach requires some knowledge of PL/SQL it is relatively simply to build a fully functional benchmark. The second approach is to modify or add code to the existing transactions under the source directory. Both approaches are described in the following sections.

### PL/SQL stubs

To install the PL/SQL stubs run the script *storedprocedures.sql* in the sql directory using sqlplus against a schema of your choice i.e.

```
[oracle@dgiles-uk sql]$ sqlplus dom/dom

SQL*Plus: Release 9.2.0.4.0 - Production on Tue Nov 11 12:48:06 2003

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> start storedprocedures.sql

Type created.

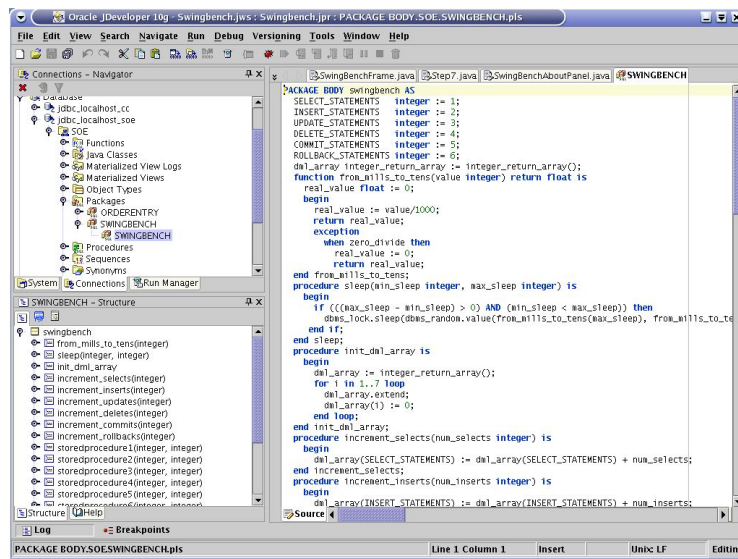
Package created.

Package body created.

SQL>
```

This will create a type called *integer\_return\_array* and a new package called *swingbench*. Users can edit the package body of *swingbench* to add transactions that implement their own functionality. By default users can use six predefined functions to create new transactions, this can be extended by modifying the java code in the java package *com.dom.benchmarking.swingbench.storedprocedures*.

The six PL/SQL functions can be edited using tools such as Oracle Enterprise Manager, Oracle Jdeveloper, TOAD etc.



Oracle JDeveloper

There are six main functions that the developer is free to modify are storedprocedure1 through storedprocedure6, as seen below

```
function storedprocedure1(min_sleep integer, max_sleep integer) return
integer_return_array is
begin
    init_dml_array();
    sleep(min_sleep, max_sleep);
    return dml_array;
end storedprocedure1;
```

The code that ships is simply a stub. It can be trivially modified to include the users own set of SQL operations or calls to other procedures/functions. For example to simple select the number of customers the code could be modified as shown below

```
function storedprocedure1(min_sleep integer, max_sleep integer) return
integer_return_array is
    number_of_customers integer := 0;
begin
    init_dml_array();
    select count(1)
    into number_of_customers
    from customers;
    increment_selects(1);
    sleep(min_sleep, max_sleep);
    return dml_array;
end storedprocedure1;
```

The *init\_dml()* call simply resets the integer array that is returned from the function. It is used to record the total number/type of DML operations that are performed in your code. The *increment\_selects()* call updates the number of select operations that you've performed in this case one. The package comes with a function appropriate for each type of standard database operation, select, insert, update, delete, commit, rollback. Whilst its not necessary to update the array it is necessary to return a integer array (*dml\_array*) at the end of each function.



## *Developing Java Transactions*

As described earlier in this document user defined transactions can be of two types either simple atomic operations or more complex processes that may have many atomic transactions contained within them. To create a user defined transaction developers must first implement a interface.

```
public interface Task {

    public static final String JDBC_CONNECTION = "jdbcConnection";
    public static final String QUERY_TIMEOUT = "queryTimeOut";

    public void init(Map param) throws SwingBenchException;
    public void execute(Map param) throws SwingBenchException;
    public void close();
    public void addTaskListener(TaskListener transListener);
    public void removeTaskListener(TaskListener transListener);
    public void processTransactionEvent(boolean transactionSuccessful,
                                       long transactionPeriod, String id);

    public String getId();
    public void setId(String newProcessName);
    public void setThinkSleepTime(long newMinSleepTime, long newMaxSleepTime);
}
```

To simplify transaction creation an abstract class `JdbcTaskImpl` implements all but the `init`, `execute` and `close` methods. This makes the creation of Tasks a fairly simple operation by allowing the developer to simply extend its functionality.

The following section illustrates a simple transaction that calls a stored procedure `orderentry.browseandupdateorders`.

```

package com.dom.benchmarking.swingbench.plsqltransactions;

import com.dom.benchmarking.swingbench.event.JdbcTaskEvent;
import com.dom.benchmarking.swingbench.kernel.SwingBenchException;
import com.dom.benchmarking.swingbench.kernel.Task;
import com.dom.benchmarking.swingbench.utilities.RandomGenerator;

import com.protomatter.syslog.Syslog;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.SQLException;

import java.util.Map;
import oracle.jdbc.OracleTypes;

import oracle.sql.ARRAY;

public class BrowseAndUpdateOrders extends OrderEntryProcess {
    public BrowseAndUpdateOrders() {}

    public void close() {}

    public void init(Map params) {
        Connection connection = (Connection)params.get(Task.JDBC_CONNECTION);
        try {
            this.getMaxandMinCustID(connection);
        } catch (SQLException se) {
            Syslog.error(this, se);
        }
    }

    public void execute(Map params) throws SwingBenchException {
        Connection connection = (Connection)params.get(Task.JDBC_CONNECTION);
        int queryTimeout = 60;
        if (params.get(Task.QUERY_TIMEOUT) != null)
            queryTimeout = ((Integer) (params.get(Task.QUERY_TIMEOUT))).intValue();
        long executeStart = System.currentTimeMillis();
        int[] dmlArray = null;
        try {
            long start = System.currentTimeMillis();
            try {
                CallableStatement cs = connection.prepareCall(
                    "{? = call                                orderentry.browseandupdateorders(?,?,?) }
");
                cs.registerOutParameter(1, OracleTypes.ARRAY, "INTEGER_RETURN_ARRAY");
                cs.setInt(2, RandomGenerator.randomInteger(MIN_CUSTID, MAX_CUSTID));
                cs.setInt(3, (int) this.getMinSleepTime());
                cs.setInt(4, (int) this.getMaxSleepTime());
                cs.setQueryTimeout(queryTimeout);
                cs.executeUpdate();
                dmlArray = ((ARRAY) cs.getArray(1)).getIntArray();
                cs.close();
            } catch (SQLException se) {
                throw new SwingBenchException(se.getMessage());
            }
            processTransactionEvent(new JdbcTaskEvent(this, getId(),
                (System.currentTimeMillis() - executeStart), true,
                dmlArray));
        } catch (SwingBenchException ex) {
            processTransactionEvent(new JdbcTaskEvent(this, getId(),
                (System.currentTimeMillis() - executeStart), false,
                dmlArray));
            throw new SwingBenchException(ex);
        }
    }
}

```

The code first initializes its values in the `init()` routine which is called at the first loading of the class. This gives the developer the chance to read in values (seed data) from the file system or database. The `execute`

method is responsible for executing the jdbc operations and telling the swingbench framework how long it took to process. The code notifies the framework of a successful or failed transaction by calling `processTransactionEvent` and passing it a `JdbcTaskEvent`.

The `execute` method is passed a hash map containing parameters for its execution. For jdbc transactions it will always contain two key pairs, the jdbc connection (`JDBC_CONNECTION`) and the query timeout (`QUERY_TIMEOUT`). Further values can also be passed by including them in the `EnvironmentVariables` element in the `swingconfig.xml`. (described earlier)

The default swingbench environment ships with the source code for 4 benchmarks

- CallingCircle
- OrderEntry (PL/SQL)
- OrderEntry (jdbc)
- PL/SQL stubs

The java source is located in the `$$SWINGBENCHHOME/source` directory along with a script (ant) to compile all of the code. The following shows its compilation of all of the supplied source.

```
[oracle@dgiles-uk source]$ ./antbuild
Buildfile: /home/oracle/java/SwingBench/swingbench/source/build.xml

init:
  [mkdir] Created dir: /home/oracle/java/SwingBench/swingbench/classes

compile:
  [javac] Compiling 26 source files to /
home/oracle/java/SwingBench/swingbench/classes

dist:
  [jar] Building jar: /
home/oracle/java/SwingBench/swingbench/lib/mytransactions.jar

BUILD SUCCESSFUL
Total time: 4 seconds
```

The script will compile all of the java under the source directory and create a file called `mytransaction.jar` which is placed in the `$$SWINGHOME/lib` directory. This file contains the transactions in compiled form (class). The default configuration file for swingbench (`swingbench.env`, `swingbenchenv.bat`) will use the transactions specified in `mytransactions` before using the default shipped code.

To use your own java transactions simply include the attribute `SourceFile` of the `Transaction` element in the `swingconfig.xml` file.

```
<TransactionList WaitTillAllLogon="true" MinDelay="250" MaxDelay="750">
  <Transaction Id="HR Transaction : Add Employee" ShortName="AE"
SourceFile="com.daves.transcation.HR.addemployee" Weight="100" Enabled="true" />
  <Transaction Id="HR Transaction : Update Employee" ShortName="AE"
SourceFile="com.daves.transcation.HR.updemployee" Weight="100" Enabled="true" />
</TransactionList>
```

## Supplied Benchmarks

Swingbench ships with three functional benchmarks and one template PL/SQL benchmark. These are described below

Benchmark Name	Description	Profile
CallingCircle	<p>The Calling Circle application represents a self-service OLTP application. The application models the customers of a telecommunications company registering, updating and inquiring on a calling circle of their most frequently called numbers in order to receive discounted call pricing. It uses three files</p> <ul style="list-style-type: none"> <li>newccprocess.txt</li> <li>qryccprocess.txt</li> <li>updccprocess.txt</li> </ul> <p>These contain seed data for each type of transaction in the benchmark. They must be recreated for each run using the ccwizard tool. The location of each file is specified in the swingconfig.xml file in the EnvironmentVariable section.</p>	<p>Large amounts of dynamic PL/SQL. Heavy CPU utilization</p> <ul style="list-style-type: none"> <li>Select 83%</li> <li>Insert 7%</li> <li>Update 10%</li> <li>Delete 0%</li> </ul>
Order Entry (PL/SQL)	<p>Models the classic order entry stress test. It has a similar profile to the TPC-C benchmark. This version models a online order entry system with users being required to log-on before purchasing goods. The benchmark uses three files</p> <ul style="list-style-type: none"> <li>names.txt</li> <li>nls.txt</li> <li>productids.txt</li> </ul> <p>These contain sample data used by each of the transactions. The location of each file is specified in the swingconfig.xml file in the EnvironmentVariable section.</p>	<p>Static PL/SQL with a small table (INVENTORY) that is heavily updated.</p> <ul style="list-style-type: none"> <li>Select 50%</li> <li>Insert 30%</li> <li>Update 20%</li> <li>Delete 0%</li> </ul>
Order Entry (jdbc)	As Above	<p>Large amounts of jdbc calls. Network/Client intensive</p> <ul style="list-style-type: none"> <li>Select 50%</li> <li>Insert 30%</li> <li>Update 20%</li> <li>Delete 0%</li> </ul>
PL/SQL stubs	Blank PL/SQL stubs provided for users own benchmarks/extensions	PL/SQL based.

**NOTE :** The only supported way to create the schema's and data for these benchmarks is using the wizards described in the following section.

## Wizards

To assist in the creation of the default benchmarks, callingcircle and order entry, swingbench ships with two wizards which step the user through the process of creating, dropping and in the case of the callingcircle benchmark generating data for each run. The wizards can be launched from the \$SWINGHOME/bin directory using the following commands (for the callingcircle benchmark).

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./ccwizard
```

And (for the order entry benchmark)

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./oewizard
```



As with the swingbench load generator the the wizards use xml configuration files to maintain a persistent record of a users/database settings for a benchmark. The author recommends that users modify these files to reflect their own configuration. These files are held in the held in the \$SWINGHOME/bin directory. The default configuration file for the callingcircle benchmark is displayed below.

```

<?xml version='1.0'?>
<WizardConfig Name="Oracle Entry Install Wizard" Mode="Interactive">
  <WizardSteps RunnableStep="5">
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step0"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step1"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step2"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step3"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step4"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step5"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step6"/>
    <WizardStep SourceFile="com.dom.benchmarking.swingbench.wizards.oe.Step7"/>
  </WizardSteps>
  <DefaultParameters>
    <Parameter Key="operation" Value="create"/>
    <Parameter Key="dbusername" Value="sys as sysdba"/>
    <Parameter Key="dbapassword" Value="manager"/>
    <Parameter Key="username" Value="soe"/>
    <Parameter Key="password" Value="soe"/>
    <Parameter Key="connectionstring" Value="//localhost/DOM102"/>
    <Parameter Key="connectiontype" Value="oci"/>
    <Parameter Key="tablespace" Value="soe"/>
    <Parameter Key="inextablespace" Value="soeindex"/>
    <Parameter Key="datatablespaceexists" Value="false"/>
    <Parameter Key="inextablespaceexists" Value="false"/>
    <Parameter Key="datafile" Value="+Data"/>
    <Parameter Key="indexdatafile" Value="+Data"/>
    <Parameter Key="tsize" Value="100"/>
    <Parameter Key="customercount" Value="25000"/>
    <Parameter Key="ordercount" Value="25000"/>
  </DefaultParameters>
</WizardConfig>

```

Users will typically change values such as their connect string to point to their own instance. From version 2.1f of swingbench users can run the wizards in character mode. This is achieved by changing the “Mode” attribute from “Interactive” to “LightsOut”. In character mode the wizard is driven entirely of the configuration file. The users can change the “operation” attribute to one of three values “create”, “drop” and “generate” to respectively either create a benchmarks schema, drop a existing schema or in the case of the callingcircle benchmark generate data for a new benchmark run.

## Appendix A (Simple Benchmark Walk Through)

This appendix describes the process of creating and running a simple benchmark using the calling circle schema. This example assumes the user is using the default install on Unix/Linux.

**NOTE :** Windows users should use the “.bat” files instead of the unix shell scripts.

### Step 1 (Set Up the Environment)

To run the swingbench framework using the default installation you must first edit the swingbench.env file in the \$SWINGHOME directory to specify the location of you Java Virtual machine and Oracle Home. An example configuration is shown below

```
#!/bin/bash
export ORACLE_HOME=/home/oracle/orabase/product/10g
export JAVAHOME=/usr/java/j2sdk1.4.2_08
export SWINGHOME=/home/oracle/swingbench
export ANTHOME=$SWINGHOME/lib
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ORACLE_HOME/lib

export CLASSPATH=$JAVAHOME/lib/rt.jar:$JAVAHOME/lib/tools.jar:
$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$SWINGHOME/lib/mytransactions.jar:$
{SWINGHOME}/lib/swingbench.jar:$ANTHOME/ant.jar
```

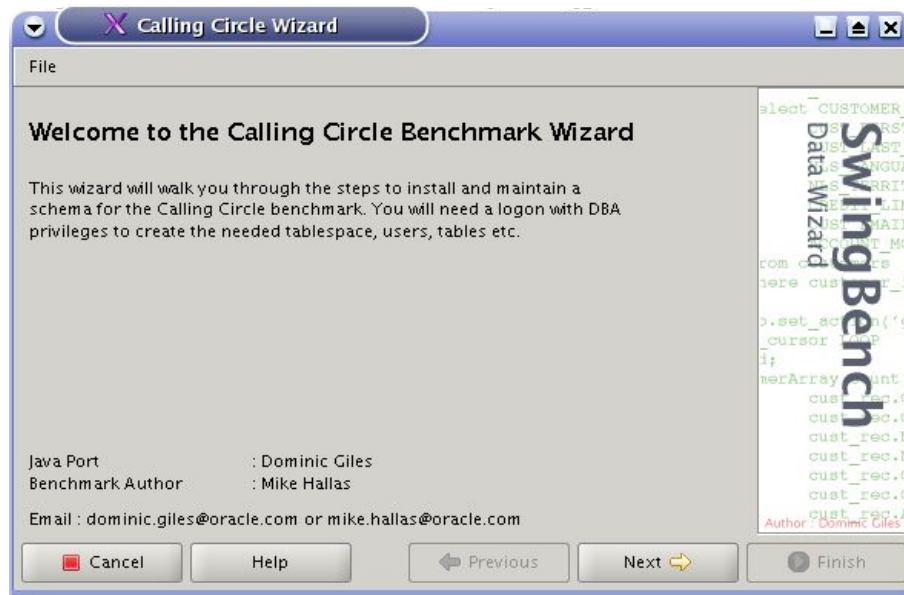
Ensure the values in red are set correctly

### Step 2 (Create the Calling Circle Schema)

The swingbench framework provides two wizards for the installation of the calling circle and order entry benchmarks. These can be found in the “bin” directory of the default install. The calling circle wizard is launched using the following commands

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./ccwizard
```

This will launch the following dialogue



Press next and select the “Create the Calling Circle Schema”. Press next again and enter the details of the database in which you wish to create the the benchmark schema. Specify a user with DBA privileges.



Press next. Enter the details of the the schema, tablespaces and datafiles that will be used to hold the tables.

**NOTE :** If your using Oracle ASM you only need to specify the name of the storage group for the datafile i.e. “+DATA”.

Press next again. The following dialogue allows the user to specify the size of the benchmark. A larger schema allows more runs before the benchmark must be rebuild. Users can increase or decrease the customer value by moving the slider (increases logarithmically).





Press next and next again and the benchmark creation will begin. This may take some time depending on the performance of your machine.

### Step 3 (Generate Data for a Benchmark Run)

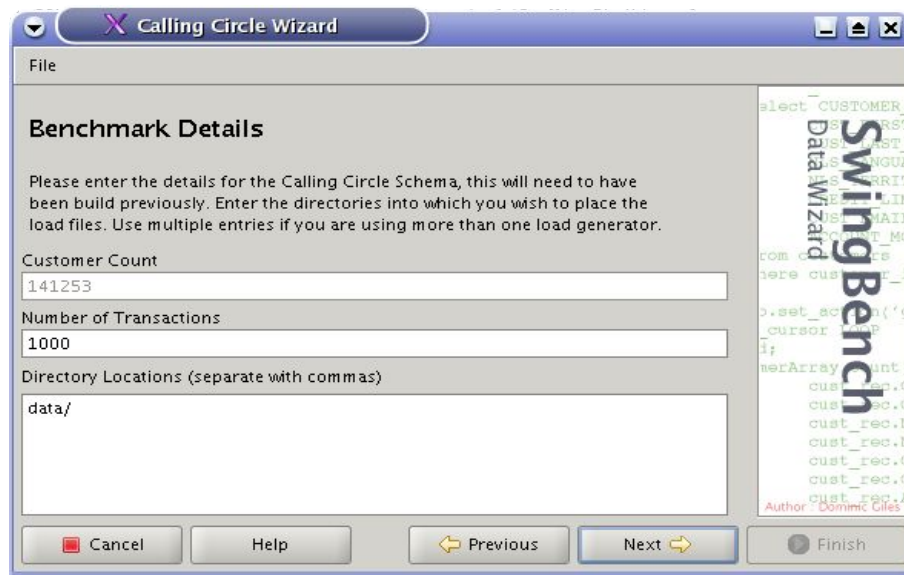
The Calling Circle benchmark requires a new set of data to be generated before each run. The CallingCircle Wizard will generate the necessary files for you. Start the wizard as detailed previously. Press next and select the “Generate Data for Benchmark Run”, press next again. The dialog shown below allows you to enter details of the schema created in Step 2.



The image shows the 'Schema Details' step of the Calling Circle Wizard. The dialog has a title bar with 'Calling Circle Wizard' and a 'File' menu. The main area is titled 'Schema Details' and contains instructions: 'Please enter the details for the Calling Circle Schema, this will need to have been build previously.' Below this are five input fields: 'Username' (containing 'CC'), 'Password' (containing '\*\*\*'), 'Connection String' (containing '//localhost/DOM102'), 'Connection Type' (a dropdown menu showing 'Type II jdbc driver (oci)'), and 'Cancel', 'Help', 'Previous', 'Next', and 'Finish' buttons at the bottom. A vertical watermark 'SwingBench Data Wizard' is visible on the right side of the dialog.

**HINT :** Edit the ccwizard.xml to specify the defaults for your environment

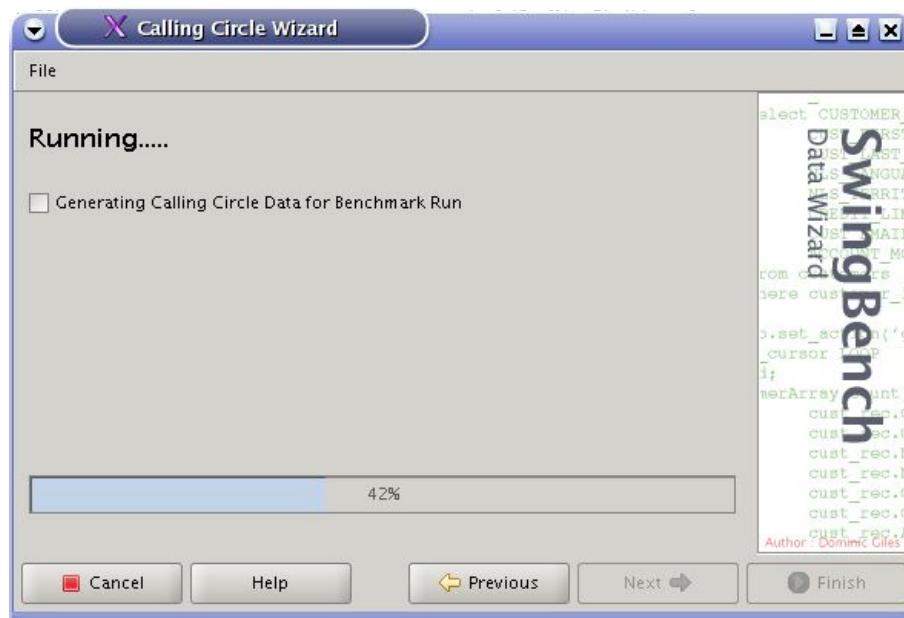
The “Benchmark Details” step allows you to enter how many transactions will be generated for a run and where the transaction data will be written to.



The image shows the 'Benchmark Details' step of the Calling Circle Wizard. The dialog has a title bar with 'Calling Circle Wizard' and a 'File' menu. The main area is titled 'Benchmark Details' and contains instructions: 'Please enter the details for the Calling Circle Schema, this will need to have been build previously. Enter the directories into which you wish to place the load files. Use multiple entries if you are using more than one load generator.' Below this are three input fields: 'Customer Count' (containing '141253'), 'Number of Transactions' (containing '1000'), and 'Directory Locations (separate with commas)' (containing 'data/'). At the bottom are 'Cancel', 'Help', 'Previous', 'Next', and 'Finish' buttons. A vertical watermark 'SwingBench Data Wizard' is visible on the right side of the dialog.

The “Number of Transactions” field specifies how many will be created per directory location. It also determines the the length of a benchmark run. On a 4 processor 1.6.Ghz Xeon Intel white box running Linux Advanced server it takes 1 minute to consume 500 transactions, 3 minutes for 2,000, 25 minutes for 20,000. As a result it is important to create enough transactions to last the length of your demonstration.

After specifying the number of transactions, or accepting the default, hit next and then next again. This will begin the benchmark data generation.



After the data generation has completed the wizard will display the hit ratio (NOTE : this is not an error message), if this ratio exceeds 30% you should consider regenerating the CC benchmark schema. There should now be 3 files located in each of the directory locations specified i.e.

```
[oracle@load1 bin]$ ls -l data
total 9872
-rw-r--r-- 1 oracle oinstall 2531642 Apr 4 10:55 newccprocess.txt
-rw-r--r-- 1 oracle oinstall 322500 Apr 4 10:55 qryccprocess.txt
-rw-r--r-- 1 oracle oinstall 7226736 Apr 4 10:55 updccprocess.txt
```

### *Optional Step (Start CPU Monitor on target system)*

It is possible to start a CPU monitor on the target system where the database is located. You'll need to make sure that the swingbench software is also installed on this system (see step 1). After this is completed invoke the cpu monitor with the following command.

```
[oracle@node1 bin]$ ./cpumonitor
CPU monitor started successfully
```

## Alternative Step 4a (Run Swingbench with command line options)

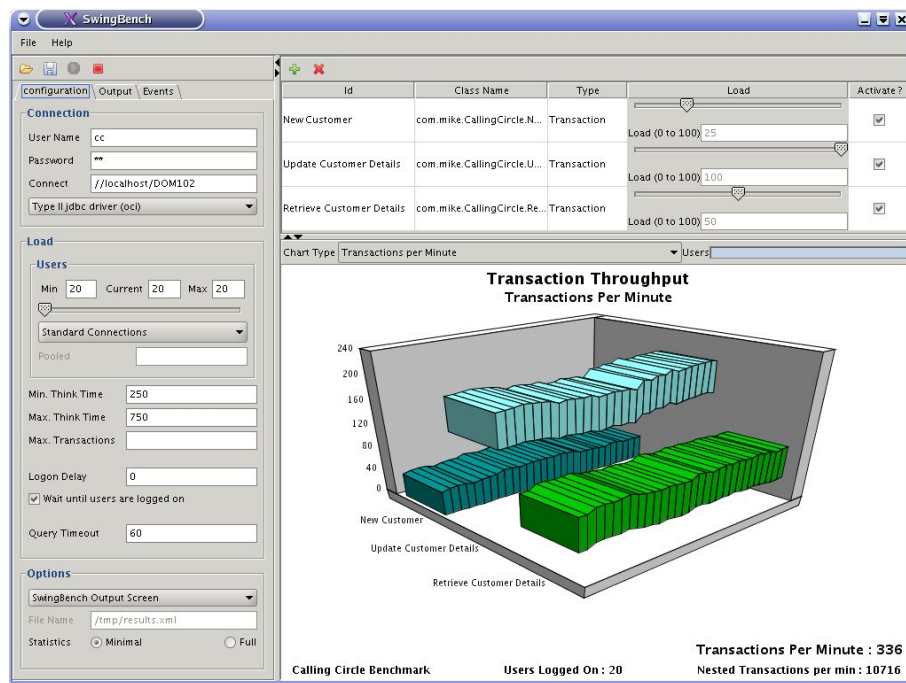
It is now possible to invoke swingbench with a command line option such as

```
[oracle@load1 bin]$ ./swingbench -c sample/ccconfig.xml -cs //node1/ORCL -dt  
oci -D CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/ -uc 20
```

or if you started a cpu monitor on the target system.

```
[oracle@load1 bin]$ ./swingbench -c sample/ccconfig.xml -cs //node1/ORCL -dt  
oci -D CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/ -uc 20 -cpuloc //  
node1/CPUMonitor
```

This invokes swingbench using the sample callingCircleconfiguration supplied by default but overriding the connectstring and location of the newly generated data files.



Alternatively you could start minibench or charbench using similar parameters

```
[oracle@load1 bin]$ ./charbench -c sample/ccconfig.xml -cs //node1/ORCL -dt  
oci -D CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/ -uc 20 -vt -a  
Author : Dominic Giles  
Version : 2.2
```

Results will be written to results.xml.

Time	Users	TPM	Nested TPM	NC	UCD	RCD
12:13:03	0	0	0	0	0	0
12:13:05	13	0	0	0	0	0
12:13:07	20	4	88	0	0	4
12:13:09	20	5	459	0	0	5
12:13:11	20	20	816	0	14	6

### *Alternative Step 4b (Modify the swingconfig.xml File)*

Swingbench supplies sample configuration files for all four benchmarks, these are located in the sample directory. To run the callingcircle benchmark first copy the file ccconfig.xml from the sample directory to swingconfig.xml in the bin directory replacing the existing file (make sure you back this up if you intend to use its values later on) . You may wish to edit this file to reflect your database i.e. Connect string, username, password, think times etc. You will also need to ensure that the environment variables specific to the callingcircle benchmark are set correctly. These are located in the following section of the configuration file

```
<EnvironmentVariables>
  <Variable Key="CC_QUERYPROCESS_FILE_LOC"
Value="/tmp/qryccprocess.txt" />
  <Variable Key="CC_UPDATEPROCESS_FILE_LOC"
Value="/tmp/updcccprocess.txt" />
  <Variable Key="CC_NEWPROCESS_FILE_LOC" Value="/tmp/newccprocess.txt" />
</EnvironmentVariables>
```

### *Step 5 (Run the Swingbench Load Generator)*

To launch the *swingbench* load generator run the following command

```
[oracle@dgiles-uk swingbench]$ cd bin/
[oracle@dgiles-uk bin]$ ./swingbench
```

Ensure the details are correct in the configuration panel and start the load with the “Start” button.

**NOTE** : the users will begin to log off when the load generator runs out of data.