Code Coverage Tool

Overview

The command line code coverage tool is used to gather and report code coverage statistics during nightly builds where using an IDE tool is not possible. The tool consists of three different pieces:

1. Flex preload SWF
2. Java Code Coverage Server
3. Java Code Coverage Reporter

The Flex portion of the tool is a SWF that is preloaded for every SWF that is run. The job of the preload SWF is to listen to the execution of lines in the running SWF. Every executed line is sent over a socket to the Java server. The Java server will write the data to disk where it can be analyzed later. After all of the SWFs have been run the Java server is shutdown to flush all the data to disk.

After the execution data has been collected the Code Coverage Reporter can be run to analyze the results. The reporting tool provides a summary of the total line coverage and total method coverage. By default, more detailed information can be found at the package, file, and method level. At the package level the following information is available:

1. Total lines executed in the package, total lines in the package, and percent coverage.
2. Total methods executed in the package, total methods in the package, and percent coverage.
3. List of files in the package (see file level details below).

At the file level the following information is available:

1. Total lines executed in the file, total lines in the file, and percent coverage.
2. Total methods executed in the file, total methods in the file, and percent coverage.
3. List of unexecuted lines and unexecuted methods.
4. List of methods in the file.

At the method level the following information is available:

1. Total lines executed in the method, total lines in the method, and percent coverage.

How to Use

All of the Code Coverage Tool's command files are located in its "bin" directory. It is recommended to add this directory to your path. The following commands are available on either the Windows or Unix platforms:

ccreporter	Runs the Code Coverage Reporter.
ccserver	Used to start and stop the Code Coverage Server.
start-ccserver	Starts the Code Coverage Server in the background.

For more information on the command line options and configuration options see the Code Coverage Server and Code Coverage Reporter sections below.

The Code Coverage Server is started before running of any tests. After the tests have completed the Code Coverage Server is shut down to flush all data to disk. The tests can be either flexunit or mustella tests or a combination of both. After all the data has been collected one or more reports can be created.

Example

This example will show how to run code coverage on a mustella test. It assumes the Code Coverage Tool's "bin" directory in on the path and the user is located in the mustella directory. First the Java server is started in the background:

```
$ start-ccserver

$ Apache Flex Code Coverage Server
```

```
Version 0.9

listening on port 9097
```

Next the mustella test is run:

>./mini_run.sh components/Button

After the mustella test finishes the Code Coverage Server is shutdown:

```
$ ccserver stop

Apache Flex Code Coverage Server

Version 0.9

stopping Apache Flex Code Coverage Server
```

Now that the raw coverage data has been captured we can generate a report to see the coverage of mx.controls.Button. The overall coverage summary is output to the console and the details are written to the report file.

```
$ ccreporter -include mx.controls.Button

Apache Flex Code Coverage Reporter

Version 0.9


Found 3 files in /Users/dloverin/ccdata

Analyzing 3 files

Results:

Line Coverage:   78%

Method Coverage: 77%

Report output to /Users/dloverin/ApacheFlex/sdk/mustella/ccreport.xml
```

Now ccreport.xml can be examined to look at the unexecuted lines and see how the tests can be improved.

If code coverage is run on multiple packages using the "-hideFiles" option is recommended to reduce the size of the report. Once packages with low coverage are found the report can be regenerated with full details but filtered to only include the classes of concern.

```
Note on Mustella Timeouts
```

When mustella tests are running under code coverage, cpu intensive tests will run slower than normal. If any of your mustella tests explicitly set the timeout attribute it is recommended to use the mustella timeout options to effectively double that timeout. For example, if you have a test that has a timeout of 5 seconds, set the mustella -set_timeout option to 5 seconds which will add 5 seconds to the timeout of every test. Setting -timeout can also be useful to prevent timeout failures:

>./mini_run.sh -timeout=10 -set_timeout=5 -all

How it works

When the Code Coverage Server (ccserver) is started it modifies the user's mm.cfg file, adding the preloadSWF entry. The preloadSWF entry is set to the path of the CodeCoveragePreloadSWF.swf in the Code Coverage's Tool lib directory. The ccserver then waits for the preload SWF to connect with it over the data port. Once a connection is established the server creates a thread to read the trace data from the preload SWF and writes the data to a file. When the next preload SWF makes a connection the previous thread is stopped and the data file is closed before create a new thread and a new data file. Each SWF that is run results in a new data file in the data directory.

The preload SWF initiates a stream of trace data from the Flash Player when it is loaded. The trace data includes the paths, package, file, and line number that is executed. The trace information is optimized and written over a socket to the ccserver. Two optimizations are performed by the Flex client:

1. Id numbers are generated for the debug file strings. The id and debug file string are sent once to the server and afterwards only the id is sent.
2. Executed lines are only sent once to the server. For the purposes of this tool we only care if a line is execute or not, not how many times it is executed.

The ccserver listens for commands on the command port. When it receives the "stop" command it closes any existing data files and exits.

After the ccserver has shutdown the Code Coverage Reporter can be run to read the data files and create a report. The Code Coverage Reporter reads a data file to get the executed lines and also reads the executed SWFs to get the total number of lines that could be executed. This allows the reporter to compare the executed lines vs. the total number of lines. If any of the SWFs have been deleted before the Reporter is run, the report will failure since that total number of lines will cannot be calculated.

Code Coverage Data File Format

At a high level the file contains a list of executed lines. To save space, executed lines are only written once to the file. The file contains the name of the SWF that was run to allow the total number of lines to examined and compared to the executed lines in the data file. There are three kinds of lines in the data file:

#i,file	Line starts with a '#' character. Defines a string id and its value; where *i* is the id of *file*. The number is used in an executed instruction line to save space. Example: #1,E:\dev\4.y\frameworks\projects\framework\src;mx\managers;SystemManager.as Where the "1" represents E:\dev\4.y\frameworks\projects\framework\src;mx\managers;SystemManager.as in executed instruction lines.
i,lineNumber	An executed line; where *i* is the id of the file and *lineNumber* is the line number executed. Example: 1,100 If id #1 is E:\dev\4.y\frameworks\projects\framework\src;mx\managers;SystemManager.as, then this line means line number 100 was executed in SystemManager.as.
@filename	Line starts with an '@' character, where filename is the URL of the SWF that was executed. Example: @file:///ApacheFlex/sdk/mustella/tests/components/Button/swfs/Button_DataBinding.swf

Code Coverage Server

The Code Coverage Server's primary job is to write code coverage data to the data directory. The server performs four different functions:

1. Write code coverage data to the data directory.
2. At start up time this server is responsible for modifying the user's mm.cfg file so a preload SWF will be loaded for every SWF that is run.
3. Provide a policy file to the Flash Player to allow code coverage data to be sent from the preload SWF to the server over the data port.
4. Accept commands on the command port. The server is shutdown by sending it a stop command over the command port.

The format of the command line is:

```
ccserver [options]* [start | stop]
```

```
start_ccserver
```

The server can be started with either the "start_ccserver" command or the "ccserver start" command. The difference is the "start_ccserver" starts the server in the background so the command line is not blocked.

The server is shutdown with the "ccserver stop" command.

Command Line Options

append	Start the server without deleting all of the code coverage data files in the dataDirectory (see configuration options). New data files will be added to the existing files. The default behavior is to delete all existing data files on start up. This option is only useful when used with the "start" command.
help	shows help

Configuration options

The ccserver.properties file is located in the Code Coverage Tool's lib directory.

Option	Default Value	Comments

host	localhost	The host Code Coverage Server is located on.
dataPort	9097	The port execution data is written to.
commandPort	9098	The port server commands are written to.
policyFilePort	9843	The port the Flash Player uses to request the policy file to allow the preload SWF to use the data port.
preloadSWF	CodeCoveragePreloadSWF.swf	The preload SWF. Located in the same directory as ccserver.jar.
mmCfgPath	The user's home directory.	Absolute path to the user's mm.cfg file. If the server is unable to modify this file the preload SWF will not be loaded and no execution data will be recorded.
dataDirectory	<user_home>/ccdata	By default a "ccdata" directory will be created if it does not exist in the user's home directory. This directory will contain a file for each SWF executed. The name of each file will be ccdata_<i>.text, where <i> is a sequential number starting at zero.

Code Coverage Reporter

The Code Coverage Reporter can be run without command line options if the ccserver is also run with the default configuration.

The format of the command line is:

```
ccreporter [options]* [filename | directory]*
```

The command is followed by zero or options and ends with a list of files and directory to read code coverage data files from. If the list of files and directory is not provided then the reports looks in the "ccdata" directory in the user's home directory (also default location of the ccserver).

Command line options

Option	Parameters	Default Value	Comments
-exclude	filter	Empty	A string to exclude package names and class names from the report. This prevents them from influencing the code coverage numbers. Simple wild card characters, '*', '?' may be used in the package and class names. For example to exclude all classes in the mx.core package use the following: -exclude mx.core.*
-help	None		Display help.
-hide-unexecuted	None	false	Determines if unexecuted line and unexecuted methods in a file are included in the report. Including this data can make the report large and is best used when the number of the files is more focused by using with the -include and -exclude options.
-hide-files	None	false	Determines if the files in a package are listed. If files are hidden then only the overall coverage summary and packages will be reported.
-hide-methods	None	false	Determines if the methods in a package are listed. If methods are hidden then only the overall coverage summary, packages, and files will be reported.
-hide-packages		false	Determines if the packages in a SWF are listed. If packages are hidden only the overall coverage summary will be reported.
-include	filter	None.	A string to determine which classes are included. Simple wilds card characters, '*', '?' may be used. To only see classes in the mx.controls package use the following: -include mx.controls.*
-output	filename	ccreport.xml	Control the output file of the report. The default file, "ccreport.xml" is output the the current working directory.
-report-factory	class name	XMLReportFactory	Allows alternate reports to be created. Specify the fully qualified class name of the factory. If the factory is in another jar, place the jar in the same directory as ccreporter.jar.