Gump3CommandLineOptions

Command line options reference

```
Utility commandline interface for Gump.

Usage:
  ./gump command [opts ...]

Available commands are:

  run             -- run pygump
  debug           -- run pygump in debug mode, attaching pdb
  debug-with-wing -- run pygump in debug mode, attaching the Wing IDE
  test            -- run the pygump unit tests
  dynagump        -- run the dynagump web application server
  webgump         -- run the webgump application server
  update-host     -- update the configuration of a gump host
  create-database -- create a new gump MySQL database
  pycompile       -- compile all pyump source files

Run

  ./gump help [command]

for more information about a particular command.

Run

  ./gump help variables

for more information about the environment variables that alter gump its
behaviour.
```

Environment variables

```
Gump needs various other programs available in order to run. You
can change which programs gump tries to use using environment variables.
In addition, several core gump settings are also customizable using
environment variables.

You can set all these variables (except for GUMP_HOME) in the file

  /home/lsimons/svn/gump/branches/Gump3/giraffe-settings.sh

the location of this file is found as follows:

  GUMP_HOME/GUMP_HOSTNAME-settings.sh

Recognized variables are:

  GUMP_HOME     -- location of the gump subversion checkout. Defaults
                   to the current working directory if possible.
  GUMP_HOSTNAME -- name of this machine. Defaults to the output from the
                   hostname command.
  GUMP_ENV_FILE -- location of the file that contains the custom
                   settings to load (i.e. the file mentioned above). You
                   can override GUMP_HOME and GUMP_HOSTNAME here, but that
                   may have some unpredictable effects.
  GUMP_PYTHON   -- the name of the python executable to use. Defaults to
                   the latest version of python that is installed. Note
                   that pygump is supported only on python2.4.
  GUMP_WORKDIR  -- the directory that pygump will generate various files in
                   (like log output). Defaults to GUMP_HOME/pygump/work.
  JAVA_HOME     -- the location of a java development kit. Gump tries to
                   work with any JDK, but results may vary (for example,
                   both ant and maven require jdk 1.2 at least).

These variables are only used by dynagump:

  JAVA_OPTIONS  -- Extra options to pass to the JVM.
  JETTY_PORT    -- Override the default port for Jetty. Defaults to 8080.
  JETTY_ADMIN_PORT -- The port where the jetty web administration should
                   bind. Defaults to 8081.
  JAVA_DEBUG_PORT -- The port the JVM debug server should listen to.
                   Defaults to 8082.

Of course, the various commands that gump issues may also behave
differently based on environment variables. For example, maven reacts to
MAVEN_HOME, many make-based build scripts respect the CC environment
variable, etc etc.
```

The 'run' command

```
Run pygump.

usage: gump run [options ...]

options:
  -h, --help            show this help message and exit
  -d, --debug           print extra information
  -q, --quiet           print as little information as possible (overrides
                        --debug)
  --homedir=HOMEDIR     the base directory for gump
  --hostname=HOSTNAME   the hostname gump will use
  --workdir=WORKDIR     the working directory gump will use
  --logdir=LOGDIR       the directory gump will write logs to
  -w WORKSPACE, --workspace=WORKSPACE
                        absolute path to the workspace gump will use
  -u, --do-updates      run cvs and svn updates
  -b, --do-builds       run builders
  --databaseserver=DATABASESERVER
                        hostname of the database server gump will connect to
  --databaseport=DATABASEPORT
                        port of the database server gump will connect to
  --databasename=DATABASENAME
                        name of the database gump will connect to
  --databaseuser=DATABASEUSER
                        username gump will use to connect to the database
  --databasepassword=DATABASEPASSWORD
                        password gump will use to connect to the database
  --color               write log output using ansi color codes
  --irc=IRC             enable an IRCbot during this run using
                        nickname@irc.freenode.net/channel
  --attach-pdb          Run within the Python Debugger (PDB)
  --attach-wingdb       Run within the Wing IDE Debugger
```

The 'dynagump' command

```
    Run Dynagump.

    Usage:
      ./gump dynagump dynagump-action [dynagump-args ...]

    The available actions are:

  run       Run in a servlet container
  admin     Run in a servlet container and turn on container web administration
  debug     Run in a servlet container and turn on JVM remote debug
  profile   Run in a servlet container and turn on JVM profiling

    If no action is specified, gump passes run as the action to
    execute.
```

The 'debug' command

```
Run pygump in debug mode.

Usage:
  ./gump debug [gump.py-args ...]

This is not the same as executing the 'run' command with a '--debug'
parameter. Using this command will actually start the command line
debugger pdb to run gump in, whereas the '--debug' option customizes
the log verbosity gump will use.

This command otherwise accepts the same arguments as the 'run'
command.
```

The 'debug-with-wing' command

```
Run pygump in debug mode.

Usage:
  ./gump debug [gump.py-args ...]

This is not the same as executing the 'run' command with a '--debug'
parameter. Using this command will actually start the debug connector
for the Wing IDE and attach it to the gump process, whereas the
'--debug' option customizes the log verbosity gump will use.

This command otherwise accepts the same arguments as the 'run'
command.
```

The 'test' command

```
    Run pygump its unit tests.

    Usage:
      ./gump test [OPTIONS]

    Available options include:
--version             show program's version number and exit
-h, --help            show this help message and exit
-a, --annotate        Page annotations
-c, --clear           Clear all .pyc and .pyo files in the project base and
                      included paths
-d, --debug           Debug run - do not catch exceptions
-q, --quiet           Quiet
-s, --stats           Give coverage stats
-v, --verbose         Verbose.
-l LOGDIR, --logdir=LOGDIR
                      Directory to write annotation log files (for use with
                      -a).
Controlling Coverage Paths:
  These options are only necessary if your project layout deviates from
  what pylid expects.
  -b DIR, --base=DIR  Project base directory. Can be passed multiple times.
                      (Default: "..")
  -e DIR, --exclude=DIR
                      Exclude path from coverage analysis. Can be passed
                      multiple times. (Default: ".")
  -i DIR, --include=DIR
                      Include path for analysis. Can be passed multiple
                      times (Default: "..")
```

The 'pydoc' command

```
    Runs a pydoc server on port 1234.

    Usage:
      ./gump pydoc

    Visit http://localhost:1234/ to see the documentation it provides
```

This document was last generated on Thu Jul 7 12:57:08 CEST 2005. It can be generated using the command 'gump help generate-reference'