Bugjar Documentation

Release 0.1.0

Russell Keith-Magee

Mar 23, 2023

Contents

Bugjar is part of the **'BeeWare suite'**_. The project website is http://pybee.org/bugjar.

Anyone who learned to code in the mid to late 80s probably spent some time with a Borland compiler – probably either Turbo Pascal or Turbo C. One of the best features of the Turbo compilers was their IDE – and in particular, a really good visual debugger that would let you inspect code while it was running.

Then we all moved to Unix, and somehow forgot what a good debugger was. GDB is perfectly functional, but isn't very intuitive. GDB gives you perfect control over the execution of your code, but bad contextual information to let you know what control you should be exercising.

Then came Python. Python's execution model contains excellent debugging hooks, and supplies PDB as a proof of concept. PDB is an interface that shares many similarities with GDB – text mode, fantastic control, but very bad contextual information.

So - enter `bugjar`. A graphical interface for debugging code. PDB, but with the context to help you step through code in a meaningful way.

CHAPTER 1

Quickstart

Bugjar can be installed with pip:

```
$ pip install bugjar
```

You can then debug a Python script by typing the following at a shell prompt:

```
$ bugjar myscript.py arg1 arg2
```

This will start a graphical interface, with `myscript.py` loaded into the source code window. You can set (or remove) breakpoints by clicking on line numbers; you can step through and into code; or you can set the program running unconstrained. Each time the debugger stops at a breakpoint, the inspector will be updated with the current contents of locals, globals, and builtins.

The Python script will run using your current environment; if you have an active virtualenv, that environment will be current.

When you quit the debugger, the script will be terminated.

1.1 Problems under Ubuntu

Ubuntu's packaging of Python omits the `idlelib` library from it's base packge. If you're using Python 2.7 on Ubuntu 13.04, you can install `idlelib` by running:

```
$ sudo apt-get install idle-python2.7
```

For other versions of Python and Ubuntu, you'll need to adjust this as appropriate.

1.2 Problems under Windows

If you're running Cricket in a virtualenv, you'll need to set an environment variable so that Cricket can find the TCL graphics library:

```
$ set TCL_LIBRARY=c:\Python27\tcl\tcl8.5
```

You'll need to adjust the exact path to reflect your local Python install. You may find it helpful to put this line in the `activate.bat` script for your virtual environment so that it is automatically set whenever the virtualenv is activated.

CHAPTER 2

Documentation

Documentation for bugjar can be found on **'Read The Docs'_**.

CHAPTER 3

Community

Bugjar is part of the **'BeeWare suite'_**. You can talk to the community through:

- @beeware@fosstodon.org on Mastodon
- Discord

We foster a welcoming and respectful community as described in our BeeWare Community Code of Conduct.

Contents:

3.1 Headless mode

Bugjar can also operate in a headless mode. This can be used to debug processes running on a remote machine (although it also works on local machines).

In headless mode, Bugjar is split into two parts:

- The **Net**: a headless backend responsible for debugging code
- The **Jar**: the GUI used to inspect code.

To debug in headless mode, you first start a headless debugger (the net) on the process that you want to debug:

```
$ bugjar-net myscript.py arg1 arg2
```

Then, on the machine that you want to visualize the debugging session, you start the user interface (the jar), and attach it to the net:

```
$ bugjar-jar –host example.com
```

If the net and the jar are running on the same machine, the `--host example.com` argument can be ommitted.

Unlike local mode, when you quit the debugger, the script will *not* be terminated by closing the jar. If you close the jar, and reopen a new session, the GUI will resume where it left off. The net is responsible for running the script; when the net is stopped, the script will be terminated.

3.2 Contributing to Bugjar

If you experience problems with bugjar, log them on GitHub. If you want to contribute code, please fork the code and submit a pull request.

3.2.1 Setting up your development environment

The recommended way of setting up your development envrionment for `bugjar` is to install a virtual environment, install the required dependencies and start coding. Assuming that you are using `virtualenvwrapper`, you only have to run:

```
$ git clone git@github.com:pybee/bugjar.git
$ cd bugjar
$ mkvirtualenv bugjar
```

bugjar uses `unittest` (or `unittest2` for Python < 2.7) for its own test suite as well as additional helper modules for testing. To install all the requirements for bugjar, you have to run the following commands within your virutal envrionment:

```
$ pip install -e .
$ pip install -r requirements_dev.txt
```

In case you are running a python version `< 2.7` please use the `requirements_dev_python2.7.txt` instead because `unittest2` is not part of the standard library for these version.

Now you are ready to start hacking! Have fun!

3.3 Bugjar Roadmap

Bugjar is a new project - we have lots of things that we'd like to do. If you'd like to contribute, providing a patch for one of these features:

- Port to Python 3
- Add the ability to browse objects in the stack
- Add the ability to add user-defined expressions to watch.

3.4 Release History

3.4.1 0.1.0 - August 31, 2013

Initial public release for DjangoCon US 2013.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search