Interactive Shell for a scientific IDE

Gaël Varoquaux
Enthought Inc.

21, August 2008

A Wx frontend for Ipython

1 IPython1/IPython0

IPython1

- Good software design.
- Testing.
- Network transparencies.
- Model-View separation (but no view implemented yet)
- No fancy ipythongy features.

IPython0

- Tortellini code.
- Strongly linked to readline.
- Many crazy side effects (some logics coded in the display hook ;))
- Magics, shell-expansion, fancy debug mode, nice tracebacks...

Short term solution: embed an IPython0 in the view, and use it to drive an IPython1 in the model.

- Running user code in the event loop blocks it.
  $\Rightarrow$ no UI refresh, no response to keyboard
- Code run in threads needs to be threadsafe.
  + GUIs toolkit are **not** thread-safe.
  $\Rightarrow$ need to teach to the user about thread.

No threads, we execute code in the event loop.

$\Rightarrow$ We need to be clever with event handling, to insert refresh event...

1 Subprocess execution

Problem

- Subprocess execution is at the core of ipython (magics: rm, ls ...).
- These processes expect to have a terminal, for IO:
  - `stdout/stderr`: we want to see the output of ls.
  - `stdin`:

```
In[1]:  rm foobar
rm:  remove file ‘foobar’?
```

Solution

- The good: no threading problem: we can do more clever refresh.
- The bad: we need a blocking call but to react to events.
- The ugly: key events translation to feed to `stdin`.

What to do with this?

2 What to do with this?

Fundamental problems with GUIs

Your GUI is living in the same Python VM as the scripts you run

$\Rightarrow$ Fragile, slow refresh...

So why bother? Go multiprocess? The terminal works well.

What do we gain with GUIs?

- Pretty look and feel
  - ☹ Doesn't make you more productive/richer: no economic or academic incentive
- Embed visualization/new views

2 Proof of principle application

- **Work lead by Evan Patterson**
- 2 user-visible processes:
  - Shell + different visualizations.
  - Editor

  Linked together through sockets.

Now we have to step back and think what is the shared data that a shell exposes: a namespace/context?

Thank you

Please join the fun

enthought-dev mailing-list