

Eclipse, Python, Git, and Vim

Oh My!

PRESENTED BY:

Jesse Keating

Senior Software Engineer, Red Hat, Inc.



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Today's Topics

- What is Eclipse?
- Developing Python in Eclipse
- Interacting with git in Eclipse
- Using Vim with Eclipse





What is Eclipse?

Eclipse is...

- not a cheesy vampire book
- not a Japanese sports car
- not a pack of gum
- an Integrated Development Environment
- (eclipse.org is much bigger than just the IDE)



A quick tour

The screenshot shows the Eclipse IDE with the Pydev plugin. The main editor window displays the code for `pyfedpkg/__init__.py`. The code includes a class `PackageModule` with a method `_findbranch`. The left sidebar shows the project structure, and the right sidebar shows the Outline view. The bottom status bar indicates the current file path.

```
1046 for line in self.__lines:
1047     gitignore_file.write(line)
1048     gitignore_file.close()
1049
1050
1051 # Create a class for package module
1052 class PackageModule:
1053     def _findbranch(self):
1054         """Find the branch we're on.
1055
1056         The goal of this function is to catch if we are on a branch we
1057         can make some assumptions about. If it doesn't match our branch regex
1058         then we raise and ask the user to specify.
1059
1060         """
1061
1062         try:
1063             localbranch = self.repo.active_branch.name
1064         except TypeError, e:
1065             raise FedpkgError('Repo in inconsistent state: %s' % e)
1066         try:
1067             merge = self.repo.git.config('--get', 'branch.%s.merge' % localbranch)
1068         except git.errors.GitCommandError, e:
1069             raise FedpkgError('Unable to find remote branch. Use --dist')
1070         # Trim off the refs/heads so that we're just working with the branch
1071         # name
1072         merge = merge.replace('refs/heads/', '')
1073         # Search for one of our known branches, raise if we can't find one
1074         # to deal with
```

Outline view:

- utils
- os
- sys
- shutil
- re
- pycurl
- subprocess
- subprocess (kitchen)
- hashlib
- koji
- rpm
- logging
- git
- ConfigParser
- stat
- StringIO
- OpenSSL
- fnmatch
- offtrac
- urllib2
- LOOKASIDE

Problems view:

Description	Resource	Path	Location	Type
0 items				

Status bar: fedora-packager/src/pyfedpkg/__init__.py



Features

- Editor
- Multiple Perspectives
- Execution Testing
- Debugging
- Team (source control) Interaction
- Plugins to add lots more!



Editor

- Multiple tabs
- Language colors
- Code Completion
- Whitespace management
- (near) Real Time error checking
- Code folding/collapsing
- Spell checking
- Much more



Pydev Perspective Views

- Navigation and information
 - Project explorer
 - Source file outline
 - Errors
 - Console
 - History
- Can be on their own or stacked
- Can minimize or maximize

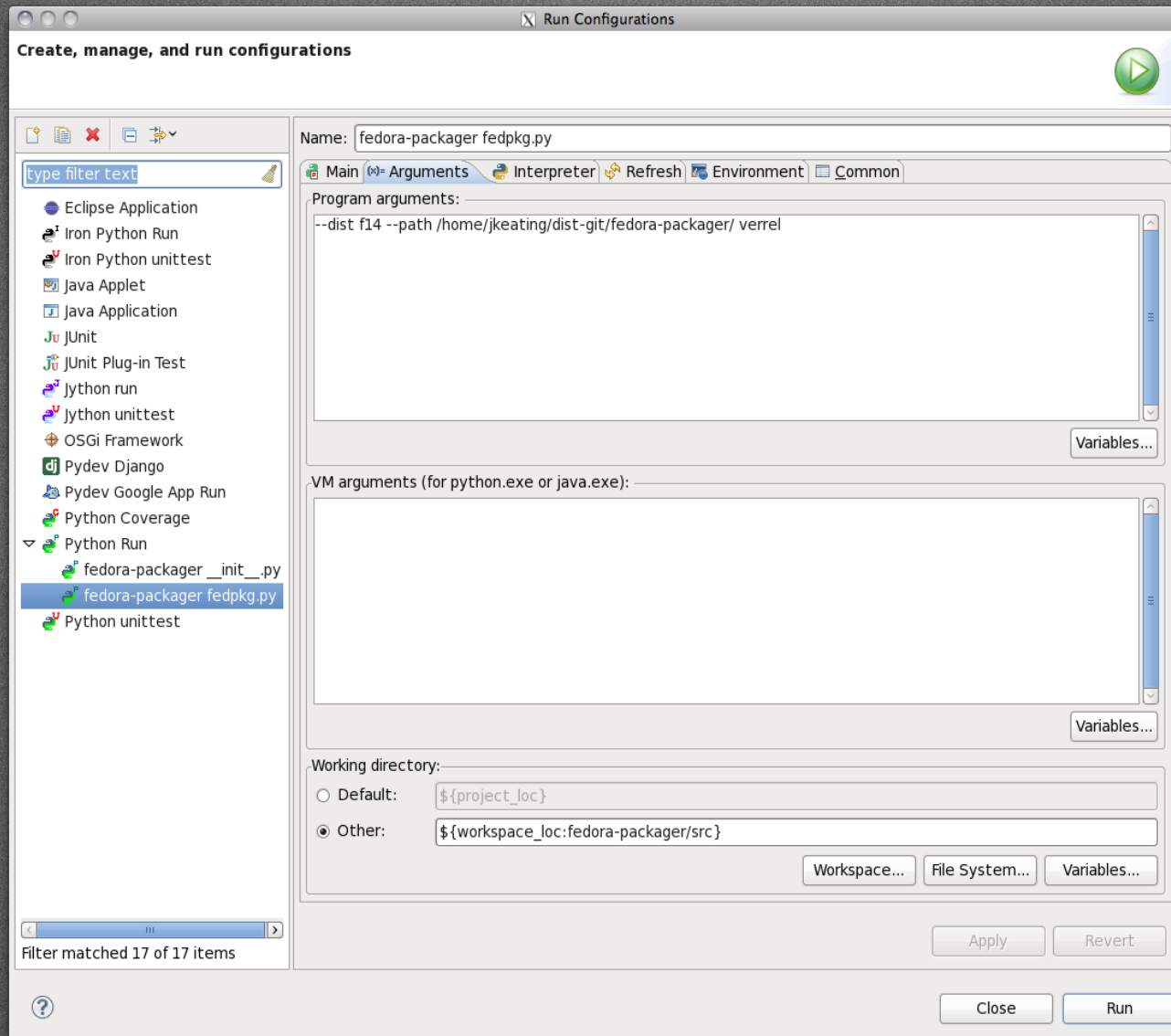


Execution Testing

- Multiple configurations
- Custom app / interpreter arguments
- Console output
- Support for code coverage
- Support for Google App Run
- More with plugins



Execution Testing



Debug Perspective

- Set breakpoints
- Inspect stack data
- Step into, over, return
- Manually pause, resume
- Multiple configurations (linked with run)
- More with plugins



Debug Perspective

The screenshot displays the Eclipse IDE in the Debug Perspective. The top menu bar includes File, Edit, Source, Refactoring, Navigate, Search, Project, Pydev, Run, Window, and Help. The toolbar contains various icons for file operations, running, and debugging.

The **Debug** view on the left shows the execution stack for the `fedora-packager` project. The selected frame is `__init__ [__init__.py:1143]` within the `MainThread - pid16415_seq1`. Below it, the stack includes `verrel [fedpkg.py:868]`, `<module> [fedpkg.py:1494]`, `run [pydevd.py:916]`, and `<module> [pydevd.py:1145]`.

The **Variables** view on the right shows the state of the program. It includes a table with the following data:

Name	Value
Globals	Global variables
dist	str: f14
path	str: /home/jkeating/dist-git/fedora-pac
self	PackageModule: <pyfedpkg.PackageM
lookaside	str: http://pkgs.fedoraproject.org/repo
lookasidehash	str: md5

The **pyfedpkg** view at the bottom shows the source code of the `__init__` method in `fedpkg.py`. The code is as follows:

```
1133 def __init__(self, path=None, dist=None):
1134     # Initiate a PackageModule object in a given path
1135     # Set some global variables used throughout
1136     if not path:
1137         path = os.getcwd()
1138     log.debug('Creating module object from %s' % path)
1139     self.path = path
1140     self.lookaside = LOOKASIDE
1141     self.lookasidehash = LOOKASIDEHASH
1142     self.spec = self.gimmespec()
1143     self.module = utils._name_from_spec(os.path.join(self.path, self.spec))
```

The **Outline** view on the right shows the structure of the `__init__` method, with the following methods listed:

- `__getlocalarch`
- `__init__` (selected)
- `build`
- `clog`
- `compile`
- `getver`
- `getrel`

The **Console** view at the bottom shows the output of the program, including a warning message: `pydev debugger: warning: psyco not available for speedups (the debugger will still work correctly, but a bit slower)` and a status message: `pydev debugger: starting`.

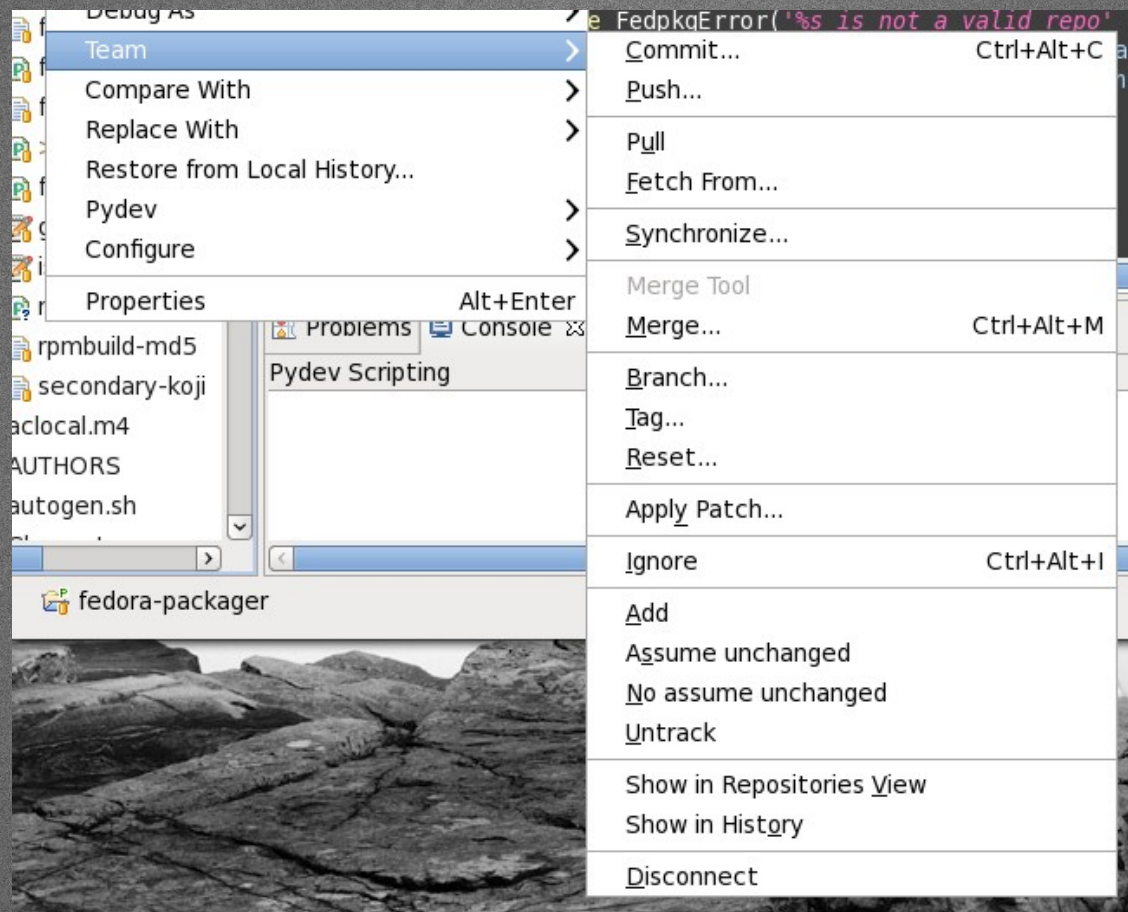


Team Controls

- Interact with source control
 - commit
 - push
 - merge
 - tag
 - More...
- Support for a variety of SCMs (with plugins)



Team Controls



The background features a dark blue horizontal band with a large, stylized Python logo. The logo is composed of two interlocking snakes, one in a light blue-grey and the other in a darker blue. The text "Developing Python" is overlaid on the right side of the band in a white, sans-serif font.

Developing Python

Create a new Project

- Create a pydev project
- Create a new python package within the project
- Create a new module within the package
- Create a the script





Sling some Code

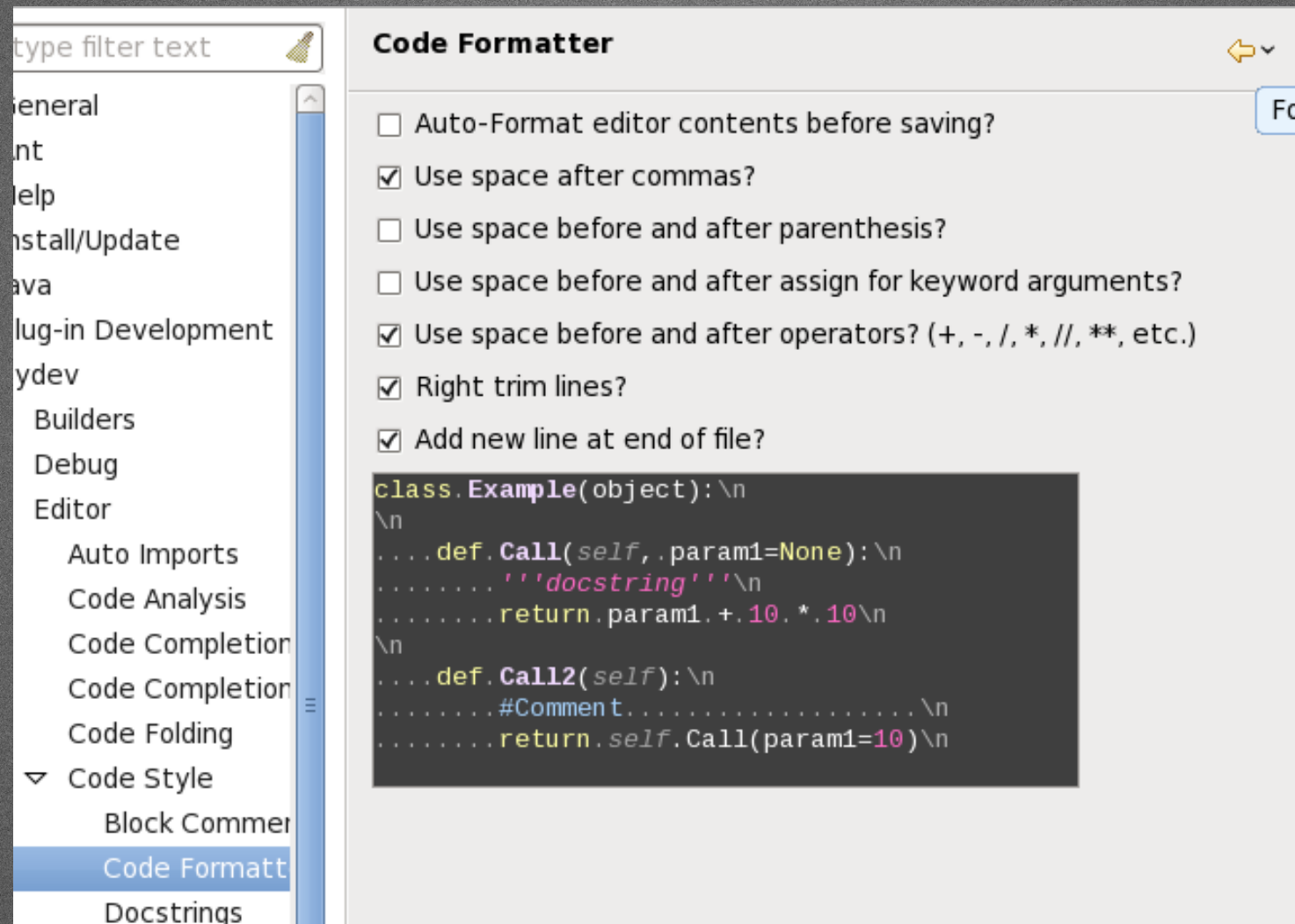
Setup a run

- Make sure script is the active tab
- “Console” view tab will automatically focus when output happens

Setup a debug

- Breakpoints are vitally important
 - Cannot be on a blank line (lost lots of time to this one...)
 - Do not have to save the file after adding a breakpoint
- Debug perspective will automatically launch as soon as a breakpoint is encountered
- Can use console to evaluate statements

Code Formatting



Diffing

- Can diff against local history
- Could diff against previous SCM commits
- Can revert all or parts

Refactoring

- Rename items
- Create new methods from existing code
- Inline / extract a variable

Interactive Console

- Use a fresh python prompt
- Send selected code to the console
- Get execfile sent to console to continue playing with symbols

Code Testing

- Support for code unittests
 - Pydev test runner
 - Nose
 - py.test
- Support for code coverage
- Support for pylint

The background of the slide features a large, stylized Git logo. The logo is composed of two interlocking shapes: a light gray one and a medium blue one, set against a dark blue background. The text "Interacting with git" is overlaid on the right side of the logo.

Interacting with git

git Interaction

- Can create new repo from existing project
- Can create new project from existing repo
- Can link existing project to existing repo

Create a git repo from project

- Share Project
- Choose git
- Create a new repository
- Profit!



Commit files to git

- No files exist in the repo by default, they have to be added/committed



Using git to aid development

- Create branches for topic work
- Diffing / committing
- Creating patches
- Resetting work
- History and repository viewing
- Merging
- Tagging





Using Vim

vrapper

- Wraps the current editor with vim like keybindings, rather than embedding vim itself
- Easy to turn on/off without restarting eclipse
- Still has command/insert modes
- Not all commands or key sequences work though, and a few bugs.



Some quick vrapper features

- Navigation (arrows or k,j,h,l)
- Searching (/ ,? ,n,N)
- Change {word,line,etc} (c{w,\$,G,gg})
- Undo / redo (u,R)
- Repeat (.)
- Yank / paste (y{...},{p,P})
- Visual mode (v)
- Command mode (:)



Some quick vrapper features

- Config file (.vrapperrc)
- Macros (q[a-z])
- Marks (m[a-z])



What's missing?

- Search and replace
- Regex searching
- Vim plugins



Summary

- Eclipse is a useful IDE
- Developing python in Eclipse is awesome
- Using git within Eclipse is handy
- Using vim within Eclipse is a godsend!



Questions?

CONTACT:
jkeating@redhat.com



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.