

# Plugin frameworks

## 3 approaches to designing plugin APIs

Noufal Ibrahim  
`noufal@nibrahim.net.in`

PyCon India 2010, MSRIT, Bangalore

25 September 2010

# About me

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language

Writing a plugin

Example

- Freelance programmer/trainer based in Bangalore
- Organiser of PyCon India 2009, 2010

# About this talk

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :

Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :

Emacs

Embedded

scripting

language

Writing a plugin

Example

- This talk about approaches to writing plugin frameworks using examples
  - 1 TRAC - The project planning/tracking tool from edgewall software.
  - 2 py.test - The test harness from the PyPy project.
  - 3 Emacs - The extensible text editor from the GNU project.
- Informed by my lifelong(?) experiences with Emacs and my work with `py.test` and `trac` over the past two years
- Inspired by a Stack Overflow discussion I had at <http://bit.ly/bznQ2g>

# Plugin frameworks

## Plugin frameworks

noufal

### Introduction

About me

About this talk

### Plugin frameworks

### Case #1 :

Trac

Trac Component architecture

Writing a plugin

Example

Pros/Cons

### Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

### Case #3 :

Emacs

Embedded

scripting

language

Writing a plugin

Example

- Controlled ways to run user code inside your application

# Plugin frameworks

## Plugin frameworks

noufal

Introduction

About me

About this talk

Plugin frameworks

Case #1 :

Trac

Trac Component architecture

Writing a plugin

Example

Pros/Cons

Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :

Emacs

Embedded

scripting

language

Writing a plugin

Example

- Controlled ways to run user code inside your application
- Used to shift the burden of writing your program to your user

# Plugin frameworks

## Plugin frameworks

noufal

### Introduction

About me  
About this talk

### Plugin frameworks

### Case #1 :

Trac

Trac Component architecture

Writing a plugin

Example

Pros/Cons

### Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

### Case #3 :

Emacs

Embedded scripting language

Example

Writing a plugin

Example

- Controlled ways to run user code inside your application
- Used to shift the burden of writing your program to your user
- Used to defer important decisions for later so that they can be made when you know enough

# Plugin frameworks

## Plugin frameworks

noufal

### Introduction

About me  
About this talk

### Plugin frameworks

### Case #1 : Trac

Trac Component architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded scripting language  
Writing a plugin  
Example

- Controlled ways to run user code inside your application
- Used to shift the burden of writing your program to your user
- Used to defer important decisions for later so that they can be made when you know enough
- Used to write *frameworks* that can be used for multiple tasks

# Plugin frameworks

## Plugin frameworks

noufal

### Introduction

About me  
About this talk

### Plugin frameworks

### Case #1 : Trac

Trac Component architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded scripting language  
Writing a plugin  
Example

- Controlled ways to run user code inside your application
- Used to shift the burden of writing your program to your user
- Used to defer important decisions for later so that they can be made when you know enough
- Used to write *frameworks* that can be used for multiple tasks
- To reduce the size of the application



# Plugin frameworks

## Plugin frameworks

noufal

### Introduction

About me  
About this talk

### Plugin frameworks

### Case #1 : Trac

Trac Component architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded scripting language  
Writing a plugin  
Example

- Controlled ways to run user code inside your application
- Used to shift the burden of writing your program to your user
- Used to defer important decisions for later so that they can be made when you know enough
- Used to write *frameworks* that can be used for multiple tasks
- To reduce the size of the application
- Generally fun to do

# Trac : Component architecture

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded

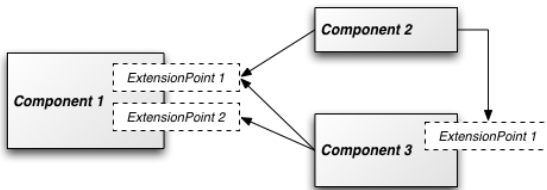
scripting

language

Writing a plugin

Example

- Designed using a *Component architecture*.
- Each component exposes “extension points” that can be plugged into by other components.



- Components are singletons
- Simplified version of the Zope Component Architecture
- Details at <http://trac.edgewall.org/wiki/TracDev/ComponentArchitecture>
- Very popular architecture. Used in other applications like Eclipse.

# Trac : Writing a plugin

Plugin  
frameworks

noufal

Introduction

About me  
About this talk  
Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture  
Writing a plugin  
Example  
Pros/Cons

Case #2 :  
py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language  
Writing a plugin  
Example

- Write a new component (subclass component).
- Interfaces are created by subclassing `Interface` and then wrapping them in an `ExtensionPoint`.
- The `Interface` will have methods defined in them that can be overridden by user created `Components` that *implement* the interface.

# Trac : Example plugin

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language

Writing a plugin

Example

```
1 # Following is the code in the application
2 # The interface and it's specification
3 class IToDoObserver(Interface):
4     def todo_added(name, description):
5         "Called when a to-do item is added."
6
7 class TodoList(Component):
8     # Declaration of an extension point
9     observers = ExtensionPoint(ITodoObserver)
10
11     def __init__(self):
12         self.todos = {}
13
14     def add(self, name, description):
15         assert not name in self.todos, 'To-do already in list'
16         self.todos[name] = description
17         for observer in self.observers: # Using the extension point
18             observer.todo_added(name, description)
19
20
21 # A plugin that prints out details when something is added
22 class TodoPrinter(Component):
23     # Stating which interface is implemented
24     implements(ITodoObserver)
25
26     def todo_added(self, name, description):
27         prinyt 'TODO:', name
28         print '      ', description
```

# Trac : Pros/Cons

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded

scripting

language

Writing a plugin

Example

- Pros
  - Very “Object Oriented”
- Cons
  - High entry barrier (need to learn “plugin API”)
  - Lots of boilerplate code

# py.test : Metaprogramming

Plugin  
frameworks

noufal

Introduction

About me  
About this talk  
Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture  
Writing a plugin  
Example  
Pros/Cons

Case #2 :  
py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language  
Writing a plugin  
Example

- Takes the “best api is no api” to the extreme.
- Relies on naming conventions to insert code into the framework.
- Searches multiple sources for “plugin” files to load (setuptools entry points, `-p` option, `conftest.py` etc.)
- Inside the plugin file, hooks can be defined using the `pytest_` prefix (e.g. `pytest_addoption` to add a command line option).

# py.test : Writing a plugin

## Plugin frameworks

noufal

### Introduction

About me  
About this talk  
Plugin frameworks

### Case #1 : Trac

Trac Component architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded scripting language  
Writing a plugin  
Example

- Implement the hooks that are necessary for the plugin to work (e.g. add a few command line options, initialise some stuff before the tests start, print out some extra information when tests are done).
- Inform `py.test` that this plugin file needs to be picked up at startup time.
- Similar in some sense to the first approach but using modules instead of singleton classes.

# py.test : Example

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :

Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :

Emacs

Embedded

scripting

language

Writing a plugin

Example

```
1 class BugZillaInteg(object):
2     def pytest_report_teststatus(self, report):
3         "Hook called when test status is created"
4         if report.failed:
5             if self.analysed(report.item.function.__doc__):
6                 return "analysed", "A", "ANALYSED"
7
8     def __init__(self, config, bugzilla):
9         self.config = config
10        self.bugzilla = bugzilla
11
12 def pytest_configure(config):
13     # Logging in into bugzilla
14     bzilla = pyzilla.BugZilla(config.getvalue("bugzilla_url"),
15                             config.getvalue("bugzilla_verbose"))
16     bzilla.login(username = config.getvalue("bugzilla_username"),
17                 password = config.getvalue("bugzilla_pw"))
18     # Register the plugin
19     config.pluginmanager.register(BugZillaInteg(config, bzilla), "bugzilla")
```



# py.test : Pros/Cons

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :

Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :

py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :

Emacs

Embedded

scripting

language

Writing a plugin

Example

## ■ Pros

- Very easy to write a plugin
- Very easy to visualise what's going on.
- High level of flexibility.

## ■ Cons

- The internals of py.test can get complex.

# Emacs : Embedded scripting language

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language

Writing a plugin

Example

- Build your application as a “library” for a scripting language.
- The primitives will be hardcoded and the rest built on top in the scripting language.
- No real “API”. The boundaries between plugin and core get blurred.
- Emacs is a successful example. It’s an “editor” that works as an email client, IRC client, PIM, web browser, twitter client, video editor, mp3 player and other things.

# Emacs : Writing a plugin

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded  
scripting  
language

Writing a plugin

Example

- Write a script just like one would do for any language
- Evaluate the script (or drop it into the init file)

# Emacs : Example

Plugin  
frameworks

noufal

Introduction

About me

About this talk

Plugin  
frameworks

Case #1 :  
Trac

Trac Component  
architecture

Writing a plugin

Example

Pros/Cons

Case #2 :  
py.test

Metaprogramming

Writing a plugin

Example

Pros/Cons

Case #3 :  
Emacs

Embedded

scripting

language

Writing a plugin

Example

```
1 (defun list-issues ()
2   "Shows all TBDs in current tree in all files of the same type as current"
3   (interactive)
4   (let ((dir (file-name-directory (buffer-file-name))))
5     (ext (file-name-extension (buffer-file-name))))
6     (rgrep "TBD" (concat ".*" ext) dir))
7     (other-window 1))
8
9
10 (global-set-key (kbd "<f11>") 'list-issues)
```

# Thoughts

## Plugin frameworks

noufal

### Introduction

About me  
About this talk  
Plugin  
frameworks

### Case #1 : Trac

Trac Component  
architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded  
scripting  
language  
Writing a plugin  
Example

General feelings on how these things should be done.

# Thoughts

## Plugin frameworks

noufal

### Introduction

About me  
About this talk  
Plugin frameworks

### Case #1 : Trac

Trac Component architecture  
Writing a plugin  
Example  
Pros/Cons

### Case #2 : py.test

Metaprogramming  
Writing a plugin  
Example  
Pros/Cons

### Case #3 : Emacs

Embedded scripting language  
Writing a plugin  
Example

General feelings on how these things should be done.

- Try to keep the non extensible core as small as possible.
- Implement the smallest subset of what you need and make the rest plugins.
- Move all non essential parts to plugin land.
- Reduce boilerplate as much as possible. Make it *easy* (and not just possible) for users to extend your software. Rely on conventions rather than writing lots of code.
- If you're going the Emacs route, embed a *real* scripting language and not just a toy or ad-hoc one.