# Python and Google App Engine

Dan Sanderson

June 14, 2012

Google Developers


python
powered

# Google App Engine

- Platform for building scalable web applications
- Built on Google infrastructure
- Pay for what you use
  - Apps, instance hours, storage, bandwidth, service calls
  - Free to start!
- Launched with Python 2.5 exclusively in 2008; then Java, Go, Python 2.7

# Google App Engine

- Easy development
- Easy deployment
- No servers to manage, no OS to update; App Engine does this for you
- Based on standard technologies: Python 2.7, WSGI

# Agenda

- Development environment
- Tools and IDEs
- Unit testing
- Remote access
- (more more more!)

# *Demo*

# Development Environment

*Make your computer look like AE.*

# Development Environment

*Make your computer look like AE.*

# Development Environment

- Python 2.7
- App Engine SDK (development server)
- Optional packages
- virtualenv and pip

# Python 2.7

- *python.org*
- 2.7.x, not 3.x
- Interpreter, standard library
- (App Engine sandbox)

# Python 2.7

- Windows
- Python 2.7.3 Windows (X86-64) Installer
- C:\Python27\
- python -V

# Python 2.7

- Mac OS X
  - 10.7 Lion: you already have it
    - python -V
  - Python 2.7.3 Mac OS X installers
  - Homebrew (*mxcl.github.com/homebrew*)

# Python 2.7

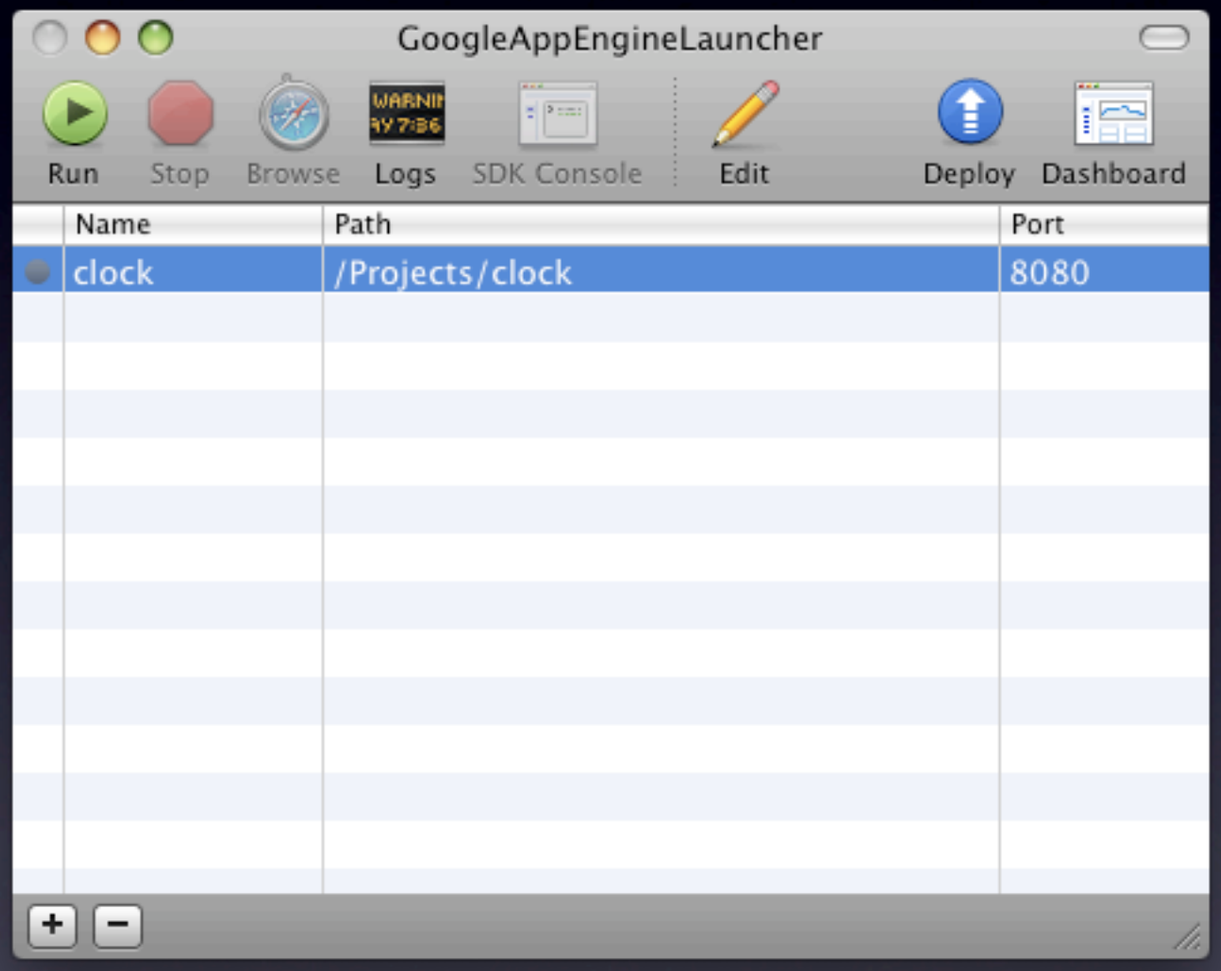
- Linux
  - You probably have it.
    - python -V
    - Ubuntu Precise: python 2.7.3
  - sudo apt-get install python

# App Engine SDK

*developers.google.com/appengine*

- Windows:
  GoogleAppEngine-1.6.6.msi
- Mac OS X:
  GoogleAppEngineLauncher-1.6.6.dmg
- Linux:
  google_appengine_1.6.6.zip

# App Engine SDK



# App Engine SDK

- dev_appserver.py *appdir*
- appcfg.py *command appdir*
- API libraries
- SDK includes all dependencies
- More tools! (webapp)
- *Not installed in site-packages*

# Optional Packages

*app.yaml:*

```
application: myapp
version: 1
runtime: python27
api_version: 1
threadsafe: true

# ...

libraries:
- name: django
  version: 1.3
- name: jinja2
  version: 2.6
- name: markupsafe
  version: 0.15
```

# Optional Packages

- Django
- Jinja2, markupsafe
- lxml
- numpy
- PIL
- pycrypto
- webapp2*
- webob
- yaml

*Special access to fast implementations on App Engine*

*Versioned access*

*Not in the SDK; install them locally, as needed*

# Optional Packages

- AE live environment:
  AE modules + sandbox + AE's Python +
  requested libraries
- Development server environment:
  AE modules + simulated sandbox +
  your Python environment
- setup.py, easy_install, pip add to
  site-packages; one big pile
- Do I have libs installed that AE doesn't have?
  What if different apps need different versions?

# virtualenv

- Create virtual Python environments, each with its own set of packages
- Imports use the active environment
- Versions isolated, dependencies controlled

# virtualenv

- Install virtualenv (once):
  `sudo easy_install virtualenv`
- Create an environment for your App Engine project:

  `virtualenv myapp_env`

- Activate the environment:
  `source ./myapp_env/bin/activate`

# pip

- Python package installation manager
- Like easy_install but better
- Python Package Index (PyPI): *pypi.python.org*
- Works well with virtualenv (no sudo!)

# pip

- Install pip (once):
  `sudo easy_install pip`
- To install a package from PyPI:
  `pip install jinja2`
- To install a specific version:
  `pip install jinja2==2.6`

# pip

- Requirements file:

```
pip install -r requirements.txt
```

- File is just a list of package names, with optional versions

*app.yaml:*

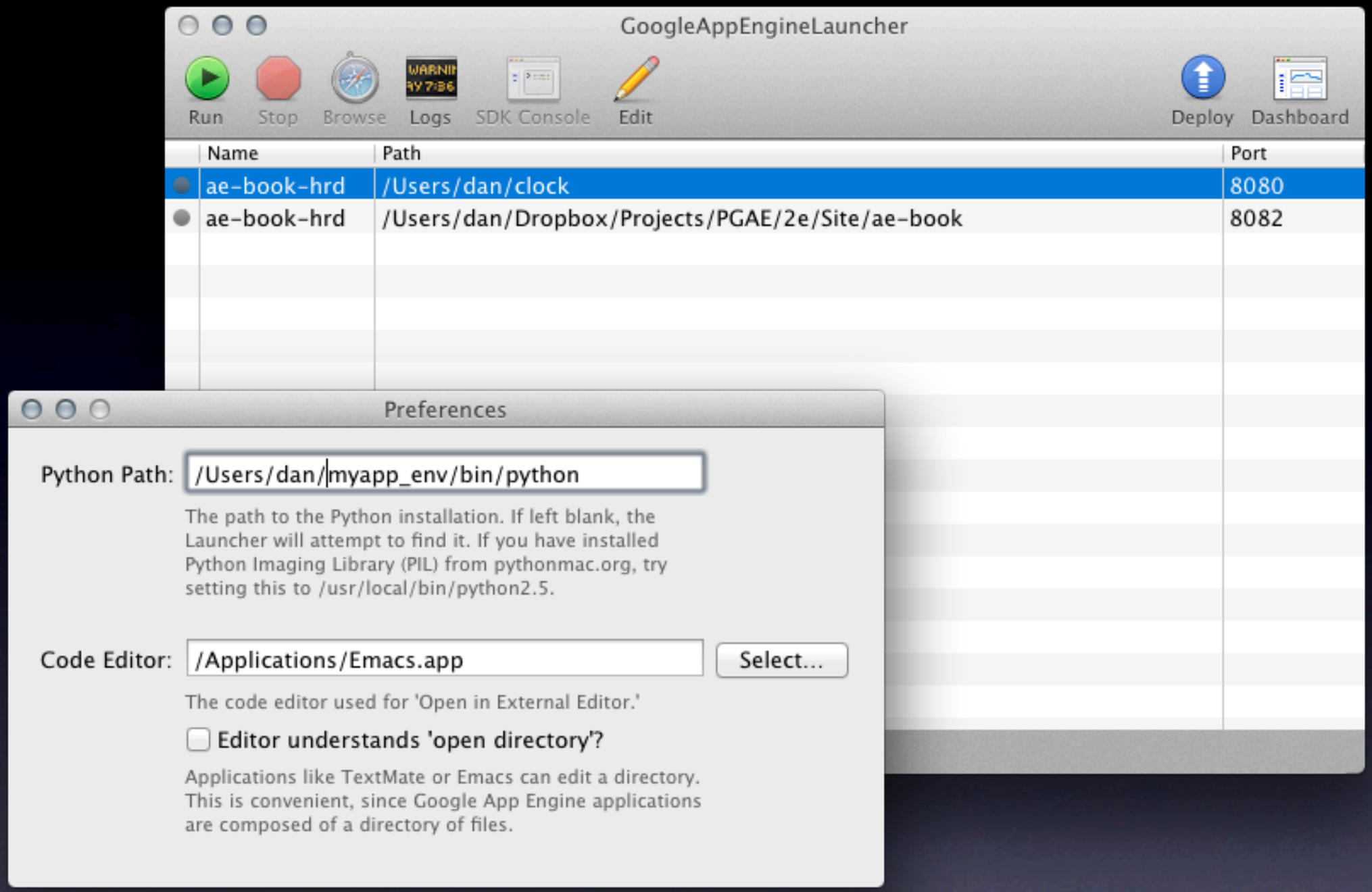
```
libraries:
- name: django
  version: 1.3
- name: jinja2
  version: 2.6
- name: markupsafe
  version: 0.15
```

*requirements.txt:*

```
django==1.3
jinja2==2.6
markupsafe==0.15
```

```
virtualenv myapp_env
source ./myapp_env/bin/activate
pip install -r requirements.txt
```

```
django==1.3
jinja2==2.6
lxml==2.3
markupsafe==0.15
numpy==1.6.1
pycrypto==2.3
PyYAML==3.10
webapp2==2.5.1
webob==1.1.1

PIL==1.1.7
```


GoogleAppEngineLauncher
Run
Stop
Browse
Logs
SDK Console
Edit
Deploy
Dashboard
Name
Path
Port
ae-book-hrd
/Users/dan/clock
8080
ae-book-hrd
/Users/dan/Dropbox/Projects/PGAE/2e/Site/ae-book
8082
Preferences
Python Path:
/Users/dan/myapp_env/bin/python
The path to the Python installation. If left blank, the Launcher will attempt to find it. If you have installed Python Imaging Library (PIL) from pythonmac.org, try setting this to /usr/local/bin/python2.5.
Code Editor:
/Applications/Emacs.app
Select...
The code editor used for 'Open in External Editor.'
Editor understands 'open directory'?
Applications like TextMate or Emacs can edit a directory. This is convenient, since Google App Engine applications are composed of a directory of files.

# Tools and IDEs

# Tools and IDEs

- dev_appserver.py, dev server console
- appcfg.py
- Google App Engine Launcher
- JetBrains PyCharm
- Aptana PyDev

# dev_appserver.py

- dev_appserver.py *appdir*
- Runs a local web server
- Watches your files for changes, re-imports modules
- Simulates runtime environment and services locally
  - Datastore, Blobstore, Memcache; MySQL
  - Mail, XMPP
  - Task queues
- Automatic datastore index generation

# Development Server Console

- *localhost:8080/_ah/admin*
- Web UI to inspect and manipulate the development environment
  - datastore, memcache; task queues; incoming mail, XMPP
  - Python console: run Python code from the browser

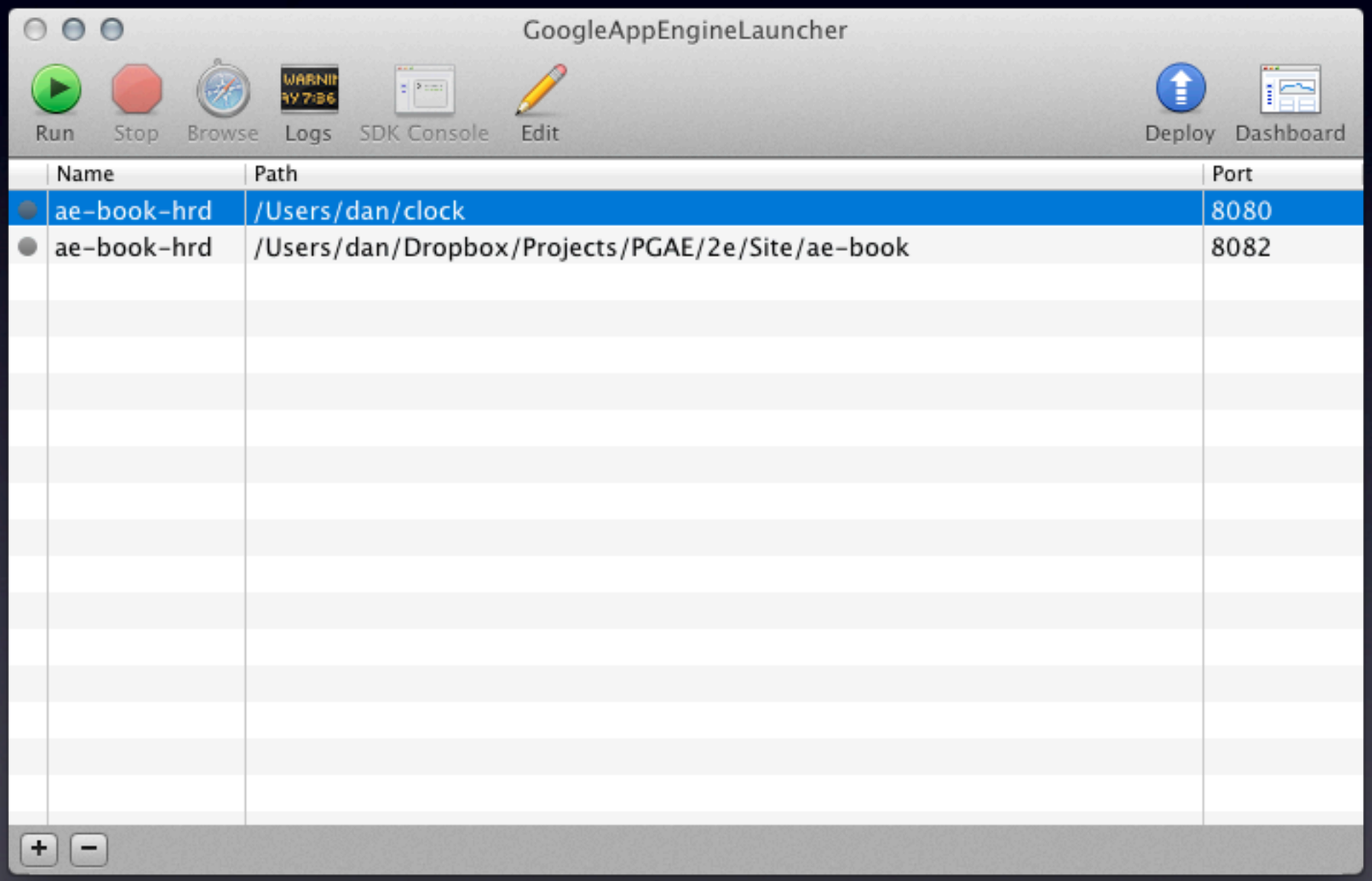
# appcfg.py

- appcfg.py *command appdir*
- Deploy your app (update)
- Manage datastore indexes (update_indexes, vacuum_indexes)
- Download app code (download_app)
- Import/export datastore data (upload_data, download_data, create_bulkloader_config)
- Download logs (request_logs)

# appcfg.py

- Use --oauth2
- Use two-factor authentication
- For tools that don't support OAuth deployment, use app-specific passwords: *google.com/settings*

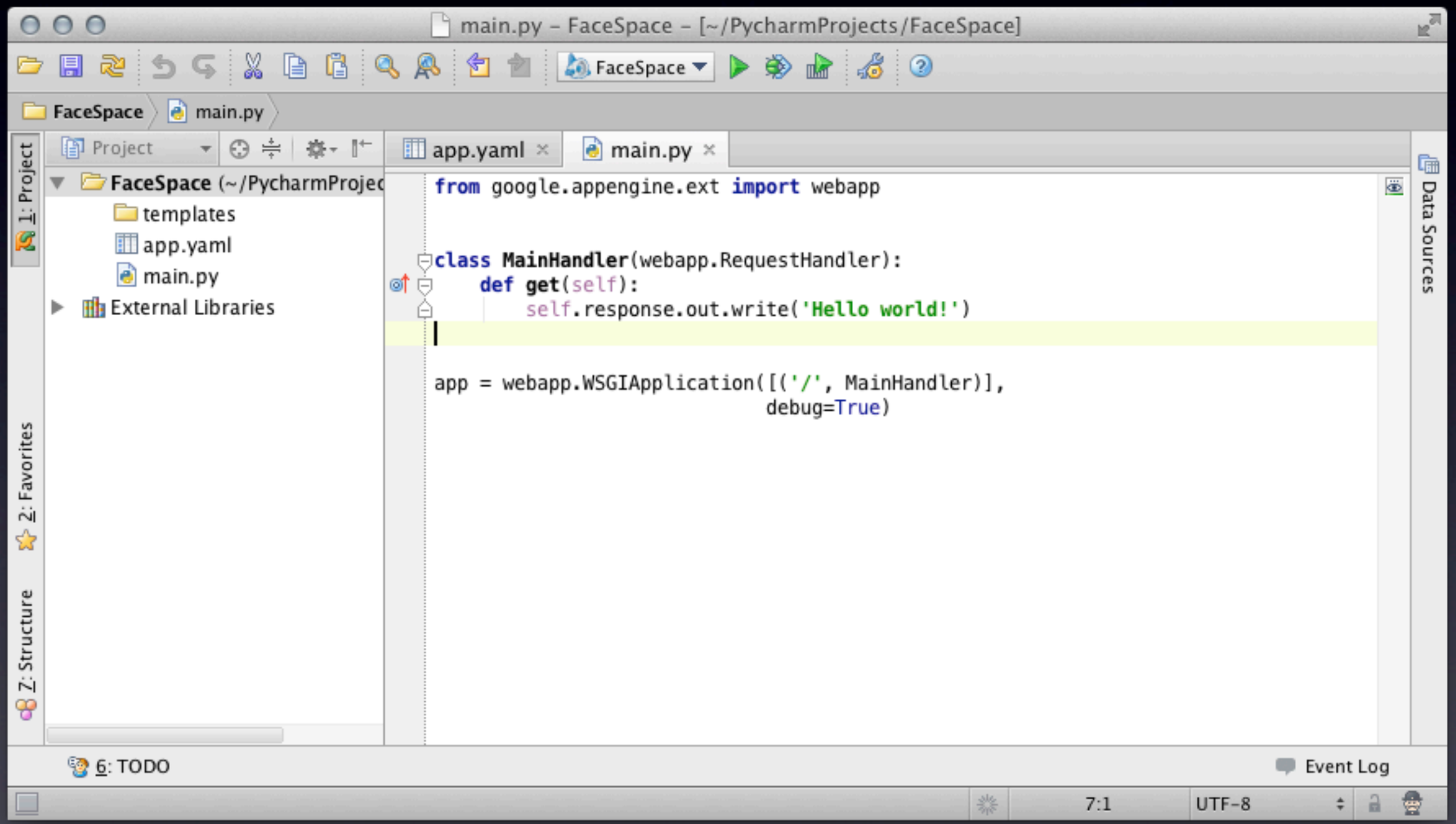
# Google App Engine Launcher



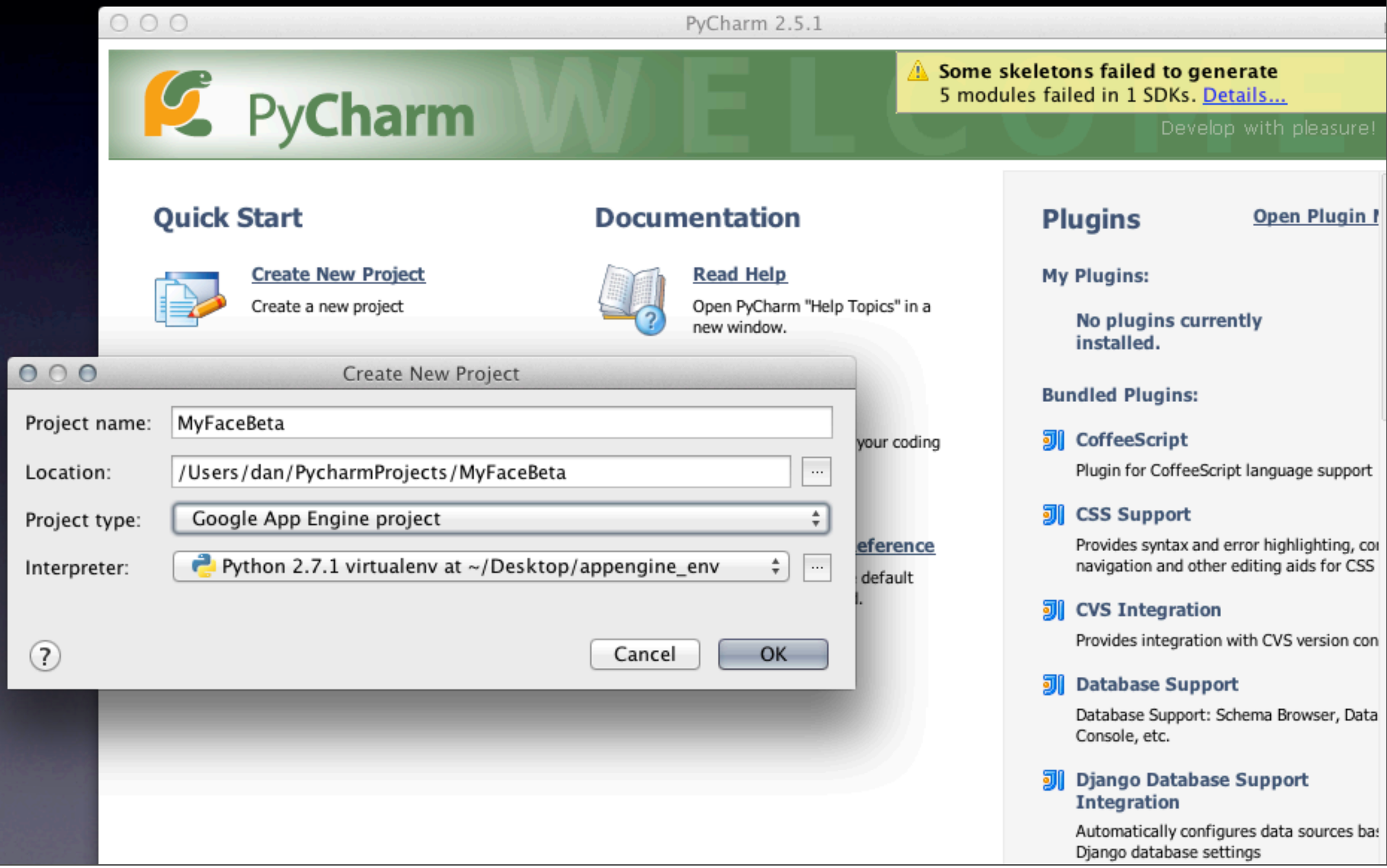
# JetBrains PyCharm

- *jetbrains.com/pycharm*
- $99 personal, $199 commercial;
  free Open Source Project license;
  30-day free trial
- code completion; interactive debugger; unit testing with coverage reports; doc strings and doc access; auto imports; refactoring
- AE project template; AE lib-aware; run dev server in debugger; deploy to App Engine, fetch logs
- Special Django, HTML, CSS, JavaScript support

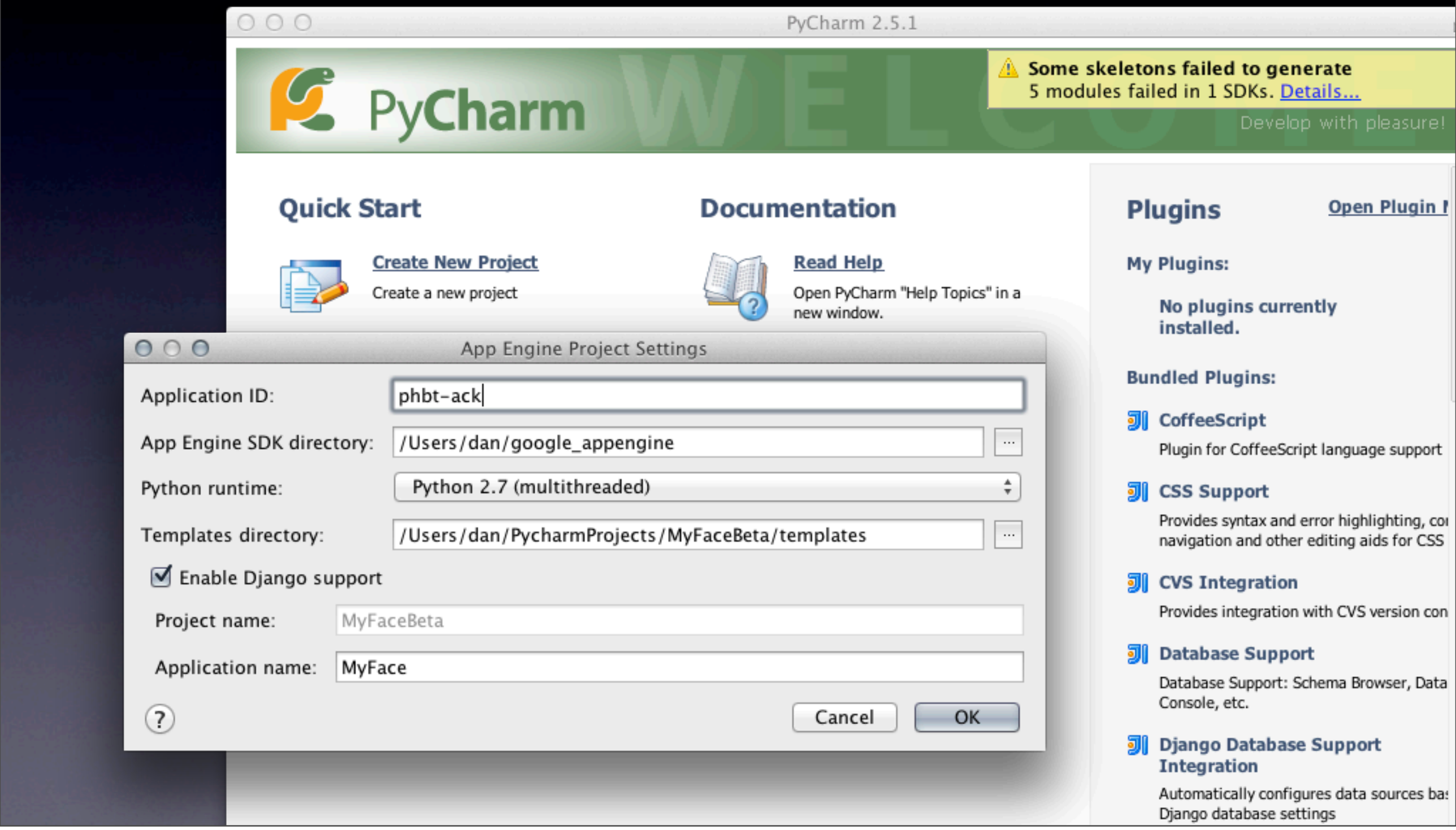
# JetBrains PyCharm



# JetBrains PyCharm



# JetBrains PyCharm

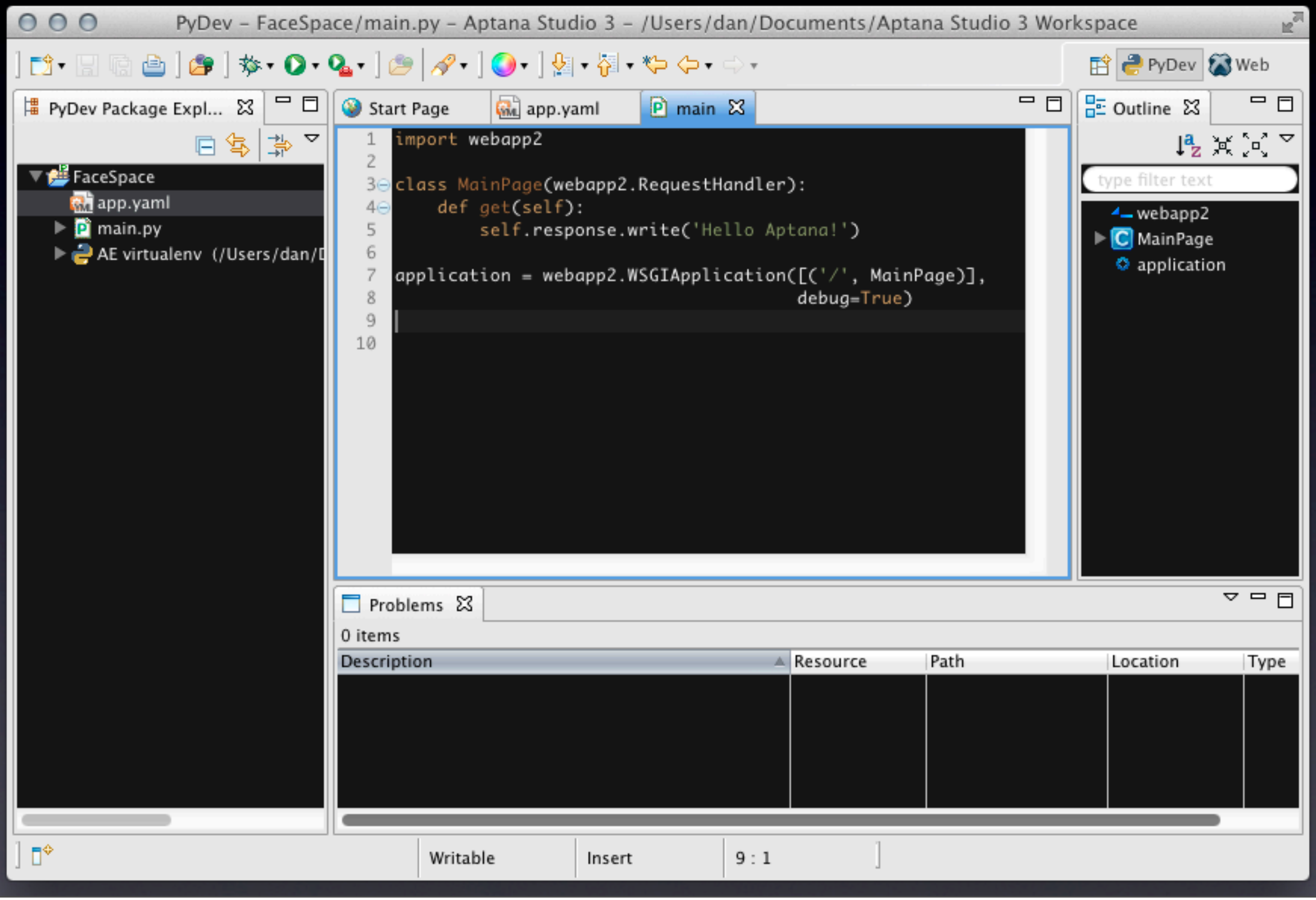


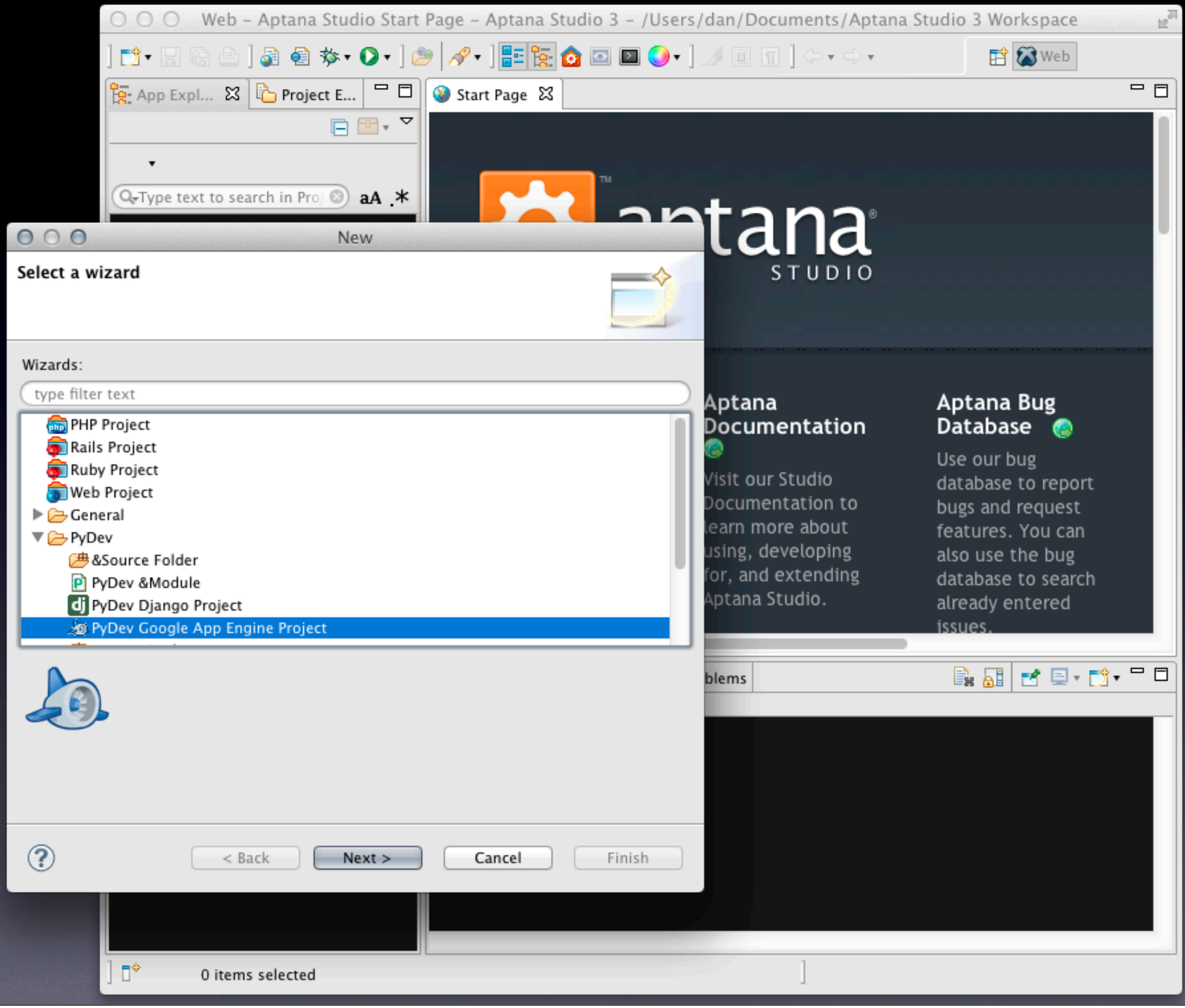
# Aptana PyDev

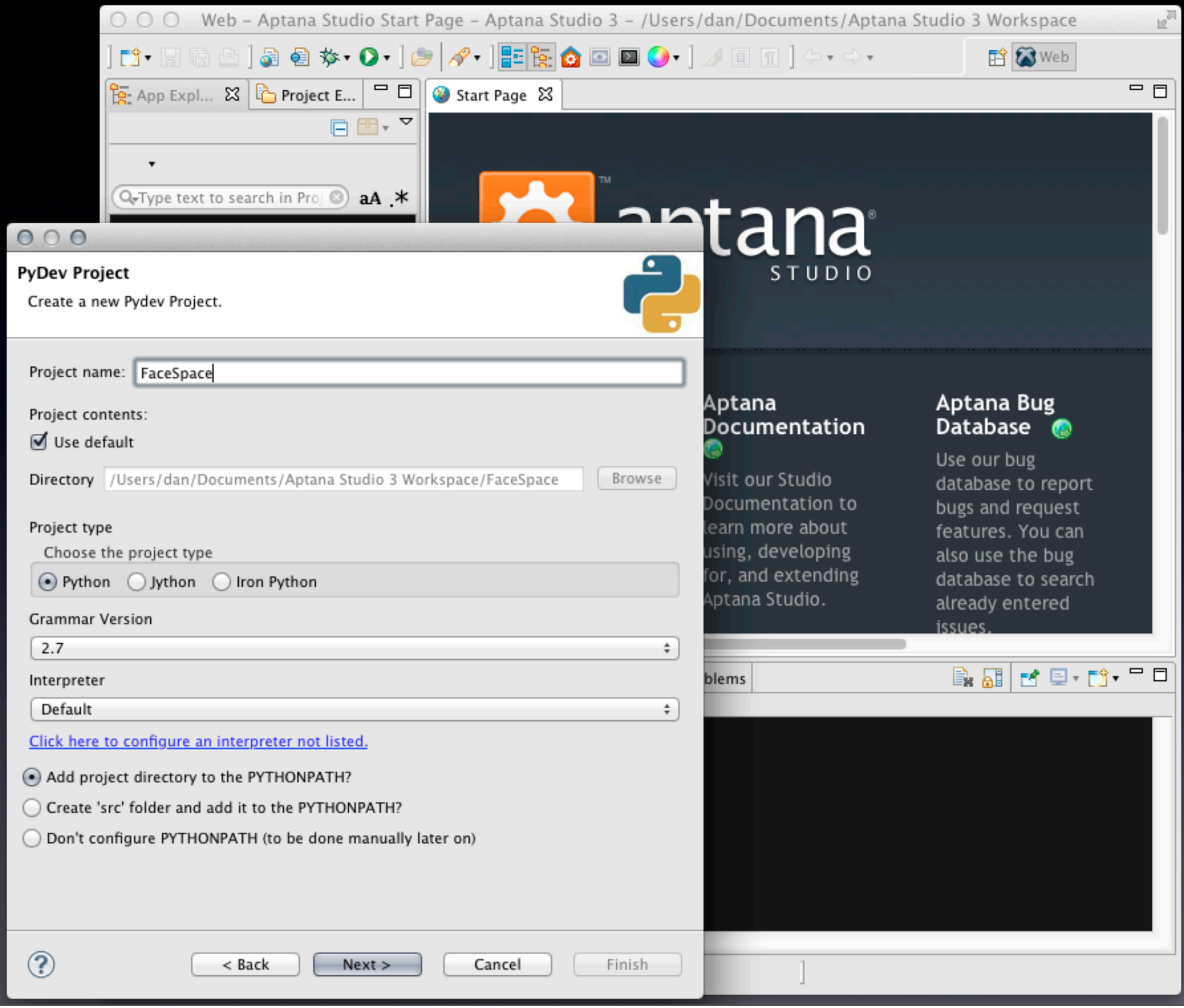
- Aptana Studio 3 with PyDev: *aptana.com*
- Eclipse + PyDev plugin: *pydev.org*
- *Free!*

- code completion; interactive debugger; doc strings and doc access; unit testing; auto imports; refactoring
- AE project set-up wizard (no starter files though); AE lib-aware; run dev server in debugger
- Special JavaScript, HTML, CSS support

# Aptana PyDev




Web – Aptana Studio Start Page – Aptana Studio 3 – /Users/dan/Documents/Aptana Studio 3 Workspace
Web
App Expl...
Project E...
Start Page
Type text to search in Pro
aA .*
aptana
STUDIO
New
Select a wizard
Wizards:
type filter text
PHP Project
Rails Project
Ruby Project
Web Project
General
PyDev
&Source Folder
PyDev &Module
PyDev Django Project
PyDev Google App Engine Project
< Back
Next >
Cancel
Finish
Aptana Documentation
Visit our Studio Documentation to learn more about using, developing for, and extending Aptana Studio.
Aptana Bug Database
Use our bug database to report bugs and request features. You can also use the bug database to search already entered issues.
blems
0 items selected


Web – Aptana Studio Start Page – Aptana Studio 3 – /Users/dan/Documents/Aptana Studio 3 Workspace
Web
App Expl...
Project E...
Start Page
Type text to search in Pro
aA .*
aptana
STUDIO
PyDev Project
Create a new Pydev Project.
Project name: FaceSpace
Project contents:
Use default
Directory /Users/dan/Documents/Aptana Studio 3 Workspace/FaceSpace
Browse
Project type
Choose the project type
Python Jython Iron Python
Grammar Version
2.7
Interpreter
Default
Click here to configure an interpreter not listed.
Add project directory to the PYTHONPATH?
Create 'src' folder and add it to the PYTHONPATH?
Don't configure PYTHONPATH (to be done manually later on)
< Back
Next >
Cancel
Finish
Aptana Documentation
Visit our Studio Documentation to learn more about using, developing for, and extending Aptana Studio.
Aptana Bug Database
Use our bug database to report bugs and request features. You can also use the bug database to search already entered issues.
blems

Preferences

type filter text

- General
- Aptana Studio
- Help
- Install/Update
- PyDev
  - Builders
  - Debug
  - Django Templates Editor
  - Editor
  - Interactive Console
  - Interpreter - Iron Python
  - Interpreter - Jython
  - Interpreter - Python
  - Logging
  - PyLint
  - PyUnit
  - Scripting PyDev
  - Task Tags
- Run/Debug
- Team

**Python Interpreters**

Python interpreters (e.g.: python.exe)

| Name | Location |
|---|---|

New...

Auto Config

Remove

Select interpreter

Enter the name and executable of your interpreter

Interpreter Name: AE virtualenv

Interpreter Executable: /Users/dan/Desktop/appengine_env/bin/python

Browse...

Cancel OK

Restore Defaults Apply

Cancel OK

PyDev Pro

Project

Project na

Project co

Use de

Directory

Project typ

Choose

Pytho

Grammar

2.7

Interprete

Please con

Add pro

Create 'src' folder and add it to the PYTHONPATH?

Don't configure PYTHONPATH (to be done manually later on)

Preferences

type filter text

- General
- Aptana Studio
- Help
- Install/Update
- PyDev
  - Builders
  - Debug
  - Django Templates Editor
  - Editor
  - Interactive Console
  - Interpreter - Iron Py
  - Interpreter - Jython
  - Interpreter - Python
  - Logging
  - PyLint
  - PyUnit
  - Scripting PyDev
  - Task Tags
- Run/Debug
- Team

**Python Interpreters**

Python interpreters (e.g.: python.exe)

| Name | Location |
|---|---|

New...

Selection needed

Select the folders to be added to the SYSTEM pythonpath!

IMPORTANT: The folders for your PROJECTS should NOT be added here, but in your project configuration.

Check:http://pydev.org/manual_101_interpreter.html for more details.

- [ ] /Applications/Aptana Studio 3/plugins/org.python.pydev_2.6.0.2012052102/pysrc
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.eg
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/site-packages/pip-1.1-py2.7.egg
- [x] /Users/dan/Desktop/appengine_env/lib/python27.zip
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/plat-darwin
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/plat-mac
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/plat-mac/lib-scriptpackages
- [x] /Users/dan/Desktop/appengine_env/Extras/lib/python
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/lib-tk
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/lib-old
- [x] /Users/dan/Desktop/appengine_env/lib/python2.7/lib-dynload
- [x] /System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7
- [ ] /System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-darwin
- [ ] /System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-tk

Select All not in Workspace | Select All | Deselect All

Cancel | OK

Cancel | OK

PyDev Pro

Project

Project na

Project co

Use de

Directory

Project typ

Choose

Pytho

Grammar

2.7

Interprete

Please con

Add pro

Create 'src' folder and add it to the PYTHONPATH?

Don't configure PYTHONPATH (to be done manually later on)

# Unit Testing

# Unit Testing

- It's just Python, right?
- Activate your virtualenv
- Make sure google_appengine and your app's root dir are in PYTHONPATH
- (Use a test runner script, or a test tool: nose-gae, gaeunit)

# Unit Testing

```
% source ./myapp_env/bin/activate
(myapp_env)% export PYTHONPATH=$PYTHONPATH:/
Users/dan/google_appengine
(myapp_env)% python
>>> from google.appengine.ext import db
>>> import models
>>> m = models.Entry()
>>> m.title = 'Test'
>>> m.put()    # (in theory)
>>>
```

# Unit Testing

```
>>> from google.appengine.api import mail
>>> mail.send_mail('a@b.com', 'b@c.com',
'subj', 'body')
...
AssertionError: No api proxy found for service
"mail"
>>>
```

# testbed

- Service API stubs for a clean test environment
- Provided by the App Engine SDK
  - google.appengine.ext.testbed
  - Mail and XMPP: records outgoing messages
  - Datastore: can fuzz test eventual consistency
  - Users: can set signed-in status, address, is-admin status

# testbed

```
>>> from google.appengine.ext import testbed
>>> tb = testbed.Testbed()
>>> tb.activate()
>>> tb.init_datastore_v3_stub()
```

```
>>> from google.appengine.ext import db
>>> import models
>>> m = models.Entry()
>>> m.title = ‘Test’
>>> m.put()   # (success!)
datastore_types.Key.from_path(u’Entry’, 1L,
_app=u’testbed-test’)
```

# testbed

```
>>> tb.activate()
>>> tb.init_mail_stub()

>>> from google.appengine.api import mail
>>> mail.send_mail('a@b.com', 'b@c.com',
'subj', 'body')    # (success!)
>>> tb.get_stub(testbed.MAIL_SERVICE_NAME)
.get_sent_messages(to='b@c.com')
[<google.appengine.api.mail.EmailMessage
object at 0x10b5115d0>]
>>>
```

# testbed

```
import unittest
from google.appengine.ext import testbed
import models

class EntryTestBase(unittest.TestCase):
  def setUp(self):
    self.tb = testbed.Testbed()
    self.tb.activate()
    self.tb.init_datastore_v3_stub()

  def tearDown(self):
    self.tb.deactivate()
```

# testbed

```
class EntryTestBase(unittest.TestCase):
  # ...

  def testEntry(self):
    m = models.Entry()
    m.title = 'Test'
    m.put()
    self.assertIsNotNone(m.create_date)


if __name__ == '__main__':
  unittest.main()
```

# testbed

```
tb.init_all_stubs()

tb.init_blobstore_stub()
tb.init_datastore_v3_stub()
tb.init_memcache_stub()
tb.init_images_stub()
tb.init_mail_stub()
tb.init_taskqueue_stub()
tb.init_urlfetch_stub()
tb.init_user_stub()
tb.init_xmpp_stub()
```

*developers.google.com/appengine/docs/python/tools/localunittesting*

# Unit Testing

```
>>> import webob
>>> import blog
>>> h = blog.FrontPageHandler()
>>> h.request = webob.Request.blank('/')
>>> h.get()    # (assuming testbed is set up as needed)
>>> h.response.status
'200 OK'
>>>
```

# WebTest

- Test helper for WSGI applications, wraps any WSGI app instance in a webob-based interface
- Third-party tool, not included
- To install WebTest (in your virtualenv):
  `pip install WebTest`

# WebTest

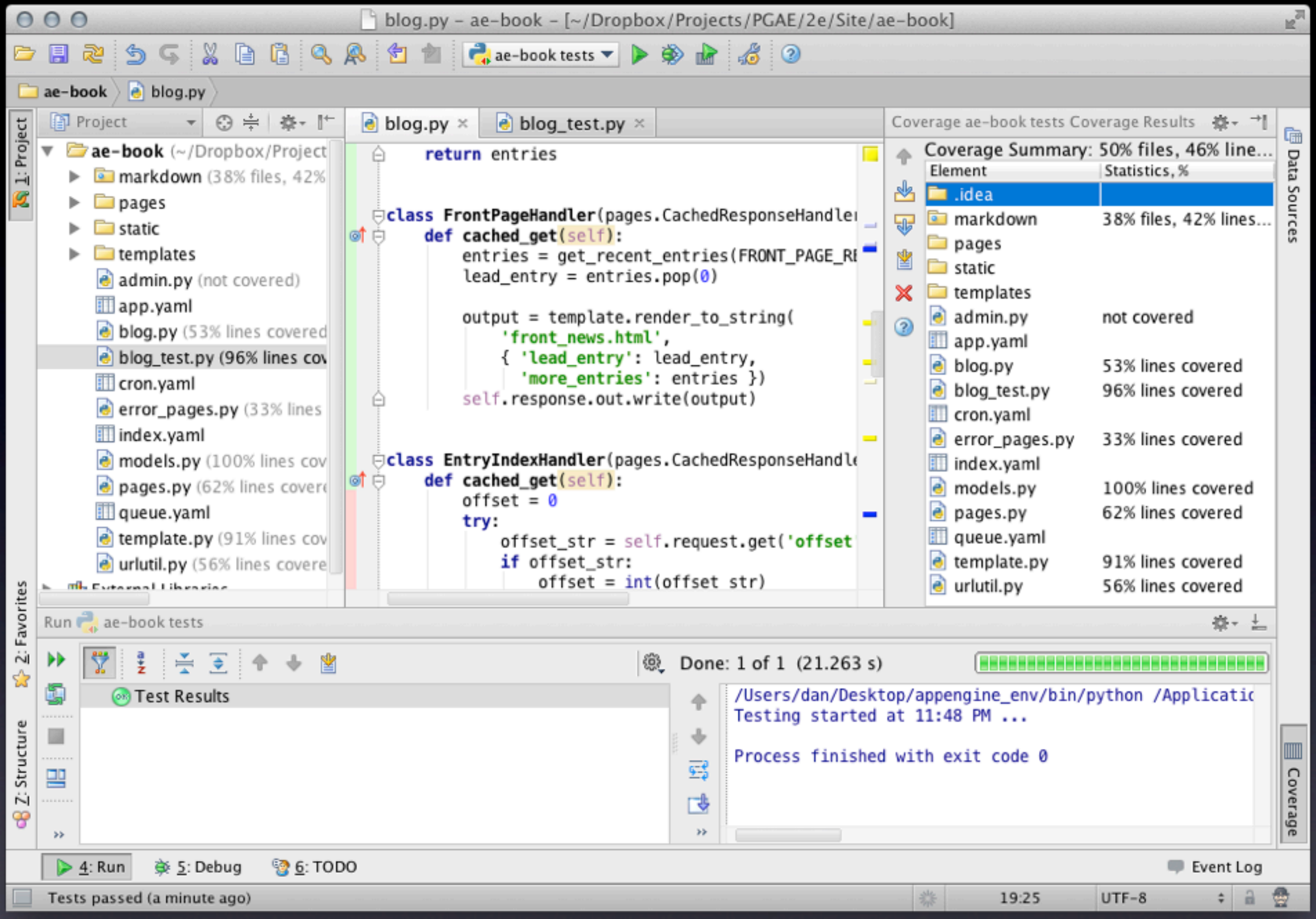
```
import unittest
import webtest

import blog
```

```
class BlogTest(unittest.TestCase):
  def setUp(self):
    # (testbed init goes here...)

    self.testapp = webtest.TestApp(
        blob.application)
```

# WebTest

```
class BlogTest(unittest.TestCase):
  # ...

  def testFrontPage(self):
    response = self.testapp.get('/')
    self.assertEqual(200, response.status_int)
    # ...
```

*webtest.pythonpaste.org*
*www.webob.org*


blog.py – ae-book – [~/Dropbox/Projects/PGAE/2e/Site/ae-book]
ae-book tests
ae-book
blog.py
1: Project
Project
ae-book (~/Dropbox/Project
markdown (38% files, 42%
pages
static
templates
admin.py (not covered)
app.yaml
blog.py (53% lines covered
blog_test.py (96% lines cov
cron.yaml
error_pages.py (33% lines
index.yaml
models.py (100% lines cov
pages.py (62% lines covere
queue.yaml
template.py (91% lines cov
urlutil.py (56% lines covere
blog.py
blog_test.py
return entries
class FrontPageHandler(pages.CachedResponseHandle
def cached_get(self):
entries = get_recent_entries(FRONT_PAGE_RE
lead_entry = entries.pop(0)
output = template.render_to_string(
'front_news.html',
{ 'lead_entry': lead_entry,
'more_entries': entries })
self.response.out.write(output)
class EntryIndexHandler(pages.CachedResponseHandle
def cached_get(self):
offset = 0
try:
offset_str = self.request.get('offset
if offset_str:
offset = int(offset_str)
Coverage ae-book tests Coverage Results
Coverage Summary: 50% files, 46% line...
Element
Statistics, %
.idea
markdown 38% files, 42% lines...
pages
static
templates
admin.py not covered
app.yaml
blog.py 53% lines covered
blog_test.py 96% lines covered
cron.yaml
error_pages.py 33% lines covered
index.yaml
models.py 100% lines covered
pages.py 62% lines covered
queue.yaml
template.py 91% lines covered
urlutil.py 56% lines covered
Data Sources
2: Favorites
7: Structure
Run ae-book tests
Done: 1 of 1 (21.263 s)
Test Results
/Users/dan/Desktop/appengine_env/bin/python /Applicatic
Testing started at 11:48 PM ...
Process finished with exit code 0
Coverage
4: Run
5: Debug
6: TODO
Event Log
Tests passed (a minute ago)
19:25
UTF-8

# Remote Access

# Remote Access

- Service API stubs: live site, development server, testbed
- Remote access: configure API stubs locally to access the live services for the app
- Uses a proxy in the app itself

# Remote Access

- Activate the remote_api builtin (/_ah/remote_api) in app.yaml:

```
builtins:
- remote_api: on
```

- remote_api_shell.py *appid*
- (Does not support OAuth, must use app-specific password: *google.com/settings*)
- Remote API calls consume app resources!

# Remote Access

```
% cd ae-book
% remote_api_shell.py ae-book
Email: ...
Password: ...
App Engine remote_api shell
Python 2.7.1 ...
s~ae-book> import models
s~ae-book> m = models.Entry.all().fetch(3)
s~ae-book> m[2].title
u’Code Samples Are Coming’
```

# Remote Access

```
#!/usr/bin/python

from google.appengine.ext.remote_api import \
  remote_api_stub

def auth_func():
  # prompt for email address and password
  return email_address, password

remote_api_stub.ConfigureRemoteApi(
    'appid', '/_ah/remote_api', auth_func)
remote_api_stub.MaybeInvokeAuthentication()
```

# Review

- Development environment
  - Python, AE SDK, dev server
  - Optional libs, virtualenv, pip; reproducable environments with requirements files
- Tools and IDEs
  - Launcher, PyCharm, Aptana PyDev
  - Using IDEs with virtualenv
  - Deployment with appcfg.py

# Review

- Unit testing
  - testbed
  - WebTest
  - PyCharm code coverage
- Remote access
  - Remote shell
  - Remote access from a script

*developers.google.com/appengine*

*appengine.google.com*

*ae-book.appspot.com*

*Programming Google App Engine, 2nd ed.*
Summer 2012

Dan Sanderson
*profiles.google.com/dan.sanderson*