codegenloader

Release 0.2

March 15, 2015

Contents

CodeGenLoader is a Python import hook that transparently runs a code generator at import time, allowing the use of generated code without a separate compilation step. The package includes an extensible base class as well as implementations for the Protocol Buffer and Thrift code generators.

CHAPTER 1

Installation

```
pip install codegenloader
```

CHAPTER 2

Prerequisites

CodeGenLoader runs on Python 2 (2.5+) and 3, including pypy and jython. The base class does not have any external dependencies, but to use the Thrift or Protocol Buffer subclasses you will need the thrift or protobuf packages installed. Note that protobuf 3.0.0 (currently in alpha) is required to use protobuf on Python 3.

Changed in version 0.2: Added Python 3 support.

CHAPTER 3

Example

Add the following two lines to `mypackage/proto/__init__.py`:

```
import codegenloader.protobuf
__path__ = codegenloader.protobuf.make_path(__name__, ".")
```

Now, assuming `foo.proto` exists in in `mypackage/proto/`, you can do:

```
from mypackage.proto.foo_pb2 import Foo
```

CHAPTER 4

Detailed docs

4.1 codegenloader.base

class codegenloader.base.**CodeGenLoader**(*path*)

Abstract base class for code generation import hooks.

The entry point for applications is to define a subclass and use the register class method to set __path__. This will make the module where __path__ was assigned a pseudo-package from which the generated code can be imported.

The interface between this class and the python interpreter is defined in PEP 302: http://www.python.org/dev/peps/pep-0302/

Constructs a CodeGenLoader.

Implements the hook protocol from PEP 302: it is called with a "path", and returns a loader if we can handle that path (i.e. if the path is actually a unique token we created in register), or raises an ImportError if not.

classmethod register(**args*, ***kwargs*)

Registers an import hook.

Arguments are passed (eventually) to initialize.

initialize(*modname*, *basedir*)

Real initialization function, independent of PEP302 requirements.

modname is the module name relative to which the generated code will be imported. basedir is the directory in which the source files for generation can be found. If it is not an absolute path, it is interpreted as relative to the file containing modname

find_module(*fullname*)

Returns a loader object for the module fullname, if it exists.

Implements the "finder" portion of the PEP 302 interface.

load_module(*fullname*)

Returns the module named fullname.

Implements the "loader" portion of the PEP 302 interface.

get_relname(*fullname*)

Converts a fully-qualified module name to a relative one.

get_contents(*relname*)

Return a tuple (is_pkg, contents) if code is stored for this module.

If the code is not found, raises KeyError.

store_contents(*relpath*, *contents*)
Store the contents of a file at relpath.

To be called from subclasses after code has been generated.

can_generate(*relname*)
Should return True if we can generate a module named `relname`.

generate(*relname*)
Generate code for module `relname`.

Should call `store_contents` for any files generated.

4.2 codegenloader.protobuf

`codegenloader.protobuf`.**make_path**(*modname*, *basedir*)
Returns a object to be set as `__path__`.

This is the visible entry point to this module. To use it, assign the result of this function to `__path__`:

```
import dropbox.codegenloader.protobuf
__path__ = dropbox.codegenloader.protobuf.make_path(__name__, "proto")
```

The first argument should always be `__name__`; the second is a directory name that contains the `.proto` files. (relative to the file where `make_path` is called).

4.3 codegenloader.thrift

`codegenloader.thrift`.**make_path**(*modname*, *basedir*)
Returns a object to be set as `__path__`.

This is the visible entry point to this module. To use it, assign the result of this function to `__path__`:

```
import dropbox.codegenloader.thrift
__path__ = dropbox.codegenloader.thrift.make_path(__name__, "thrift")
```

The first argument should always be `__name__`; the second is a directory name that contains the thrift files. (relative to the file where `make_path` is called).

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

C

C

F

G

I

L

M

R

S