jtypes.rubicon

Release 0.1.0a4

Adam Karpierz

Oct 17, 2021

CONTENTS:

CHAPTER

ONE

README

Currently only as placeholder (because a base package jtypes.jvm is still in development)

1.1 jtypes.rubicon

A bridge between the Java Runtime Environment and Python.

1.2 Overview

> **jtypes.rubicon** is a bridge between Python and Java, allowing these to intercommunicate.
> It is an effort to allow Python programs full access to Java class libraries.
>
> PyPI record.

> **jtypes.rubicon** is a lightweight Python package, based on the *ctypes* or *cffi* library.
> It is an almost fully compliant implementation of Steve Menard's **JPype** package by reimplementing whole its functionality in a clean Python instead of C/C++.

1.2.1 About Rubicon-Java:

Borrowed from the original website:

> **Rubicon-Java** is a bridge between the Java Runtime Environment and Python. It enables you to:
>
> - Instantiate objects defined in Java,
> - Invoke static and instance methods on objects defined in Java,
> - Access and modify static and instance fields on objects defined in Java, and
> - Write and use Python implementations of interfaces defined in Java.

1.2.2 Quickstart

Rubicon-Java consists of three components:

1. A Python library,
2. A JNI library, and
3. A Java JAR file.

A `Makefile` has been provided to compile the JNI and JAR components. Type:

```
$ make
```

to compile them. The compiled output will be placed in the `dist` directory.

Cross platform support

This Makefile currently only works under OS/X; however, the build commands aren't complicated; it should be fairly easy to reproduce the build on other platforms. Pull requests to make the `Makefile` cross-platform are welcome.

To use Rubicon-Java, you'll need to ensure:

1. `rubicon.jar` is in the classpath when you start your Java VM.
2. The Rubicon library file is somewhere that it will be found by dynamic library discovery. This means:
   a. Under OS X, put the directory containing `librubicon.dylib` is in your `DYLD_LIBRARY_PATH`
   b. Under Linux, put the directory containing `librubicon.so` is in your `LD_LIBRARY_PATH`
   c. Under Windows.... something :-)
3. The `rubicon` Python module is somewhere that can be added to a `PYTHONPATH`. You can install rubicon using:

   ```
   $ pip install rubicon-java
   ```

   If you do this, you'll need to reference your system Python install when setting your `PYTHONPATH`.

The Rubicon bridge starts on the Java side. Import the Python object:

```
import org.pybee.rubicon.Python;
```

Then start the Python interpreter, and run a Python file:

```
# Initialize the Python VM
String pythonHome = "/path/to/python";
String pythonPath = "/path/to/dir1:/path/to/dir2";
if (Python.start(pythonHome, pythonPath, null) != 0) {
    System.out.println("Error initializing Python VM.");
}

# Start a Python script
if (Python.run("/path/to/script.py") != 0) {
    System.out.println("Error running Python script.");
}

# Shut down the Python VM.
Python.stop();
```

The PYTHONPATH you specify must enable access to the rubicon Python module.

In your Python script, you can then reference Java objects:

```
>>> from rubicon.java import JavaClass

# Wrap a Java class
>>> URL = JavaClass("java/net/URL")

# Then instantiate the Java class, using the API
# that is exposed in Java.
>>> url = URL("http://pybee.org")

# You can then call methods on the Java object as if it
# were a Python object.
>>> print url.getHost()
pybee.org
```

It's also possible to provide implementations of Java Interfaces in Python. For example, lets say you want to create a Swing Button, and you want to respond to button clicks:

```
>>> from rubicon.java import JavaClass, JavaInterface

# Wrap the Java interface
>>> ActionListener = JavaInterface('java/awt/event/ActionListener')

# Define your own implementation
>>> class MyActionListener(ActionListener):
...     def actionPerformed(self, event):
...         print "Button Pressed"

# Instantiate an instance of the listener
>>> listener = MyActionListener()

# Create a button, and set the listener
>>> Button = JavaClass('javax/swing/JButton')
>>> button = Button('Push it')
>>> button.setActionListener(listener)
```

Of course, this sample code won't work unless it's in the context of a larger application starting a Swing GUI and so on.

1.2.3 Testing

To run the Rubicon test suite:

1. Configure your shell environment so that the Python, Java, and Rubicon dynamic libraries can be discovered by the dynamic linker.
   - On OSX, using Python 2.7.7 built under Homebrew:

     ```
     export DYLD_LIBRARY_PATH=/usr/local/Cellar/python/2.7.7_2/Frameworks/Python.
     ↪framework/Versions/2.7/lib/:`/usr/libexec/java_home`/jre/lib/server:./dist
     ```

2. Build the libraries:

```
$ make clean
$ make all
```

3. Run the test suite:

```
$ java org.pybee.rubicon.test.Test
```

This is a Python test suite, invoked via Java.

1.2.4 Community

Rubicon is part of the BeeWare suite. You can talk to the community through:

- @pybeeware on Twitter
- The pybee/general channel on Gitter.

We foster a welcoming and respectful community as described in our BeeWare Community Code of Conduct.

1.2.5 Contributing

If you experience problems with this backend, log them on GitHub. If you want to contribute code, please fork the code and submit a pull request.

1.3 Installation

Prerequisites:

- Python 2.7 or Python 3.5 or later
  - http://www.python.org/
  - 2.7 and 3.7 are primary test environments.
- pip and setuptools
  - http://pypi.python.org/pypi/pip
  - http://pypi.python.org/pypi/setuptools

To install run:

```
python -m pip install --upgrade jtypes.rubicon
```

To ensure everything is running correctly you can run the tests using:

```
python -m jt.rubicon.tests
```

1.4 Development

Visit development page

Installation from sources:

Clone the sources and run:

```
python -m pip install ./jtypes.rubicon
```

or on development mode:

```
python -m pip install --editable ./jtypes.rubicon
```

Prerequisites:

- Development is strictly based on *tox*. To install it run:

  ```
  python -m pip install tox
  ```

1.5 License

1.6 Authors

- Adam Karpierz <adam@karpierz.net>

CHAPTER

TWO

CHANGELOG

2.1 0.1.0a4 (2019-07-10)

- Last release for Python2.

2.2 0.1.0a3 (2018-11-08)

- Update of the required setuptools version.
- Minor setup and tests improvements.

2.3 0.1.0a0 (2016-11-30)

- Initial version.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- search