pyGraphML Documentation

Release 0.1

Hadrien Mary

July 26, 2011

CONTENTS

pyGraphML is a GraphML parser written in Python. GraphML is a comprehensive and easy-to-use file format for graphs.

CHAPTER

ONE

CONTENTS

1.1 Getting started

1.1.1 Using the bindings from the Python Interpreter

The Tulip Python bindings can also be used through the classical Python Interpreter. But some setup has to be done before importing the `tulip` module.

First, the path to the `tulip` module must be provided to Python. In the following, <tulip_install_dir> represents the root directory of a Tulip installation. The Tulip Python module is installed in the following directory according to your system :

- Linux : <tulip_install_dir>/lib
- Windows : <tulip_install_dir>/bin

This path has to be added to the list of Python module search path. To do so, you can add it in the **PYTHONPATH** environment variable or add it to the `sys.path` list.

Second, your system must be able to find the Tulip C++ libraries in order to use the bindings. These libraries are also installed in the directory provided above. You have to add this path to the **LD_LIBRARY_PATH** environment variable on Linux or to the **PATH** environment variable on Windows.

You should now be able to import the `tulip` module through the Python shell. Issue the following command at the shell prompt to perform that task:

```
>>> from tulip import *
```

Important, if you want to use Tulip algorithms implemented as plugins written in C++ (e.g. graph layout algorithms), you have to load them before being able to call them (see `tlp.applyAlgorithm()`, `tlp.Graph.computeLayoutProperty()`, ...). To load all the Tulip plugins written in C++, you have to execute the following sequence of command:

```
>>> tlp.initTulipLib()
>>> tlp.loadPlugins()
```

1.2 `pygraphml` – API documentation

class `Graph.`**`Graph`**(*name=''*)

Main class which represent a Graph

Parameters

- **name** – name of the graph

BFS(*root=None*)
Breadth-first search.

See Also:

Wikipedia descritpion

Parameters

- **root** – first to start the search

Returns list of nodes

DFS_prefix(*root=None*)
Depth-first search.

See Also:

Wikipedia descritpion

Parameters

- **root** – first to start the search

Returns list of nodes

add_edge(*n1*, *n2*, *directed=False*)

add_edge_by_label(*label1*, *label2*)

add_node(*label=''*)

children(*node*)

edges()

get_depth(*node*)

nodes()

root()

set_root(*node*)

set_root_by_attribute(*value*, *attribute='label'*)

show(*show_label=False*)

class Node.**Node**

children()

edges()

parent()

class Edge.**Edge**(*node1*, *node2*, *directed=False*)

child()

directed(*dir*)

node(*node*)
Return the other node

parent()

set_directed(*dir*)

class Attribute.**Attribute**(*name*, *value*, *type='string'*)

class Item.**Item**

attributes()

class Point.**Point**(*x=0*, *y=0*, *z=0*)

vectorize(*point*)

class GraphMLParser.**GraphMLParser**

parse(*fname*)

write(*graph*, *fname*)

CHAPTER

TWO

INDICES AND TABLES

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

PYTHON MODULE INDEX

p

INDEX