Tema AdapterLib

Date: 07.12.2010

AdapterLib is a python library for building keyword-based test automation tools that are compatible with TEMA test engine.

Table of Contents

Installation

tema-adapterlib is a normal python library and can be installed with normal commands:

```
python setup.py build
python setup.py install
```

Alternatively Debian or RPM packages can be used.

Debian installation

Adapter can be installed with following commands:

```
dpkg -i python-adapterlib-VERSION.deb
```

Packaging

Source packages

Makefile is included that can be used to build source distribution packages:

```
make source
```

Alternatively setup.py can be used directly to generate source package:

```
python setup.py sdist
```

Generated packages will be in directory **dist**.

Debian packages

Source package includes necessary files for building Debian packages. Makefile is included that can be used to build Debian packages:

```
make builddeb
```

Generated packages will be in directory **debbuild**.

RPM packages

Source package includes necessary files for building RPM packages. Makefile is included that can be used to build RPM packages:

```
make buildrpm
```

Generated packages will be in directory **dist**.

API Documentation

API documentation can be generated with epydoc:

```
make apidoc
```

Epydoc writes html-documentation to directory *apidoc*.

Creating new adapters

Best way to start is to look at existing adapters *tema-androidadapter*, *tema-atspiadapter* and *tema-onlinerobotadapter*.

Keywords

All keywords must be inherited from Keyword-class.

Another option is to inherit from KeywordProxy-class. KeywordProxy allows to use adapter as proxy for another keyword-based tool. For example tema-onlinerobotadapter is done using KeywordProxy.

TestRunner

New testrunner must be inherited from TestRunner in adapterlib.testrunner . In subclass at least methods _setupTestAutomation and _cleanupTestAutomation must be implemented.

All keywords, that will be used, must be imported in TestRunner.

Main

Short main program must be written. Main program needs to create AdapterMain from adapterlib and TestRunner from adapter.

Example main:

```
from ExampleAdapter.testrunner import TestRunner
from adapterlib.main import AdapterMain
if __name__ == "__main__":
    am = AdapterMain()
    options, args = am.parseArguments()
    if options:
        testRunner = TestRunner(args,options.delay,options.record)
        am.runAdapter(testRunner,options)
```