file:///D:/myTools/Python/Python36/Lib/site-packages/win32com/HTML/GeneratedSupport.html

Generated Python COM Support

This file describes how the Python COM extensions support "generated files". The information contained here is for expert Python users, and people who need to take advantage of the advanced features of the support. More general information is available in the Quick Start to Client Side COM documentation.

Introduction

Generated Python COM support means that a .py file exists behind a particular COM object. This .py file is created by a generation process from a COM type library.

This documentation talks about the process of the creation of the .py files.

Design Goals

The main design goal is that the Python programmer need not know much about the type library they wish to work with. They need not know the name of a specific Python module to use a type library. COM uses an IID, version and LCID to identify a type library. Therefore, the Python programmer only need know this information to obtain a Python module.

How to generate support files

Support files can be generated either "off-line" by the makepy utility, or in custom Python code.

Using makepy is in many ways far simpler - you simply pick the type library and you are ready to go! The Quick Start to Client Side COM documentation describes this process.

Often however, you will want to use code to ensure the type library has been processed. This document describes that process.

Usage

The win32com.client.gencache module implements all functionality. As described above, if you wish to generate support from code, you need to know the IID, version and LCID of the type library.

The following functions are defined. The best examples of their usage is probably in the Pythonwin OCX Demos, and the COM Test Suite (particularly testMSOffice.py)

Note that the gencache.py file supports being run from the command line, and provides some utilities for managing the cache. Run the file to see usage options.

Using makepy to help with the runtime generation

makepy supports a "-i" option, to print information about a type library. When you select a type library, makepy will print out 2 lines of code that you cant paste into your application. This will then allow your module to generate the makepy .py file at runtime, but will only take you a few seconds!

win32com.client.gencache functions

def MakeModuleForTypelib(typelibCLSID, lcid, major, minor, progressInstance = None):

Generate support for a type library.
Given the IID, LCID and version information for a type library, generate and import the necessary support files.

Returns

The Python module. No exceptions are caught.

Params

typelibCLSID
IID of the type library.
major
Integer major version.
minor
Integer minor version.
lcid
Integer LCID for the library.
progressInstance
A class instance to use as the progress indicator, or None to use the default GUI one.

def EnsureModule(typelibCLSID, lcid, major, minor, progressInstance = None):

Ensure Python support is loaded for a type library, generating if necessary.

Given the IID, LCID and version information for a type library, check and if necessary generate, then import the necessary support files.

Returns:

The Python module. No exceptions are caught during the generate process.

Params

typelibCLSID
IID of the type library.

major
Integer major version.

minor
Integer minor version.

lcid
Integer LCID for the library.

progressInstance
A class instance to use as the progress indicator, or None to use the default GUI one.

def GetClassForProgID(*progid*):

Get a Python class for a Program ID
Given a Program ID, return a Python class which wraps the COM object

Returns

The Python class, or None if no module is available.

Params

progid
A COM ProgramID or IID (eg, "Word.Application")

def GetModuleForProgID(progid):

Get a Python module for a Program ID
Given a Program ID, return a Python module which contains the class which wraps the COM object.

Returns

The Python module, or None if no module is available.

Params:

progid
A COM ProgramID or IID (eg, "Word.Application")

def GetModuleForCLSID(clsid):

Get a Python module for a CLSID
Given a CLSID, return a Python module which contains the class which wraps the COM object.

Returns

The Python module, or None if no module is available.

Params

progid
A COM CLSID (ie, not the description)

def GetModuleForTypelib(typelibCLSID, lcid, major, minor):

Get a Python module for a type library ID

Returns

An imported Python module, else None

Params:

typelibCLSID
IID of the type library.
major
Integer major version.
minor
Integer minor version
lcid
Integer LCID for the library.