Introduction to Python ISAPI support

See also

- The isapi related modules
- The isapi related objects

Note: if you are viewing this documentation directly from disk, most links in this document will fail - you can also find this document in the CHM file that comes with pywin32, where the links will work

Introduction

This documents Python support for hosting ISAPI exensions and filters inside Microsoft Internet Information Server (IIS). It assumes a basic understanding of the ISAPI filter and extension mechanism.

In summary, to implement a filter or extension, you provide a Python module which defines a Filter and/or Extension class. Once your class has been loaded, IIS/ISAPI will, via an extension DLL, call methods on your class.

A filter and a class instance need only provide 3 methods - for filters they are called `GetFilterVersion`, `HttpFilterProc` and `TerminateFilter`. For extensions they are named `GetExtensionVersion`, `HttpExtensionProc` and `TerminateExtension`. If you are familiar with writing ISAPI extensions in C/C++, these names and their purpose will be familiar.

Most of the work is done in the `HttpFilterProc` and `HttpExtensionProc` methods. These both take a single parameter - an HTTP_FILTER_CONTEXT and EXTENSION_CONTROL_BLOCK object respectively.

In addition to these components, there is an 'isapi' package, containing support facilities (base-classes, exceptions, etc) which can be leveraged by the extension.

Base classes

There are a number of base classes provided to make writing extensions a little simpler. Of particular note is `isapi.threaded_extension.ThreadPoolExtension`. This implements a thread-pool and informs IIS that the request is progressing in the background. Your sub-class need only provide a `Dispatch` method, which is called on one of the worker threads rather than the thread that the request came in on.

There is base-class for a filter in `isapi.simple`, but there is no equivilent threaded filter - filters work under a different model, where background processing is not possible.

Samples

Please see the `isapi/samples` directory for some sample filters and extensions.

Implementation

A Python ISAPI filter extension consists of 2 main components:

- A DLL used by ISAPI to interface with Python.
- A Python script used by that DLL to implement the filter or extension functionality

Extension DLL

The DLL is usually managed automatically by the isapi.install module. As the Python script for the extension is installed, a generic DLL provided with the isapi package is installed next to the script, and IIS configured to use this DLL.

The name of the DLL always has the same base name as the Python script, but with a leading underscore (_), and an extension of .dll. For example, the sample "redirector.py" will, when installed, have "_redirector.dll" created in the same directory.

The Python script may provide 2 entry points - methods named __FilterFactory__ and __ExtensionFactory__, both taking no arguments and returning a filter or extension object.

Using py2exe and the isapi package

You can instruct py2exe to create a 'frozen' Python ISAPI filter/extension. In this case, py2exe will create a package with everything you need in one directory, and the Python source file embedded in the .zip file.

In general, you will want to build a seperate installation executable along with the ISAPI extension. This executable will be built from the same script. See the ISAPI sample in the py2exe distribution.