Apache モジュール mod_so

説明:	起動時や再起動時に実行コードとモジュールをサーバにロードする
ステータス:	Extension
モジュール識別子:	so_module
ソースファイル:	mod_so.c
互換性:	このモジュールは Window では（常に含まれている）Base モジュールです

概要

いくつかのオペレーティングシステムでは、サーバの再コンパイルをする代わりに、 このモジュールを使用して 動的共有オブジェクト[1]（DSO）機構により、実行時に Apache にモジュールを読み込ませることが できます。

Unix 上では、読み込まれるコードは通常は共有オブジェクトファイル（普通 .so という拡張子が付いています）からです。 Windows 上ではこのモジュールの拡張子は .so か .dll です。

> 警告
>
> Apache 1.3 のモジュールを直接 Apache 2.0 で使うことはできません - モジュールは Apache 2.0 用に動的にロードされるか、 直接組み込まれるために修正されなければなりません。

トピック

ディレクティブ

Windows 用のロード可能なモジュールを作成する

> 注
>
> Apache 1.3.15 と 2.0 とで Windows のモジュール名の形式は変更されました - モジュールは mod_foo.so という名前になりました。
>
> まだ mod_so で ApacheModuleFoo.dll という名前のモジュールも ロードされますが、新しい名前の付け方を使う方が好まれます。モジュールを 2.0 用に移植しているのであれば、2.0 の習慣に合うように名前を 修正してください。

Apache のモジュール API は UNIX と Windows 間では変更されていません。 多くのモジュールは全く変更なし、もしくは簡単な変更により Windows で実行できるようになります。ただし、それ以外の Windows には無い Unix アーキテクチャーの機能に依存したモジュールは動作しません。

モジュールが実際に動作するときは、 二つの方法のどちらかでサーバに追加することができます。まず、Unix と同様にサーバにコンパイルして組み込むことができます。Windows 用の Apache は Unix 用の Apache にある Configure プログラムがありませんので、モジュールのソースファイルを ApacheCore プロジェクトファイルに追加し、シンボルを os¥win32¥modules.c ファイルに追加する必要があります。

二つ目はモジュールを DLL としてコンパイルする方法です。 DLL は共有ライブラリで、実行時に LoadModule ディレクティブによりサーバに読み込むことができます。これらのモジュール DLL はそのまま配布することが可能で、サーバを再コンパイルすることなく、Windows 用の Apache のすべてのインストールで実行することができます。

モジュール DLL を作成するためには、 モジュールの作成に小さな変更を行なう必要があります。 つまり、モジュールのレコード（これは後で作成されます。 以下を参照してください）が DLL からエクスポートされなければなりません。 これを行なうには、AP_MODULE_DECLARE_DATA (Apache のヘッダファイルで定義されています) をモジュールのモジュールレコード 定義の部分に追加してください。たとえば、モジュールに

```
module foo_module;
```

があるとすると、それを次のもので置き換えてください。

```
module AP_MODULE_DECLARE_DATA foo_module;
```

Unix 上でもこのモジュールを 変更無しで使い続けられるように、このマクロは Windows 上でのみ効力を持ちます。.DEF ファイルの方を良く知っているという場合は、 代わりにそれを使ってモジュールレコードを エクスポートすることもできます。

さあ、あなたのモジュールの DLL を作成しましょう。これを、 libhttpd.lib 共有ライブラリがコンパイルされたときに作成された ibhttpd.lib エクスポートライブラリとリンクしてください。この時に、 Apache のヘッダファイルが正しい位置にあるように、 コンパイラの設定を変える必要があるかもしれません。 このライブラリはサーバルートの modules ディレクトリにあります。 ビルド環境が正しく設定されるように、既存のモジュール用の .dsp を 取ってくるのが一番良いでしょう。もしくは、あなたの .dsp と コンパイラとリンクのオプションを比較する、というものでも良いです。

これで DLL 版のモジュールが作成されているはずです。 サーバルートの modules ディレクトリにモジュールを置いて、 LoadModule ディレクティブを使って読み込んでください。

LoadFile ディレクティブ

説明:	指定されたオブジェクトファイルやライブラリをリンクする
構文:	LoadFile filename [filename] ...
コンテキスト:	サーバ設定ファイル
ステータス:	Extension
モジュール:	mod_so

LoadFile ディレクティブは、サーバが起動されたときや再起動されたときに、 指定されたオブジェクトファイルやライブラリをリンクします。 これはモジュールが動作するために

必要になるかもしれない追加の コードを読み込むために使用されます。Filename は絶対パスか、ServerRoot [2] からの相対パスです。

例:

```
LoadFile libexec/libxmlparse.so
```

LoadModule ディレクティブ

説明:	オブジェクトファイルやライブラリをリンクし、使用モジュールの リストに追加する
構文:	LoadModule module filename
コンテキスト:	サーバ設定ファイル
ステータス:	Extension
モジュール:	mod_so

LoadModule ディレクティブは filename というオブジェクトファイルおよびライブラリをリンクし、module という名前のモジュールの構造をアクティブなモジュールのリストに追加します。 Module はファイル中の module 型の外部変数の名前で、モジュールのドキュメントに モジュール識別子 [3]として書かれているものです。例 :

```
LoadModule status_module modules/mod_status.so
```

これは ServerRoot の modules サブディレクトリから指定された名前の モジュールをロードします。

URI References

[1] http://httpd.apache.org/docs-2.1/dso.html
[2] http://httpd.apache.org/docs-2.1/mod/core.html#serverroot
[3] http://httpd.apache.org/docs-2.1/mod/module-dict.html#moduleidentifier