自信を持って Apacheを操るために

内部構造からたどるWebサーバ設定のキモ

本テキストに関するご質問、その他ご用命については

小山浩之　<oyama@module.jp>

http://module.jp/

までご連絡ください。

概念	50分
休憩	10分
設定例	50分
休憩	10分
標準モジュールの動作	50分
質疑応答	？

スライドについて

お手元のテキスト スライド番号

スライドについて

テキストから修正があるもの

スライドについて

テキストには無いスライド

- マニュアルは最低１回、目を通す
- 設定ファイルはコンテキストを意識する
- まずデフォルトのhttpd.confを捨てる
- Apacheは”モジュール”が命

Apacheの概念

Apache 1.3.x

- 基本的にバグフィックスのみ
- 新機能の追加は(基本的に)行わない
- 枯れたソフトウェアでセキュリティホールも少ない

Apache 2.0.x

- より多機能化
  - 新機能の追加
  - モジュール化の徹底
- “基本的には”1.3.xと大きな違いは無い

Apache 2.2.0

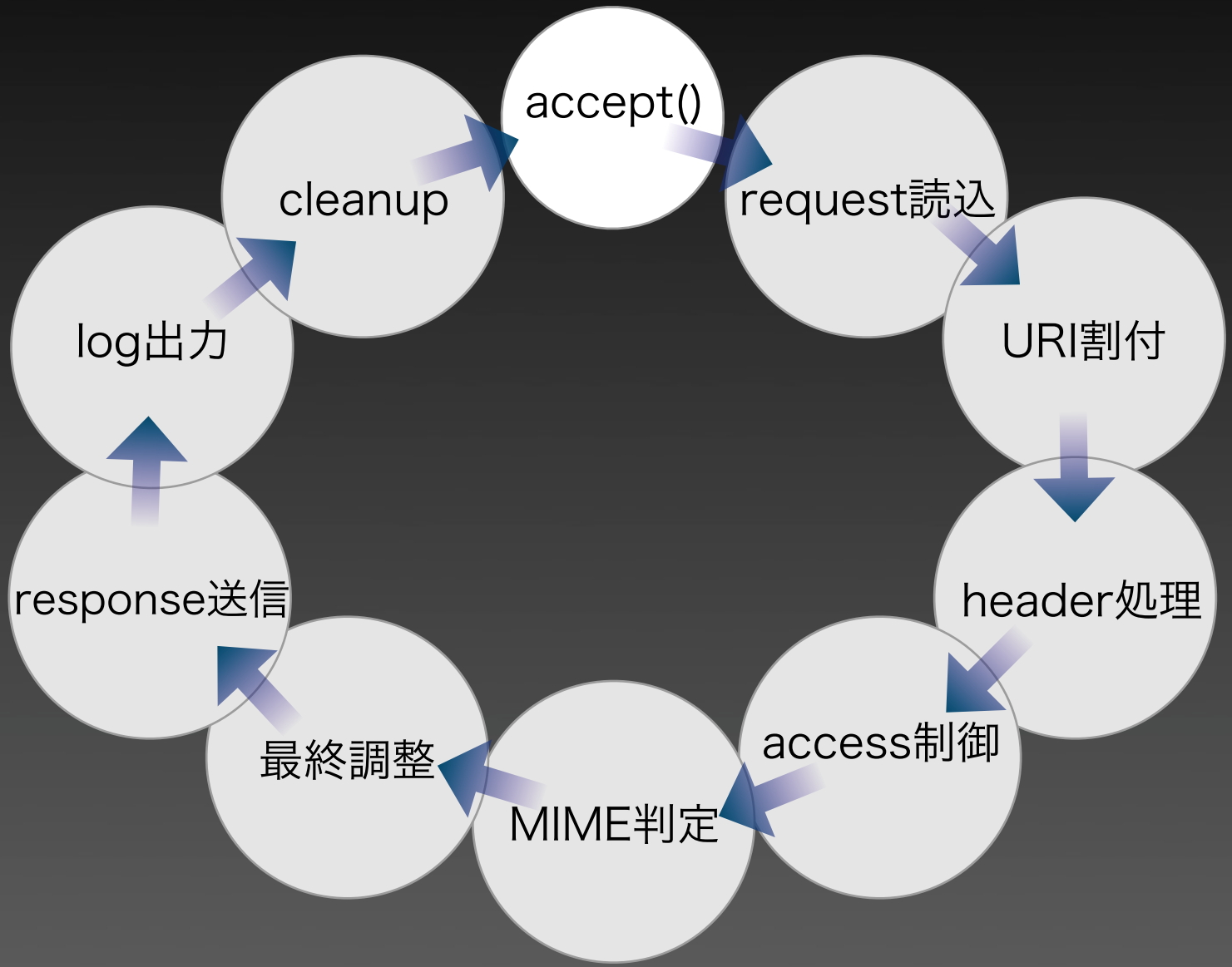
- 2005年12月現在の最新安定板
- 基本的な構造は2.0.xと同じ
- 更に多機能化
  - proxyベースのロードバランサ
  - AJP13サポート
  - キャッシュ
- 認証・承認系アーキテクチャの刷新

- 本家のドキュメントだけでも十分です
- 必ず最低１回すべてに目を通してください
- (開発以外に)必要な事はすべて書いてあります

Apacheの仕事は...

- リクエストを受け付ける
- コンテンツにマッピングする
- アクセスの可否を制御する
- コンテンツを配信する
- 作業内容の記録(ログ)

リクエスト処理ループ


accept()
request読込
URI割付
header処理
access制御
MIME判定
最終調整
response送信
log出力
cleanup

- 一つのリクエスト毎に、リクエスト処理ループを実行する
- リクエスト処理中はプロセス/スレッドが占有される
- ループ中の処理内容はURIによって変化する
- ループ中の処理内容は組み込んでいるモジュールによって変化する

- Apacheはモジュール化されたサーバ
- モジュールで機能をpluginする
- モジュールの無いApacheは貧弱

モジュールの機能

- アクセス制御
- ユーザ認証
- ログの出力
- 動的なコンテンツの配信
- Proxy

その他、ほぼ全ての機能がモジュールによるもの

- モジュールを知る == Apacheを知る
- どうやって知る？
  - モジュールのソースコードを読む
  - ドキュメントを読む

- モジュール一覧
  - http://httpd.apache.org/docs/2.2/mod/
- ディレクティブクイックリファレンス
  - http://httpd.apache.org/docs/2.2/mod/quickreference.html

- 設定要素”ディレクティブ”は、モジュールの動作を決定するもの
- モジュールを組み込むと、そのモジュール用のディレクティブが追加される
  - つまりモジュールを組み込んでいないと利用できないディレクティブがある
  - ディレクティブには、それを利用できる”コンテキスト”が決まっている

- ディレクティブの大文字小文字は無視
  - 引数は意識するかも
- ディレクティブの前(左)の空白は無視
- # はコメント
  - 文の後ろにコメントはNG
  - Listen 8080 # これはダメ

コンテキスト?

- サーバ設定ファイル
  - httpd.conf
- バーチャルホスト
  - <VirtualHost>
- ディレクトリ
  - <Directory>, <Location>, <Files>
- .htaccess

```
 1:    Listen 80
 2:    ServerRoot /var/www
 3:    DocumentRoot /var/www/htdocs
 4:
 5:    User  nobody
 6:    Group nobody

       ~~~

17:    MaxRequestsPerChild 0
18:    ErrorLog logs/error_log
19:
20:    <Directory />
21:        Options FollowSymLinks
22:        AllowOverride None
23:    </Directory>
```

サーバ設定

ディレクトリ

Listen ディレクティブ

説明:	サーバが listen するIP アドレスとポート番号
構文:	Listen [*IP-address*:]*portnumber*
コンテキスト:	サーバ設定ファイル
ステータス:	MPM
モジュール:	beos, leader, mpm_netware, mpm_winnt, mpmt_os2, perchild, prefork, threadpool, worker
互換性:	Apache 2.0 から必要なディレクティブ

Listen ディレクティブは Apache が特定の IP アドレスやポート番号だけを listen するように指定します。 デフォルトでは全ての IP インターフェースのリクエストに応答します。 Listen ディレクティブは 現在は必須のディレクティブと

AuthUserFile ディレクティブ

説明:	認証に使用するユーザとパスワードの一覧が格納されている、 テキストファイルの名前を設定する
構文:	AuthUserFile *file-path*
コンテキスト:	ディレクトリ, .htaccess
上書き:	AuthConfig
ステータス:	Base
モジュール:	mod_auth

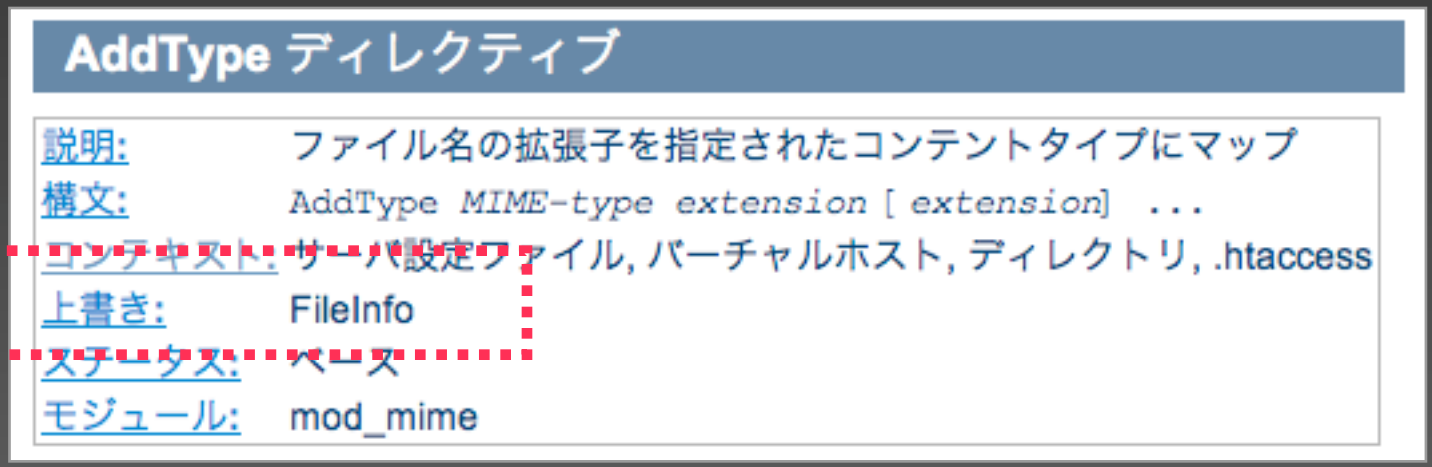
AuthUserFile ディレクティブは、 ユーザ認証のためのユーザとパスワードの一覧を格納した テキストファイルの名前を設定します。 *file-path* はユーザファイルへのパスです。 もし絶対パスでなければ (つまり スラッシュで始まらないパスで

- コンテキストを意識することで、設定の幅が広がる
  - URIごとの設定 <Location>
  - ディレクトリごとの設定 <Directory>
- コンテキストごとに個別の設定が可能

- <Directory>, <DirectoryMatch>
- <Location>, <LocationMatch>
- <Files>, <FilesMatch>
- <VirtualHost>
- <IfModule>

- 設定は入れ子に継承される
- <Directory /> 基本の設定
- <Directory /var/www/htdocs> 更に具体的な設定

- .htaccessは<Directory>の延長
- AllowOverrideで記述可能か決定する
  - マニュアルの”上書き”を調べる



AddType ディレクティブ

説明:	ファイル名の拡張子を指定されたコンテントタイプにマップ
構文:	AddType *MIME-type extension* [*extension*] ...
コンテキスト:	サーバ設定ファイル, バーチャルホスト, ディレクトリ, .htaccess
上書き:	FileInfo
ステータス:	ベース
モジュール:	mod_mime

- ちょっと違うもの<IfModule ...>
- 指定したモジュールが組み込まれている場合にのみ評価される
  - <IfModule proxy_module>...</IfModule>
  - <IfModule mod_proxy.c>...</IfModule>
- 構成の違う複数のApacheを、一つのhttpd.confで管理したい場合などに便利
  - 実際はIncludeを使った方が便利かも？

Apacheの設定例

1.3 & 2.0の設定

- デフォルトのhttpd.confは捨てる
  - 理解・運用の妨げ
  - 冗長な記述
  - 使わないパラメータ
- highperformance.confから始める

- highperformance.confは...
  - わずか70行 (httpd.confは1,070行)
  - わずか15の設定
  - 設定の土台として最適
    - 肉付けは必要

```
# cp highperformance.conf  httpd.conf
```

```
 1:   Listen 80
 2:   ServerRoot /var/www
 3:   DocumentRoot /var/www/htdocs
 4:
 5:   User  nobody
 6:   Group nobody
 7:
 8:   <IfModule worker.c>
 9:       StartServers          2
10:       MaxClients          150
11:       MinSpareThreads      25
12:       MaxSpareThreads      75
13:       ThreadsPerChild      25
14:       MaxRequestsPerChild   0
15:   </IfModule>
16:
17:   MaxRequestsPerChild 0
18:   ErrorLog logs/error_log
19:
20:   <Directory />
21:       Options FollowSymLinks
22:       AllowOverride None
23:   </Directory>
```

2.2の設定

- デフォルトのhttpd.confが大幅に簡素化
  - 補助的な設定は conf/extra/*.conf へ
- とはいえ基本は一緒で「いったん全部消す」
  - もしくはコメントを全削除

2.0から2.2の移行

- 基本的にそのまま移行可能
  - 2.0のモジュールは2.2でも動作する
  - httpd.confも流用可能
- mod_accessモジュールの廃止 (mod_authz_host)
  - configureのオプションとLoadModuleの引数が変わった

足りないものは？

- アクセスログ
- ディレクトリへのアクセス時の配慮
- 問題のあるクライアントへの配慮

アクセスログの出力

- mod_log_config.cを追加

```
LoadModule log_config_module modules/mod_log_config.so

LogFormat “%h %l %u %t \"%r\" %>s %b” common
CustomLog logs/access_log  common
```

LogFormatとCustomLogの組み合わせで、
自由にログの出力が可能。
書式は

http://httpd.apache.org/docs/2.2/mod/mod_log_config.html

ディレクトリの配慮

- mod_dir.cを追加

```
LoadModule dir_module modules/mod_dir.so

DirectoryIndex index.html
```

mod_dir.cが無いとディレクトリへのアクセスを、自動的にindex.htmlに割り振る機能が働かない。

問題clientへの配慮

- mod_setenvif.cを組み込む

```
LoadModule setenvif_module modules/mod_setenvif.so

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
...
```

必要ならば設定を追加

- バーチャルホスト
- .htaccessによる設定の許可
- 動的コンテンツのハンドリング
- SSL
- アクセス制御, ユーザ認証
- WebDAV
- エラー画面のカスタマイズ
- 仮想URIのマッピング

モジュールが無い？

- バイナリパッケージ管理システムの多くはDSOが前提
- ソースコードからビルドする場合もDSOがデフォルト
  - ただし全てのモジュールがビルドされているわけではない

標準モジュール

- access, auth, include, log-config, env, setenvif, mime, status, autoindex, asis, cgi/cgid, negotiation, dir, imap, actions, userdir, alias
- auth-anon, auth-dbm, auth-digest, cache, ldap, auth-ldap, deflate, log-forensic, mime-magic, cern-meta, expires, headers, usertrack, unique-id, proxy, ssl

2.2の標準モジュール

- authn-*, authz-*, auth-basic, include, filter, log-config, env, setenvif, mime, status, autoindex, asic, cgi/cgid, negotiation, dir, actions, userdir, alias, so
- ./configure --helpで判別できる
  - --disable-NAME ... 標準で組み込まれる
  - --enable-NAME ... 指定すれば組み込まれる

+

```
./configure --disable-include
# 明示的にmod_includeを外す

./configure --enable-proxy --enable-rewrite
# mod_proxy, mod_rewriteを追加する

./configure --enable-mods-shared=all
# 全てのモジュールをビルドする(実験的なモジュールなどは除く)
```

- マルチプロセッシングモジュールの選定
  - コンパイル時に決定し、後から変更はできない
  - --with-mpm=worker
  - 特別な理由が無い限りworkerを選ぶ
    - FreeBSD 4.xはprefork。5.xはOK

- コンテンツハンドラの設定
  - コンテンツに応じた処理方法

```
AddHandler  default  .gif
AddHandler  cgi-script .cgi
AddHandler  php-script .php
```

  - 場所(URI)に応じた処理方法

```
<Location /status>
    SetHandler server-status
</Location>


ScriptAlias /cgi-bin/  /var/www/cgi-bin/
```

- マッピング
  - Apacheは基本的にDocumentRootをそのまま公開する
  - DocumentRoot以外のディレクトリや、ファイルを柔軟に公開できる
    - Alias, RewriteRule etc...

```
Alias /virtual/  /home/oyama/htdocs/

RewriteEngine On
RewriteRule ^(.+)¥.html$  $1.php
```

request_rec構造体

- リクエスト処理ループ内で処理中のリクエスト情報を格納する
- リクエスト処理ループの各ステップで、モジュールがrequest_recの値を書き換えて動作をカスタマイズしている
- リクエスト/レスポンスの要

request_rec構造体

- リクエストされたURI
- マッピンスされたファイル名
- MIMEタイプ
- コンテンツハンドラ名
- リクエスト/レスポンスヘッダ

その他リクエスト/レスポンスに関わる全ての情報

Apache
標準モジュールの動作

- 標準モジュール約30数個
- 実際に利用しているのは極一部
  - Mapper
  - AAA
  - Metadata
  - Generators
  - Filters
  - Logger

HTTPモジュール

- HTTPの処理にまつわるcoreモジュール
- http_core.c, mod_mime.c

Mappersモジュール

- URIとファイルシステムのマッピング
- mod_alias.c, mod_dir.c, mod_negotiation.c, mod_rewrite.c, mod_userdir.c, mod_vhost_alias.c etc...
- 最も豊富で重要？

AAAモジュール

- アクセス制御やユーザ認証などを行う
- Authorization And Authentication
- mod_access.c, mod_auth.c etc.

2.2のAAAモジュール

- 2.0までの認証系モジュールは機能が一枚岩で開発者に不評
  - 認証と承認機能を任意に組み合わせられる構造に
- Authenticationモジュール
  - mod_authn_file, mod_authn_dbm...
- Authorizationモジュール
  - mod_authz_user, mod_authz_host...
- mod_accessは廃止

+

Metadataモジュール

- Apacheの内部動作・レスポンスヘッダなどの付加情報の操作
- mod_env.c, mod_expires.c, mod_headers.c, mod_mime_magic.c, mod_setenvif.c, mod_unique_id.c, mod_usertrack.c etc...

Generatorsモジュール

- コンテンツを生成する
- “コンテンツハンドラ”
- SetHandler, AddHandlerに呼応して起動する
- mod_autoindex.c, mod_cgi.c etc...

Filtersモジュール

- 送信するコンテンツを加工する
- mod_deflate.c, mod_ext_filter.c, mod_include.c

Loggersモジュール

- 動作ログの出力
- mod_log_config.c etc...

- モジュールたちがrequest_rec構造体を通じ連携し、一つのリクエストを処理
- 利用するモジュールがどの時点でrequest_rec構造体を参照・操作するかが動作のキモ
- Apacheのモジュールアーキテクチャは自由度が高く、必ずしも行儀の良いモジュールばかりではない

モジュールの動作

モジュールAPI	Apache内部の関数
ap_hook_translate_name()	ap_run_translate_name()
ap_hook_map_to_storage()	ap_run_map_to_storage()
ap_hook_header_parser()	ap_run_header_parser()
ap_hook_access_checker()	ap_run_access_checker()
ap_hook_check_user_id()	ap_run_check_user_id()
ap_hook_auth_checker()	ap_run_auth_checker()
ap_hook_type_checker()	ap_run_type_checker()
ap_hook_fixups()	ap_run_fixups()
ap_hook_handler()	ap_run_handler()
ap_hook_log_transaction()	ap_run_log_transaction()

- 各モジュールのmodule構造体を辿ると、モジュールの動作を理解しやすい
  - Directiveの定義
  - Apacheのフックに登録する関数の定義
  - etc...

- ディレクティブの定義についてはマニュアルを読んだ方が手軽。
    - ただし引数の解釈方法を厳密に把握したい場合はソースコードを読んだほうが手軽で確実
- モジュールの動作の詳細はソースコードを読まなければ分からない

mod_dir.c

- DirectoryIndex, DirectorySlashディレクティブ
- ディレクトリアクセス時のindexファイルへの自動内部リダイレクト
- /(スラッシュ)無しのディレクトリアクセス時、/ 付きURLへの自動外部リダイレクト
- ap_hook_fixups()で介入している

mod_dir.c

- 一般的なWebサイトをホストするサーバでは必須？
- ディレクトリ以外へのアクセス時はほとんど処理は行わない
- DirectoryIndexはマッチしやすい名前を左側に記述する

mod_userdir.c

- UserDirディレクティブ
- /~(チルダ)で始まるURIを、ローカルユーザの公開ディレクトリにマッピングする
- ap_hook_translate_name()で介入している
  - mod_alias.c, mod_vhost_alias.cも同様

AAAの動作

- 他のモジュールとは一部異なる
- mod_auth_*
  - ユーザ認証の手段を実装(Basic/Digest認証)
- mod_authz_*
  - アクセスの可否ルールを実装(ホストアドレス, ユーザ,グループ etc...)
- mod_authn_*
  - 認証データベースのアクセス用プラグイン(file,dbm etc...)

mod_authn_file.c

- AuthUserFileディレクティブ
- ユーザ認証が必要な場面で、受け取ったユーザ名を元にパスワードファイルのエントリを検索・パスワード比較を行う
- mod_auth_basic/mod_auth_digest共通のプラグインとして動作する

mod_authn_file.c

- キャッシュやインデックスを使用せずに、ファイル先頭からシーケンシャルに検索する
- 当然パスワードファイルが大きい場合はパフォーマンス的に不利
  - mod_authn_dbm.c, mod_authnz_ldap.c, mod_authn_dbd.cなどの選択肢がある

アクセス制御は簡単

```
 1: static int my_access_check(request_rec *r)
 2: {
 3:     const char *cookie;
 4:
 5:     cookie = apr_table_get(r->headers_in,
 6:                            “Cookie”);
 7:     if (cookie != NULL
 8:         && strcmp(cookie, “key=value”) != 0) {
 9:             return OK;
10:     }
11:
12:     return HTTP_FORBIDDEN;
13: }
```

mod_expires.c

- ExpiresAcitve, ExpiresByType, ExpiresDefault ディレクティブ
- レスポンスヘッダにコンテンツの有効期限”Expires:”と”Cache-Control:”をセットするだけ。
- 実はFilter系モジュールなので、コンテンツ配信時に起動する

mod_expires.c

- 既にExpiresヘッダがセットされている場合はそのまま使用する
- ファイルのmtimeまたは、アクセス時刻を基準に有効期限を計算
  - ExpiresByType text/css　M3600
  - ExpiresByType text/gif　A3600

mod_header.c

- Header, RequestHeaderディレクティブ
- 任意のリクエスト/レスポンスヘッダを付与する
- mod_expires.cに似ているが、Filterに加えてap_hook_fixups()も使用している

mod_cgi.c

- ScriptLog, ScriptLogLength, ScriptLogBuffer ディレクティブ
- fork() & exec()して外部プログラムを実行
- ap_hook_hander()で介入。”cgi-script”ハンドラを登録

mod_cgi.c

- cgi-scriptハンドラのリクエストか？
- HTTP GET/POSTメソッドか？
- ExecCGIな環境か？stat()は取れてる？
- 子プロセスをfork() & exec()
- リクエストbodyをpipe (stdin)にwrite
- pipe (stdout)からレスポンスをread
- レスポンスヘッダを解析。Location:ヘッダがある場合はリダイレクト
- レスポンスをクライアントに配信

mod_phpX.c

- php_value, php_flag, php_admin_valueディレクティブ
- 処理の流れ自体はmod_cgi などと似ている。スクリプトエンジン内包。
- ap_hook_handler()

mod_proxy_ajp.c

- AJP13でサービスしてるアプリケーションへのProxy
- 他のWebサーバを”マウント”するのと同様に、AJP13サーバをマウントする。
- AJP13でリクエストを転送し、レスポンスをクライアントに配送するだけ。

- 多くのカスタマイズ用件はApache三種の神器で対応可能
  - mod_setenvif
  - mod_rewrite
  - mod_log_config
- これらモジュール活用のボキャブラリがカギ
- ユーザ認証系もAAAモジュールのリファクタリングで柔軟性が増した
- ちょっと変わったアクセス制御系は独自にモジュールを実装するのが吉

- モジュールのソースコードを読むのが一番手っ取り早い
- 込み入ったデバッグが必要な場合も、コードを読めば当たりがつけやすい
- 最低限のC言語の知識で楽に読める

本テキストに関するご質問、その他ご用命については

小山浩之　<oyama@module.jp>

http://module.jp/

までご連絡ください。