コンテントネゴシエーション

Apache は HTTP/1.1 の規格に記述されているコンテントネゴシエーションを サポートしています。 ブラウザにより提供されたメディアタイプ、 言語、文字セット、エンコーディングの優先傾向に基づいて、 最適なリソースの表現を選択できます。 また、不完全なネゴシエーション情報を送ってくるブラウザからのリクエストを もっと賢く取り扱えるよう、いくつか機能も実装してあります。

コンテントネゴシエーションは mod_negotiation モジュールによって 提供されていて、デフォルトで組み込まれています。

トピック

コンテントネゴシエーションについて

リソースは、幾つか異なった表現で利用できる場合があります。 例えば、異なる言語や異なるメディアタイプ、 またはそれらの組み合わせで利用できるかも知れません。 もっとも適した選択をする方法の一つには、インデックスページを ユーザに見せて、ユーザに選んでもらう方法があります。 しかし、サーバが自動的に選ぶことができる場合が多くあります。 これは、ブラウザがリクエスト情報毎の情報の一部に、 どの表現を嗜好するかを送ることで動作しています。 例えばブラウザは、可能ならフランス語で情報を見たい、 不可能ならその代わりに英語でもよいと、 自分の嗜好を知らせることができます。 ブラウザはリクエストのヘッダで自分の優先傾向を知らせます。 フランス語のみの表現を要求する場合は、ブラウザは次を送ります。

```
Accept-Language: fr
```

この優先傾向は、選択可能な表現が存在して、 言語によって様々な表現がある場合にのみ適用される ということに注意してください。

もっと複雑なリクエストの例を挙げましょう。 このブラウザはフランス語と英語を受け付ける、しかしフランス語を好む、 そして様々なメディアタイプを受け付けるが、 プレインテキストや他のタイプよりは HTML を好む、 他のメディアタイプよりは GIF や JPEG を好む、しかし最終手段として 他のメディアタイプも受け付ける、と設定されています。

```
Accept-Language: fr; q=1.0, en; q=0.5
Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6, image/jpeg; q=0.6,
image/*; q=0.5, */*; q=0.1
```

Apache は HTTP/1.1 規格で定義されている 'server driven' コンテントネゴシエーションをサポートしています。 Accept, Accept-Language, Accept-Charset, Accept-Encoding リ

クエストヘッダを完全にサポートしています。Apache は 'transparent' コンテントネゴシエーションもサポートしていますが、 これは RFC 2295 と RFC 2296 で定義されている試験的な ネゴシエーションプロトコルです。 これらの RFCで定義されている 'feature negotiation' はサポートしていません。

リソースとは URI で特定される概念上のもののことです (RFC 2396)。 Apache のような HTTP サーバは、その名前空間の中での リソースの表現へのアクセスを提供します。 それぞれの表現は 定義されたメディアタイプ、文字セット、エンコーディング等の 付属した、バイト列の形式です。 それぞれのリソースはある時点で 0 個、1 個、それ以上の表現と関連付けられる可能性があります。複数の表現が利用できる場合は、 リソースはネゴシエーション可能であるとされ、 個々の表現は variant と呼ばれます。 ネゴシエーション可能なリソースの variant が異なる、 その状態を指して、 ネゴシエーションの次元と呼びます。

Apache におけるネゴシエーション

リソースをネゴシエーションするためには、 サーバは variant それぞれについての情報を知っておく必要があります。 これは以下の二つの方法のどちらかで行われます。

- タイプマップ (すなわち *.var ファイル) を使う方法。 これは variant を明示的に挙げているファイルを指定します。
- 'Multiviews' を使って、サーバが暗黙の内にファイル名にパターン照合を 行なってその結果から選択する方法。

type-map ファイルを使う

タイプマップは type-map ハンドラ (もしくは、古い Apache の設定と下位互換である mime タイプ application/x-type-map) に関連付けられたドキュメントです。 この機能を使うためには、あるファイルの拡張子を type-map として定義するようなハンドラを、 設定ファイル中に置く必要があることに注意してください。 これは

```
AddHandler type-map .var
```

をサーバ設定ファイル中に書くことが一番良い方法です。

タイプマップファイルは記述するリソースと同じ名前を持っていて、 利用可能な variant それぞれのエントリを持っている必要があります。 そして、このエントリは連続した HTTP のヘッダ行で構成されます。 異なる variant のためのエントリは空行で区切られています。 エントリ中に空行が複数あってはいけません。 習慣的には、マップファイルは全体を結合したもののエントリから始まります (しかしこれは必須ではなく、あったとしても無視されるものです)。 次に例を示します。このファイルはリソース foo を記述しているので、foo.var という名前になります。

```
URI: foo

URI: foo.en.html
Content-type: text/html
Content-language: en

```

```
URI: foo.fr.de.html
Content-type: text/html;charset=iso-8859-2
Content-language: fr, de
```

たとえ MultiViews を使用するようになっていたとしても、 ファイル名の拡張子よりタイプマップの方が優先権を持つということにも 注意してください。 variant の品質が違うときは、この画像のように (JPEG, GIF, ASCII アートがあります) メディアタイプの "qs" パラメータで指定されます。

```
URI: foo

URI: foo.jpeg
Content-type: image/jpeg; qs=0.8

URI: foo.gif
Content-type: image/gif; qs=0.5

URI: foo.txt
Content-type: text/plain; qs=0.01
```

qs 値の範囲は 0.000 から 1.000 です。qs 値が 0.000 の variant は決して 選択されないことに注意してください。'qs' 値のない variant は qs 値 1.0 を 与えられます。qs パラメータはクライアントの能力に関係無く、他の variant と 比較したときの variant の相対的な「品質」を示します。 例えば、写真を表現しようとしているときは JPEG ファイルの方が普通は ASCII ファイルよりも高い品質になります。しかし、リソースが元々 ASCII アートで表現されているときは、ASCII ファイルの 方が JPEG ファイルよりも高い品質になります。このように、qs は 表現されるリソースの性質によって variant 毎に特有の値を取ります。

認識されるヘッダの一覧は mod_negotiation [1] ドキュメントにあります。

Multiviews

MultiViews はディレクトリ毎のオプションで、 <Directory>, <Location>, <Files> や、(AllowOverride が適切な値に 設定されていると) .htaccess ファイルで Options ディレクティブによって設定することができます。 Options All は MultiViews をセットしないことに注意してください。明示的に その名前を書く必要があります。

MultiViews の効果は以下のようになります: サーバが /some/dir/foo へのリクエストを受け取り、/some/dir で MultiViews が有効であって、 /some/dir/foo が存在しない場合、 サーバはディレクトリを読んで foo.* にあてはまる全てのファイルを探し、 事実上それらのファイルをマップするタイプマップを作ります。 そのとき、メディアタイプとコンテントエンコーディングは、そのファイル名を 直接指定したときと同じものが割り当てられます。それからクライアントの要求に一番合うものを選びます。

サーバがディレクトリの索引を作ろうとしている場合、 MultiViews は DirectoryIndex ディレクティブで指定されたファイルを探す過程にも 適用されます。設定ファイルに

```
DirectoryIndex index
```

が書かれていて、index.html と index.html3 が 両方存在していると、サーバはその中からどちらかを適当に選びます。 もしその両方が存在せずに index.cgi が存在していると、 サーバはそれを実行します。

もしディレクトリを読んでいる際に、 文字セット、コンテントタイプ、言語、エンコーディングを 指定するための mod_mime で認識できる拡張子を持たないファイルが見つかると、結果は MultiViewsMatch ディレクティブの設定に依存します。このディレクティブは ハンドラ、フィルタ、他のファイル拡張子タイプのどれが MultiViews ネゴシエーションで使用できるかを決定します。

ネゴシエーション方法

Apache はリソースの variant の一覧を、タイプマップファイルか ディレクトリ内のファイル名からかで取得した後、 「最適な」 variant を決定するために二つの方法の どちらかを起動します。 Apache のコンテントネゴシエーションの機能を使うために、 どのようにしてこの調停が行われるか詳細を知る必要はありません。 しかしながら、この文書の残りでは関心のある人のために、 使用されている方法について説明しています。

ネゴシエーション方法は二つあります。

1. 通常は Apache のアルゴリズムを用いた Server driven negotiation が使用されます。Apache のアルゴリズムは後に詳細に説明されています。 このアルゴリズムが使用された場合、Apache はより良い結果になるように、特定の次元において品質の値を 「変える」ことができます。Apache が品質の値を変える方法は後で詳細に説明されています。
2. RFC 2295 で定義されている機構を用いてブラウザが特に指定した場合、 transparent content negotiation が使用されます。このネゴシエーション方法では、「最適な」 variant の決定をブラウザが完全に制御することができます。 ですから、結果はブラウザが使用しているアルゴリズムに依存します。 Transparent negotiation の処理の過程で、ブラウザは RFC 2296 で 定義されている 'remote variant selection algorithm' を実行するように頼むことができます。

ネゴシエーションの次元

次元	説明
メディアタイプ	ブラウザは Accept ヘッダフィールドで優先傾向を指定します。 アイテムそれぞれは、関連した品質数値を持つことができます。 variant の説明も品質数値を持つことができます ("qs" パラメータをご覧下さい)。
言語	ブラウザは Accept-Language ヘッダフィールドで優先傾向を指定します。 要素それぞれに品質数値を持たせることができます。 variants は 0 か 1 つかそれ以上の言語と 関連づけることができます。
エンコーディング	ブラウザは Accept-Encoding ヘッダフィールドで優先傾向を指定します。 要素それぞれに品質数値を持たせることができます。
文字セット	ブラウザは Accept-Charset ヘッダフィールドで優先傾向を指定します。 要素それぞれに品質数値を持たせることができます。 variant はメディアタイプのパラメータとして文字セットを 指定することもできます。

Apache ネゴシエーションアルゴリズム

ブラウザに返す「最適な」variant を（もしあれば）選択するように Apache は次のアルゴリズムを使うことができます。 このアルゴリズムを設定により変更することはできません。 次のように動作します:

1. まずはじめに、ネゴシエーションの次元それぞれについて適切な Accept* ヘッダフィールドを調べ、 variant それぞれに品質を割り当てます。 もしある次元の Accept* ヘッダでその variant が許容できないことが示されていれば、それを削除します。 variant が一つも残っていなければ、ステップ 4 に行きます。
2. 消去法で「最適な」 variant を選びます。 次のテストが順番に適用されます。 テストで選択されなかった variant は削除されていきます。 テスト後 variant が一つだけ残っていれば、それを最適なものとして ステップ 3 に進みます。 複数 variant が残っていれば、次のテストに進みます。
    1. variant のメディアタイプの品質数値と Accept ヘッダの品質数値との積を計算して、最高値の variant を選びます。
    2. 言語品質数値が最高の variant を選びます。
    3. （もしあれば） Accept-Language ヘッダの言語順か、（もしあれば） LanguagePriority ディレクティブの言語順で最適な言語の variant を選びます。
    4. 最高「レベル」のメディアパラメータ（text/html メディアタイプのバージョンを与えるために使われます）を持つ variant を選びます。
    5. Accept-Charset ヘッダ行で与えられている最高の文字セット メディアパラメータを持つ variant を選びます。 明示的に除外されていない限り、ISO-8859-1 が許容されるようになっています。 text/* メディアタイプであるけれども 特定の文字セットに明示的に関連づけられているわけではない variant は ISO-8859-1 であると仮定されます。
    6. ISO-8859-1 ではない文字セットメディアパラメータと 関連づけられている variant を選びます。 そのような variant がない場合は、代わりに全ての variant を選びます。
    7. 最適なエンコーディングの variant を選びます。 もし user-agent が許容するエンコーディングがあれば、 その variant のみを選びます。 そうではなく、もしエンコードされたものとそうでない variant が混ざって存在していたらエンコードされていない variant のみを選びます。 variant が全部エンコードされているか variant が全部エンコードされていないという場合は、 全ての variant を選びます。
    8. 内容の最も短い variant を選びます。
    9. 残っている variant の最初のものを選びます。 タイプマップファイルの最初にリストされているか、 variant がディレクトリから最初に読み込まれる時にASCII順でソートしてファイル名が先頭になったか、のどちらかです。
3. アルゴリズムを使って一つの「最適な」variant を選びましたので、 それを応答として返します。ネゴシエーションの次元を指定するために HTTP レスポンスヘッダ Vary が設定されます（リソースのキャッシュをする時に、 ブラウザやキャッシュはこの情報を使うことができます)。 以上で終わり。
4. ここに来たということは、variant が一つも選択されなかった（ブラウザが許容するものがなかったため）ということです。 406 ステータス（"No Acceptable representation" を意味する）が、利用可能な variant のリストのついた HTML ドキュメントとともに返されます。 相違の次元を示す HTTP Vary ヘッダも設定されます。

品質の値を変える

上記の Apaceh ネゴシエーションアルゴリズムの厳格な解釈で 得られるであろう値から、Apache は品質数値を時々変えます。 これは、このアルゴリズムで完全ではない、あるいは正確でない情報を送る ブラウザ向けによりよい結果を得るために行われます。 かなりポピュラーなブラウザで、もしないと間違った variant を選択する結果になってしまうような Accept ヘッダ情報を送るものもあります。 ブラウザが完全で正しい情報を送っていれば、この数値変化は適用されません。

メディアタイプとワイルドカード

Accept: リクエストヘッダはメディアタイプの優先傾向を指定します。 これはまた、"image/*" や "*/*" といった「ワイルドカード」メディアタイプを含むことができます。ここで * は任意の文字列にマッチします。 ですから、次の:

```
Accept: image/*, */*
```

を含むリクエストは、"image/" ではじまるタイプ全てが許容できる、 そして他のどんなタイプも許容できる（この場合はじめの "image/*" は冗長になります）ことを示します。 扱うことのできる明示的なタイプに加えて、機械的に ワイルドカードを送るブラウザもあります。例えば:

```
Accept: text/html, text/plain, image/gif, image/jpeg, */*
```

こうすることの狙いは、明示的にリストしているタイプが優先されるけれども、 異なる表現が利用可能であればそれでも良い、ということです。 しかしながら、上の基本的なアルゴリズムでは、 */* ワイルドカードは他の全てのタイプと全く同等なので優先されません。 ブラウザは */* にもっと低い品質（優先）値を付けてリクエストを送るべきなのです。例えば:

```
Accept: text/html, text/plain, image/gif, image/jpeg, */*; q=0.01
```

明示的なタイプには品質数値が付けられていませんので、 デフォルトの 1.0（最高値）の優先になります。 ワイルドカード */* は低い優先度 0.01 を与えられているので、 明示的にリストされているタイプに合致する variant がない場合にのみ、 他のタイプが返されます。

もし Accept: ヘッダが q 値を全く含んでいなければ、 望みの挙動をするために、 Apache は "*/*" があれば 0.01 の q 値を設定します。 また、"type/*" の形のワイルドカードには 0.02 の q 値を設定します（ですからこれらは "*/*" のマッチよりも優先されます）。もし Accept: ヘッダ中のメディアタイプのどれかが q 値を含んでいれば、これらの特殊な値は適応されず、 正しい情報を送るブラウザからのリクエストは期待通りに 動作するようになります。

言語ネゴシエーションの例外処理

Apache 2.0 では新たに、言語ネゴシエーションが適合するものを 見つけるのに失敗した時に、優雅にフォールバックできるような ネゴシエーションアルゴリズムが幾つか追加され

ました。

サーバのページをクライアントがリクエストしたけれども、 ブラウザの送ってきた Accept-Language に合致するページが一つも 見つからなかった場合に、サーバは "No Acceptable Variant" か "Multiple Choices" レスポンスをクライアントに返します。 これらのエラーメッセージを返さないように、 このような場合には Apache が Accept-Language を無視して、 クライアントのリクエストに明示的には合致しないドキュメントを 提供するように設定できます。 ForceLanguagePriority ディレクティブは、これらのエラーの一つか両方をオーバーライドするために 使用できて、 LanguagePriority ディレクティブの内容を使ってサーバの判断を代行するようにできます。

サーバは他に適合するものが見つからなければ、 言語サブセットで適合するものを試そうともします。 例えばクライアントが英国英語である en-GB 言語で ドキュメントをリクエストした場合、サーバは HTTP/1.1 規格では、単に en とマークされているドキュメントをマッチするものとすることは通常は許されていません。 (英国英語は理解できるけど一般的な英語は理解できないという読み手は 考えられないので、Accept-Language ヘッダで en-GB を含んで en を含まないのはほぼ確実に設定の間違いである、 ということに注意してください。 ですが不幸なことに、多くのクライアントではデフォルトで このような設定になっています。) しかしながら、他の言語にはマッチせず、"No Acceptable Variants" エラーを返したり、 LanguagePriority にフォールバックしようとしているときは、 サブセット指定を無視して、en-GB を en にマッチします。 Apache はクライアントの許容言語リストに暗黙に 非常に低い品質値の親言語を加えることになります。 しかし、クライアントが "en-GB; qs=0.9, fr; qs=0.8" とリクエストして、 サーバが "en" と "fr" と設計されたドキュメントを持っている場合は、 "fr" ドキュメントが返されることに注意してください。 このような処理は、HTTP 1.1 規格との整合性を維持して、 適切に設定されたクライアントともきちんと動作するために 必要です。

Transparent Content Negotiation の拡張

Apache は transparent content negotiation プロトコル (RFC 2295) を次のように拡張しています。 特定のコンテントエンコーディングのみが利用可能である variant に印を付けるために、新たに {encoding ..} 要素を variant リスト中に使っています。 リスト中のエンコードされた variant を認識し、 Accept-Encoding リクエストヘッダに従って許容される エンコードをもった variant は、どれでも候補 variant として使用するように、RVSA/1.0 アルゴリズム (RFC 2296) の実装が拡張されました。 RVSA/1.0 の実装では、最適な variant が見つかるまで、 計算した品質数値は小数点以下 5 桁まで丸めません。

リンクと名前の変換に関する注意点

言語ネゴシエーションを使っている場合は、 ファイルが一つ以上の拡張子を持てて、 拡張子の順番は通常は考慮されない (詳細は mod_mime [2] を参照) ので、 幾つかの異なる名前の変換を選べることになります。

典型的なファイルでは、MIME タイプ拡張子 (例えば html) を持っていて、エンコーディング拡張子 (例えば gz) を持っているかもしれなくて、 このファイルに異なる言語 variant を用意していれば、 もちろん言語拡張子 (例えば en) を持っているでしょう。

例:

- foo.en.html
- foo.html.en
- foo.en.html.gz

ファイル名と、それに対して使えるリンクと使えないリンクの例です:

ファイル名	使えるリンク	使えないリンク
foo.html.en	foo foo.html	-
foo.en.html	foo	foo.html
foo.html.en.gz	foo foo.html	foo.gz foo.html.gz
foo.en.html.gz	foo	foo.html foo.html.gz foo.gz
foo.gz.html.en	foo foo.gz foo.gz.html	foo.html
foo.html.gz.en	foo foo.html foo.html.gz	foo.gz

上の表を見て、拡張子なしのリンク（例えば foo）がいつでも使えることに気が付くでしょう。 この利点は、ドキュメントとして応答するファイルの 実際のファイルタイプを隠蔽して、リンクの参照を変更することなく 後からファイルを変更できる、 例えば html から shtml に、あるいは cgi に変更できる点です。

リンクに MIME タイプを使い続けたい（例えば foo.html)時は、言語拡張子は（エンコーディング拡張子もあればそれも含めて）MIME タイプ拡張子の右側になければなりません（例えば foo.html.en)。

キャッシュに関する注意事項

キャッシュが一つの表現を保存しているときは、 リクエスト URL と関連づけられています。 次にその URL がリクエストされた時に、キャッシュは 保存されている表現を使用できます。しかし、 リソースがサーバでネゴシエーション可能であれば、 最初のリクエストでキャッシュされて続くキャッシュヒットでは 間違った応答を返してしまうということになりかねません。 これを防ぐために、Apache はコンテントネゴシエーションの 後に返された応答全てに、HTTP/1.0 クライアントでは キャッシュ不可能の印をつけます。 また、ネゴシエーションされた応答のキャッシュを可能にする HTTP/1.1 プロトコルの機能も Apache はサポートします。

HTTP/1.0 準拠のクライアントからのリクエストに対しては、（ブラウザであろうとキャッシュであろうと）ネゴシエーションを受けた応答のキャッシュを許すために、CacheNegotiatedDocs ディレクティブを使用できます。 このディレクティブは、サーバ設定ファイルやバーチャルホストに書くことができ、 引数をとりません。 HTTP/1.1 クライアントからのリクエストには効力を持ちません。

追加情報

コンテントネゴシエーションに関する追加情報は、 Alan J. Flavell さんのLanguage Negotiation Notes[3] をご覧下さい。ですが、 Apache 2.0 での変更点を含むためには更新されていないかもしれない ということに注意してください。

URI References

[1] http://httpd.apache.org/docs-2.1/mod/mod_negotiation.html#typemaps
[2] http://httpd.apache.org/docs-2.1/mod/mod_mime.html#multipleext
[3] http://ppewww.ph.gla.ac.uk/~flavell/www/lang-neg.html