NAME

MIME::Types − Definition of MIME types

INHERITANCE

```
 MIME::Types
   is a Exporter
```

SYNOPSIS

```
 use MIME::Types;
 my $mt    = MIME::Types−>new(...);    # MIME::Types object
 my $type  = $mt−>type('text/plain');  # MIME::Type  object
 my $type  = $mt−>mimeTypeOf('gif');
 my $type  = $mt−>mimeTypeOf('picture.jpg');
 my @types = $mt−>httpAccept('text/html, application/json;q=0.1')
```

DESCRIPTION

MIME types are used in many applications (for instance as part of e−mail and HTTP traffic) to indicate the type of content which is transmitted. or expected. See RFC2045 at *https://www.ietf.org/rfc/rfc2045.txt*

Sometimes detailed knowledge about a mime-type is need, however this module only knows about the filename extensions which relate to some filetype. It can also be used to produce the right format: types which are not registered at IANA need to use ’x−’ prefixes.

This object administers a huge list of known mime-types, combined from various sources. For instance, it contains **all IANA** types and the knowledge of Apache. Probably the most complete table on the net!

MIME::Types and daemons (fork)

If your program uses fork (usually for a daemon), then you want to have the type table initialized before you start forking. So, first call

```
 my $mt = MIME::Types−>new;
```

Later, each time you create this object (you may, of course, also reuse the object you create here) you will get access to **the same global table** of types.

METHODS

Constructors

MIME::Types−>**new**(%options)

Create a new `MIME::Types` object which manages the data. In the current implementation, it does not matter whether you create this object often within your program, but in the future this may change.

```
 −Option         −−Default
  db_file          <installed source>
  only_complete    <false>
  only_iana        <false>
  skip_extensions  <false>
```

db_file => FILENAME

The location of the database which contains the type information. Only the first instantiation of this object will have this parameter obeyed.

[2.10] This parameter can be globally overruled via the `PERL_MIME_TYPE_DB` environment variable, which may be needed in case of PAR or other tricky installations. For PAR, you probably set this environment variable to ‘‘inc/lib/MIME/types.db’’

only_complete => BOOLEAN

Only include complete MIME type definitions: requires at least one known extension. This will reduce the number of entries −−and with that the amount of memory consumed— considerably.

In your program you have to decide: the first time that you call the creator (`new`) determines whether you get the full or the partial information.

only_iana => BOOLEAN
Only load the types which are currently known by IANA.

skip_extensions => BOOLEAN
Do not load the table to map extensions to types, which is quite large.

Knowledge

`$obj->`**addType**($type, ...)
Add one or more TYPEs to the set of known types. Each TYPE is a `MIME::Type` which must be experimental: either the main-type or the sub-type must start with `x-`.

Please inform the maintainer of this module when registered types are missing. Before version MIME::Types version 1.14, a warning was produced when an unknown IANA type was added. This has been removed, because some people need that to get their application to work locally... broken applications...

`$obj->`**extensions**()
Returns a list of all defined extensions.

`$obj->`**listTypes**()
Returns a list of all defined mime-types by name only. This will**not** instantiate MIME::T ype objects. See **types()**

`$obj->`**mimeTypeOf**($filename)
Returns the `MIME::Type` object which belongs to the FILENAME (or simply its filename extension) or `undef` if the file type is unknown. The extension is used and considered case-insensitive.

In some cases, more than one type is known for a certain filename extension. In that case, the preferred one is taken (for an unclear definition of preference)

example: use of **mimeTypeOf()**

```
 my $types = MIME::Types->new;
 my $mime = $types->mimeTypeOf('gif');

 my $mime = $types->mimeTypeOf('picture.jpg');
 print $mime->isBinary;
```

`$obj->`**type**($string)
Returns the `MIME::Type` which describes the type related to STRING. [2.00] Only one type will be returned.

[before 2.00] One type may be described more than once. Different extensions may be in use for this type, and different operating systems may cause more than one `MIME::Type` object to be defined. In scalar context, only the first is returned.

`$obj->`**types**()
Returns a list of all defined mime-types. For reasons of backwards compatibility, this will instantiate MIME::Type objects, which will be returned. See **listTypes()**.

HTTP support

`$obj->`**httpAccept**($header)
[2.07] Decompose a typical HTTP-Accept header, and sort it based on the included priority information. Returned is a sorted list of type names, where the highest priority type is first. The list may contain '*/*' (accept any) or a '*' as subtype.

Ill-formated typenames are ignored. On equal qualities, the order is kept. See RFC2616 section 14.1

example:

```
  my @types = $types->httpAccept('text/html, application/json;q=0.9');
```

`$obj`->**httpAcceptBest**($accept|\@types, `@have`)

[2.07] The `$accept` string is processed via **httpAccept()** to order the types on preference. You may also provide a list of ordered `@types` which may have been the result of that method, called earlier.

As second parameter, you pass a LIST of types you `@have` to offer. Those need to be MIME::Type objects. The preferred type will get selected. When none of these are accepted by the client, this will return `undef`. It should result in a 406 server response.

example:

```
   my $accept = $req->header('Accept');
   my @have   = map $mt->type($_), qw[text/plain text/html];
   my @ext    = $mt->httpAcceptBest($accept, @have);
```

`$obj`->**httpAcceptSelect**($accept|\@types, `@filenames`|\@filenames)

[2.07] Like **httpAcceptBest()**, but now we do not return a pair with mime-type and filename, not just the type. If `$accept` is `undef`, the first filename is returned.

example:

```
   use HTTP::Status ':constants';
   use File::Glob   'bsd_glob';    # understands blanks in filename

   my @filenames   = bsd_glob "$imagedir/$fnbase.*;
   my $accept      = $req->header('Accept');
   my ($fn, $mime) = $mt->httpAcceptSelect($accept, @filenames);
   my $code        = defined $mime ? HTTP_NOT_ACCEPTABLE : HTTP_OK;
```

FUNCTIONS

The next functions are provided for backward compatibility with MIME::Types versions [0.06] and below. This code originates from Jeff Okamoto *okamoto@corp.hp.com* and others.

by_mediatype(TYPE)

This function takes a media type and returns a list or anonymous array of anonymous three-element arrays whose values are the file name suffix used to identify it, the media type, and a content encoding.

TYPE can be a full type name (contains '/', and will be matched in full), a partial type (which is used as regular expression) or a real regular expression.

by_suffix(FILENAME|SUFFIX)

Like `mimeTypeOf`, but does not return an `MIME::Type` object. If the file +type is unknown, both the returned media type and encoding are empty strings.

example: use of function **by_suffix()**

```
  use MIME::Types 'by_suffix';
  my ($mediatype, $encoding) = by_suffix('image.gif');

  my $refdata = by_suffix('image.gif');
  my ($mediatype, $encoding) = @$refdata;
```

import_mime_types()

This method has been removed: mime-types are only useful if understood by many parties. Therefore, the IANA assigns names which can be used. In the table kept by this `MIME::Types` module all these names, plus the most often used temporary names are kept. When names seem to be missing, please contact the maintainer for inclusion.

SEE ALSO

This module is part of MIME-Types distribution version 2.22, built on October 27, 2021. Website: *http://perl.overmeer.net/CPAN/*

LICENSE