NAME

MIME::Type − description of one MIME type

SYNOPSIS

```
use MIME::Types;
my $mimetypes = MIME::Types->new;
my MIME::Type $plaintext = $mimetypes->type('text/plain');
print $plaintext->mediaType;   # text
print $plaintext->subType;     # plain

my @ext = $plaintext->extensions;
print "@ext"                   # txt asc c cc h hh cpp

print $plaintext->encoding     # 8bit
if($plaintext->isBinary)       # false
if($plaintext->isAscii)        # true
if($plaintext->equals('text/plain') {...}
if($plaintext eq 'text/plain') # same

print MIME::Type->simplified('x-appl/x-zip') #  'appl/zip'
```

DESCRIPTION

MIME types are used in MIME entities, for instance as part of e−mail and HTTP traffic. Sometimes real knowledge about a mime-type is need. Objects of `MIME::Type` store the information on one such type.

OVERLOADED

overload: **string comparison**

When a MIME::Type object is compared to either a string or another MIME::Type, the **equals()** method is called. Comparison is smart, which means that it extends common string comparison with some features which are defined in the related RFCs.

overload: **stringification**

The stringification (use of the object in a place where a string is required) will result in the type name, the same as **type()** returns.

example: use of stringification

```
my $mime = MIME::Type->new('text/html');
print "$mime\n";   # explicit stringification
print $mime;       # implicit stringification
```

METHODS

Initiation

MIME::Type−>**new**(%options)

Create (*instantiate*) a new MIME::Type object which manages one mime type.

```
−Option    −−Default
 encoding    <depends on type>
 extensions  []
 simplified  <derived from type>
 system      undef
 type        <required>
```

encoding => ’7bit’|’8bit’|’base64’|’quoted−printable’

How must this data be encoded to be transported safely. The default depends on the type: mimes with as main type `text/` will default to `quoted−printable` and all other to `base64`.

extensions => REF-ARRAY

An array of extensions which are using this mime.

simplified => STRING
The mime types main− and sub-label can both start with `x−`, to indicate that is a non-registered name. Of course, after registration this flag can disappear which adds to the confusion. The simplified string has the `x−` thingies removed and are translated to lower-case.

system => REGEX
Regular expression which defines for which systems this rule is valid. The REGEX is matched on `$^O`.

type => STRING
The type which is defined here. It consists of a *type* and a *sub-type*, both case-insensitive. This module will return lower-case, but accept upper-case.

Attributes

`$obj`−>**encoding**()
Returns the type of encoding which is required to transport data of this type safely.

`$obj`−>**extensions**()
Returns a list of extensions which are known to be used for this mime type.

`$obj`−>**simplified**([$string])
MIME::Type−>**simplified**([$string])
Returns the simplified mime type for this object or the specified STRING. Mime type names can get officially registered. Until then, they have to carry an `x−` preamble to indicate that. Of course, after recognition, the `x−` can disappear. In many cases, we prefer the simplified version of the type.

example: results of **simplified()**

```
my $mime = MIME::Type->new(type => 'x-appl/x-zip');
print $mime->simplified;                    # 'appl/zip'

print $mime->simplified('text/PLAIN');      # 'text/plain'
print MIME::Type->simplified('x-xyz/x-abc'); # 'xyz/abc'
```

`$obj`−>**system**()
Returns the regular expression which can be used to determine whether this type is active on the system where you are working on.

`$obj`−>**type**()
Returns the long type of this object, for instance `'text/plain'`

Knowledge

`$obj`−>**equals**($string|$mime)
Compare this mime-type object with a STRING or other object. In case of a STRING, simplification will take place.

`$obj`−>**isAscii**()
Old name for **isText()**.

`$obj`−>**isBinary**()
Returns true when the type is not known to be text. See**isT ext()**.

`$obj`−>**isExperimental**()
[2.00] Return `true` when the type is defined for experimental use; the subtype starts with `x.`

`$obj`−>**isPersonal**()
[2.00] Return `true` when the type is defined by a person for private use; the subtype starts with `prs.`

`$obj`−>**isRegistered**()
Mime-types which are not registered by IANA nor defined in RFCs shall start with an `x−`. This counts for as well the media-type as the sub-type. In case either one of the types starts with `x−` this method will return false.

`$obj->`**isSignature**()
: Returns true when the type is in the list of known signatures.

`$obj->`**isText**()
: [2.05] All types which may have the charset attribute, are text. However, there is currently no record of attributes in this module... so we guess.

`$obj->`**isVendor**()
: [2.00] Return `true` when the type is defined by a vendor; the subtype starts with `vnd.`

`$obj->`**mediaType**()
: The media type of the simplified mime. For `'text/plain'` it will return `'text'`.

  For historical reasons, the `'mainType'` method still can be used to retrieve the same value. However, that method is deprecated.

`$obj->`**subType**()
: The sub type of the simplified mime. For `'text/plain'` it will return `'plain'`.

DIAGNOSTICS

Error: Type parameter is obligatory.
: When a MIME::Type object is created, the type itself must be specified with the `type` option flag.

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