NAME

MojoX::MIME::Types − MIME Types for Mojolicious

INHERITANCE

```
MojoX::MIME::Types
  is a Mojo::Base
```

SYNOPSIS

```
use MojoX::MIME::Types;

# set in Mojolicious as default
$app->types(MojoX::MIME::Types->new);
app->types(MojoX::MIME::Types->new);   # ::Lite

# basic interface translated into pure MIME::Types
$types->type(foo => 'text/foo');
say $types->type('foo');
```

DESCRIPTION

[Added to MIME::Types 2.07] This module is a drop-in replacement for Mojolicious::Types, but with a more correct handling plus a complete list of types... a huge list of types.

Some methods ignore information they receive: those parameters are accepted for compatibility with the Mojolicious::Types interface, but should not contain useful information.

Read the ''DETAILS'' below, about how to connect this module into Mojolicious and the differences you get.

METHODS

Constructors

MojoX::MIME::Types->**new**(%options)

Create the 'type' handler for Mojolicious. When you do not specify your own MIME::Type object ($mime_type), it will be instantanted for you. You create one yourself when you would like to pass some parameter to the object constructor.

```
-Option    --Default
 mime_types  <created internally>
 types       undef
```

mime_types => MIME::Types−object

Pass your own prepared MIME::Types object, when you need some instantiation parameters different from the defaults.

types => HASH

Ignored.

example:

```
$app->types(MojoX::MIME::Types->new);

# when you need to pass options to MIME::Types->new
my $mt    = MIME::Types->new(%opts);
my $types = MojoX::MIME::Types->new(mime_types => $mt);
$app->types($types);
```

Attributes

`$obj`->**mapping**([\%table])

In Mojolicious::Types, this attribute exposes the internal administration of types, offering to change it with using a clean abstract interface. That interface mistake bites now we have more complex internals.

Avoid this method! The returned HASH is e xpensive to construct, changes passed via `%table` are

ignored: MIME::Types is very complete!

`$obj->`**mimeTypes**()
Returns the internal mime types object.

Actions

`$obj->`**content_type**($controller, \%options)
Set a content type on the controller when not yet set. The `%options` contains `ext` or `file` specify an file extension or file name which is used to derive the content type. Added and marked EXPERIMENTAL in Mojo 7.94.

`$obj->`**detect**(`$accept`, [$prio])
Returns a list of filename extensions. The`$accept` header in HTTP can contain multiple types, with a priority indication ('q' attributes). The returned list contains a list with extensions, the extensions related to the highest priority type first. The `$prio`-flag is ignored. See **MIME::Types::httpAccept()**.

This **detect()** function is not the correct approach for the Accept header: the ''Accept'' may contain wildcards ('*') in types for globbing, which does not produce extensions. Better use **MIME::Types::httpAcceptBest()** or **MIME::Types::httpAcceptSelect()**.

example:

```
my $exts = $types->detect('application/json;q=9');
my $exts = $types->detect('text/html, application/json;q=9');
```

`$obj->`**file_type**($filename)
Return the mime type for a filename. Added and marked EXPERIMENTAL in Mojo 7.94.

`$obj->`**type**(`$ext`, [$type|\@types])
Returns the first type name for an extension `$ext`, unless you specify type names.

When a single `$type` or an ARRAY of `@types` are specified, the `$self` object is returned. Nothing is done with the provided info.

DETAILS

Why?

The Mojolicious::Types module has only very little knowledge about what is really needed to treat types correctly, and only contains a tiny list of extensions. MIME::Types tries to follow the standards very closely and contains all types found in various lists on internet.

How to use with Mojolicious

Start your Mojo application like this:

```
package MyApp;
use Mojo::Base 'Mojolicious';

sub startup {
   my $self = shift;
   ...
   $self->types(MojoX::MIME::Types->new);
}
```

If you have special options for **MIME::Types::new()**, then create your own MIME::Types object first:

```
my $mt    = MIME::Types->new(%opts);
my $types = MojoX::MIME::Types->new(mime_types => $mt);
$self->types($types);
```

In any case, you can reach the smart MIME::Types object later as

```
my $mt    = $app->types->mimeTypes;
my $mime  = $mt->mimeTypeOf($filename);
```

How to use with Mojolicious::Lite

The use in Mojolicious::Lite applications is only slightly different from above:

```
app->types(MojoX::MIME::Types->new);
my $types = app->types;
```

Differences with Mojolicious::Types

There are a few major difference with Mojolicious::Types:

- the tables maintained by MIME::Types are complete. So: there shouldn't be a need to add your own types, not via `types()`, not via `type()`. All attempts to add types are ignored; better remove them from your code.
- This plugin understands the experimental flag 'x-' in types and handles casing issues.
- Updates to the internal hash via **types()** are simply ignored, because it is expensive to implement (and won't add something new).
- The **detect()** is implemented in a compatible way, but does not understand wildcards ('*'). You should use **MIME::Types::httpAcceptBest()** or **MIME::Types::httpAcceptSelect()** to replace this broken function.

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