NAME

Date::Manip::Obj − Base class for Date::Manip objects

SYNOPSIS

The Date::Manip::Obj class is the base class used for the following Date::Manip classes:

Date::Manip::Base
Date::Manip::TZ
Date::Manip::Date
Date::Manip::Delta
Date::Manip::Recur

This module is not intended to be called directly and performs no useful function by itself. Instead, use the various derived classes which inherit from it.

DESCRIPTION

This module contains a set of methods used by all Date::Manip classes listed above.

You should be familiar with the Date::Manip::Objects and Date::Manip::Config documentation.

In the method descriptions below, Date::Manip::Date objects will usually be used as examples, but (unless otherwise stated), all of the classes listed above have the same methods, and work in the same fashion.

METHODS FOR CREATING OBJECTS

In the examples below, any variable named some variation of `$date` (`$date`, `$date1`, `$date2`, ...) is a Date::Manip::Date object. Similarly, `$delta`, `$recur`, `$tz`, and `$base` refer to objects in the appropriate class.

Any `$obj` variable refers to an object in any of the classes.

new

There are two ways to use the new method. They are:

```
$obj2  = new CLASS ($obj1,$string,@parse_opts,\@opts);
$obj2  = $obj1->new($string,@parse_opts,\@opts)
```

In both cases, all arguments are optional.

Both methods are used to create a new object of a given class. In the first case, **CLASS** is the class of the new object. For example:

```
$date  = new Date::Manip::Date;
$delta = new Date::Manip::Delta;
```

In the second method, the class of the new object will be derived from the first object. For example:

```
$date1 = new Date::Manip::Date;
$date2 = $date1->new();
```

the class of the second object (`$date2`) is Date::Manip::Date because that is the class of the object (`$date1`) used to create it.

In both first method (when a `$obj1` is passed in) and always in the second method, the new object will share as much information from the old object (`$obj1`) as possible.

For example, if you call either of these:

```
$date2 = new Date::Manip::Date $date1;
$date2 = $date1->new();
```

the new date object will use the same embedded Date::Manip::TZ and Date::Manip::Base objects.

When specifying CLASS and including an old object, objects do not need to be of the same class. For example, the following are all valid:

```
$date = new Date::Manip::Date $delta;
$date = new Date::Manip::Date $tz;
```

You can even do:

```
$date = new Date::Manip::Date $base;
```

but this will have to create a completely new Date::Manip::TZ object, which means that optimal performance may not be achieved if a Date::Manip::TZ object already exists.

There are two special cases. Either of the following will create a new Date::Manip::Base object for handling multiple configurations:

```
$base2 = new Date::Manip::Base $base1;
$base2 = $base1->new();
```

Either of the following will create a new Date::Manip::TZ object with the same Date::Manip::Base object embedded in it:

```
$tz2   = new Date::Manip::TZ $tz1;
$tz2   = $tz1->new();
```

The new base object will initially have the same configuration as the original base object, but changing it's configuration will not affect the original base object.

If the `\@opts` argument is passed in, it is a list reference containing a list suitable for passing to the **config** method (described below). In this case, a new Date::Manip::Base object (and perhaps Date::Manip::TZ object) will be created. The new Base object will start as identical to the original one (if a previously defined object was used to create the new object) with the additional options in `@opts` added.

In other words, the following are equivalent:

```
$date  = new Date::Manip::Date $obj,\@opts;

$base  = $obj->base();
$base2 = $base->new();
$date = new Date::Manip::Date $base2;
$date->config(@opts);
```

It should be noted that the options are applied to the NEW Date::Manip::Base object, not the old one.

An optional string (`$string` and parse opts `@parse_opts`) may be passed in only when creating a Date::Manip::Date, Date::Manip::Delta, or Date::Manip::Recur object. If passed in when creating a Date::Manip::TZ or Date::Manip::Base object, a warning will be issued, but execution will continue.

If the string is included, it will be parsed to give an initial value to the object. This will only be done AFTER any options are handled, so the following are equivalent:

```
$date = new Date::Manip::Date $string,@parse_opts,\@opts;

$date = new Date::Manip::Date;
$date->config(@opts);
$date->parse($string,@parse_opts);
```

Once a Date::Manip::Date object (or any object in any other Date::Manip class) is created, it should always be used to create additional objects in order to preserve cached data for optimal performance and memory usage.

The one caveat is if you are working with multiple configurations as described in the Date::Manip::Objects document. In that case, you may need to create completely new objects to allow multiple Date::Manip::Base objects to be used.

new_config

```
$obj2 = $obj1->new_config($string,\@opts);
```

This creates a new instance with a new Date::Manip::Base object (and possibly a new Date::Manip::TZ object).

For example,

```
$date2 = $date1->new_config();
```

creates a new Date::Manip::Date object with a new Date::Manip::TZ (and Date::Manip::Base) object. Initially, it is the same configuration as the original object.

If the object is a Date::Manip::Base object, the following are equivalent:

```
$base2 = $base1->new_config();

$base2 = $base1->new();
```

Both `$string` and `\@opts` are optional. They are used in the same way they are used in the new method.

new_date
new_delta
new_recur

These are shortcuts for specifying the class. The following sets of calls are all equivalent:

```
$date  = $obj->new_date();
$date  = new Date::Manip::Date($obj);

$delta = $obj->new_delta();
$delta = new Date::Manip::Date($obj);
```

These methods all allow optional `($string,\@opts)` arguments.

OTHER METHODS

base
tz

```
$base = $obj->base();
```

This returns the Date::Manip::Base object associated with the given object.

If `$obj` is a Date::Manip::Base object, nothing is returned (i.e. it doesn't create a new copy of the object).

```
$tz = $obj->tz();
```

This returns the Date::Manip::TZ object associated with the given object. If `$obj` is a Date::Manip::TZ or Date::Manip::Base object, nothing is returned.

config

```
$obj->config($var1,$val1,$var2,$val2,...);
```

This will set the value of any configuration variables. Please refer to the Date::Manip::Config manual for a list of all configuration variables and their description.

get_config

```
@var = $obj->get_config();
$val = $obj->get_config($var1);
@val = $obj->get_config($var1,$var2,...);
```

This queries the current config values. With no argument, it will return the list of config variables (all lowercase).

With one or more arguments, it returns the current values for the config variables passed in (case

insensitive).

err

```
$err = $obj->err();
```

This will return the full error message if the previous operation failed for any reason.

```
$obj->err(1);
```

will clear the error code.

is_date
is_delta
is_recur

```
$flag = $obj->is_date();
```

Returns 0 or 1, depending on the object. For example, a Date::Manip::Date object returns 1 with the is_date method, and 0 for the other two.

version

```
$vers = $obj->version($flag);
```

This returns the version of Date::Manip.

If `$flag` is passed in, and `$obj` is not a Date::Manip::Base object, the version and timezone information will be passed back.

KNOWN BUGS

None known.

BUGS AND QUESTIONS

Please refer to the Date::Manip::Problems documentation for information on submitting bug reports or questions to the author.

SEE ALSO

Date::Manip − main module documentation

LICENSE

AUTHOR

Sullivan Beck (sbeck@cpan.org)