

# **1 ModPerl::MM -- A "subclass" of ExtUtils::MakeMaker for mod\_perl 2.0**

## 1.1 Synopsis

```

use ModPerl::MM;

# ModPerl::MM takes care of doing all the dirty job of overriding
ModPerl::MM::WriteMakefile(...);

# if there is a need to extend the default methods
sub MY::constants {
    my $self = shift;
    $self->ModPerl::MM::MY::constants;
    # do something else;
}

# or prevent overriding completely
sub MY::constants { shift->MM::constants(@_); };

# override the default value of WriteMakefile's attribute
my $extra_inc = "/foo/include";
ModPerl::MM::WriteMakefile(
    ...
    INC => $extra_inc,
    ...
);

# extend the default value of WriteMakefile's attribute
my $extra_inc = "/foo/include";
ModPerl::MM::WriteMakefile(
    ...
    INC => join " ", $extra_inc, ModPerl::MM::get_def_opt('INC'),
    ...
);

```

## 1.2 Description

ModPerl::MM is a "subclass" of ExtUtils::MakeMaker for mod\_perl 2.0, to a degree of sub-classability of ExtUtils::MakeMaker.

When ModPerl::MM::WriteMakefile() is used instead of ExtUtils::MakeMaker::WriteMakefile(), ModPerl::MM overrides several ExtUtils::MakeMaker methods behind the scenes and supplies default WriteMakefile() arguments adjusted for mod\_perl 2.0 build. It's written in such a way so that normally 3rd party module developers for mod\_perl 2.0, don't need to mess with *Makefile.PL* at all.

## 1.3 MY::: Default Methods

ModPerl::MM overrides method *foo* as long as *Makefile.PL* hasn't already specified a method *MY::foo*. If the latter happens, ModPerl::MM will DWIM and do nothing.

In case the functionality of `ModPerl::MM` methods needs to be extended, rather than completely overridden, the `ModPerl::MM` methods can be called internally. For example if you need to modify constants in addition to the modifications applied by `ModPerl::MM::MY::constants`, call the `ModPerl::MM::MY::constants` method (notice that it resides in the package `ModPerl::MM::MY` and not `ModPerl::MM`), then do your extra manipulations on constants:

```
# if there is a need to extend the methods
sub MY::constants {
    my $self = shift;
    $self->ModPerl::MM::MY::constants;
    # do something else;
}
```

In certain cases a developers may want to prevent from `ModPerl::MM` to override certain methods. In that case an explicit override in *Makefile.PL* will do the job. For example if you don't want the `constants()` method to be overridden by `ModPerl::MM`, add to your *Makefile.PL*:

```
sub MY::constants { shift->MM::constants(@_); };
```

`ModPerl::MM` overrides the following methods:

### 1.3.1 *ModPerl::MM::MY::constants*

This method makes sure that everything gets installed relative to the `Apache2/` subdir if `MP_INST_APACHE2=1` was used to build `mod_perl 2.0`.

### 1.3.2 *ModPerl::MM::MY::post\_initialize*

This method makes sure that everything gets installed relative to the `Apache2/` subdir if `MP_INST_APACHE2=1` was used to build `mod_perl 2.0`.

## 1.4 *WriteMakefile()* Default Arguments

`ModPerl::MM::WriteMakefile` supplies default arguments such as `INC` and `TYPEMAPS` unless they weren't passed to `ModPerl::MM::WriteMakefile` from *Makefile.PL*.

If the default values aren't satisfying these should be overridden in *Makefile.PL*. For example to supply an empty `INC`, explicitly set the argument in *Makefile.PL*.

```
ModPerl::MM::WriteMakefile(
    ...
    INC => '',
    ...
);
```

If instead of fully overriding the default arguments, you want to extend or modify them, they can be retrieved using the `ModPerl::MM::get_def_opt()` function. The following example appends an extra value to the default `INC` attribute:

## 1.5 Public API

```
my $extra_inc = "/foo/include";
ModPerl::MM::WriteMakefile(
    ...
    INC => join " ", $extra_inc, ModPerl::MM::get_def_opt('INC'),
    ...
);
```

ModPerl::MM supplies default values for the following ModPerl::MM::WriteMakefile attributes:

### ***1.4.1 CCFLAGS***

### ***1.4.2 LIBS***

### ***1.4.3 INC***

### ***1.4.4 OPTIMIZE***

### ***1.4.5 LDDLFLAGS***

### ***1.4.6 TYPEMAPS***

### ***1.4.7 dynamic\_lib***

#### **1.4.7.1 OTHERLDFLAGS**

```
dynamic_lib => { OTHERLDFLAGS => ... }
```

### ***1.4.8 macro***

#### **1.4.8.1 MOD\_INSTALL**

```
macro => { MOD_INSTALL => ... }
```

arranges for modules to be installed under the subdir *Apache2/* if *mod\_perl* was built with *MP\_INST\_APACHE2=1*.

## **1.5 Public API**

The following functions are a part of the public API. They are described elsewhere in this document.

### ***1.5.1 WriteMakefile( )***

```
ModPerl::MM::WriteMakefile(...);
```

### ***1.5.2 get\_def\_opt( )***

```
my $def_val = ModPerl::MM::get_def_opt($key);
```



## Table of Contents:

1	ModPerl::MM -- A "subclass" of ExtUtils::MakeMaker for mod_perl 2.0 . . . . .	1
1.1	Synopsis . . . . .	2
1.2	Description . . . . .	2
1.3	MY:: Default Methods . . . . .	2
1.3.1	ModPerl::MM::MY::constants. . . . .	3
1.3.2	ModPerl::MM::MY::post_initialize. . . . .	3
1.4	WriteMakefile() Default Arguments . . . . .	3
1.4.1	CCFLAGS . . . . .	4
1.4.2	LIBS . . . . .	4
1.4.3	INC . . . . .	4
1.4.4	OPTIMIZE. . . . .	4
1.4.5	LDDLFLAGS . . . . .	4
1.4.6	TYPEMAPS. . . . .	4
1.4.7	dynamic_lib. . . . .	4
1.4.7.1	OTHERLDFLAGS . . . . .	4
1.4.8	macro. . . . .	4
1.4.8.1	MOD_INSTALL. . . . .	4
1.5	Public API . . . . .	4
1.5.1	WriteMakefile() . . . . .	5
1.5.2	get_def_opt() . . . . .	5