NAME

MooseX::App::Meta::Role::Class::Base − Meta class role for application base class

DESCRIPTION

This meta class role will automatically be applied to the application base class. This documentation is only of interest if you intend to write plugins for MooseX-App.

ACCESSORS

app_messageclass

Message class for generating error messages. Defaults to MooseX::App::Message::Block. The default can be overwritten by altering the `_build_app_messageclass` method. Defaults to MooseX::App::Message::Block

app_namespace

Usually MooseX::App will take the package name of the base class as the namespace for commands. This namespace can be changed.

app_base

Usually MooseX::App will take the name of the calling wrapper script to construct the program name in various help messages. This name can be changed via the app_base accessor. Defaults to the base name of `$0`

app_fuzzy

Boolean flag that controls if command names and attributes should be matched exactly or fuzzy. Defaults to true.

app_command_name

Coderef attribute that controls how package names are translated to command names and attributes. Defaults to &MooseX::App::Utils::class_to_command

app_commands

Hashref with command to command class map.

app_strict

Boolean flag that controls if an application with superfluous/unknown positional parameters should terminate with an error message or not. If disabled all extra parameters will be copied to the extra_argv command class attribute.

app_prefer_commandline

By default, arguments passed to new_with_command and new_with_options have a higher priority than the command line options. This boolean flag will give the command line an higher priority.

app_permute

Boolean flag that controls if command line arguments that take multiple values (ie ArrayRef or HashRef type constraints) can be permuted.

METHODS

command_check

Runs sanity checks on options and parameters. Will usually only be executed if either HARNESS_ACTIVE or APP_DEVELOPER environment are set.

command_register

```
$self->command_register($command_moniker,$command_class);
```

Registers an additional command

command_get

```
my $command_class = $self->command_register($command_moniker);
```

Returns a command class for the given command moniker

command_class_to_command

```
my $command_moniker = $meta->command_class_to_command($command_class);
```

Returns the command moniker for the given command class.

command_message

```
my $message = $meta->command_message(
   header  => $header,
   type    => 'error',
   body    => $message
);
```

Generates a message object (using the class from app_messageclass)

command_usage_attributes

```
my @attributes = $meta->command_usage_attributes($metaclass);
```

Returns a list of attributes/command options for the given meta class.

command_usage_command

```
my @messages = $meta->command_usage_command($command_metaclass);
```

Returns a list of messages containing the documentation for a given command meta class.

command_usage_description

```
my $message = $meta->command_usage_description($command_metaclass);
```

Returns a messages with the basic command description.

command_usage_global

```
my @messages = $meta->command_usage_global();
```

Returns a list of messages containing the documentation for the application.

command_usage_header

```
my $message = $meta->command_usage_header();
my $message = $meta->command_usage_header($command_meta_class);
```

Returns a message containing the basic usage documentation

command_find

```
my @commands = $meta->command_find($user_command_input);
```

Returns a list of command names matching the user input

command_candidates

```
my $commands = $meta->command_candidates($user_command_input);
```

Returns either a single command or an arrayref of possibly matching commands.

command_proto

```
my ($result,$errors) = $meta->command_proto($command_meta_class);
```

Returns all parsed options (as hashref) and erros (as arrayref) for the proto command. Is a wrapper around command_parse_options.

command_args

```
my ($options,$errors) = $self->command_args($command_meta_class);
```

Returns all parsed options (as hashref) and erros (as arrayref) for the main command. Is a wrapper around command_parse_options.

command_parse_options

```
my ($options,$errors) = $self->command_parse_options(\@attribute_metaclasses);
```

Tries to parse the selected attributes from `@ARGV`.

command_scan_namespace

```
my %namespaces = $self->command_scan_namespace($namespace);
```

Scans a namespace for command classes. Returns a hash with command names as keys and package names as values.

command_process_attribute

```
my @attributes = $self->command_process_attribute($attribute_metaclass,$matches)
```

TODO ###Returns a list of all attributes with the given type

command_usage_options

```
my $usage = $self->command_usage_options($metaclass,$headline);
```

Returns the options usage as a message object

command_usage_parameters

```
my $usage = $self->command_usage_parameters($metaclass,$headline);
```

Returns the positional parameters usage as a message object

command_check_attributes

```
$errors = $self->command_check_attributes($command_metaclass,$errors,$params)
```

Checks all attributes. Returns/alters the `$errors` arrayref

command_parser_hints

```
$self->command_parser_hints($self,$metaclass)
```

Generates parser hints as required by MooseX::App::ParsedArgv