NAME

MooseX::Types::VariantTable::Declare − Declarative sugar for MooseX::Types::VariantTable based methods.

SYNOPSIS

```
package Awesome;
use Moose;

variant_method dance => Item => sub {
    # Item is the least derived type in the hierarchy,
    # every other type subtypes it
    # this is in effect a fallback
    return "fallback";
};

# a more specific type
variant_method dance => Ballerina => sub {
    my ( $self, $ballerina, @args ) = @_;

    $ballerina; # a value that passed the TC named "Ballerina"

    return "pretty!";
};

# also works with objects
variant_method dance => $type_object => sub { ... };
```

DESCRIPTION

This module provides declarative sugar for defining Moose::Meta::Method::VariantTable methods in your Moose classes and roles.

These methods have some semantics:

Declaration

The order of the declarations do not matter in most cases.

It is the type hierarchy that defines the order in which the constraints are checked and items dispatched.

However, in the case that two constraints without an explicit relationship between them (one is a subtype of the other) both accept the same value, the one that was declared earlier will win. There is no way around this issue, so be careful what types you use especially when mixing variants form many different sources.

Adding the same type to a variant table twice is an error.

Inheritence

When dispatching all of the subclass's variants will be tried before the superclass.

This allows shadowing of types from the superclass even using broader types.

Roles

... are currently broken.

Don't use variant table methods in a role, unless that's the only definition, because in the future variant table merging will happen at role composition time in a role composition like way, so your code will not continue to work the same.