Attribute types in Perl classes when using Moose

OOP | Moose | object oriented | class | object | instance | constructor | getter | setter | accessor | attribute

Prev Next

In a simple Perl script we usually don't care much about the types of the values, but as the application grows, a type system can improve the correctness of the application.

Moose lets you define a type for each attribute and then enforces the types via the setters.

After the the first introduction to Object Oriented Perl with Moose, you should probably get familiar with the type-checking system of Moose.

Set the type to be Int

This is our first sample script:

```
use strict;
use warnings;
use v5.10;

use Person;

my $student = Person->new( name => 'Joe' );
$student->year(1988);
say $student->year;
$student->year('23 years ago');
```

After loading the Person module (the class), we create the $student object by calling the "new" constructor of the class. Then we call the "year" accessor and set the value to 1988. After printing the value we try to set it to "23 years ago".

In the module you can see two attributes. The "year" attribute has an `isa` entry with a value `Int`. Because of this, the setter Moose creates will restrict the values it accepts to integers.

```
package Person;
use Moose;

has 'name' => (is => 'rw');
has 'year' => (isa => 'Int', is => 'rw');

1;
```

Save the module in "somedir/lib/Person.pm" and save the script in "somedir/bin/app.pl" then from within the somedir directory run the script with "perl -Ilib bin/app.pl"

After printing the value 1988, we get the following error:

```
Attribute (year) does not pass the type constraint because:
   Validation failed for 'Int' with value "23 years ago"
       at accessor Person::year (defined at lib/Person.pm line 5) line 4
   Person::year('Person=HASH(0x19a4120)', '23 years ago')
       called at bin/app.pl line 13
```

This error message shows that Moose did not accept the string "23 years ago" as an Integer.

Another class as a type constraint

Besides the default type constraints, Moose also allows you to use the name of any existing class as a type constraint.

For example you can declare that the "birthday" attribute must be a DateTime object.

```
package Person;
use Moose;

has 'name'     => (is => 'rw');
has 'birthday' => (isa => 'DateTime', is => 'rw');

1;
```

Try the example script:

```
use strict;
use warnings;
use v5.10;

use Person;
use DateTime;
```

```
    my $student = Person->new( name => 'Joe' );
    $student->birthday( DateTime->new( year => 1988, month => 4, day => 17) );
10. say $student->birthday;
11. $student->birthday(1988);
```

You can see, the first call to the "birthday" setter receives a DateTime object created on the spot. This call works well. The second call receives the number 1988 and throws an exception similar to the previous one:

```
Attribute (birthday) does not pass the type constraint because:
    Validation failed for 'DateTime' with value 1988
       at accessor Person::birthday (defined at lib/Person.pm line 5) line 4
    Person::birthday('Person=HASH(0x2143928)', 1988)
       called at bin/app.pl line 14
```