

Tracing Code With Aspect.pm

Tom Hukins

Aspect Oriented Programming

Tracing Code

- `Devel::Trace`
- `perl -d`
- `Devel::ebug`

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

Pointcuts

```
package SocketDebug;
```

```
use Aspect;
```

```
# use the module before the code that pointcuts it  
use IO::Socket::INET;
```

```
my $pointcut = call qr{\AIO::Socket};
```

```
before {
```

```
    my $context = shift;          # See Aspect::AdviceContext
```

```
    warn 'Calling ', $context->sub_name, "\n";
```

```
} $pointcut;
```

```
1;
```

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

Pointcuts

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext
    warn 'Calling ', $context->sub_name, "\n";
} $pointcut;

1;
```

A Program

```
#!/usr/bin/perl
```

```
use strict;  
use warnings;
```

```
use LWP::Simple;
```

```
getprint 'http://another.dull.example.com/';
```

Pointcut Meets Program

```
% perl -Mlib=. -MSocketDebug lwp_get
Calling IO::Socket::INET::new
Calling IO::Socket::new
Calling IO::Socket::INET::configure
Calling IO::Socket::INET::_sock_info
Calling IO::Socket::INET::INADDR_ANY
Calling IO::Socket::INET::_sock_info
Calling IO::Socket::INET::_get_addr
Calling IO::Socket::INET::inet_aton
Calling IO::Socket::INET::AF_INET
Calling IO::Socket::socket
Calling IO::Socket::INET::INADDR_ANY
Calling IO::Socket::INET::SOCK_STREAM
Calling IO::Socket::INET::pack_sockaddr_in
Calling IO::Socket::INET::connect
Calling IO::Socket::connect
Calling IO::Socket::INET::peerport
```

Another Pointcut

```
package SocketDebug;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;

my $pointcut = call qr{\AIO::Socket};
before {
    my $context = shift;          # See Aspect::AdviceContext

    warn 'Calling ', $context->sub_name, "\n";

} $pointcut;

1;
```

Another Pointcut

```
package SocketConfigure;  
  
use Aspect;  
  
# use the module before the code that pointcuts it  
use IO::Socket::INET;  
  
my $pointcut = call qr{\AIO::Socket::INET::configure};  
before {  
    my $context = shift;          # See Aspect::AdviceContext  
    use Data::Dumper;  
    warn 'Calling ', $context->sub_name, ' with ',  
        Dumper[$context->params];  
} $pointcut;  
  
1;
```

Another Trace

```
% perl -Mlib=. -MSocketConfigure lwp_get
Calling IO::Socket::INET::configure with $VAR1 = [
    bless( \*Symbol::GEN0, 'LWP::Protocol::http::Socket'
),
    {
        'Proto' => 'tcp',
        'PeerAddr' => 'another.dull.example.com',
        'PeerPort' => 80
    }
];
... page content goes here...
```

Another Trace

```
% perl -Mlib=. -MSocketConfigure lwp_get
Calling IO::Socket::INET::configure with $VAR1 = [
    bless( \*Symbol::GEN0, 'LWP::Protocol::http::Socket'
),
    {
        'Proto' => 'tcp',
        'PeerAddr' => 'another.dull.example.com',
        'PeerPort' => 80
    }
];
... page content goes here...
```

A Pointcut That Changes Things

```
package SocketChange;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;
my $pointcut = call qr{\AIO::Socket::INET::configure};

before {
    my $context = shift;          # See Aspect::AdviceContext
    my @param    = $context->params;
    $param[1]{PeerAddr} = 'localhost';
    $context->params(@param);
} $pointcut;

1;
```

A Pointcut That Changes Things

```
package SocketChange;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;
my $pointcut = call qr{\AIO::Socket::INET::configure};

before {
    my $context = shift;          # See Aspect::AdviceContext
    my @param    = $context->params;
    $param[1]{PeerAddr} = 'localhost';
    $context->params(@param);
} $pointcut;

1;
```

A Pointcut That Changes Things

```
package SocketChange;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;
my $pointcut = call qr{\AIO::Socket::INET::configure};

before {
    my $context = shift;          # See Aspect::AdviceContext
    my @param    = $context->params;
    $param[1]{PeerAddr} = 'localhost';
    $context->params(@param);
} $pointcut;

1;
```

A Pointcut That Changes Things

```
package SocketChange;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;
my $pointcut = call qr{\AIO::Socket::INET::configure};

before {
    my $context = shift;          # See Aspect::AdviceContext
    my @param    = $context->params;
    $param[1]{PeerAddr} = 'localhost';
    $context->params(@param);
} $pointcut;

1;
```

A Pointcut That Changes Things

```
package SocketChange;

use Aspect;

# use the module before the code that pointcuts it
use IO::Socket::INET;
my $pointcut = call qr{\AIO::Socket::INET::configure};

before {
    my $context = shift;          # See Aspect::AdviceContext
    my @param    = $context->params;
    $param[1]{PeerAddr} = 'localhost';
    $context->params(@param);
} $pointcut;

1;
```

**Enough Contrived
Examples!**

More Pointcuts

```
package Not::The::RSPCA;

use Aspect;

# use the modules before the code that pointcuts them
use Animal::All;
my $pointcut = call qr{\AAnimal::.*::new\z};

after {
    my $context = shift;          # See Aspect::AdviceContext
    my($animal) = $context->return_value;
    $animal->pull_its_legs_off;
    $context->return_value($animal);
} $pointcut;

1;
```

More Pointcuts

```
package Not::The::RSPCA;

use Aspect;

# use the modules before the code that pointcuts them
use Animal::All;
my $pointcut = call qr{\AAnimal::.*::new\z};

after {
    my $context = shift;          # See Aspect::AdviceContext
    my($animal) = $context->return_value;
    $animal->pull_its_legs_off;
    $context->return_value($animal);
} $pointcut;

1;
```

More Pointcuts

```
package Not::The::RSPCA;

use Aspect;

# use the modules before the code that pointcuts them
use Animal::All;
my $pointcut = call qr{\AAnimal::.*::new\z};

after {
    my $context = shift;          # See Aspect::AdviceContext
    my($animal) = $context->return_value;
    $animal->pull_its_legs_off;
    $context->return_value($animal);
} $pointcut;

1;
```

More Pointcuts

```
package Not::The::RSPCA;

use Aspect;

# use the modules before the code that pointcuts them
use Animal::All;
my $pointcut = call qr{\AAnimal::.*::new\z};

after {
    my $context = shift;          # See Aspect::AdviceContext
    my($animal) = $context->return_value;
    $animal->pull_its_legs_off;
    $context->return_value($animal);
} $pointcut;

1;
```

More Pointcuts

```
package Not::The::RSPCA;

use Aspect;

# use the modules before the code that pointcuts them
use Animal::All;
my $pointcut = call qr{\AAnimal::.*::new\z};

after {
    my $context = shift;          # See Aspect::AdviceContext
    my($animal) = $context->return_value;
    $animal->pull_its_legs_off;
    $context->return_value($animal);
} $pointcut;

1;
```

Recursive Pointcuts

```
after {  
    my $context = shift;          # See Aspect::AdviceContext  
    my($animal) = $context->return_value;  
  
    if ($animal->is_ugly) {  
        $animal = $context->original($context->params);  
    }  
  
    $context->return_value($animal);  
} $pointcut;
```

Recursive Pointcuts

```
after {
  my $context = shift;          # See Aspect::AdviceContext
  my($animal) = $context->return_value;

  if ($animal->is_ugly) {
    $animal = $context->original($context->params);
  }

  $context->return_value($animal);
} $pointcut;
```

Behind The Curtain

- `Aspect::Hook::LexWrap`
- based on `Hook::LexWrap`
- Alters the symbol table so you don't have to

The End

tom@eborcom.com