# "Merkwürdigkeiten" - eval und $@

In Ausgabe 9 von $foo habe ich unter "Tipps & Tricks" gezeigt, wie man mit `eval` ein "try-catch" nachbauen kann. Kurz danach, habe ich eine Mail bekommen, dass das nicht gehen würde. Der Code, um den es dabei ging, war folgender:

```
use strict;
use warnings;

eval {
    my $obj = Meine::Klasse->new();
    die "Ein Fehler!\n";
};

if($@) {
    print "Exception caught: $@\n";
}
else {
    print "No exception occurred\n"
}
```

Auf den ersten Blick ist alles klar, oder? Das gibt ganz eindeutig *Exception caught: Ein Fehler!* aus. Das wurde aber nicht ausgegeben, sondern *No exception occured*. Was war passiert? Ein Bug in Perl? Merkwürdig, oder?

Bei der Suche nach dem Fehler zeigte sich schnell, dass das kein Bug in Perl ist, sondern wieder einer der Fallen, mit denen man sich selbst in den Fuß schießen kann. Der Code von `Meine::Klasse` sieht (beispielhaft) so aus:

```
package Local::Class;

use strict;
use warnings;

sub new { bless {}, shift }

sub DESTROY {
    eval{
        my $a = 2;
        # Code zum aufraeumen
    };
}

1;
```

Und Fehler erkannt? Das Problem ist, dass Perls *Garbage Collection* zuschlägt und der Destruktor der Klasse ausgeführt wird. Der Destruktor ist die Subroutine `DESTROY`. Diese wird von Perl automatisch ausgeführt, wenn ein Objekt zerstört wird. Mehr zum Thema "Garbage Collection in Perl" wird es in der nächsten Ausgabe von $foo geben.

Der Ablauf ist somit folgender:
Perl "betritt" den `eval`-Block im Skript. Darin wird ein Objekt der Klasse `Meine::Klasse` erzeugt. Als nächstes wird das `die` ausgeführt und die (globale) Variable `$@` wird gesetzt. Als nächstes wird der `eval`-Block verlassen und Perls Garbage Collection wird angeschmissen. Dabei wird das Objekt von `Meine::Klasse` zerstört, wodurch der Destruktor der Klasse aufgerufen wird. Darin kommt wieder ein C<eval>-Block vor, der aber problemlos durchläuft - also wird die (globale) Variable `$@` auf *undef* gesetzt.

Jetzt kommt der *if-else*-Teil im Hauptskript. Durch den Destruktor ist ja `$@` auf *undef* gesetzt worden, also wird der `else`-Teil ausgeführt.

Wie kann man das Problem lösen?
Sehr einfach: Man muss in dem Destruktor die (globale) Variable `$@` lokalisieren:

```
sub DESTROY {
    local $@;
    eval{
        my $a = 2;
        # Code zum aufraeumen
    };
}
```

Damit gelten die Änderungen, die innerhalb von `DESTROY` an `$@` gemacht werden nur innerhalb dieser Subroutine und den Subroutinen, die in `DESTROY` aufgerufen werden (siehe auch `perldoc -f local`). Verlässt Perl `DESTROY`, wird der Wert von `$@` wieder auf den Wert gesetzt, den die Variable vor der Lokalisierung hatte.

Jetzt wird auch wirklich *Exception caught: Ein Fehler!* ausgegeben.