



# **Closures**

## **beim Umstieg von CGI nach mod\_perl mit Apache::Registry**

**Steffen Schwigon**  
**Perl Mongers Dresden**

# Themen

- Exposition: Closures allgemein
  - ◆ Beispiel
  - ◆ Apache::Registry
- Durchführung: Closures in verschachtelten Subs
  - ◆ Normales Programm
  - ◆ Rahmen-Subfunction
  - ◆ Rahmen-Subfunction und wiederholter Aufruf (I)
  - ◆ Rahmen-Subfunction und wiederholter Aufruf (II)
  - ◆ Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (I)
  - ◆ Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (II)
- Reprise: Sinnvoller Einsatz von Closures
  - ◆ Unabhängige Zufallszahlengeneratoren



# Exposition: Closures allgemein

# Beispiel

- aus "Advanced Perl Programming"

```
sub generate_greeting {  
    my ($greeting) = "hello world";  
    return sub { print $greeting };  
}
```

```
$rs = generate_greeting();  
&$rs(); # Prints "hello world"
```

- Erzeugte SUB behält Umgebung zum Erzeugungszeitpunkt

# Apache::Registry (I)

- <http://affe.de/perl/affe.pl>

- CGI

```
ScriptAlias /perl/ /var/www/lichtschwert/perl
```

- auch bekannt durch

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
```

- mod\_perl mit Apache::Registry

```
Alias /perl/ /var/www/lichtschwert/perl  
PerlModule Apache::Registry
```

```
<Location /perl>  
    SetHandler perl-script  
    PerlHandler Apache::Registry  
    Options ExecCGI  
</Location>
```

# Apache::Registry (II)

- Skripte (z.B. affe.pl) werden compiliert
- als Body einer Subfunction
- und ausgeführt



# Durchführung: Closures in verschachtelten Subs

# Normales Programm

```
my $affe = 100;

sub affe_minus_20 {
    $affe -= 20;
}

sub affe_minus_30 {
    $affe -= 30;
}

print "                (start) affe: $affe\n";

affe_minus_20();
print "                (-20) affe: $affe\n";

affe_minus_30();
print "                (-30) affe: $affe\n";
```

# Normales Programm

- Ausgabe

Closures 1 - normales Programm.

```
(start) affe: 100  
(-20) affe: 80  
(-30) affe: 50
```

- Code (komprimiert)

```
my $affe = 100;  
sub affe_minus_20 { $affe -= 20; }  
sub affe_minus_30 { $affe -= 30; }  
affe_minus_20();  
affe_minus_30();
```

# Rahmen-Subfunktion

```
sub rahmen {
  my $affe = 100;

  sub affe_minus_20 {
    $affe -= 20;
  }

  sub affe_minus_30 {
    $affe -= 30;
  }

  print "                (start) affe: $affe\n";

  affe_minus_20();
  print "                (-20) affe: $affe\n";

  affe_minus_30();
  print "                (-30) affe: $affe\n";
}

rahmen();
```

# Rahmen-Subfunktion

- Ausgabe

```
Variable "$affe" will not stay shared at ../closures2.pl line 11  
Variable "$affe" will not stay shared at ../closures2.pl line 11  
Closures 2 - Rahmen-Sub drumrum.
```

```
    (start) affe: 100  
      (-20) affe: 80  
      (-30) affe: 50
```

- Code (komprimiert)

```
sub rahmen {  
  my $affe = 100;  
  sub affe_minus_20 { $affe -= 20; }  
  sub affe_minus_30 { $affe -= 30; }  
  affe_minus_20();  
  affe_minus_30();  
}  
rahmen();
```

# Rahmen-Subfunction und wiederholter Aufruf (I)

```
sub rahmen {
  my $affe = 100;

  sub affe_minus_20 {
    $affe -= 20;
  }

  sub affe_minus_30 {
    $affe -= 30;
  }
  print "                (start) affe: $affe\n";

  affe_minus_20();
  print "                (-20) affe: $affe\n";

  affe_minus_30();
  print "                (-30) affe: $affe\n";
}

rahmen();
rahmen();
```

# Rahmen-Subfunktion und wiederholter Aufruf (I)

- Ausgabe

```
Variable "$affe" will not stay shared at ../closures3.pl line 11  
Variable "$affe" will not stay shared at ../closures3.pl line 11  
Closures 3 - Rahmen-Sub und wiederholter Aufruf (I).
```

```
      (start) affe: 100  
      (-20) affe: 80  
      (-30) affe: 50  
      (start) affe: 100  
      (-20) affe: 100  
      (-30) affe: 100
```

- Code (komprimiert)

```
sub rahmen {  
  my $affe = 100;  
  sub affe_minus_20 { $affe -= 20; }  
  sub affe_minus_30 { $affe -= 30; }  
  affe_minus_20();  
  affe_minus_30();  
}  
rahmen();  
rahmen();
```

# Rahmen-Subfunktion und wiederholter Aufruf (II)

```
sub rahmen {
  my $affe = 100;

  sub affe_minus_20 {
    $affe -= 20;
    printf " -20: affe: %2d", $affe;
  }

  sub affe_minus_30 {
    $affe -= 30;
    printf " -30: affe: %2d", $affe;
  }

  printf "          (start) affe: %2d\n", $affe;

  affe_minus_20();
  printf "      (-20) affe: %2d\n", $affe;

  affe_minus_30();
  printf "      (-30) affe: %2d\n", $affe;
}

rahmen();
rahmen();
```

# Rahmen-Subfunktion und wiederholter Aufruf (II)

- Ausgabe

```
Variable "$affe" will not stay shared at ../closures3a.pl line 1
Variable "$affe" will not stay shared at ../closures3a.pl line 1
Closures 3a - Rahmen-Sub und wiederholter Aufruf (II).
```

```
                (start) affe: 100
-20: affe: 80      (-20) affe: 80
-30: affe: 50      (-30) affe: 50
                (start) affe: 100
-20: affe: 30      (-20) affe: 100
-30: affe: 0       (-30) affe: 100
```

- Code (komprimiert)

```
sub rahmen {
  my $affe = 100;
  sub affe_minus_20 { $affe -= 20; }
  sub affe_minus_30 { $affe -= 30; }
  affe_minus_20();
  affe_minus_30();
}
rahmen();
rahmen();
```

# Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (I)

```
sub rahmen {
  use vars qw($affe);
  local $affe = 100;

  sub affe_minus_20 {
    $affe -= 20;
  }

  sub affe_minus_30 {
    $affe -= 30;
  }

  print "                (start) affe: $affe\n";

  affe_minus_20();
  print "                (-20) affe: $affe\n";

  affe_minus_30();
  print "                (-30) affe: $affe\n";
}

rahmen();
rahmen();
```

# Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (I)

- Ausgabe

```
Closures 4 - Rahmen-Sub und wiederholter Aufruf (Korrigiert) (I)
      (start) affe: 100
        (-20) affe: 80
        (-30) affe: 50
      (start) affe: 100
        (-20) affe: 80
        (-30) affe: 50
```

- Code (komprimiert)

```
sub rahmen {
  use vars qw($affe);
  local $affe = 100;
  sub affe_minus_20 { $affe -= 20; }
  sub affe_minus_30 { $affe -= 30; }
  affe_minus_20();
  affe_minus_30();
}
rahmen();
rahmen();
```

# Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (II)

```
sub rahmen {  
  
    use vars qw($affe);  
    local $affe = 100;  
  
    sub affe_minus_20 {  
        $affe -= 20;  
        printf " -20: affe: %2d", $affe;  
    }  
  
    sub affe_minus_30 {  
        $affe -= 30;  
        printf " -30: affe: %2d", $affe;  
    }  
  
    printf "                (start) affe: %2d\n", $affe;  
  
    affe_minus_20();  
    printf "                (-20) affe: %2d\n", $affe;  
  
    affe_minus_30();  
    printf "                (-30) affe: %2d\n", $affe;  
}  
rahmen();  
rahmen();
```

# Rahmen-Subfunction und wiederholter Aufruf (Korrigiert) (II)

- Ausgabe

Closures 4 - Rahmen-Sub und wiederholter Aufruf (Korrigiert) (II)

```
(start) affe: 100
-20: affe: 80      (-20) affe: 80
-30: affe: 50      (-30) affe: 50
(start) affe: 100
-20: affe: 80      (-20) affe: 80
-30: affe: 50      (-30) affe: 50
```

- Code (komprimiert)

```
sub rahmen {
  use vars qw($affe);
  local $affe = 100;
  sub affe_minus_20 { $affe -= 20; }
  sub affe_minus_30 { $affe -= 30; }
  affe_minus_20();
  affe_minus_30();
}
rahmen();
rahmen();
```

# Reprise: Sinnvoller Einsatz von Closures

# Unabhängige Zufallszahlengeneratoren

- Bsp. aus "Advanced Perl Programming"

```
# Returns a randomn number generator function
sub my_srand {
    my ($seed) = @_ ;

    my $rand = $seed ;
    return sub {
        # Compute a new pseudo-random number based on its old value
        # This number is constrained between 0 and 1000.
        $rand = ($rand*21+1) % 1000 ;
    } ;
}

$random_iter1 = my_srand (100) ;
$random_iter2 = my_srand (1099) ;
for ($i = 0 ; $i < 15 ; $i++) {
    print &$random_iter1(), " ", &$random_iter2(), "\n" ;
}
```

# Unabhängige Zufallszahlengeneratoren

- Ausgabe

101 80  
122 681  
563 302  
824 343  
305 204  
406 285  
527 986  
68 707  
429 848  
10 809  
211 990  
432 791  
73 612  
534 853  
215 914

Ende

