# Klein, schön, schnell

## mit Curses den eigenen Scripten eine schöne Oberfläche verleihen

Alexander Kluth

15. deutscher Perl-Workshop 2013

# Gliederung

# Gliederung

# Wieso eigentlich? Motivation!

- Zusammengehackten Tools einfacher benutzbar machen
- Benutzung für eventuell andere Nutzer intiutiver machen
- Optische Aufwertung

# Wieso eigentlich? Motivation!

- Zusammengehackten Tools einfacher benutzbar machen
- Benutzung für eventuell andere Nutzer intiutiver machen
- Optische Aufwertung

# Wieso eigentlich? Motivation!

- Zusammengehackten Tools einfacher benutzbar machen
- Benutzung für eventuell andere Nutzer intiutiver machen
- Optische Aufwertung

# Wieso eigentlich? Motivation!

- Curses::UI ist einfach
- Gute Dokumentation
- Geringer Overhead
- Auch auf Umgebungen ohne X-Server nutzbar

# Wieso eigentlich? Motivation!

- Curses::UI ist einfach
- Gute Dokumentation
- Geringer Overhead
- Auch auf Umgebungen ohne X-Server nutzbar

# Wieso eigentlich? Motivation!

- Curses::UI ist einfach
- Gute Dokumentation
- Geringer Overhead
- Auch auf Umgebungen ohne X-Server nutzbar

# Wieso eigentlich? Motivation!

- Curses::UI ist einfach
- Gute Dokumentation
- Geringer Overhead
- Auch auf Umgebungen ohne X-Server nutzbar

# Gliederung

# Warum Curses, warum nicht Tk? Warum keine GUI?

**K**

urze Antwort
Weil wir nix Klicki-Bunti, wir sind Perl-Hacker die mit der Konsole arbeiten

# Warum Curses, warum nicht Tk? Warum keine GUI?

Lange Antwort:

- Ziel ist nicht höchste Benutzerfreundlichkeit
- Eigentlichen Sinn des Programms nicht verschleiern
- Möglichkeit der Ausführung auf Rechnern ohne X-Server

# Warum Curses, warum nicht Tk? Warum keine GUI?

Lange Antwort:

- Ziel ist nicht höchste Benutzerfreundlichkeit
- Eigentlichen Sinn des Programms nicht verschleiern
- Möglichkeit der Ausführung auf Rechnern ohne X-Server

# Warum Curses, warum nicht Tk? Warum keine GUI?

Lange Antwort:

- Ziel ist nicht höchste Benutzerfreundlichkeit
- Eigentlichen Sinn des Programms nicht verschleiern
- Möglichkeit der Ausführung auf Rechnern ohne X-Server

# Warum aber dann ausgerechnet Curses(::UI)?

- Bekannt und »beliebt«
- Gute Dokumentation
- Wenig Code, große Wirkung

# Warum aber dann ausgerechnet Curses(::UI)?

- Bekannt und »beliebt«
- Gute Dokumentation
- Wenig Code, große Wirkung

# Warum aber dann ausgerechnet Curses(::UI)?

- Bekannt und »beliebt«
- Gute Dokumentation
- Wenig Code, große Wirkung

# Gliederung

# Curses::UI und der große Bruder

- Curses::UI basiert auf dem Paket Curses
- Curses ist das Interface zur curses(3) Library des Systems

# Curses::UI und der große Bruder

- Curses::UI basiert auf dem Paket Curses
- Curses ist das Interface zur curses(3) Library des Systems

# Curses::UI hat viel zu bieten

- Objektorientiertes Interface
- Abstraktionsschicht zu Curses
- Viele Out-of-the-Box Widgets

# Curses::UI hat viel zu bieten

- Objektorientiertes Interface
- Abstraktionsschicht zu Curses
- Viele Out-of-the-Box Widgets

# Curses::UI hat viel zu bieten

- Objektorientiertes Interface
- Abstraktionsschicht zu Curses
- Viele Out-of-the-Box Widgets

# Curses::UI und der große Bruder

Curses::UI stellt ein objektorientiertes Interface und eine Abstaktionsschicht zu Curses dar

# Gliederung

# Am Anfang war das Fenster - Curses::UI::Window

- Das Window-Widget selbst ist ein Widget
- Weitere Widgets können dem Window-Widget hinzugefügt werden
- Stellt den Applikations«rahmen« dar

```
use Curses::UI;
my $cui = new Curses::UI;
my $win = $cui->add(
  'mainWindow',
  'Window'
  -fg => 'green'
);
```

# Am Anfang war das Fenster - Curses::UI::Window

- Das Window-Widget selbst ist ein Widget
- Weitere Widgets können dem Window-Widget hinzugefügt werden
- Stellt den Applikations«rahmen« dar

```
use Curses::UI;
my $cui = new Curses::UI;
my $win = $cui->add(
 'mainWindow',
 'Window'
 -fg => 'green'
);
```

# Text mich voll - Curses::UI::Label

- Klassische Label wie in der »großen« GUI-Programmierung
- Darstellung von Text innerhalb eines Window-Widgets

```
my $label = $win->add(
    'helloLabel',
    'Label',
    -text => »Hallo Welt!«,
    -x => 1, -y => 1
);
```

# Text mich voll - Curses::UI::Label

- Klassische Label wie in der »großen« GUI-Programmierung
- Darstellung von Text innerhalb eines Window-Widgets

```
my $label = $win->add(
    'helloLabel',
    'Label',
    -text => »Hallo Welt!«,
    -x => 1, -y => 1
);
```

# Und nun alles zusammen - Hallo Welt!

Es könnte einfacher nicht sein:

```
#!/usr/bin/perl
use strict;
use warnings;
use Curses::UI;

my $cui = new Curses::UI;
my $win = $cui->add('mainWindow', 'Window', '-fg =>
'green');
my $label = $win->add('helloLabel', 'Label', -text =>
»Hallo Welt!«, -x => 1, -y => 1);

$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
$cui->mainloop();
```

# Und nun alles zusammen - Hallo Welt!

Es könnte einfacher nicht sein:

```
#!/usr/bin/perl
use strict;
use warnings;
use Curses::UI;

my $cui = new Curses::UI;
my $win = $cui->add('mainWindow', 'Window', '-fg =>
'green');
my $label = $win->add('helloLabel', 'Label', -text =>
»Hallo Welt!«, -x => 1, -y => 1);

$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
$cui->mainloop();
```

# Gliederung

# Alles geordnet - Curses::UI::ListBox

- Ordnen von Elementen in einer Liste

```
my $list = $win->add(
    "listBox",
    "Listbox",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Elementen via ->values(\@arrayMitContent)

# Alles geordnet - Curses::UI::ListBox

- Ordnen von Elementen in einer Liste

```
my $list = $win->add(
    "listBox",
    "Listbox",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Elementen via ->values(\@arrayMitContent)

# Alles geordnet - Curses::UI::ListBox

- Ordnen von Elementen in einer Liste

```
my $list = $win->add(
    "listBox",
    "Listbox",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Elementen via ->values(\@arrayMitContent)

# Textsicher - Curses::UI::TextViewer

- Darstellung von Freitext in einer Box

```
my $text = $main->add(
    "textBox",
    "TextViewer",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Text mit ->text($hinzuzufügenderText)

# Textsicher - Curses::UI::TextViewer

- Darstellung von Freitext in einer Box

```
my $text = $main->add(
    "textBox",
    "TextViewer",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Text mit ->text($hinzuzufügenderText)

# Textsicher - Curses::UI::TextViewer

- Darstellung von Freitext in einer Box

```
my $text = $main->add(
    "textBox",
    "TextViewer",
    -fg => "white",
    -border => 1
);
```

- Hinzufügen von Text mit ->text($hinzuzufügenderText)

# Zusammenfassung - Widgets

- Widgets haben meist den Aufbau ID, Typ, Optionen
- Neben den vorgestellten viele Weitere (Kalender, Filebrowser, Progressbar, Radiobuttonbox, TextEditor)

# Zusammenfassung - Widgets

- Widgets haben meist den Aufbau ID, Typ, Optionen
- Neben den vorgestellten viele Weitere (Kalender, Filebrowser, Progressbar, Radiobuttonbox, TextEditor)

# Gliederung

# Auf Knopfdruck - Bindings

- Reaktionen auf Tastendruck kann für verschiedene Widgets eingestellt werden
- Auf Knopfdruck wird eine vordefinierte Funktion aufgerufen
- Vorrang hat das Widget, welches den Fokus hat

```
$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
```

# Auf Knopfdruck - Bindings

- Reaktionen auf Tastendruck kann für verschiedene Widgets eingestellt werden
- Auf Knopfdruck wird eine vordefinierte Funktion aufgerufen
- Vorrang hat das Widget, welches den Fokus hat

```
$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
```

# Auf Knopfdruck - Bindings

- Reaktionen auf Tastendruck kann für verschiedene Widgets eingestellt werden
- Auf Knopfdruck wird eine vordefinierte Funktion aufgerufen
- Vorrang hat das Widget, welches den Fokus hat

```
$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
```

# Auf Knopfdruck - Bindings

- Reaktionen auf Tastendruck kann für verschiedene Widgets eingestellt werden
- Auf Knopfdruck wird eine vordefinierte Funktion aufgerufen
- Vorrang hat das Widget, welches den Fokus hat

```
$cui->set_binding(sub { exit 0; }, »\cC«, »q«);
```

# In Szene gesetzt - Dialoge

- Vielzahl von Out-of-the-Box-Dialogen (Question, Status, Filebrowser, Calendar)
- Eigener Dialog mit eigenen Widgets möglich
- Pro Dialog-Typ eigene Methode

# In Szene gesetzt - Dialoge

- Vielzahl von Out-of-the-Box-Dialogen (Question, Status, Filebrowser, Calendar)
- Eigener Dialog mit eigenen Widgets möglich
- Pro Dialog-Typ eigene Methode

# In Szene gesetzt - Dialoge

- Vielzahl von Out-of-the-Box-Dialogen (Question, Status, Filebrowser, Calendar)
- Eigener Dialog mit eigenen Widgets möglich
- Pro Dialog-Typ eigene Methode

# In Szene gesetzt - Dialoge

- Beispiel:

```
$question = $cui->question(»Alles gut hier?«);
$status = $cui->status(»Alles gut hier!«);
```

# In Szene gesetzt - Dialoge

- Beispiel:

```
$question = $cui->question(»Alles gut hier?«);
$status = $cui->status(»Alles gut hier!«);
```

# In Szene gesetzt - Dialoge

- Beispiel:

```
$question = $cui->question(»Alles gut hier?«);
$status = $cui->status(»Alles gut hier!«);
```

# In Szene gesetzt - Dialoge

- Komplexere Dialoge sind über die Methode ->dialog() realisierbar:

```
my $ja = $cui->dialog(
    -message => "Wirklich alles gut?",
    -buttons => [»Ja«, «Nein«],
    -values => [1,0],
    -title => »Jetzt will ich's aber wissen«,
);
if ($ja) { # Alles gut, Wahnsinnscode folgt }
```

# Zusammenfassung

- Die erste Hauptbotschaft des Vortrags in ein bis zwei Zeilen.
- Die zweite Hauptbotschaft des Vortrags in ein bis zwei Zeilen.
- Eventuell noch eine dritte Botschaft, aber das reicht dann auch.

- Ausblick
  - Etwas, was wir noch nicht lösen konnten.
  - Noch etwas, das wir noch nicht lösen konnten.

# Weiterführende Literatur I


A. Autor.
*Einführung in das Präsentationswesen*.
Klein-Verlag, 1990.


S. Jemand.
On this and that.
*Journal on This and That*. 2(1):50–100, 2000.