

# Mojo::UserAgent

parallele HTTP-Anfragen, Kommandozeilen-Verwendung

Uwe Völker

XING AG

05.03.2012

- 1 Einführung
- 2 Mojo::UserAgent
- 3 HTML/JSON-Verarbeitung
- 4 auf der Kommandozeile

- 1 Einführung
  - HTTP-Requests in Perl?
  - parallele HTTP-Requests
  - Mojo::UserAgent
- 2 Mojo::UserAgent
- 3 HTML/JSON-Verarbeitung
- 4 auf der Kommandozeile

# HTTP-Requests in Perl?

- LWP::UserAgent
- LWP::Simple

# HTTP-Requests in Perl?

- LWP::UserAgent
- LWP::Simple
- HTTP::Lite
- HTTP::Client (nur GET!)

# parallele HTTP-Requests

- LWP::Parallel::UserAgent (sehr alt - 2004)
- HTTP::Async
- AnyEvent::HTTP, POE::Component::Client::HTTP
- WWW::Curl::Multi

# Mojo::UserAgent

Wer sich nicht daran stört, Mojolicious zu installieren, bekommt einen mächtigen HTTP-Klienten:

- asynchron
- HTTP 1.1
- WebSocket-Unterstützung

- 1 Einführung
- 2 Mojo::UserAgent**
  - Überblick
  - non-blocking
  - parallele Anfragen
- 3 HTML/JSON-Verarbeitung
- 4 auf der Kommandozeile



# Überblick

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

```
use strict;
```

```
use warnings;
```

```
use Mojo::UserAgent;
```

```
my $ua=Mojo::UserAgent->new( max_redirects=>5);
```

```
my $tx=$ua->get( 'http://www.perlworkshop.de/' );
```

```
print $tx->res->body;
```

# HTTP-Methoden

- get
- post, post\_form
- put
- delete
- head

# transaction-Objekt

- req - request-Objekt
- res - response-Objekt

# transaction-Objekt

- req - request-Objekt
- res - response-Objekt
- local\_address, local\_port
- remote\_address, remote\_port

# non-blocking

```
$ua->get('http://perl.org' => sub {  
    my ($ua, $tx) = @_  
    print $tx->res->body;  
    Mojo::IOLoop->stop;  
});  
Mojo::IOLoop->start;
```

# Mojo::IOLoop

- minimalistic IO loop
- building block for non-blocking TCP clients and servers
- start, stop

## parallele Anfragen

```
my $delay = Mojo::IOLoop->delay;  
foreach my $url ( 'mojolicio.us', 'cpan.org' ) {  
    $delay->begin;  
    $ua->get($url => sub {  
        my ($ua, $tx) = @_;  
        $delay->end($tx->res->dom->at('title')->text);  
    });  
}  
my @titles = $delay->wait;
```

# Mojo::IOLoop::Delay

- synchronize events
- begin - increase event counter
- end - decrease event counter (and set return values)
- wait - wait for "finish" event (and return results)



- 1 Einführung
- 2 Mojo::UserAgent
- 3 HTML/JSON-Verarbeitung**
  - Mojo::DOM
  - Mojo::JSON::Pointer
- 4 auf der Kommandozeile

# Mojo::DOM

- HTML5/XML DOM-Parser
- klingt wie HTML::TreeBuilder?

# Mojo::DOM

- HTML5/XML DOM-Parser
- klingt wie HTML::TreeBuilder?
- Unterstützung fuer CSS3-Selektoren
- mehr wie HTML::TreeBuilder::XPath

# Mojo::DOM

```
use Mojo::DOM;

my $dom = Mojo::DOM->new(
    '<div id="a">A</div><div id="b">B</div>' );

print $dom->at( '#b' )->text;

$dom->div->[1]->replace_content( 'foo' );

print $dom;
# <div id="a">A</div><div id="b">foo</div>
```

# Mojo::DOM

- new/parse - DOM aufbauen
- Suche: at, find
- Bewegung: root, children, parent, div, p (Tags)
- Extraktion: attr, text
- Manipulation: append, prepend, replace

# Mojo::JSON::Pointer

```
use Mojo::JSON::Pointer;  
  
my $p = Mojo::JSON::Pointer->new;  
  
say $p->get({foo => [23, 'bar']}, '/foo/1');  
  
if ($p->contains({foo => [23, 'bar']}, '/foo')) {  
    say 'Contains "/foo".';  
}
```

# Mojo::JSON::Pointer

- `http://tools.ietf.org/html/draft-pbryan-zyp-json-pointer-02`
- `get` - extract parts of the data structure
- `contains` - check if a sub structure is present

- 1 Einführung
- 2 Mojo::UserAgent
- 3 HTML/JSON-Verarbeitung
- 4 auf der Kommandozeile**
  - mojo get
  - CSS-Selektoren
  - JSON-Pointer



# mojo get

- `mojo get -r perl.org`
- `mojo get -M POST -c body example.org`
- `mojo get -r perl.org '#short_lists .list p a' 0`
- `mojo get http://search.twitter.com/search.json /error`

# CSS-Selektoren

- zusätzliche Parameter nach der URL
- Selektor(en)
- Kommandos (text, all, attr, id)

# JSON-Pointer

- Content-Type muss JSON sein
- / ... / ...
- Hash-Key oder Array-Index

# Mojo::UserAgent

- Fragen?
- <http://groups.google.com/group/mojolicious>
- #mojo on irc.perl.org
- Einfach ausprobieren!