WWW::Mechanize

Web-Automatisierung mit WWW::Mechanize
Eine Einführung

Web-Automatisierung

- Oft sind wichtige Informationen, Daten oder Dateien nur über aufwendig manuell aus zu füllende Formulare zu erreichen.
- Solche Tätigkeiten lassen sich meist schnell und kostengünstig automatisieren (Web-Scraping).
- WWW::Mechanize ist ein besonders gut geeignetes Framework zur Erstellung von Web-Scrapern.

Law and Order

Bevor man Informationen von Webseiten Dritter automatisch ermittelt und weiterverarbeitet, sollte man prüfen, ob dies rechtlich überhaupt zulässig ist oder ob der Betreiber der Website damit einverstanden ist.

Höflichkeit

- Computerprogramme können Websites wesentlicher schneller und öfter aufrufen als Menschen
  - Bandbreite (Downloads)
  - Tageszeit (cron, Task-Planner)
  - sleep($naptime);

Grundlagen

WWW::Mechanize verhält sich wie ein handelsüblicher Browser.

WWW::Mechanize Features

- WWW::Mechanize kann
  - (passwortgeschützte) Seiten aufrufen
  - Links ermitteln und folgen
  - Formulare ausfüllen und absenden
  - Dateien downloaden
  - u.v.m.
- und natürlich alle Perl-Features nutzen.

Support für SSL (https)

Ja, dazu muss eines der beiden CPAN-Module Crypt::SSLeay oder IO::Socket::SSL installiert werden.

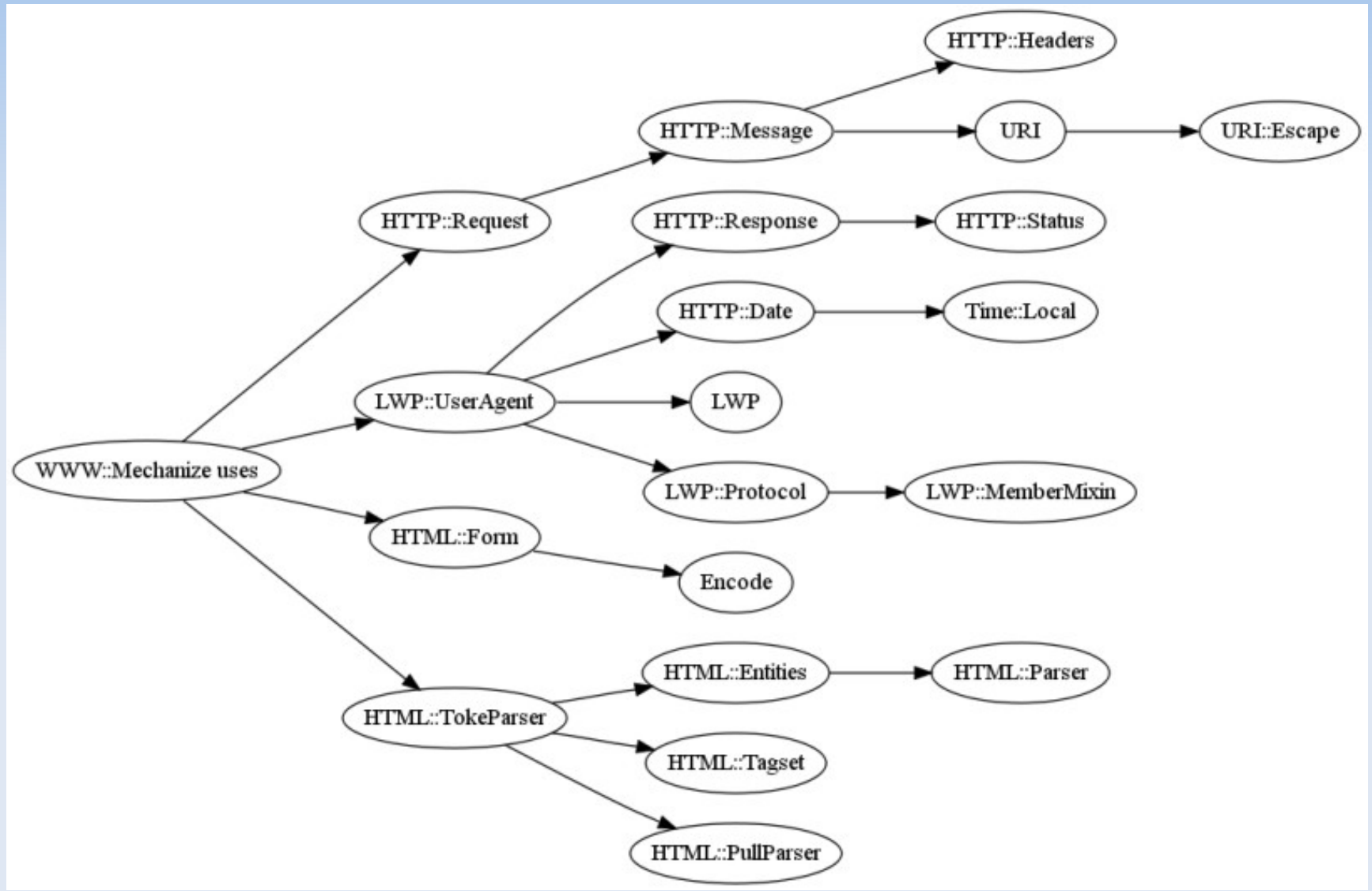
JavaScript, Flash, Java Applets

Nein, JavaScript, Flash-Formulare oder Java Applets werden nicht unterstützt.

Architektur

WWW::Mechanize ist komplex, aber einfach zu bedienen.

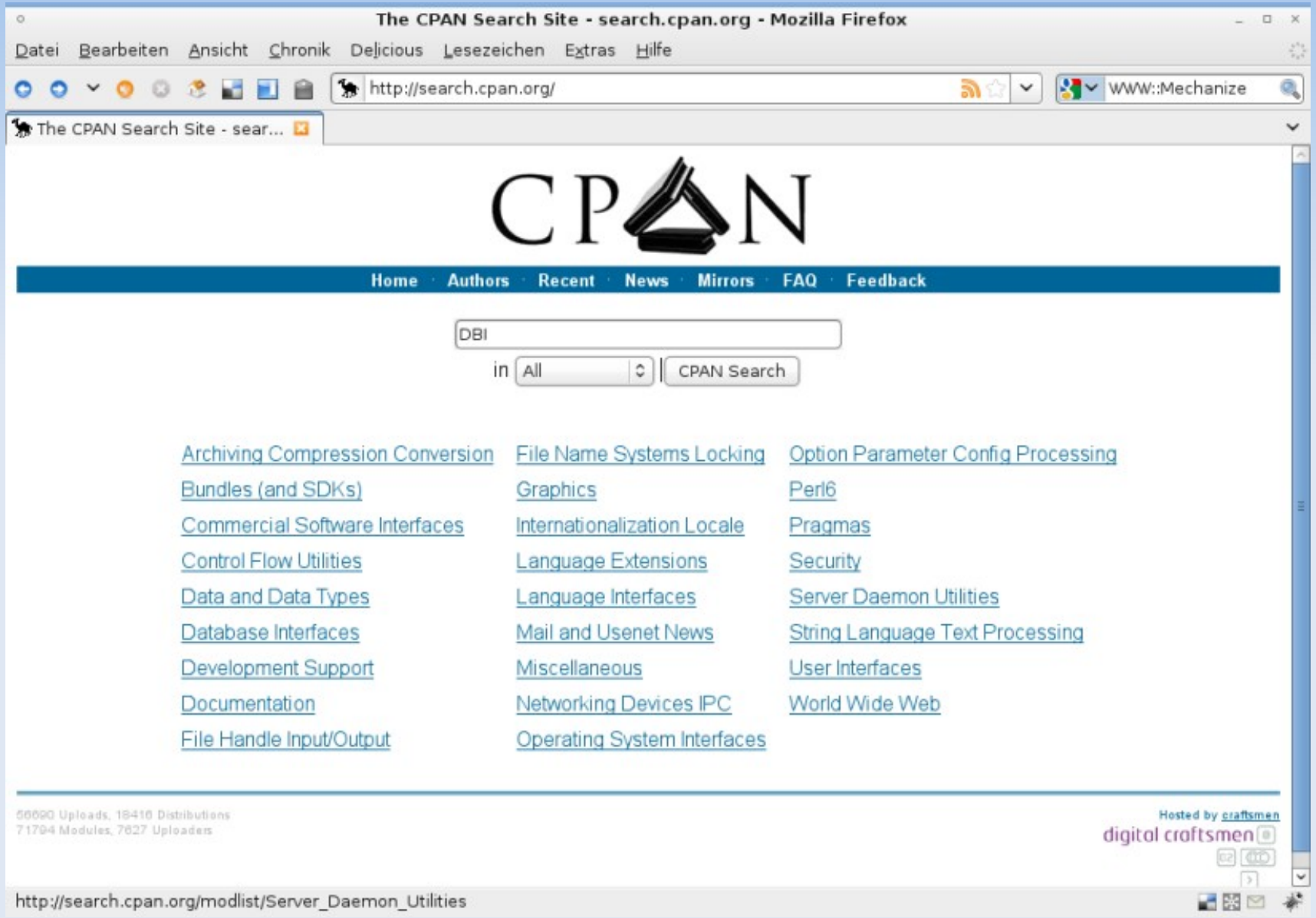
WWW::Mechanize nutzt zahlreiche CPAN-Module



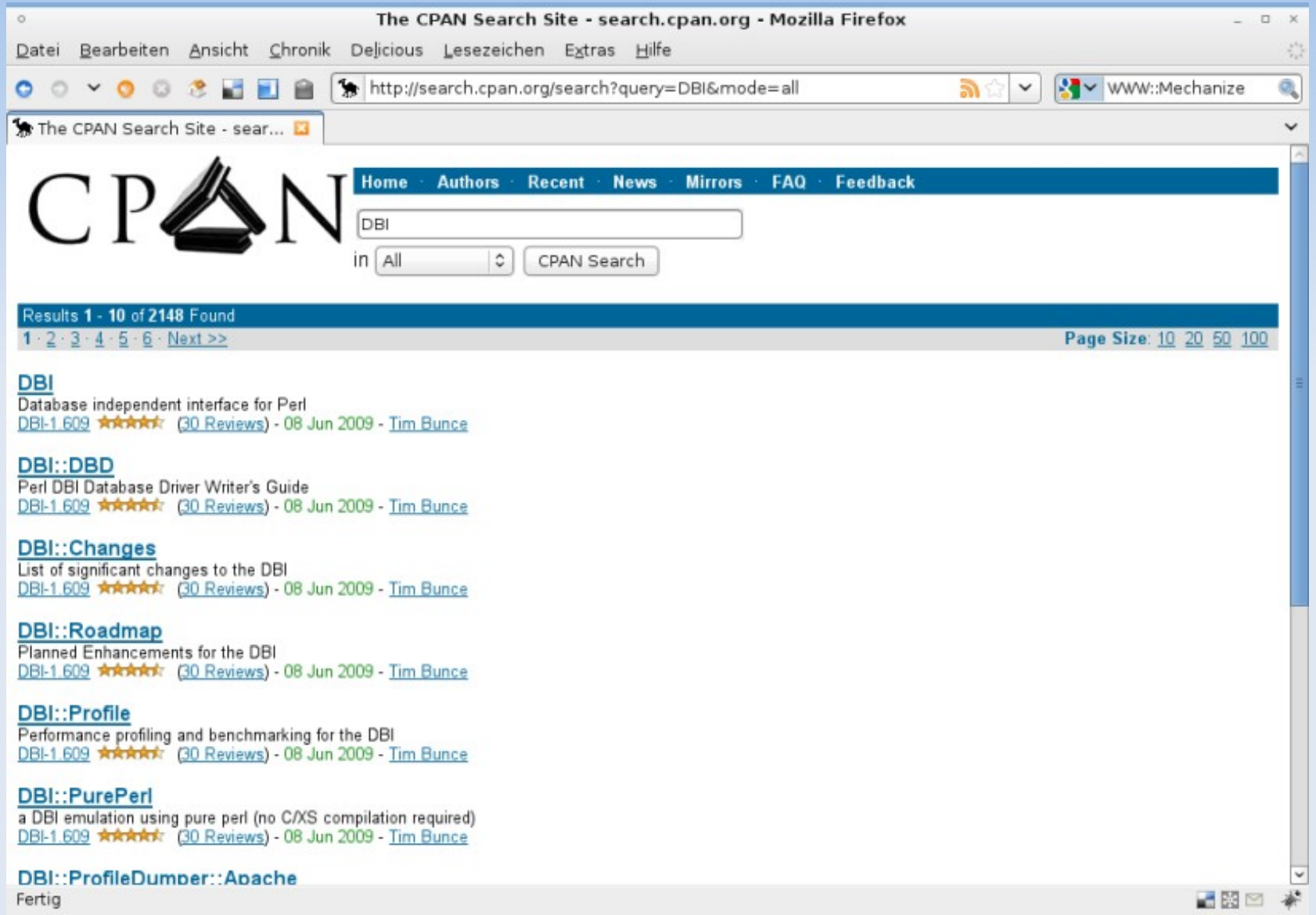
Beispiel cpansearch.pl

- Aufgaben des Programms
  - Gehe zu http://search.cpan.org
  - Suche nach "DBI"
    - Formular ausfüllen
    - Submit Button klicken
  - Hole die Suchergebnis-Seite
  - Ermittle den Link zur Modul-Seite

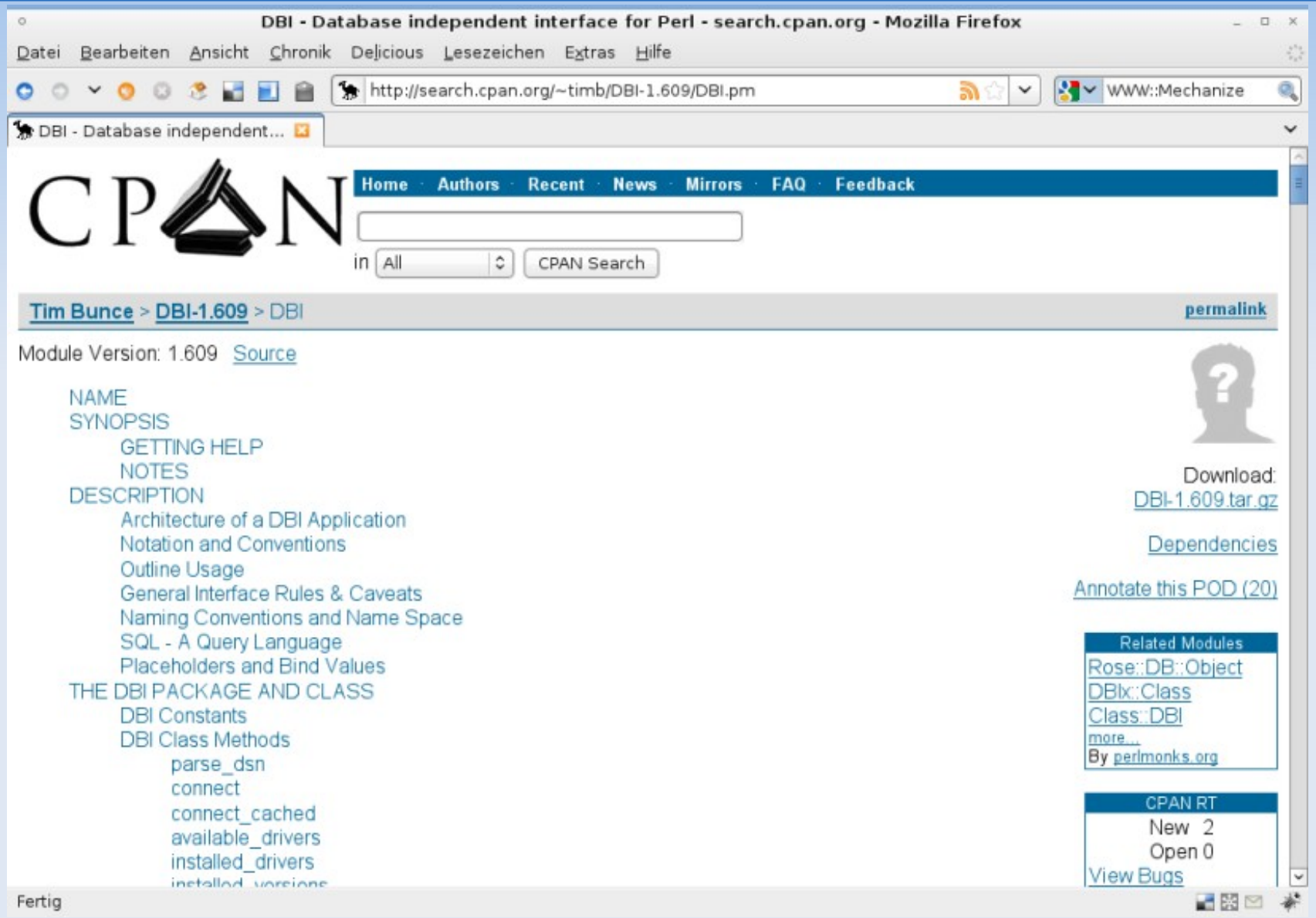
Schritt I und II



Schritt III


The CPAN Search Site - search.cpan.org - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Delicious Lesezeichen Extras Hilfe
http://search.cpan.org/search?query=DBI&mode=all
WWW::Mechanize
The CPAN Search Site - sear...
CPAN
Home · Authors · Recent · News · Mirrors · FAQ · Feedback
DBI
in All
CPAN Search
Results 1 - 10 of 2148 Found
1 · 2 · 3 · 4 · 5 · 6 · Next >>
Page Size: 10 20 50 100
DBI
Database independent interface for Perl
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::DBD
Perl DBI Database Driver Writer's Guide
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::Changes
List of significant changes to the DBI
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::Roadmap
Planned Enhancements for the DBI
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::Profile
Performance profiling and benchmarking for the DBI
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::PurePerl
a DBI emulation using pure perl (no C/XS compilation required)
DBI-1.609 (30 Reviews) - 08 Jun 2009 - Tim Bunce
DBI::ProfileDumper::Apache
Fertig

Schritt IV


DBI - Database independent interface for Perl - search.cpan.org - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Delicious Lesezeichen Extras Hilfe
http://search.cpan.org/~timb/DBI-1.609/DBI.pm
WWW::Mechanize
DBI - Database independent...
CPAN
Home · Authors · Recent · News · Mirrors · FAQ · Feedback
in All
CPAN Search
Tim Bunce > DBI-1.609 > DBI
permalink
Module Version: 1.609 Source
NAME
SYNOPSIS
GETTING HELP
NOTES
DESCRIPTION
Architecture of a DBI Application
Notation and Conventions
Outline Usage
General Interface Rules & Caveats
Naming Conventions and Name Space
SQL - A Query Language
Placeholders and Bind Values
THE DBI PACKAGE AND CLASS
DBI Constants
DBI Class Methods
parse_dsn
connect
connect_cached
available_drivers
installed_drivers
Download:
DBI-1.609.tar.gz
Dependencies
Annotate this POD (20)
Related Modules
Rose::DB::Object
DBIx::Class
Class::DBI
more...
By perlmonks.org
CPAN RT
New 2
Open 0
View Bugs
Fertig

Source Code: cpansearch.pl

```
#!/usr/bin/perl
use strict;
use warnings;
my $module_name = $ARGV[0] or die "Bitte den Namen des gesuchten
  CPAN-Moduls eingeben!\n";

use WWW::Mechanize;
my $browser = WWW::Mechanize->new();

$browser->get("http://search.cpan.org/");

$browser->form_number(1);
$browser->field("query", $module_name);
$browser->click();

$browser->follow_link( text_regex => qr/$module_name/ );

my $url = $browser->uri;

print "$url\n";
```

new() I

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new();
```

new() II - Defaults

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
       stack_depth => 8675309,      # infinite History
      timeout => 180,               # Network inactivity
      autocheck => 1,               # auto die on errors
      agent => "WWW-Mechanize/x.xx", # Versionsnummer
      cookie_jar  => {},            # Cookies akzeptieren
);
```

new() III

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
      stack_depth => 0,            # no History
      timeout => 180,              # Network inactivity
      autocheck => 1,              # auto die on errors
      agent => "WonderBot 5.0",    # Versionsnummer
      cookie_jar  => {},           # Cookies akzeptieren
);

# Das Herabsetzen des History-Speichers senkt den
# Hauptspeicherverbrauch (RAM) erheblich!
```

timeout => ???

- Der Parameter timeout legt eine Netzwerk Inaktivitäts Zeit in Sekunden fest.
- Sobald ein einziges Byte übertragen wurde, wird der Wert zurückgesetzt.
- Eine zeitliche Begrenzung für Netzwerk-verbindungen kann durch die Perl-Funktion alarm() realisiert werden.
- Details und Beispiele im LWP FAQ 1.0.3.

agent_alias()

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
      stack_depth => 0,            # no History
      timeout => 180,              # Network inactivity
      autocheck => 1,              # auto die on errors
);
my $alias = 'Windows IE 6';
$browser->agent_alias( $alias );

print join(", ",$browser->known_agent_aliases()) , "\n";

#    'Windows IE 6', 'Windows Mozilla'
#    'Mac Safari' , 'Mac Mozilla'
#    'Linux Mozilla' ,'Linux Konqueror'
```

get()

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
      stack_depth => 0,            # no History
      timeout => 180,              # Network inactivity
      autocheck => 1,              # auto die on errors
);

$browser->agent_alias( 'Windows IE 6' ) ;

my $start_url = 'http://search.cpan.org';

$browser->get( $start_url );
```

get() II autocheck => 1

```
my $start_url = 'http://www.somewhere.tld';

$browser->get( $start_url );

$ Error GETing http://www.somewhere.tld: Can't connect to
  www.somewhere.tld:80 (Bad hostname 'www.somewhere.tld')
  at xyz.pl line nn
```

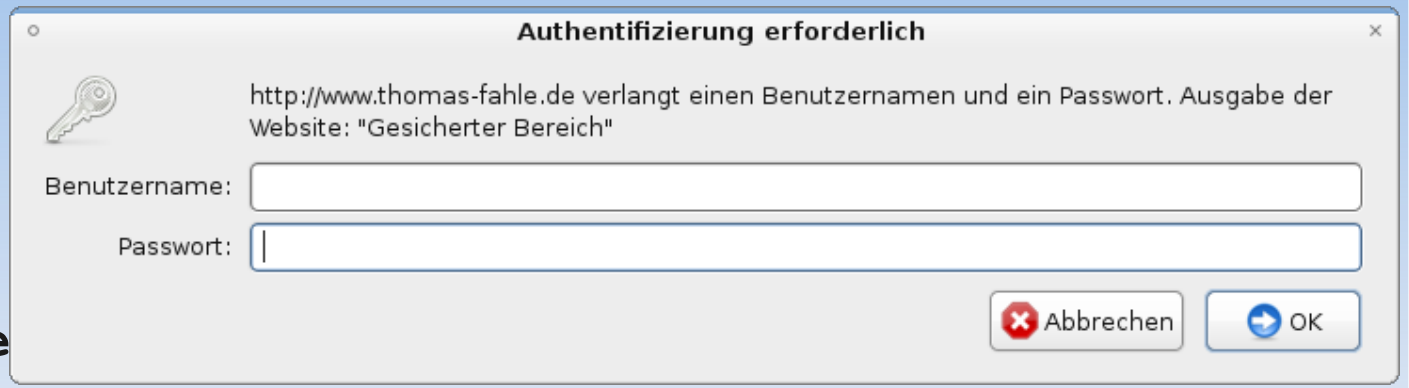
Datei-Download

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
      stack_depth => 0,            # no History
      timeout => 180,              # Network inactivity
      autocheck => 1,              # auto die on errors
);

my $start_url = 'http://search.cpan.org';

my $file = './cpan.html';

$browser->get( $start_url, ':content_file' =>
  $file );
```

Dateidownload II

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
      stack_depth => 0,            # no History
      timeout => 180,              # Network inactivity
      autocheck => 1,              # auto die on errors
);

my $pdf_url = 'http://perl-
  nachrichten.de/index.cgi/news/552/pdf';

my $file = './new.pdf';

$browser->get( $pdf_url, ':content_file' => $file );
```

Passwortgeschützte Seiten




```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize

my $browser = WWW::Mechanize->new();

my $start_url = 'http://www.passwort-geschuetzt.de/';

my $username = 'USER';
my $password = 'PASSWORD';

$browser->credentials( $username, $password );

$browser->get( $start_url );
```

Status

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new();

my $start_url = 'http://search.cpan.org';

$browser->get( $start_url );

print $browser->status(), "\n";
# status() gibt eine dreistellige Zahl zurück,
# den HTTP-Status-Code, z.B. 200, 500 usw.
```

Fehler manuell prüfen

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new(
                  autocheck => 0,
);
```

success()

```
#!/usr/bin/perl
use strict;
use warnings;
use WWW::Mechanize;
my $browser = WWW::Mechanize->new(
        autocheck => 0,
);
my $start_url = 'http://search.cpan.org';
$browser->get( $start_url );

print $browser->status(), "\n";

if ( $browser->success() ) {
     print "Alles Klar\n";
   # MachWeiter();
}
else {
     warn "Oops Fehler\n";
   # behandleFehler();
}
```

response()

```
#!/usr/bin/perl
use strict;
use warnings;
use WWW::Mechanize;
my $browser = WWW::Mechanize->new(   autocheck => 0 );

my $start_url = 'http://gibt-es-nicht.tld';

$browser->get( $start_url );
if ( $browser->success() ) {
    print "Alles Klar\n";
}else {
   print $browser->response->status_line(), "\n";
   # HTTP::Response
}
# Ausgabe:
# 500 Can't connect to gibt-es-nicht.tld:80
# (Bad hostname 'gibt-es-nicht.tld')
```

Alle Links ermitteln

```
#!/usr/bin/perl
use strict;
use warnings;
use WWW::Mechanize;
my $browser = WWW::Mechanize->new();
$browser->get("http://search.cpan.org/");

my @link_objects = $browser->links( );

# @link_objects enhält Objekte der Klasse
# WWW::Mechanize::Link

foreach my $link (@link_objects) {
        print $link->url() , "\n";     # URL from the link
        print $link->url_abs() , "\n";# Abs URL
        print $link->text(), "\n";     # Text of the link
        print "\n";
}
```

Links folgen - follow_links()

```
# 3. Link, der mit 'download' beschriftet ist
$browser->follow_link( text => 'download', n => 3 );

# 1. Link, dessen Text 'download' enthält
$browser->follow_link( text_regex => qr/download/i, );
# n = 1  (default)

# 1. Link, dessen URL 'download' enthält.
$browser->follow_link( url_regex => qr/download/i,
                       n => 1  );

# einfach dem 3. Link auf der Seite folgen
$browser->follow_link( n => 3 );
```

Links folgen – follow_links() II

```
# UND Verknüpfung

# Link, der mit 'download' beschriftet ist
# und dessen url 'download' enthält

$browser->follow_link( text => 'download',
                       url_regex => qr/download/i,
);

# n = 1
```

Links folgen III – Rückgabewert prüfen

```
#!/usr/bin/perl
use strict;
use warnings;
use WWW::Mechanize;

my $browser = WWW::Mechanize->new( autocheck => 1 );

$browser->get("http://search.cpan.org/");

$browser->follow_link( text => 'Database Interfaces' )
          or die "Kann Link nicht finden\n";

print $browser->title(), "\n"; # <title>(.*)</title>
```

Links finden und folgen

```
#!/usr/bin/perl
use strict;
use warnings;
use WWW::Mechanize;
my $browser = WWW::Mechanize->new( autocheck => 1 );
$browser->get("http://search.cpan.org/");

my $link = $browser->find_link(
           text => 'Database Interfaces', n => 1 );
# $link ist ein Objekt der Klasse WWW::Mechanize::Link

if ( defined $link ) {
        print $link->url(), "\n";
        $browser->get( $link );
        print $browser->title(), "\n";
}else {
        print "Ooops: Kein Link gefunden\n";
}
```

Formulare

Alle Formulare ermitteln

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;
my $browser = WWW::Mechanize->new();

$browser->get("http://search.cpan.org/");

# Formulare auf der Seite ermitteln
my @formulare = $browser->forms();

foreach my $formular ( @formulare ) {
        # $formular ist ein Object der Klasse HTML::Form
        print $formular->dump(), "\n";
}
```

mech-dump

- mech-dump, das mit WWW::Mechanize geliefert wird, ermittelt alle Formulare einer Webseite

```
$ mech-dump http://search.cpan.org

GET http://search.cpan.org/search [f]

  query=                                (text)

  mode=all                              (option)    [*all/All|
  module/Modules|dist/Distributions|author/Authors]

  <NONAME>=CPAN Search                  (submit)
```

Formulare auswählen

- $browser->form_number($number)
  - Wählt das Formular mit der Nummer $number
  - Das erste Formular hat die Nummer 1
- $browser->form_name($name)
  - Wählt das Formular mit dem Namen $name

Formulare ermitteln

```
$ /mech-dump http://heise.de

GET http://www.heise.de/suche/
  q=                                  (text)
  search_submit=Suche                 (submit)
  rm=search                           (hidden readonly)

GET http://www.heise.de/preisvergleich/
  fs=                                 (text)
  in=                                 (option)   [*/in allen
Kategorien|1/in Hardware|2/in Software|3/in Games|4/in
Video/Foto/TV|5/in Telefon & Co|6/in Audio/HIFI|7/in DVD|8/
in Haushalt]
  <NONAME>=Suche                      (submit
```

Formular ausfüllen

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

my $browser = WWW::Mechanize->new();

$browser->get("http://www.heise.de/");

# Das zweite Formular auswaehlen (Preisvergleich)
$browser->form_number(2);
my $produkt = 'Half-Life';
$browser->field('fs', $produkt);
$browser->field('in', 3); # Games
```

Formular absenden

- $browser->click();
  - Klickt auf den Button.
- $browser->submit();
  - Schickt das Formular ab, ohne auf einen Button zu klicken.

Murphy's Law: it doesn't work

- **$browser->response->status_line();**
- **use LWP::Debug qw(+);**

Murphy's Law II

```
#!/usr/bin/perl
use strict;
use warnings;

use WWW::Mechanize;

use LWP::Debug qw(+);

my $browser = WWW::Mechanize->new();

$browser->get(' http://www.example.tld ');

print "Status: ", $browser->response->status_line(), "\n";
```

Murphy's Law III

```
$ perl  alice-dsl.pl
LWP::UserAgent::new: ()
LWP::UserAgent::request: ()
HTTP::Cookies::add_cookie_header: Checking www.example.tld for cookies
HTTP::Cookies::add_cookie_header: Checking .example.tld for cookies
HTTP::Cookies::add_cookie_header: Checking example.tld for cookies
HTTP::Cookies::add_cookie_header: Checking .tld for cookies
LWP::UserAgent::send_request: GET http://www.example.tld
LWP::UserAgent::_need_proxy: Not proxied
LWP::Protocol::http::request: ()
LWP::Protocol::collect: read 264 bytes
LWP::UserAgent::request: Simple response: Found
LWP::UserAgent::request: ()
HTTP::Cookies::add_cookie_header: Checking alicesuche.aol.de for cookies
HTTP::Cookies::add_cookie_header: Checking .aol.de for cookies
HTTP::Cookies::add_cookie_header: Checking aol.de for cookies
HTTP::Cookies::add_cookie_header: Checking .de for cookies
LWP::UserAgent::send_request: GET http://alicesuche.aol.de/aol/afe_x?
  s_it=500error_alice&query=www.example.tld
LWP::UserAgent::_need_proxy: Not proxied
LWP::Protocol::http::request: ()
LWP::Protocol::collect: read 463 bytes
LWP::Protocol::collect: read 1896 bytes
LWP::Protocol::collect: read 1380 bytes
LWP::Protocol::collect: read 80 bytes
LWP::Protocol::collect: read 1380 bytes
LWP::Protocol::collect: read 1380 bytes
LWP::Protocol::collect: read 1380 bytes
LWP::Protocol::collect: read 1140 bytes
LWP::Protocol::collect: read 818 bytes
HTTP::Cookies::extract_cookies: Set cookie clickstreamid => 1910176145299038985
HTTP::Cookies::extract_cookies: Set cookie userid => 1910176145299038987
LWP::UserAgent::request: Simple response: OK
Status: 200 OK
```

Tools

- Logging
- HTML parsen
- HTTP::Recorder
- WWW::Mechanize::Shell
- Firefox Extensions

Logging

- Logging ist unerlässlich, nicht nur für Crawler
  - Crawler laufen oft automatisch und unbeobachtet, z.B. via cron
  - Websiten ändern ihr Design oder Struktur
  - Netzwerkkomponenten fallen aus oder sind vorübergehend nicht verfügbar
- Log4perl ++

HTML parsen

- HTML-Tabellen
  - HTML::TableExtract
- HTML Quelltext
  - HTML::TreeBuilder, HTML::Tree
- HTML-Header
  - HTML::HeadParser

HTTP::Recorder

- HTTP::Recorder arbeitet als Proxy für den Browser, zeichnet Aktionen des Nutzers auf und generiert daraus WWW::Mechanize Progamme.
- Diese einfachen Programme müssen meist noch bearbeitet werden.
- Ausführlicher Artikel auf Perl.com: Web Testing with HTTP::Recorder - http://www.perl.com/lpt/a/845

WWW::Mechanize::Shell

- WWW::Mechanize::Shell von Max Maischein bietet einen einfachen Shell-Zugang zu WWW::Mechanize
- Aus der Sitzung können einfache WWW::Mechanize Programme generiert werden

WWW::Mechanize::Shell Beispiel

```
$ perl -MWWW::Mechanize::Shell -e shell
 (no url)>get http://search.cpan.org
Retrieving http://search.cpan.org(200)
    http://search.cpan.org>forms
Form [1]
GET http://search.cpan.org/search [f]
  query=                          (text)
  mode=all                        (option)   [*all/All|module/Modules|dist/Distributions|
   author/Authors]
  <NONAME>=CPAN Search            (submit)
http://search.cpan.org>fill
(text)query> [] WWW::Mechanize::Shell
all|module|dist|author (option)mode> [all] module
http://search.cpan.org>submit
200
     http://search.cpan.org/search?query=WWW%3A%3AMechanize%3A%3AShell&mode=module>links
.....
http://search.cpan.org/search?query=WWW%3A%3AMechanize%3A%3AShell&mode=module>o
  WWW::Mechanize::Shell
Found 18 (200)
http://search.cpan.org/~corion/WWW-Mechanize-Shell-0.48/lib/WWW/Mechanize/Shell.pm>links
.....
http://search.cpan.org/~corion/WWW-Mechanize-Shell-0.48/lib/WWW/Mechanize/Shell.pm>get  WWW-
  Mechanize-Shell-0.48.tar.gz
Retrieving WWW-Mechanize-Shell-0.48.tar.gz(404)
http://search.cpan.org/~corion/WWW-Mechanize-Shell-0.48/lib/WWW/Mechanize/WWW-Mechanize-
  Shell-0.48.tar.gz>script
```

WWW::Mechanize::Shell Script

```
#!/usr/bin/perl -w
use strict;
use WWW::Mechanize;
use WWW::Mechanize::FormFiller;
use URI::URL;

my $agent = WWW::Mechanize->new( autocheck => 1 );
my $formfiller = WWW::Mechanize::FormFiller->new();
$agent->env_proxy();

$agent->get('http://search.cpan.org');
$agent->form_number(1) if $agent->forms and scalar @{$agent-
  >forms};
$formfiller->add_filler( 'query' => Fixed =>
  'WWW::Mechanize::Shell' );
$formfiller->add_filler( 'mode' => Fixed => 'module' );
$formfiller-fill_form($agent->current_form);
$agent->submit();
$agent->follow_link('text' => 'WWW::Mechanize::Shell');
$agent->get('WWW-Mechanize-Shell-0.48.tar.gz');
```

Firefox Extensions

- Firebug
- Web Developer
- Live HTTP-Headers

Watchlist CPAN-Module

- CPAN-Module für die Watchlist
  - Web::Scraper
  - WWW::Scripter

Web::Scraper

- Möglicher Nachfolger von WWW::Mechanize
- Perlport des Ruby-Projekts scrAPI
- unterstützt CSS-Selektoren und Xpath
- unzureichend dokumentiert

WWW::Scripter

- WWW::Scripter - For scripting web sites that have scripts
  - WWW::Scripter::Plugin::JavaScript
  - WWW::Scripter::Plugin::Ajax
- Erbt von WWW::Mechanize
- Noch im frühen Entwicklungsstadium

Literatur

- Burke, Aas: Perl & LWP. O'Reilly, Juli 2002, ISBN: 0-596-001789.
- Hemenway, Calishain: Spidering Hacks. O'Reilly, November 2003, ISBN: 0-596-005776
- Christiansen, Torkington: Perl-Kochbuch. O'Reilly, 2.A. Februar 2004, ISBN: 3-897-213664, Kapitel 20 - Web-Automatisierung

Weiterführende Links I

- WWW-Mechanize: http://search.cpan.org/dist/WWW-Mechanize/
  - WWW::Mechanize::Cookbook
  - WWW::Mechanize::Examples
  - WWW::Mechanize::FAQ

Weiterführende Links II

- LWP:http://search.cpan.org/perldoc?LWP
  - LWP::UserAgent
  - lwpcook (The libwww-perl cookbook)
  - lwptut (An LWP Tutorial)
- LWP FAQ 1.0.3: http://www.mail-archive.com/libwww@perl.org/msg00405.html

Weiterführende Links III

- Bildschirm-Abzieher (Linux-Magazin, März 2004): http://perlmeister.com/snapshots/200403/index.h
- perl.com: Screen-scraping with WWW::Mechanize: http://www.perl.com/pub/a/2003/01/22/mechaniz
- Automating Web-based Data Retrieval with Perl: http://www.developer.com/lang/other/article.php/3454041

Weiterführende Links IV

- Schöner leben mit WWW::Mechanize: http://www.kryger.de/www.mechanize/
- Perl 2002 Advent Calendar: WWW::Mechanize: http://www.perladvent.org/2002/16th/
- Web scraping with WWW::Mechanize (Apr 03): http://www.stonehenge.com/merlyn/LinuxMag/c
- Perl.com: Web Testing with HTTP::Recorder: http://www.perl.com/lpt/a/845

Weiterführende Links V

- A pragmatic approach to writing a music crawler: http://code.google.com/p/perlhobby/wiki/architec
- Gathering Information from the Web with WWW::Mechanize: http://miltonkeynes.pm.org/talks/2006/03/tom_hu
- Max Maischein - Web::Scraper - Daten aus Webseiten extrahieren: http://datenzoo.de/pub/gpw2008/web-scraper/web-scraper-talk.html

Weiterführende Links VI

- Automated Testing with WWW::Mechanize: http://www.webgui.org/uploads/Z1/6Y/Z16YD0q
- Automated Testing of Large Projects With Perl: http://petdance.com/perl/large-project-testing.pd
- IBM Developer Works: Secure Web site access with Perl: http://www.ibm.com/developerworks/linux/library/wa-perlsecure.html

Fragen

Fragen?

Danke

Danke!