Contents

1 Das Modul WebService::Leanpub

Abstract

Dieser Vortrag beschreibt das genannte Modul und seinen Einsatz. Mit diesem Modul ist es möglich das Web-API von Leanpub in Perl-Programmen oder auf der Kommandozeile zu nutzen. Es ist vor allem für Autoren bei Leanpub interessant und für ihre Helfer bei der Buchproduktion.

Was ist das Leanpub-API

Das Leanpub API ist ein Web-API, beschrieben in https://leanpub.com/help/api.

Für die meisten Funktionen benötige ich einen API-Schlüssel, diesen kann ich als Autor bei Leanpub bekommen. Mit dem API-Schlüssel kann ich nur Bücher bearbeiten, deren Autor ich bin.

Ein weitere wesentliche Komponente ist der sogenannte Slug, das ist der Teil der URL hinter `https://leanpub.com/`, der den Pfad zum Buch ausmacht. So ist zum Beispiel die URL des Buches *Using the Leanpub API with Perl*, https://leanpub.com/using-the-leanpub-api-with-perl und der Slug zu diesem Buch ist `using-the-leanpub-api-with-perl`.

Nebenbei: Sollte jemand auf Grund dieses Vortrages dieses Buch erstehen wollen, rate ich dazu, E-Mail-Benachrichtigungen für Neuausgaben abzustellen, weil ich das Buch zum Testen des APIs verwende.

Was kann ich damit machen?

Mit dem API kann ich

- Vorschauen für das ganze Buch erzeugen, für Teile des Buches oder einzelne Dateien,
- das Buch veröffentlichen,
- eine Zusammenfassung des Buches bekommen,
- die Verkaufsdaten abfragen,
- Rabattcoupons verwalten.

Wie verwende ich das Perl-Modul?

Der Dreh- und Angelpunkt des Moduls ist ein Objekt vom Typ `WebService::Leanpub`, das für Aktionen rund um genau ein Buch verwendet wird. Aus diesem Grund gebe ich beim Aufruf von `new()` den API-Key und den Slug an.

```
use WebService::Leanpub;

my $wl = WebService::Leanpub->new($api_key, $slug);
```

Mit diesem Objekt rufe ich verschiedene Methoden auf, je nachdem, was ich machen möchte. Diese Methoden geben als Antworttext den Text des Web-APIs aus (JSON-Format), den ich ausgeben oder interpretieren kann.

Das Erzeugen vollständiger und partieller Vorschauen ist sehr einfach, dafür gibt es je eine Methode. Welche Dateien für die Vorschau herangezogen werden, steht in den Dateien *Book.txt* beziehungsweise *Subset.txt*, Autoren bei Leanpub wissen, wie diese Dateien auszusehen haben.

```
$pv = $wl->preview();

$pv = $wl->subset();
```

Eine Vorschau für eine einzelne Datei zu erzeugen, ist etwas komplizierter, weil die Datei via POST-Request an das Web-API gesendet wird. Das folgende Beispiel zeigt, wie eine Datei geöffnet wird, ihr Inhalt in eine skalare Variable eingelesen und dann zum Web-API geschickt wird.

```
if (open(my $input, '<', $filename)) {
  local $/;
  undef $/;
  my $content = <$input>;
  close $input;

  $pv = $wl->single({ content => $content });
}
```

Ein Buch zu veröffentlichen ist einfacher, in der Hashreferenz `$opt`, die ich als Argument angebe, kann ich mitteilen, ob die Leser eine E-Mail bekommen sollen und was darin stehen soll. Details stehen in der Handbuchseite des Moduls.

```
$pv = $wl->publish( $opt );
```

Da das Erzeugen einer Vorschau und das Veröffentlichen lediglich durch das Web-API gestartet werden und dann asynchron innerhalb von Leanpub weiter laufen, will ich vielleicht wissen, wie der Stand der Dinge ist. Dazu kann ich den Status des letzten Jobs abfragen:

```
$pv = $wl->get_job_status();
```

Um die Buchzusammenfassung zu erhalten, rufe ich die Funktion `summary()` auf. Das Web-API benötigt hierfür keinen API-Key, das Perl-Modul für `new()` schon. Will ich wissen, wie die Ausgabe ohne API-Key aussieht, kann ich beim Aufruf von `WebService::Leanpub->new()` einen falschen API-Key angeben.

```
$pv = $wl->summary();
```

Für die Verkaufsdaten gibt es zwei Methoden. Die eine liefert mir eine Zusammenfassung, die andere die Daten zu den einzelnen Verkäufen. Die zweite Funktion liefert mir die Daten zu den 50 letzten Verkäufen. Will ich ältere Daten haben, gebe ich an, die wievielte Seite ich sehen möchte.

```
$pv = $wl->get_sales_data();

$pv = $wl->get_individual_purchases( { } );

$pv = $wl->get_individual_purchases( { page => 2 } );
```

Rabattcoupons kann ich erzeugen, ändern und auflisten. Details stehen in der Modul-Dokumentation.

```
$pv = $wl->create_coupon(\%lopt);

$pv = $wl->update_coupon(\%lopt);

$pv = $wl->get_coupon_list()
```

Ich will nichts programmieren

Das brauche ich auch nicht (mehr), denn mit dem Kommandozeilenprogramm `leanpub`, das in der Distribution enthalten ist, kann ich das Web-API recht einfach in Makefiles nutzen. Das Programm nutzt `Getopt::Long` und `Pod::Usage`, kann also einen kurzen Hilfetext oder die komplette Handbuchseite ausgeben, wenn man es darum bittet.

```
$ leanpub --help
Usage:
     leanpub [options] command [command options]
...
```

Generell teile ich dem Programm beim Aufruf mit einem Befehl mit, was ich will. Die Befehle heißen in etwa, wie die Methoden. Vor dem Befehl kommen globale Optionen, dahinter Befehlsoptionen. Die Handbuchseite ist da sehr ausführlich.

```
Options:
  -api_key=key
          Provide the Leanpub API key to be used for all actions.
  ...
  -really State that you really intend to do the command (e.g. publish).
  ...
  -slug=your_book
          Provide the book's slug.
  ...
```

Von den globalen Optionen sind `-api_key` und `-slug` so wichtig, dass ich sie nicht immer eingeben muss. Nämlich genau dann, wenn ich sie in einer Konfigurationsdatei namens *.leanpub* angegeben habe.

```
$ cat .leanpub
# configuration for leanpub
#
api_key = my_api_key_from_leanpub
slug    = using-the-leanpub-api-with-perl
```

Etwas kurios mag vielleicht die Option `-really` anmuten. Also wirklich? Ich hätte diese auch `-do_as_I_say` nennen können, aber das erschien mir zu lang. Diese Option ist nur beim Befehl `publish` wirksam und genau dann ist sie auch erforderlich. Das ist so etwas wie eine Notbremse, damit man nicht unbeabsichtigt das Buch veröffentlicht, wenn es eigentlich noch nicht bereit ist.

Fazit

Das Modul *WebService::Leanpub*, genauer gesagt, das darin enthaltene Kommanodzeilenprogramm `leanpub` ist bei mir zusammen mit einem Makefile bei der Buchproduktion im täglichen Einsatz. Es fügt sich nahtlos in meinen Workflow ein, der aus einem Editorfenster, einer Shell und einem PDF-Betrachter besteht. Ich vermeide damit die Unterbrechung (und mögliche Ablenkung) durch das Umschalten zum Webbrowser.

Über den Autor

Mathias Weidner ist seit vielen Jahren Systemadministrator für Linux und Netzwerke in Wittenberg. Vorher arbeitete er in der Softwareentwicklung, zuletzt an Serversoftware für diverse Protokolle.

Nebenbei schreibt er Bücher und Perl-Module. Letztere sind am einfachsten auf CPAN zu finden.