Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Diplom Thesis Informatics

Mit der Trello-API rummuckeln

Sebastian Engel

24th July 2012

Referees

Name Erstgutachter
(Bioinformatik)
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Name Zweitgutachter
(Biologie/Medizin)
Medizinische Fakultät
Universität Tübingen

Engel, Sebastian:
Titel der Arbeit
Diplom Thesis Bioinformatics
Eberhard Karls Universität Tübingen
Thesis period: 13th march to 11th September 2012

Abstract

Trello is a collaboration webservice to manage projects and assign their todo items to co-workers. There are many collaboration tools today, but most of them are very basic. Trello is very extensive and it is optimal fo small businesses. But although it works fine like it's supposed to it has its limits. Trello as its state now is a closed system. Nothing gets in or out unless you use Trello itself. But sometimes it would be handy if you were able to get content from Trello out into other applications. For example a CMS which should contain completed theses which you are already managing in Trello.

So this thesis addresses small scripts which let Trello interact with other webservices and applications. For this purpose I wrote a wrapper of the Trello API in Ruby to accomplish this task in the most dynamic way possible.

Acknowledgements

Write here your acknowledgements.

Contents

List of Figures

List of Tables

Nomenclature

API	Application Programming Interface
CLI	Command Line Interface
CMS	Content Management System
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
REST	Representational State Transfer
URI	Uniform resource identifier
XML	Extensible Markup Language

Chapter 1

Introduction

Blablabla.....

Die Arbeit gliedert sich dazu wie folgt: Die Grundlagen von BlaBlaBla werden in Kapitel 1 erarbeitet. ... Eine Diskussion und ein kurzer Ausblick im Kapitel **??** beschließen diese Arbeit.

Bevor wir uns der Auswertung bzw. Bewertung der gewonnenen Primärdaten zuwenden, wollen wir zunächst einige grundlegende Begriffe der deskriptiven Statistik wiederholen.

Chapter 2

Principles

2.1 Trello

2.1.1 How Trello works

Trello is a webservice by the New York City based web corporation Fog Creek Software[1]. It is a collaboration tool to manage projects, launched in 2011[2].

Figure 2.1: A Trello board.

There is the concept of so called *boards* which contents several configurable lists. Figure 2.1 shows a board with the three standard lists *To Do*, *Doing* and *Done*. In these lists the user can create todo items. These todo items are called *cards*. The cards can contain several additional data. Each card has a title and maybe a description, some asigned members, a due date, some

[1] Official Fog Creek Software website: http://www.fogcreek.com
[2] The original launch post in the Trello blog: http://blog.trello.com/launch/

labels, votes, checklists, comments and attachments. The creator of the board is the owner in the first place and the owner can add other Trello users to his boards and cards. So everone who's working on a project can see whats going on at the moment. Users who are assigned to a board can even create new todo items by themselves. If somebody works at more than one company with many projects each there is the concept of *organizations*. This is useful in order to ensure a clear separation.

2.1.2 Why Trello

Trello is not just one of hundreds of thousands of todo applications. It is streamlined for the purposes of small businesses. So for the needs in the university with small groups of people working on the same things it is perfect. Trello has proofed its value several months already. The Trello website is written in HTML 5 with the use of AJAX where it makes sense. Trello provides an iOS [tre12a] and Android [tre12c] app. Both are constantly evolving. So the system is state-of-the-art. In addition the company behind Trello is not just a start-up with three employees. Thats important, too. A product of a small business, which is based just on the enthusiasm of the founders often doesn't last long. Fog Creek Software is over ten years old and has several products.

The first wish was to see the due dates of the cards anybody is assigned to in Google Calendar. But thinking about that there were many other use cases for small scripts which could run as cron jobs on a server to serve several regular tasks. These scripts are described in more detail in Chapter 3.

2.1.3 Trello API

Trello has an API which is still in beta at the moment I'm writing this. But it is already very extensive. [tre12d]

REST

The Trello API is a *RESTful* web API. That means that the API is conform to the REST design model. REST is a common style of software architecture for distributed systems. An implementation of a REST web service follows four basic design principles:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.

- Transfer XML, JSON, or both.

[res12]

Authentification

Though the scripts which are used here need access to private boards in Trello there has to be any kind of authentification. For user applications with a frontend the Trello API provides OAuth2. But because of the concept of OAuth2 the user is required to enter his Trello username and password. [oau12] My scripts are supposed to run on servers as cron jobs. There is no user who could manually enter data. For this kind of applications Trello provides a key/token-system. Every user has a private key. Whith this key the user can generate a token. This token will be send along every request to the Trello API. The token tells Trello which scope the request can see. While generating a token one can specify the scope of the token and when it will expire. The possible expriations of a token are between one day and never. In our case we will use *never*. To generate a token one has to visit a special URL: `https://trello.com/1/authorize?key=SUBSTITUTEWITHYOURPRIVATEKEY &name=My+Application&expiration=never&response_type=token &scope=read,write` In this example the token would never expire and could read and write everything the user can access with the API. Other valid values instead of `never` for expiration would be `1day`, `30days`. `30days` is the default value. [tre12b]

2.2 JSON

All the responses to Trello API calls use JSON. It is a subset of the JavaScript programming language. Despite its relation to JavaScript it's language independent. JSON is da data-interchange format like XML. But JSON is built on two structures. One is a list of key/value pairs. In most programming languages this is realised as an hash, struct, object or associative array. The other structure is an ordered list of values. This is realised as array, list, vector or sequence in popular programming languages. In JSON itself these structures are called *object* and *array*. Objects start and end with braces. Each key is followed by a colon and the key/value pairs are separated by commas. Arrays start and end with brackets. The values are separated by commas. Both can be arbitrary nested. At every point one of my script saves content at any other place than Trello it's in the JSON format, too. That's because it guarantees easy compatibility with Trello. JSON can be saved in files, too. A JSON file has the suffix `.json`. [jso12]

2.3 Ruby

Ruby is a modern general-purpose object-oriented programming language. Its big difference to most other languages is that it focuses on humans rather than computers.

Yukihiro Matsumoto, the designer of Ruby, said once:

> Ruby is simple in appearance, but is very complex inside, just like our human body.[rub00]

That means, that Ruby is very easy readable and is intuitive for humans even so it can perform complex tasks. This is achieved with English keywords instead of brackets and braces. The result for the programmer of this consistent philosophy is a very easy to read language which is also very plain. Because of the English words instead of abstract characters Ruby is easy understandable. Even non-programmers understand mostly whats going on. So programmers produce way less errors while writing the code. A wrongly spelled word is more intuitive recognisable than a missing bracket or semicolon. [rub12a] More about Ruby can be found at `http://www.ruby-lang.org`.

2.3.1 RubyGems

Ruby has a good amount of methods and classes every Ruby installation provides. But there are hundreds of extensions for special use cases – to communicate with RESTful Web APIs for example – made by third party developers. In Ruby such extensions are called *gems*. To manage and publish these third party libraries theres is the standard *RubyGems*. It provides a standard format for third party libraries for Ruby, a tool to manage the installation of gems and a server for distributing the gems. [rub12c] Some Ruby distributions are delivered with several gems. Gems can be added to an existing Ruby installation at any time.

To install an additional gem on a Unix operating system the following command can be used:

```
gem install gemname
```

Where `gemname` is the name of the respective gem. If the installation performed without errors, the gem is ready to use. [rub12d]

To use an installed gem in a Ruby script the following code at the top of the script before the code starts is necessary:

Again, *gemname* stands for the name of the respective gem. If any gems are used in these script which are not part of the Ruby standard distribution,

Listing 2.1: Using the gem *gemname*

```
1 require 'gemname'
```

they are listed at the beginning of the related description. Further information at http://doc.rubygems.org and http://rubyforge.org/projects/rubygems/.

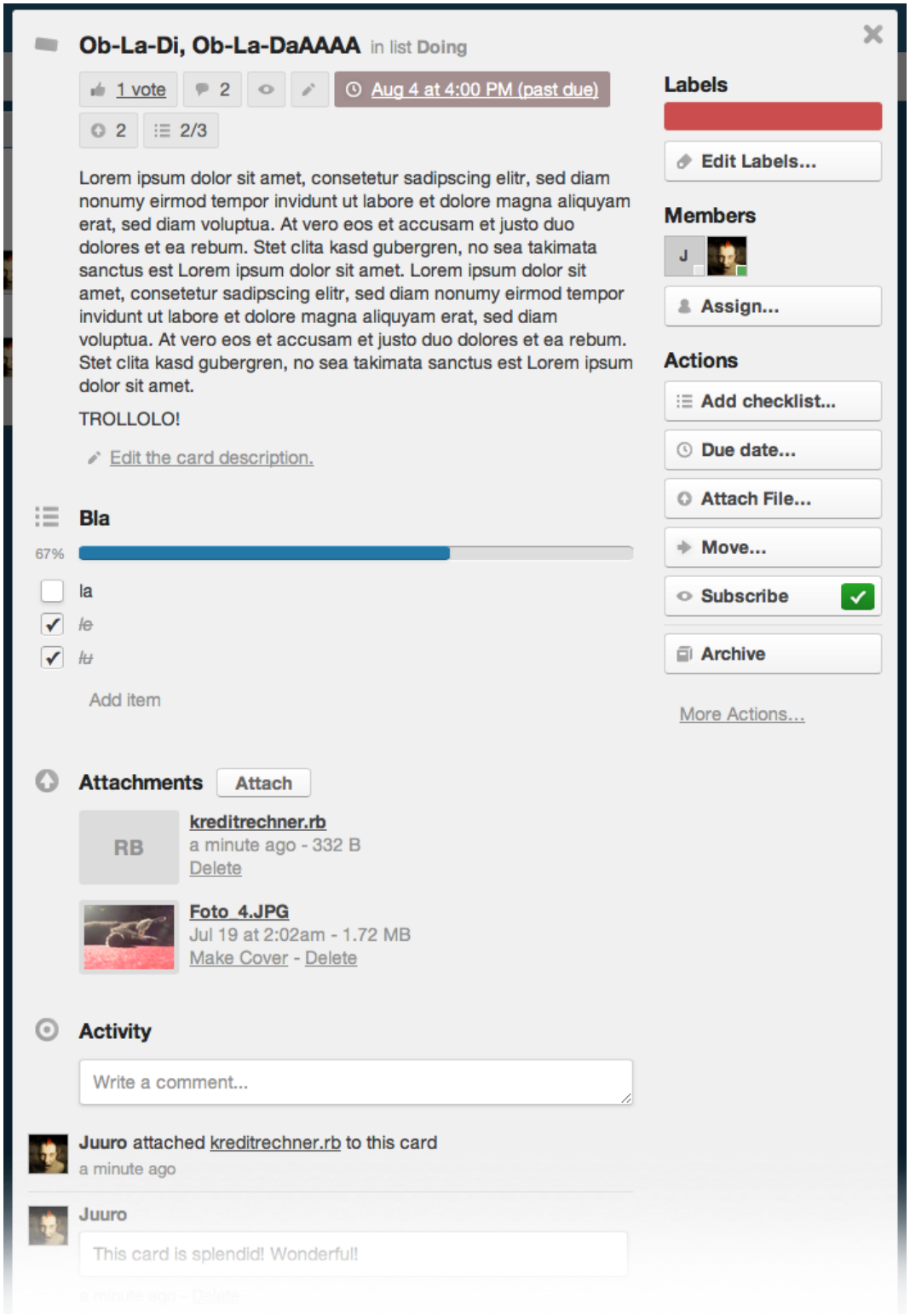

Figure 2.2: A opened card in Trello.

Chapter 3

Applications

3.1 Trello API wrapper

These scripts fulfill very different tasks, but they have also much in common. For example almost every script loads single cards. At least potentially. So I wrote a set of functions and classes which represent Trello for Ruby. This is kind of a translation of Trello to Ruby and vice versa. Additionally now the scripts can use the functions and in consequence they can stay very lightweight and clean. Almost everything that's possible whith the Trello API is possible with this API wrapper, too. But due to the fact that the Trello API is still in beta, there can always be errors and missing features.

The API wrapper has also functions to pre-process data for Ruby. From a developers point of view, Trello is all about cards. Cards are the only things in Trello with real data, not just meta data. So if the task is so get a board from the API it means to get the cards of the board. There is an API call to get all cards which are in a specific board. But with this call the developer doesn't get all information about the cards. So the API wrapper has to execute the API call for a single card to cummulate all information about all cards of the board. This is the function of the API wrapper to keep the actual script clean. So the developer can work with the data and hasn't to worry about determining them.

3.2 Command Line Interface

Almost every script needs some informations. An information which every scripts need is key and token of the user which account should be used for the access to Trello. The scripts have to know which cards, lists and boards they have to look at. So these information has to be passed to the scripts, too. At first we set this information at the top of the script. But it emerged that it's

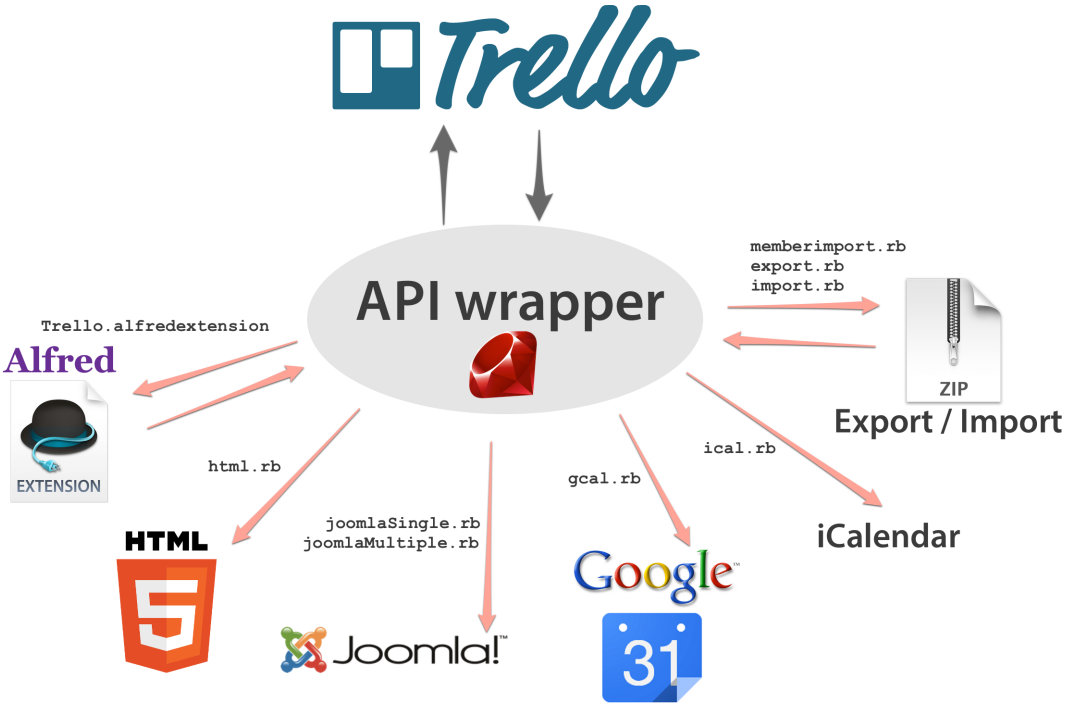

Figure 3.1: Connections between Trello, the API wrapper and the actual features. [rub][htm][joo12][goo12]

very unpractical to hard code this in each script. So it would be impossible to use the same Ruby file with several Trello accounts. For every Trello account the user has to generate a dedicated file. The solution for this problem is a command-line interface (CLI). With a CLI the user can pass information to the script in a predefined format, so the script knows exactly what to do. For every other call the user can specify different information for one and the same script.

The Ruby class OptionParser[rub12b] provides easy customisable command-line option analysis. The developer is able to specify its own options for each script. For this purpose a dedicated class is used. In order to let the actual script *know* about the CLI arguments the developer has to require the respective CLI class with the command-line option definitions.

Listing 3.1: Example usage of a script with CLI.

```
1 ruby html.rb -c 4ffd78a2c063afeb066408b8
```

An example usage of a script with CLI would look like Listing 3.1. The `-c` is an comman-line option. If there is a string behind the option, like in this case, the string is a so called *argument*. But there are command-line options which stand for its own. Those are called *flags*. Flags are only for polar decisions.

Listing 3.2: Definition of a command-line option

```
1 # Trello list(s)
2 opts.on("-l", "--lists x,y,z", Array, "Ids of one or more
    Trello lists.") do |lists|
3   options.lists = lists
4 end
```

Listing 3.2 shows the definition of the option `-l` for passing one or more IDs of lists to a script. In Line 2 the word `Array` casts the list argument to an Array object.

OptionParse provides an automated help option. If the user types

```
ruby script.rb -h
```

he gets the explanation the developer wrote in the CLI class for this script with all possible options. This list ist automatically generated by the definitions of the command-line options like in Listing 3.2. It works with `-help` and `--help` instead of `-h`, too.

Listing 3.3: Output of the `-h` option.

```
1  Usage: ical.rb [options]
2  Select the input cards with -c, -l, -b or -a
3
4  Specific options:
5   -a, --[no-]all          Set this if all due dates of all
       cards of all boards this user can see shall be used.
6   -l, --lists x,y,z       Ids of one or more Trello lists.
7   -b, --boards x,y,z      Ids of one or more Trello boards.
8   -c, --cards x,y,z       Ids of one or more Trello cards.
9   -k MANDATORY, --key     Your Trello key.
10  -t MANDATORY, --token   The Trello token.
```

Listing 3.3 shows the Output of `ruby ical.rb -h`. These are the basic CLI commands used by every script. For some scripts there are additional commands. They are explained in their respective sections.

3.3 Export to HTML

Used gems:

- `erb`
- `json`
- `open-uri`
- `pp`
- `kramdown`

The `html.rb` script exports the data of one ore more cards to an HTML file. The resulting HTML file lists the cards one below another. The order is determined by the order in the command-line argument. If the command-line looks like listing 3.4 the script will process the list with the id `4ffd78ff7f0c71780cc5aa1c` at first. That means in the HTML file are all cards in this list and below these the single card with the id `4ffd78a2c063afeb066408b8`. Each card is displayed with all their information. This includes title, description, members, due date, labels, votes, checklists, comments and attachments.

Listing 3.4: Example for a `html.rb` call.

```
1 ruby html.rb -l 4ffd78ff7f0c71780cc5aa1c
     -c 4ffd78a2c063afeb066408b8
```

Trello itself distinguishes between photos and other attachments. Normal attachments are linked under the description. Photos are embedded in the HTML code as thumbnails. The `html.rb` script does the same.

3.3.1 Markdown

Markdown is a small lightweight plain text formatting snytax, designed by John Gruber. It's designed for the use with blogs and CMS. In these use cases HTML is often too much. Markdown represents most of the features of HTML that are needed for writing. The designer of Markdown, had the goal that a text, written in Markdown, is still easy to read. John Gruber provides a software tool, written in the Perl programming language, that converts the Markdown formatted text to valid HTML. [mar04]

Listing 3.5 shows a small example of Markdown. The `###` in line 1 is a header equal to `<h3>` in HTML. The first list item of the ordered list in line 3 contains italic and bold words. In line 6 is a horizontal line. After a normal line of text a code block starts in line 10. At the end in line 14 there is a picture with title and alt texts. After the conversion it looks like listing 3.6 in HTML. The

Listing 3.5: Example for a text written in Markdown.

```
### iCloud:

1.   Shared Photo Streams Now you can *share* just the **
   photos** you want, with just the people you choose.
2.   Reminder

--------------------------------

Here is an example of AppleScript:

    tell application "Foo"
        beep
    end tell

![Apple logo](http://upload.wikimedia.org/wikipedia/
   commons/f/fa/Apple_logo_black.svg "Apple logo")
```

appearance, of course, depends on the used CSS on the respective websites. The appearance in Trello is like in figure 3.2.

Meanwhile Markdown became quite popular. Many blogging platforms support it, at least there are Markdown plug-ins for most platforms. Trello supports it in the description of cards. In the unlikely case that markdown reaches its limits inline HTML can be used. The only restriction ist, that HTML block-level-elements have to be separated to the previous and following Markdwon blocks.

For converting Markdown to HTML the gem kramdown is used. Figure 3.3 decribes the convert options of kramdown. It converts to LaTeX and a special kramdown format, too. The kramdown format is an extended Markdown syntax. As input formats it accepts HTML and kramdown besides standard Markdown. [kra12] These additional features of kramdown might be useful for future approaches. To generate lists bibliographies for scripts, papers or books.

Listing 3.6: Listing 3.5 converted to HTML.

```
<h3>iCloud:</h3>

<ol>
  <li>Shared Photo Streams Now you can <em>share</em>
      just the <strong>photos</strong> you want, with
     just the people you choose.</li>
  <li>Reminder</li>
</ol>

<hr>

<p>Here is an example of AppleScript:</p>

<pre>
  <code>tell application "Foo"
  beep
  end tell</code>
</pre>

<p><img alt="Apple logo" title="Apple logo" src="http
   ://upload.wikimedia.org/wikipedia/commons/f/fa/
   Apple_logo_black.svg"></p>
```

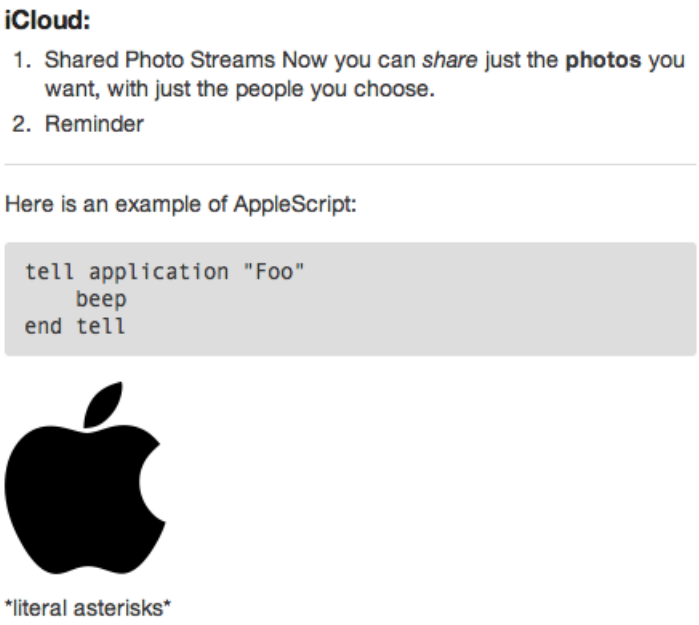

Figure 3.2: The browser view of the HTML converted from the Markdown in listing 3.5

3.3.2 Twitter Bootstrap Framework

3.3.3 HTML 5

3.3.4 CSS 3 / LESS

3.3.5 ERB / Templating

3.4 One way sync to Google Calendar

3.5 Export to iCalendar

3.6 One way sync to Joomla

3.6.1 For every card an article

3.6.2 All cards in one article

3.7 One way sync to WordPress

3.8 Backup

3.8.1 Export

3.8.2 Import

Filename option

The `-n` (or `-name`) argument for this script stands for the filename of the backup file which contains the exported Trello data. With `-n` the user can

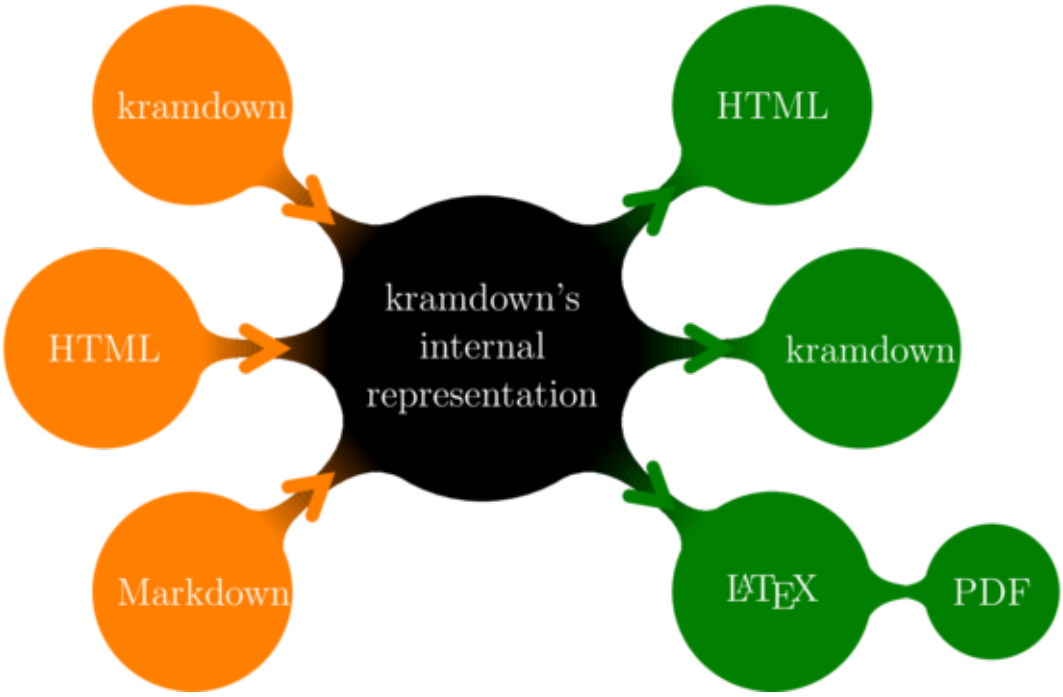

Figure 3.3: Overview about kramdowns converting options. [kra12]

specify a file to import. While processing the script first checks if the user has passed this argument. If not, it aborts. If the `-n` argument is given, the scipt proofes if the file is a ZIP file. For that it soesn't use the filename but the MIME type of the file.

TODO: listing design

Listing 3.7: Checking if the file has the MIME type "application/zip"

```
1 if `file -Ib #{@filename}`.gsub(/;.*\n/, "") != "
    application/zip"
2   puts "ERROR: The backup\index{Backup} file has to be a
      ZIP\index{ZIP} file!"
3   abort
4 end
```

In line 1 the `file -Ib #{filename}` is a bash call for receiving the MIME type of a file. Ruby executes it and with the gsub-Method it cuts the MIME part out of the received string. This shell script part in a ruby file is a bit dirty. But only for this small case it would be elaborately to use a seperate gem.

TODO: What's a MIME type?

3.8.3 Member import

Chapter 4

Conclusion

Chapter 5

Outlook

5.1 Trello Alfred Extension

Alfred [alf12] is a small Mac application which simplifies the way one can search the web or access all sorts of applications. It constist just of a input field which one cann access with a keystroke combination. It's like an extended Spotlight (on Mac) or Windows Search (on Windows). Developers can write extensions to access other webservices and applications with Alfred. It's even possible to run scripts with Alfred. With that possibility given it's perfect for acessing Trello while working in a fast and easy way.

There are three commands to add or read cards with this extension:

1. `trello board-name` will return the card-names and statuses of this board.

2. `trello board-name list-name` will return card-names and statuses of this list in this board.

3. `trello board-name text for a new card` will add a new card with the specified text to the first list of this board.

4. `trello board-name list-name text for a new card` will add a new card with the specified text to this list of this board.

If you enter `trello Berlin Visit the Reichstag` in Alfred the extension looks for a board called *Berlin*. If it finds nothign it looks for *Berlin Visit* and so on. So your board names shouldn't end with an imperative. The thought behind this operating principle is that it's very unlikely that a board name ends with an imperative and that imperatives are often used for card titles because cards are sort of a command.

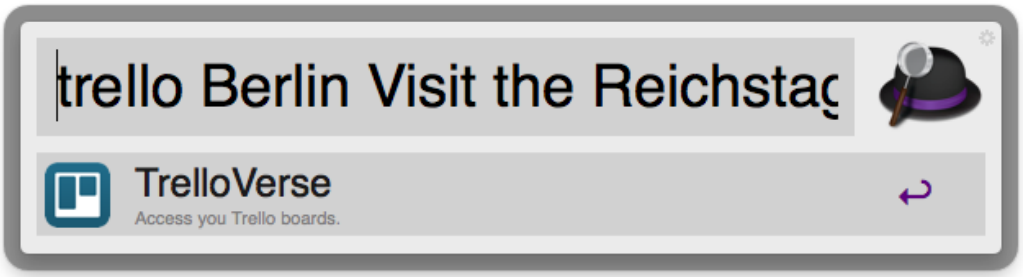

Figure 5.1: Alfred Extension for Trello: This command would add a card with the name *Visit the Reichstag* to the board called *Berlin*.

If you omit the text after the board name the extension will show you all card names of this board and its statuses.

Sometimes there are several boards with similar board names. In this case the extension will pick the "last"match. So if you have two boards called *Berlin* and *Berlin sightseeing* the extension will would pick *Berlin sightseeing*. This approach makes sense because if the extension would pick the first match, in this case *Berlin*, it wouldn't be possible to access *Berlin sightseeing*. In the case that one wants to access *Berlin* and add a new card beginning with *sightseeing* one has to put this board name betweet tick marks.

TODO: Code this and verify the practicability.

5.2 Native applications

Although Trello is an extremely good web-app, I'm of the opinion that a native application is always the better solution. The first reason is because it's a dedicated app and so it's integrated with the operationg system. Especially for todo-applications it's an advantage that they can access the systems notification system, or that they could completely vanish in the background so they don't bother the user while working. There are mobile applications for iOS [tre12a] and Android [tre12c] by Trello itself. But theres no Mac, Windows or Linux application.

A native application would even speed up the Alfred extension because the application could cache the data. So there hasn't to be an actual HTTP request for every command by the Alfred extension. And if a HTTP request necessary the user hasn't to wait because the application will handle the command in the background.

Bibliography

[alf12] Alfred app. http://www.alfredapp.com/, 08 2012.

[goo12] http://www.google.com, 08 2012.

[htm] W3c html5 logo. http://www.w3.org/html/logo/.

[joo12] www.joomla.de. http://www.joomla.de/, 08 2012.

[jso12] Json. http://www.json.org/, 08 2012.

[kra12] kramdown. http://kramdown.rubyforge.org/, 06 2012.

[mar04] Daring fireball: Markdown. http://daringfireball.net/projects/markdown/, 12 2004.

[oau12] Oauth community site. http://oauth.net/, 08 2012.

[res12] Restful web services: The basics. https://www.ibm.com/developerworks/webservices/library/ws-restful/, 08 2012.

[rub] Iap: Caffeinated crash course in ruby. http://sipb.mit.edu/iap/ruby/.

[rub00] [ruby-talk:02773] re: More code browsing questions. http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773, 05 2000.

[rub12a] About ruby. http://www.ruby-lang.org/en/about/, 08 2012.

[rub12b] Class: Optionparser (ruby 1.9.3). http://ruby-doc.org/stdlib-1.9.3/libdoc/optparse/rdoc/OptionParser.html, 08 2012.

[rub12c] Rubyforge: Rubygems: Projektinfo. http://rubyforge.org/projects/rubygems/, 08 2012.

[rub12d] Rubygems manuals. http://docs.rubygems.org/, 08 2012.

[tre12a] App store - trello. http://itunes.apple.com/us/app/trello/id461504587?mt=8, 08 2012.

[tre12b] Getting started — trello documentation. https://trello.com/docs/gettingstarted/index.html, 08 2012.

[tre12c] Trello android app available for download! — trello blog. http://blog.trello.com/trello-android-app-available-for-download/, 08 2012.

[tre12d] Trello documentation — trello documentation. https://trello.com/docs/index.html, 08 2012.

Index

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Diplomarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift