wahlversprechen Documentation

Release 1.0

Sebastian Theophil

November 01, 2014

Contents

1 Setup Guide **3**
1.1 Prerequisites . . . 3
1.2 Database Configuration . . . 3
1.3 wahlversprechen Configuration . . . 3
1.4 Start Play . . . 3
1.5 What to do next? . . . 4

2 Using wahlversprechen as an Editor **5**
2.1 Markdown Formatting . . . 5
2.2 Embedding Images . . . 5
2.3 Embedding Charts . . . 6

3 Importing Data **7**
3.1 The Data Model . . . 7
3.2 Importing from Google Spreadsheets . . . 7

4 Exporting Data in JSON **9**
4.1 JSON Endpoints . . . 9

5 Heroku Deployment Guide **11**

6 Indices and tables **13**

Contents:

CHAPTER 1

Setup Guide

1.1 Prerequisites

- Install a JDK >= 1.6 and PostgresSQL >= 9.3. I use postgresapp.com on the Mac.
- Download Play Framework 2.2 and unzip it to a folder of your choice.

1.2 Database Configuration

- Start the PostgresSQL console `psql`
- Create a new user `CREATE ROLE wahlversprechen WITH LOGIN PASSWORD ''` or a password of your choice instead of an empty password
- Create a database: `CREATE DATABASE wahlversprechen WITH OWNER wahlversprechen;`
- And create a database to run the test suite: `CREATE DATABASE wahlversprechen_test WITH OWNER wahlversprechen;`

1.3 wahlversprechen Configuration

- Clone the repository including submodules *git clone https://github.com/stheophil/wahlversprechen.git –recursive*
- *(optional) In ‘conf/application.conf‘ search for ‘db.default.password‘ and set it to the password you’ve assigned your wahlversprechen user if it wasn’t an empty password*
- *(optional) if your database is running on another machine, search for ‘db.default.url‘ and replace ‘localhost‘ with that machine’s hostname*

1.4 Start Play

- From the terminal, `cd` to the folder you’ve cloned the wahlversprechen repository into.
- Start `play` from this folder by typing your equivalent of `pathtoplay2.2/play`
- In the play console that should appear, type `run -Dhttp.port=9000 -Dhttps.port=9001`

- In the browser, go to localhost:9000 and you should see a message *"Database 'default' needs evolution", press "Apply this script now"*. This will create the necessary database tables.
- Now you should see a completely empty version of the app.

1.5 What to do next?

- See *Importing Data* on how to import data
- Edit the Admin preferences at localhost:9001/admin/prefs

That's it.

CHAPTER 2

Using wahlversprechen as an Editor

2.1 Markdown Formatting

When writing updates on campaign promises, you can use Markdown to format your texts. Markdown is a very simple text format that remains legible, even when it is not rendered as HTML. See the reference guide for more information. Here is an example:

```
# This would be a top level heading

And this is a short first paragraph with *italic* text and **bold** text.

## This is a second-level heading

Links, e.g., [CNN](http://www.cnn.com) are quick to write as well.

- When you need lists,
- just write a list.
```

Rendered as HTML it looks like this:

2.1.1 This would be a top level heading

And this is a short first paragraph with *italic* text and **bold** text.

This is a second-level heading

Links, e.g., CNN are quick to write as well.

- When you need lists,
- just write a list.

2.2 Embedding Images

The site stylesheets have minimal support for embedding images. At the moment, the site does not support image upload, so you have to store all images yourself, e.g. on Dropbox. Use the following template to get a centered image with caption text:

```
<div class="img">
        <img src="http://linktoyourimagehere.com">
        <p>Insert caption here</p>
</div>
```

2.3 Embedding Charts

While most text formatting can and should be done with Markdown, HTML can be used to embed charts e.g. Datawrapper.de can be used to create embeddedable, interactive charts to visualize important statistics.

Once you've created a chart with datawrapper or most other charting websites, the chart can be embedded using HTML code similar to the following:

```
<iframe src="http://cf.datawrapper.de/VLnBD/1/"
...
width="600" height="400">
</iframe>
```

where `http://cf.datawrapper.de/VLnBD/1/` is the address of the chart you've just created. For best results, this should be changed slightly to

```
<iframe src="http://cf.datawrapper.de/VLnBD/1/"
...
width="100%" height="400">

Please <a href="http://cf.datawrapper.de/VLnBD/1/">click here to see the chart</a>.

</iframe>
```

Note that `width="600"` was changed to `width="100%"` which allows the chart to always occupy 100% of the available screen width on a large computer screen as well as on a small phone screen. See here for an example. The height must be a fixed value. 300 or 400 work fine.

The additional line `Please <a href="http://cf.datawrapper.de/VLnBD/1/">click here to see the chart</a>.` is shown in environments that will not display the embedded charts, for example most RSS feed readers. It refers the reader to the website where he can see the chart.

CHAPTER 3

Importing Data

Currently, large amounts of data can only be imported from Google spread sheets. There is an open issue on github to fix this.

- Go to localhost:9001/admin/import
- Login when necessary. On a default installation use user name `test@test.net` and password `secret`
- Import some test data from a publicly shared Google Sheet. You can use e.g. `0AuS3i7YOiV-wdHM1dHhTY3lfTUJTa1VBNzJ0eDdpekE` which is this sheet

3.1 The Data Model

You can import arbitrary *statements*, e.g., campaign promises, that you want to track. A *statement* has an *author* who made this statement or promise. There is a very simple hierarchy among *authors*. An *author* is a *top level* author or it is not. This is used to represent campaign promises in coalition governments, where several parties competed in the elections based on their own programs and then form a coalition, often with a formalized coalition treaty. The coalition treaty would be the *top level author* and the individual party programs would be second level authors.

A statement from a second level author can reference a single statement from a top level author, i.e., a campaign promise from party A could refer to the statement in the coalition treaty it is most closely related to.

It is currently impossible, to edit the list of authors in the admin preferences. The list of authors is set when the application is started for the first time. See `app/Global.scala`. Again, there is an open issue on github.

3.2 Importing from Google Spreadsheets

If you want to import your own data, your Google spreadsheet needs to have the same column names as the test data:

- *titel* - the title of your campaign promise
- *zitat* - the exact quote that you want to track in Markdown format.
- *quelle* - the source of the quote in Markdown format.
- *ressort* - the promise’s category, e.g. “Foreign Affairs”, “Economics” etc
- *tags* - a comma separated list of tags
- *links* - when importing statements for a *top-level author*, a comma separated list of statement ids of second level authors. When the list includes more than one statement id for a single second level author, only a single id will be imported.

Yes, there is also an open issue to internationalize all texts in the application, including these column names.

You can import the same table multiple times. Previously imported statements are identified based on their title. Thus, you cannot change these after the first import. All other fields of already existing statements are updated.

CHAPTER 4

Exporting Data in JSON

4.1 JSON Endpoints

- `/json/tags` a list of all existing tags:

```
[
        {"id":937,"name":"EU","important":false},
        ...
]
```

- `/json/categories` an list of all existing categories:

```
[
        {"id":2,"name":"Education","ordering":2},
        ...
]
```

- `/json/authors` a list of all authors (e.g. “Government”, “Election Program 2012”):

```
[
        {"id":1,"name":"Coalition Treaty","ordering":1,"top_level":true,"color":"#ffffff","backg
        ...
]
```

- `/json/items/{author.name}` get the list of all promises made by `author` with the given name:

```
[
        {
                "id":680,
                "title":"This statement’s title",
                "quote":"\"Whatever he said\"",
                "quote_src":"[Here’s where he or she said it](http://cnn.com)",
                "author":"Coalition Treaty",
                "category":"Education",
                "tags":["EU","Research","Foreign Languages"],
                "ratings":[
                        {"rating":"InTheWorks","date":"2014-01-01T01:00Z"}
                        {"rating":"PromiseKept","date":"2014-01-23T17:12Z"}
                ],
                "linked_to":""
        },
        ...
]
```

- `/json/item/{id}` promise with `id`, also returns the text *entries* posted on the site:

```
{
        "id":680,
        "title":"This statement's title",
        "quote":"\"Whatever he said\"",
        "quote_src":"[Here's where he or she said it](http://cnn.com)",
        "author":"Coalition Treaty",
        "category":"Education",
        "tags":["EU","Research","Foreign Languages"],
        "ratings":[
                {"rating":"InTheWorks","date":"2014-01-01T01:00Z"}
                {"rating":"PromiseKept","date":"2014-01-23T17:12Z"}
        ],
        "entries":[
                {
                        "id":77,
                        "content":"# The text entry in Markdown formatting.\n May contain raw HT
                        "date":"2014-01-31T11:09Z",
                        "user":"Sebastian"
                }
        ],
        "linked_to":""
}
```

- `/json/item/{id}/relatedurls?[limit=x&offset=y]` get the list of web pages related to promise `id`, returns at most `limit` results starting at result `offset` in chronologically backwards order. Both `limit` and `offset` are optional.:

```
[
        {
                "id":488,
                "stmt_id":682,
                "title":"Shocker: Many Europeans speak several languages",
                "url":"http://cnn.com/blabla",
                "confidence":3.03125,
                "lastseen":"2014-05-29T08:46Z"
        },
        ...
]
```

- `/json/relatedurls?from=from_date[&to=to_date]` get the list of web pages related to campaign promises found between `from_date` (inclusive) and `to_date` (exclusive) ordered by last seen date in ascending order. `from_date` and `to_date` must be of the form "YYMMDD". Thus requesting `/json/relatedurls?from=20140529&to=20140910` might result in

```
[
  {
                "id":488,
                "stmt_id":682,
                "title":"Shocker: Many Europeans speak several languages",
                "url":"http://cnn.com/blabla",
                "confidence":3.03125,
                "lastseen":"2014-05-29T08:46Z"
        },
        ...
]
```

CHAPTER 5

Heroku Deployment Guide

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*