Public Feed-Extension for Hivebrite

Technical Documentation

Revision: 1.1, Draft
Date: 28.05.2018

This web application allows to easily embed information coming from multiple Hivebrite communities on websites. The concept is similar to the Facebook Page Plugin.
The following document describes the technical backend as well as the usage of the feed-extension.

Architecture

The architecture of the application is split into several modular parts:

- ➔ API Layer
- ➔ Storage Layer
- ➔ Presentation Layer

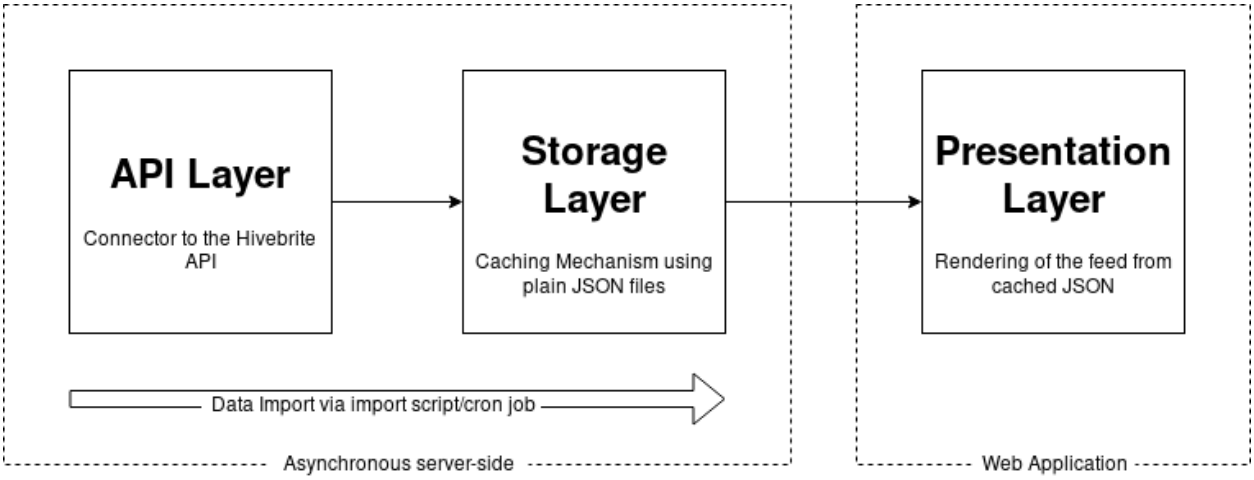

Image 1: Schematic of the architecture.

This layers are encapsulated separately and used by the web application to update and render the feed. Through the module nature of the architecture, it is easily possible to extend the functionality.

API Layer

The API layer implements parts of the Hivebrite API, that are needed for this project to function properly. It can handle data from multiple Hivebrite Communities with one instance of the web application.

This development is based on the Hivebrite Admin API documentation.

Authentication

To authenticate against the API, username, password, UID and secret key are needed. The latter two can be obtained from the Hivebrite backoffice plattform. In exchange for this information, a token is issued. This token is then used to obtain further information.

Importing Data from Hivebrite

Due to limitations of the Hivebrite API, only events will be imported in the first stage of development. This can be extended to include more data as the Hivebrite API evolves.

To improve performance of the feed for the end user, the data will be imported on a regular basis. The import script can be triggered via command line or URL. It is recommended to use a cron job to keep data up to date. It then fetches and processes data provided by the Hivebrite API. Using the storage layer, the processed data will then be stored and rendered by the presentation layer.
The script can manually be called from the command line via *./bin/console feed:import* or via the URL *https://feed.yourdomain.io/import*.

Storage Layer

Data fetched by the cron job will be stored within the application using the storage layer. Due to the nature of the data used by the application, the storage layer uses the JSON format. The gathered information will be cached in plain files directly on the filesystem..


```
{
  "communityName": "Ashoka Sandbox",
  "communityUrl": "https://ashoka-sandbox.preprod.hivebrite.com",
  "lastFetched": "2018-06-13T12:56:55+00:00",
  "feedItems": [
    {
      "type": "event",
      "published": "2018-05-22T21:00:00+00:00",
      "title": "Stakeholder Workshop",
      "body": "<p>Cool Stakeholder Workshop</p>\r\n",
      "url": "https://ashoka-sandbox.preprod.hivebrite.com/networks/events/1434",
      "meta": {
        "organizer": "Impact Transfer by Ashoka (Sandbox)",
        "location": "Vienna, Austria",
        "venue": "",
        "start": "2018-05-22T21:00:00Z",
        "end": "2018-05-22T22:00:00Z",
        "public": false
      }
    }
  ]
}
```


Image 2: The cached JSON object.

Each Hivebrite community is represented by one JSON file. This file holds all information, the web application needs to render the feed and is used directly when rendering the feed on project websites.

Storage functionality can easily be replaced/extended by a database in the future, if necessary.

Presentation Layer

The presentation layer will render a feed. This feed can then be implemented on project websites. It will be implemented either via iframe or JavaScript.

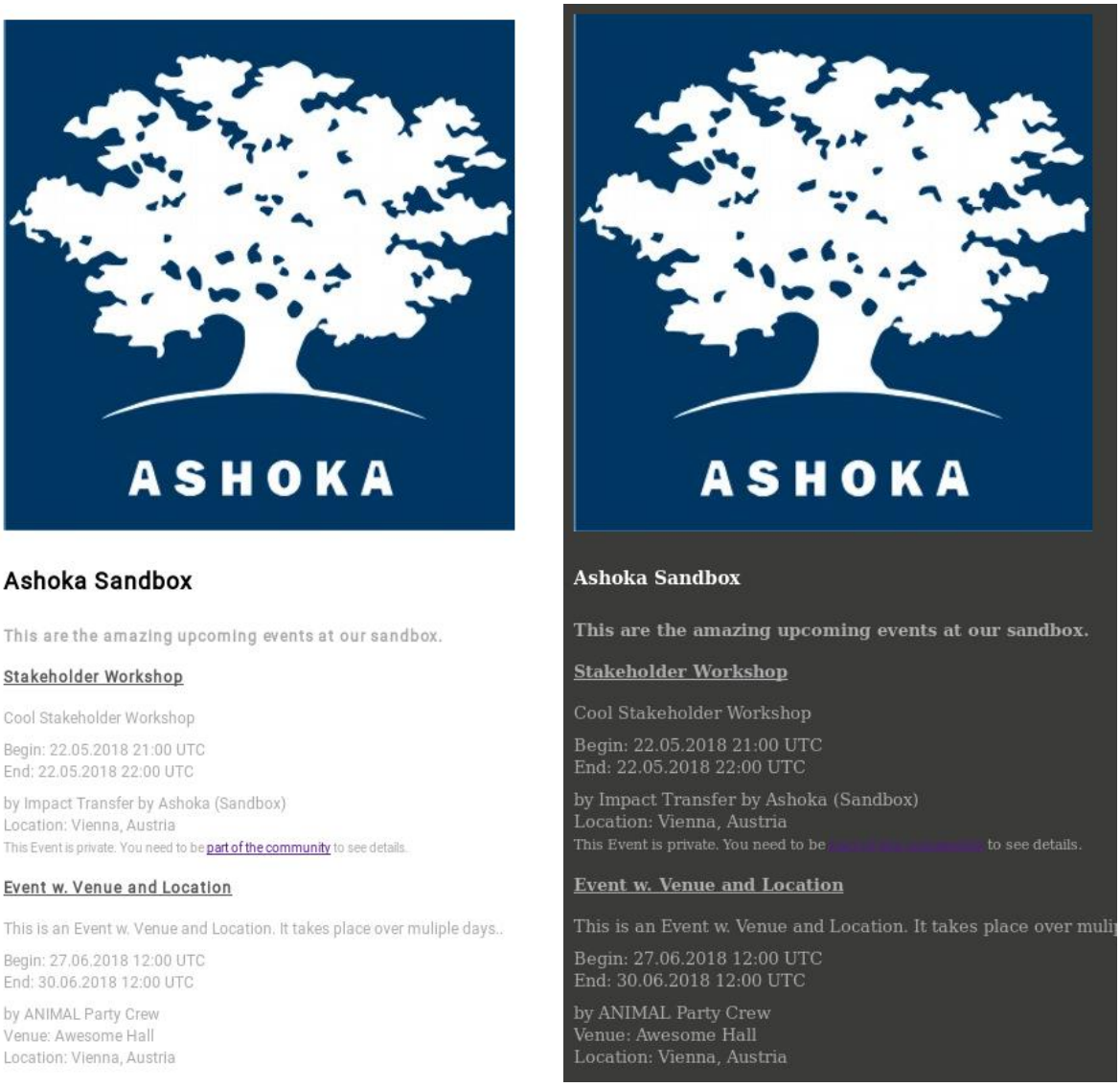

Image 3: The rendered feed with default styles.
Image 4: The rendered feed customized to fit into the Impact Transfer website.

Configuration

A basic set of rules can be configured when implementing the feed on the website of a project. The configuration will include style (colors, etc..) and functional(community, etc..) elements. See Customisation.

Installation and Setup

The web application needs to installed in a web server. The public part of the application must be publicly accessible in order to render the feed. One instance of the application can power multiple feeds from multiple Hivebrite communities.

Requirements

In order to host the application, a web server must fulfill certain requrements. It is build upon Symfony 4 and therefore requires PHP 7.1.3 or higher to run, in addition to other minor requirements. The hosting provider must also allow the setup of cron jobs (through command line or URL) to asynchronous fetch, process and store the data.
No database setup is required.

Symfony 4 is released unter MIT License.

Installation on the Web Server

All files of the source code must be transferred to the web server.

The web application uses composer to manage its external dependencies. Depending on the type of access to the server, the composer installation routine must be ran either locally or directly on the server. The **--no-dev** option should be used while installing to avoid the installation of unnecessary dependencies. In case of a local installation, the resulting vendor folder needs to be transferred to the server too.

The domain (i.e. https://feed.yourdomain.io) must point to the *public/* folder.

Configuration

The application allows simple configuration through the *hivebrite.yaml* file, located in *config/api/*. It the file is not present, copy *hivebrite.yaml.dist* to *hivebrite.yaml* or create a new file with that name. The file must contain valid YAML for the web application to function correctly. The configuration allows for multiple communities to be configured as YAML array.

```
communities:
    - name: Ashoka Sandbox
      description: This are the amazing upcoming events at our sandbox.
      logo: https://pbs.twimg.com/profile_images/495093403/ashokaindia2_400x400.jpg
      slug: ashoka-sandbox
      url: https://<your-community>.hivebrite.com

      email: email@yourserver.io
      password: Super$ecretPassw0rt
      uid: <secret-uid-hash>
      secret: <secret-key-hash>
```

Image 5: Configuration of the web application.

The following options are available:

- **name** Human-readable name of the community
- **description** A short introduction of the community, shown below the name *(optional)*
- **logo** A logo to be shown above name and description *(optional)*
- **slug** Machine-readable name of the community. Should only contain lower-case letters and hyphens (-). The value needs to be unique and is used to identify the community
- **url** Publicly accessible URL of the community
- **email** Email of the account that is used to gather the data from the API
- **password** Password of the account that is used to gather the data from the API
- **uid** UID of the community. This information can be found on the Hivebrite backoffice portal
- **secret** App secret of the community. This information can be found on the Hivebrite backoffice portal

Cron Setup

The cron job can be set up to run a command or access a website on a regular basis. It is recommended to execute the cron job at least one time per hour.

Via Command line:
0 * * * * /<path-to-the-web-application>/bin/console feed:import

Via URL:
0 * * * * wget https://feed.yourdomain.io/import

Usage and Implementation in existing Websites

The web application generate a valid HTML webpage, that can be imported into any existing website via a IFrame.

Customisation

In order to blend into existing websites, the design of the feed can be modified by using parameters in the URL.

The following options are available:

- **community** Slug (machine-readable name) of the community. Can be found the the configuration of the web application
- **font** A valid Google font name (i.e. *Roboto*)
- **backgroundColor** Bachground color of the feed (i.e. *#3B3B39*)
- **primaryFontSize** Primary font size (i.e. *14px*)
- **primaryFontColor** Primary font color (i.e. *#ffffff*)
- **secondaryFontSize** Secondary font size (i.e. *13px*)
- **secondaryFontColor** Secondary font color (i.e. *#AEAEAE*)
- **tertiaryFontSize** Tertiary font size (i.e. *13px*)
- **tertiaryFontColor** Tertiary font color (i.e. *#AEAEAE*)

The resulting URL should look something like this:

https://feed.yourdomain.io/?**community**=impact-transfer&**font**=Roboto&**primaryFontSize**=14px&**primaryFontColor**=%23ffffff&b=13px&**secondaryFontColor**=%23AEAEAE&**tertiaryFontSize**=13px&**tertiaryFontColor**=%23AEAEAE&**backgroundColor**=%233B3B39

Example Implementation

An example implementation of the feed via IFrame can be found below. The IFrame can be styled using CSS (i.e. height of the feed or border).

```
<iframe
    style="border: 0px none; height: 200px;"
    src="https://feed.yourdomain.io/?community=impact-transfer&font=Roboto&primaryFontSize=14px&primaryFontColor=%23ffffff&b=13px&secondaryFontColor=%23AEAEAE&tertiaryFontSize=13px&tertiaryFontColor=%23AEAEAE&backgroundColor=%233B3B39"
```

/>

Extended Usage

As an alternative tp IFrames, the web application provides data also as plain JSON files. This files can be used to render the feed using another programming language (i.e. JavaScript or PHP). Using this method a deeper integration into existing projects can be accomplished.

The JSON object of a community can be simple accessed via a public URL.
Example: *https.//feed.mydomain.io/feeds/<slug-of the-community>.json*

The slug of the community can be found in the configuration of the web application.