Open Data Events Documentation

Release 0.1

Makina Corpus

Sep 27, 2017

Contents

ODE stands for Open Data Events.

It's a RESTful web API that allows clients to interact with two kinds of resources: events and sources. Events are human events such as concerts, conferences, exhibitions, etc. Sources are URLs pointing to event data streams. To collect event data from sources, we provide a `harvest` script, which may be invoked as a cron job.

ODE is written in Python using the Pyramid web framework and Cornice, a REST framework for Pyramid. It is tested against both Python 2.7 and Python 3.3.

CHAPTER 1

Development install

We provide a `Makefile` to help with setting up ODE on your machine. You probably want to do this in a virtual environement.

Grab the source code:

```
$ git clone https://github.com/makinacorpus/ODE.git
```

Install the app with its dependencies:

```
$ make develop
```

Run the development server:

```
$ make serve
```

CHAPTER 2

Web API

The ODE API is based on Collection+JSON, a JSON-based read/write hypermedia-type designed to support management and querying of simple collections. Events can also be represented in iCalendar and CSV formats. Note that we don't use the Collection+JSON format for error reports because it doesn't allow us to specify a different error message for each field. Errors are therefore reported in Cornice format.

Assuming you've setup your web server to serve this API at the root of your domain name, your endpoints will be:

- /v1/events
- /v1/sources

Collection+JSON and ODE follow typical RESTful conventions:

- POST to add a new item, eg. `POST /v1/events` to add a new event
- PUT to update and existing item, eg. `PUT /v1/sources/123` to modify the source with id `123`
- GET to get a list of items or a specific item, eg. `GET /v1/sources` to get a list of sources or `GET /v1/events/123` to get a representation of the event with id `123`.
- The `Accept` request header to specify which format you want to retrieve, eg. `Accept: text/calendar` to get events in iCalendar format.
- The `Content-Type` request header to specify which format you provide, eg. `Content-Type: text/csv` to inform the server that you're sending comma-separated values.

To make POST and PUT requests, you must provide an HTTP header that will identify the event provider:

```
X-ODE-Provider-Id: xyz123
```

Collection+JSON contains URLs pointing to individual resources. If you need to customize how those URLs get generated, you may provide an HTTP header specifiying the mount point of the API:

```
X-ODE-API-Mount-Point: http://example.com/api
```

Collection requests accept query string parameters:

- `limit`: maximum number of items

- `offset`: index of the first item
- `sort_by`: name of an attribute to sort the collection
- `sort_direction`: either `asc` (ascending order) or `desc` (descending order). default to `asc`.
- `provider_id`: filter by provider id

For example, if you'd like to retrive events 20 to 30 sorted by start time in descending order, you'd use a URL such as:

```
/v1/events?offset=20&limit=10&sort_by=start_time&sort_direction=desc
```

Operations

Méthode	Ressource	Description
GET	/v1/events	Collection of events
POST	/v1/events	Add a new event
GET	/v1/events/{id}	Get an event by {id}
PUT	/v1/events/{id}	Update an event
DELETE	/v1/events/{id}	Delete an event
GET	/v1/sources	Collection of sources
POST	/v1/sources	Add a new source
GET	/v1/sources/{id}	Get a source by {id}
PUT	/v1/sources/{id}	Update a source
DELETE	/v1/sources/{id}	Delete a source

Representation formats

Nom	Mimetype	More info
Colllection+JSON	`application/vnd.collection+json`	Collection+JSON - Hypermedia Type
iCalendar	`text/calendar`	RFC 5545
CSV	`text/csv`	Comma-separated values

Event data fields

Data fields available for Collection+JSON and text/csv representations.

Field name	Required	Type/Format
`id`		String
`title`	Yes	String
`email`		String (email address)
`description`		String
`language`		String
`price_information`		String
`organiser`	Yes	String
`performers`		String (comma-separated names)
`press_url`		String (URL)
`target`		String

Table 2.1 – continued from previous page

Field name	Required	Type/Format
location_name		String
location_address		String
location_post_code		String
location_town		String
location_country		String
location_capacity		String
start_time	Yes	String (ISO 8601)
end_time	Yes	String (ISO 8601)
publication_start		String (ISO 8601)
publication_end		String (ISO 8601)
press_contact_email		String (email address)
press_contact_name		String
press_contact_phone_number		String
ticket_contact_email		String (email address)
ticket_contact_name		String
ticket_contact_phone_number		String
categories		List of strings
tags		List of strings
videos		List of dictionaries with attributes url (string) and license (‘CC BY’ or ‘
photos		List of dictionaries with attributes url (string) and license (‘CC BY’ or ‘
sounds		List of dictionaries with attributes url (string) and license (‘CC BY’ or ‘

Not that this list of fields doesn’t apply to the iCalendar format for which the specification dictates which fields are available.

Source data fields

Sources have a single field, url, which is the URL of a data stream in iCalendar or Collection+JSON format.

CHAPTER 3

Harvesting

We provide a `harvest` script which collects data from sources and updates the ODE database. It takes a Pyramid configuration file as its only argument:

```
$ harvest development.ini
```