BULDING REST APIS WITH INVENIO

Best practices and provided helpers

API DESIGN

Few Rules:

- One Resource => One URL
- No verbs in URL, use HTTP methods
- Use Header properties when possible (Location, Content-Type, ETag...)

URL STRUCTURE

/RESOURCELIST/RESOURCEID /RESOURCELI...

Create a Resource

```
curl -XPOST -H "Content-Type:application/json" \
'invenio.com/api/records' --data '{ "driver": "McF
```

GET a Resouce

```
curl -XGET -H'Accept:application/json' 'invenio.co
```

Modify a Resouce

```
curl -H 'Content-Type:application/json-patch+json'
-XPATCH '127.0.0.1:5000/api/records/1' \
--data '[{ "op": "replace", "path": "/year", "valu
```

API VERSIONING

- Solution 1: Version in URL

```
/api/v1/records/...
```

- Solution 2: Custom field in header

```
X-API-Version: v1
```

- Solution 3: Accept Headers

```
Accept: application/vnd.github.v3+json
```

- Solution 3.1 (chosen): Accept Headers Parameter

```
Accept: application/json; version: 1
```

CONTENT-TYPE NEGOTIATION

Accept Type for json GET ...

```
GET -H "Accept:application/json" .../api/records/1
> { "title": "Back to the Future" }
```

... and exports.

```
GET -H "Accept:application/marc" .../api/records/1
> ...marc record...
```

Content-Type for Imports ...

```
POST -H "Content-Type:application/marc" .../api/re
--data @metadata.marc
```

... and different operations.

```
PATCH -H 'Content-Type:application/json-patch+json
PATCH -H 'Content-Type:application/json' ... => re
```

ETAGS

Versioning Resources

```
GET .../api/records/1
> ETag: "2"
> ...
```

For caching resources

```
GET -H `If-None-Match: "2"` .../api/records/1
> HTTP/1.0 304 NOT MODIFIED
> ETag: "2"
> ...
```

And controlled modifications

```
PATCH -H `If-Match: "1"` .../api/records/1
> HTTP/1.0 412 PRECONDITION FAILED
> ...
```

HTTP ERRORS

Custom messages only when needed (400).

Can support content-negotiation too.

```
{
    "message": "Invalid JSON PATCH",
    "code": 400
}
```

```
<error>
    <message>Invalid JSON PATCH</message>
    <code>400</code>
<error>
```

Versioned with the rest of the API.

WHY NOT USING FLASK-RESTFUL?

We used it before, but...

Input handling will be replaced by Marshmallow (#335)

Bad error handling #520

Incomplete documentation #521

Flask provides *MethodView* for nice HTTP method dispatching

CONTENTNEGOTIATEDMETHODVIEW

```
class RecordResource(ContentNegotiatedMethodView):
    def __init__(self, *args, **kwargs):
        super(RecordResource, self).__init__(*args
        self.serializers = {
            'application/json': record_to_json_ser
        }

    def get(self, record_id, **kwargs):
        try:
            record = Record.get_record(record_id)
        except NoResultFound:
            abort(404)

        self.check_etag(str(record.model.version_i

        return record
```

EXAMPLE: INVENIO-RECORDS-REST

GET, PUT, PATCH, POST one record

```
/api/records/<PID>
```

```
< HTTP/1.0 200 OK
< Content-Type: application/json
< location: http://127.0.0.1:5000/api/records/1
< ETag: "1"
< ...
<
{
  "data": {
    "author": "McFly",
    "year": 1985
  },
  "id": 1,
  "links": {
    "self": "http://127.0.0.1:5000/api/records/1"
  }
}
```

FUTURE WORK

- Global
  - Security/Authentication
  - More helpers (pagination, ...)
- invenio-records-rest
  - Search api integration

```
GET /api/records?author=..
```

  - Api versioning

While keeping modules independent

API VERSION DISPATCHER

Dispatch to different applications.

Pros: separate everything, clean versioning of error handling.

Cons: not possible to dispatch to multiple applications.

```
GET -H 'accept:application/json;version=1, applica
```

MODULE STRUCTURE

CookieCutter

```
- my-rest-api
  - ...
  - my_rest_api
    - ext.py
    - restful.py
```

Versioning

```
- my-rest-api
  - ...
  - my_rest_api
    - v1
      - ext.py
      - restful.py
    - v2
      - ...
```

COOL DOCUMENTATION

Test and chose one

Tripit/Slate

sphinxcontrib-httpdomain