

An Introduction to {json:api}

Zhuo-Fei Hui
@zfhui
Blinkist

RESTful APIs using JSON

This does not tell us about how to design
our APIs.

Problem #1

Bikeshedding

Futile investment of time and energy in
discussion of marginal technical issues.

— Wiktionary

Problem #2

Overloading

Different clients prefer different structures.

Solution

{ json:api }

A Specification for Building APIs in JSON

jsonapi.org

How a **client** should request for resources
to be fetched or modified.

How a **server** should respond to those
requests.

History

2013-05-03 Yehuda Katz released initial the draft

2013-07-21 registration of the media type:

application/vnd.api+json

2015-05-29 v1.0stable released

Today v1.1 still in draft

A Simple Resource Object

User(id: integer, name: string)

Fetching a User

GET /users/1 HTTP/1.1

Accept: application/vnd.api+json

HTTP/1.1 200 OK

Content-Type: application/vnd.api+json,

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" }  
  }  
}
```

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" }  
  }  
}
```

Creating a User

POST /users HTTP/1.1

Accept: application/vnd.api+json

Content-Type: application/vnd.api+json,

```
{  
  "data": {  
    "type": "users",  
    "attributes": { "name": "Yehuda Katz" }  
  }  
}
```

Creating a User

HTTP/1.1 201 Created

Location: http://example.com/users/2

Content-Type: application/vnd.api+json,

```
{  
  "data": {  
    "id": "2",  
    "type": "users",  
    "attributes": { "name": "Yehuda Katz" }  
  }  
}
```

Updating a User

PATCH /users/2,

```
{  
  "data": {  
    "id": "2",  
    "type": "users",  
    "attributes": { "name": "Dan Gebhardt" }  
  }  
}
```

Fetching a List of Users

GET /users

GET /users

```
{  
  "data": [  
    {  
      "id": "1",  
      "type": "users",  
      "attributes": { "name": "Steve Klabnik" }  
    },  
    {  
      "id": "2",  
      "type": "users",  
      "attributes": { "name": "Dan Gebhardt" }  
    }  
  ]  
}
```

Deleting a User

DELETE /users/1

Relationships

```
User(id: integer, name: string)
```

```
Article(  
    id: integer,  
    title: string,  
    content: text,  
    user_id: integer  
)
```

Fetching a User

GET /users/1

GET /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    }  
  }  
}
```

GET /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    }  
  }  
}
```

GET /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    }  
  }  
}
```

Compound Documents

n+1 requests

GET /users/1

GET /articles/2

GET /articles/5

...

1 request

GET /users/1?include=articles

GET /users/1?include=articles

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    },  
    "included": [  
      {  
        "id": "2",  
        "type": "articles",  
        "attributes": { "title": "Intro to JSON API", "content": "Lorem opossum ..." }  
      },  
      {  
        "id": "5",  
        "type": "articles",  
        "attributes": { "title": "Anti-Bikeshedding", "content": "Marsupial fur trees ..." }  
      }  
    ]  
  }  
}
```

GET /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    },  
    "included": [  
      {  
        "id": "2",  
        "type": "articles",  
        "attributes": { "title": "Intro to JSON API", "content": "Lorem opossum ..." }  
      },  
      {  
        "id": "5",  
        "type": "articles",  
        "attributes": { "title": "Anti-Bikeshedding", "content": "Marsupial fur trees ..." }  
      }  
    ]  
  }  
}
```

GET /users/1?include=articles

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    },  
    "included": [  
      {  
        "id": "2",  
        "type": "articles",  
        "attributes": { "title": "Intro to JSON API", "content": "Lorem opossum ..." }  
      },  
      {  
        "id": "5",  
        "type": "articles",  
        "attributes": { "title": "Anti-Bikeshedding", "content": "Marsupial fur trees ..." }  
      }  
    ]  
  }  
}
```

GET /users/1?include=articles

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    },  
    "included": [  
      {  
        "id": "2",  
        "type": "articles",  
        "attributes": { "title": "Intro to JSON API", "content": "Lorem opossum ..." }  
      },  
      {  
        "id": "5",  
        "type": "articles",  
        "attributes": { "title": "Anti-Bikeshedding", "content": "Marsupial fur trees ..." }  
      }  
    ]  
  }  
}
```

Sparse Fieldsets

GET /users/1?include=articles&**fields[articles]=title**

GET /users/1?include=articles&fields[articles]=title

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Steve Klabnik" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "2", "type": "articles" },  
          { "id": "5", "type": "articles" }  
        ]  
      }  
    },  
    "included": [  
      {  
        "id": "2",  
        "type": "articles",  
        "attributes": { "title": "Intro to JSON API" }  
      },  
      {  
        "id": "5",  
        "type": "articles",  
        "attributes": { "title": "Anti-Bikeshedding" }  
      }  
    ]  
  }  
}
```

Updating a User with Relationships

PATCH /users/1



PATCH /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Dan Gebhardt" },  
    "relationships": {  
      "articles": {  
        "data": [  
          { "id": "3", "type": "articles" },  
          { "id": "6", "type": "articles" }  
        ]  
      }  
    }  
  }  
}
```

👉 PATCH /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Dan Gebhardt" },  
    "relationships": {  
      "articles": {  
        "data": []  
      }  
    }  
  }  
}
```



PATCH /users/1

```
{  
  "data": {  
    "id": "1",  
    "type": "users",  
    "attributes": { "name": "Dan Gebhardt" },  
    "relationships": {  
      "articles": {  
        "data": []  
      }  
    }  
  }  
}
```

Manipulating a User's Relationships

POST and DELETE on Relationship Links:

/users/1/relationships/articles

Adding Relationships to a User

POST /users/1/relationships/articles

```
{  
  "data": [  
    { "id": "3", "type": "articles" },  
    { "id": "7", "type": "articles" }  
  ]  
}
```

Deleting a User's Relationships

DELETE /users/1/relationships/articles

```
{  
  "data": [  
    { "id": "3", "type": "articles" },  
    { "id": "7", "type": "articles" }  
  ]  
}
```



There is More ...

- meta objects, links objects
- pagination, sorting, filtering
- error objects
- n:m relationships
- does not support creating nested resources

Tooling

gems codifying { json:api }



active_model_serializers



fast_jsonapi



jsonapi_resources



jsonapi_suite

Client libraries

Implementations

Client libraries

JavaScript

TypeScript

iOS

Ruby

PHP

Dart

Perl

Java

Android

R

Elm

.NET

Python

Elixir

Server libraries

Swift

PHP

Node.js

Ruby

Python

Go

.NET

Java

Scala

Elixir

Haskell

Perl

JavaScript

- [ember-data](#) is one of the original exemplar implementations. There is now an [official adapter](#) to support json-api.
- [backbone-jsonapi](#) is a Backbone adapter for JSON API. Supports fetching Models & Collections from a JSON API source.
- [backbone-relational-jsonapi](#) is a parsing layer for Backbone.Relational. Entities specified in JSON API are automatically parsed to be injected into Backbone.Relational relations.
- [orbit.js](#) is a standalone library for coordinating access to data sources and keeping their contents synchronized. Orbit's Common Library includes [JSONAPISource](#) for accessing JSON API servers. Orbit can be used independently or with Ember.js through the [ember-orbit](#) integration library.
- [YAYSON](#) is an isomorphic library for serializing and reading JSON API data. Extend it to fit your models or just use it with plain objects.
- [Ember JSON API Resources](#) is an [Ember CLI Addon](#) for a lightweight solution for data persistence in an [Ember.js](#) application.
- [hapi-json-api](#) Plugin for the hapi framework; enforces Accept/Content-type rules and rewrites Boom errors to be spec compliant.
- [jsonapi-datastore](#) is a lightweight standalone library for reading, serializing, and synchronizing relational JSON API data.
- [json-api-store](#) A lightweight JavaScript library for using JSON API in the browser.
- [superagent-jsonapify](#) A really lightweight (50 lines) JSON-API client addon for [superagent](#), the isomorphic ajax client.
- [angular-jsonapi](#) An AngularJS JSON API client
- [redux-json-api](#) A library which integrated JSON APIs with Redux store

To Summarise ...

- anti-bikeshedding
 - one endpoint to serve different client needs
 - tight coupling between API and underlying data structure
 - community and tooling support
- 🍰 leverages HTTP content negotiation mechanism

References

[Website](#) [Media Type Specs](#)

[Talk](#) [JSON API: convention driven API design](#) by Steve Klabnik

[Talk](#) [Past, Present and Future of JSON API](#) by Steve Klabnik

[Talk](#) [The Road to JSON API 1.0](#) by Steve Klabnik

[Talk](#) ["The JSON API Spec"](#) by Marco Otto-Witte

[Talk](#) ["Pragmatic JSON API Design"](#) by Jeremiah Lee

[Podcast](#) ["Dan Gebhard - json-api, jsonapi-resources, orbit.js & Ember Data"](#) by Byle Daigle

[Podcast](#) ["Data Loading Patterns with the JSON API with Balint Erdi"](#) by The Frontside Podcast

[Podcast](#) ["JSON API and API Design"](#) by The Changelog

[Images](#) [Bikeshed, Devices, Hamster BandConfetti](#)

Thanks!