# Learning and Implementing HATEOAS

**Kevin Dockx**
ARCHITECT

@KevinDockx https://www.kevindockx.com

## Coming Up

HATEOAS (Hypermedia as the Engine of Application State)

## Hypermedia as the Engine of Application State

Helps with evolvability and self-descriptiveness

Hypermedia drives how to consume and use the API

```
{ "id": "5b1c2b4d-48c7-402a-80c3-cc796ad49c6b",
  "title": “Commandeering a ship without getting caught",
  "description": “Commandeering a ship in rough waters ...",
  "authorId": "d28888e9-2ba9-473a-a40f-e38cb54f9b35"


}
```

## Issues Without HATEOAS

**Intrinsic knowledge of the API contract is required**

```
{ "id": "5b1c2b4d-48c7-402a-80c3-cc796ad49c6b",
  "title": “Commandeering a ship without getting caught",
  "description": “Commandeering a ship in rough waters ...",
  "authorId": "d28888e9-2ba9-473a-a40f-e38cb54f9b35",
  "numberOfAvailablePlaces": 10
}
```

## Issues Without HATEOAS

**Intrinsic knowledge of the API contract is required**

```
{ "id": "5b1c2b4d-48c7-402a-80c3-cc796ad49c6b",
  "title": “Commandeering a ship without getting caught",
  "description": “Commandeering a ship in rough waters ...",
  "authorId": "d28888e9-2ba9-473a-a40f-e38cb54f9b35",
  "numberOfAvailablePlaces": 10,
  "content": "mature"}
```

## Issues Without HATEOAS

**Intrinsic knowledge of the API contract is required**

**An additional rule, or a change of a rule, breaks consumers of the API**

**The API cannot evolve separately of consuming applications**

```
{ …
  "numberOfAvailablePlaces": 10,
  "content": "mature“,
  “links":
```

## Supporting HATEOAS

```
{ …

  "numberOfAvailablePlaces": 10,

  "content": "mature“,

  “links": [
      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": "self",
        "method": "GET"
      },
```

## Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": “update-course-full",
        "method": “PUT"
      },
```

## Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": “update-course-full",
        "method": “PUT"
      },

      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": “update-course-partial",
        "method": “PATCH"
      },
```

## Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": “delete-course",
        "method": “DELETE"
      }
```

## Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/courses/{courseId},
        "rel": “delete-course",
        "method": “DELETE"
      },

      {
        "href": "http://host/api/coursereservations,
        "rel": “reserve-course",
        "method": “POST"
      }]

}
```

## Supporting HATEOAS

"You can't have evolvability if clients have their controls baked into their design at deployment. Controls have to be learned on the fly. That's what hypermedia enables."

Roy Fielding (http://bit.ly/2hBPQXi)

## Supporting HATEOAS

**This is how the HTTP protocol works: leveraging hypermedia**

- Links, forms, … drive application state

```
<a href="uri",
   rel="type",
   type="media type">
```

## Supporting HATEOAS

**HTML represents links with the anchor element**

- href: contains the uri
- rel: describes how the link relates to the resource
- type: describes the media type

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/coursereservations,
        "rel": “reserve-course",
        "method": “POST"
      }]

}
```

## Supporting HATEOAS

`method` **defines the method to use**

`rel` **identifies the type of action**

`href` **contains the URI to be invoked to execute this action**

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/coursereservations,
        "rel": “reserve-course",
        "method": “POST"
      }]

}
```

## Supporting HATEOAS

`method` **defines the method to use**

`rel` **identifies the type of action**

`href` **contains the URI to be invoked to execute this action**

```
{
  "value": [ {author}, { author} ],
  "links": [ … ]
}
```

## Supporting HATEOAS for Collection Resources

**Envelope is required to avoid invalid JSON**

**This isn't RESTful when using media type application/json… but we're fixing that later on ☺**

## Demo Introduction: Supporting HATOEAS

**Logic for creating links depends on business rules – requires custom code**

- PUT, DELETE, … but also:
- POST to /coursereservations

# Demo Introduction – Supporting HATOEAS

## Statically typed approach

Base class (with links) and wrapper class

Inherit base class for single resources

Use wrapper class for collection resources

## Dynamically typed approach

Anonymous types & ExpandoObject

Add links to ExpandoObject for single resources

Use anonymous type for collection resources

Demo

Implementing HATEOAS support for a single resource

Demo

Implementing HATEOAS support after POSTing

## Demo

Implementing HATEOAS support for a collection resource

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors?pageNumber=1&pageSize=10",
        "rel": “previous-page",
        "method": “GET"
      },

      {
        "href": "http://host/api/authors?pageNumber=3&pageSize=10",
        "rel": “next-page",
        "method": “GET"
      }]

}
```

## Using HATEOAS for Pagination Links

Demo

Using HATEOAS for pagination links

Demo

Working towards self-discoverability with a root document

## Other Approaches and Options

**HAL (Hypertext Application Language)**

- https://bit.ly/2YAyrUc
- Provides a set of conventions for expressing hyperlinks in either JSON or XML

**Siren (Structured Interface for Representing Entities)**

- https://github.com/kevinswiber/siren
- Link format and descriptions of what to send to those links

## Other Approaches and Options

**Json-LD**

- http://json-ld.org/
- Lightweight linked data format

**Json-API**

- https://jsonapi.org/
- Specification for building JSON APIs

**OData**

- http://www.odata.org/
- Effort to standardize REST APIs

## Summary

**HATEOAS**

- Hypermedia, like links, drive how to consume and use the API, and the functionality of the consuming application: its state

**HATEOAS diminishes the need for intrinsic API knowledge**

- Even if functionality and business rules change, client applications won't break