Getting Started with HATEOAS

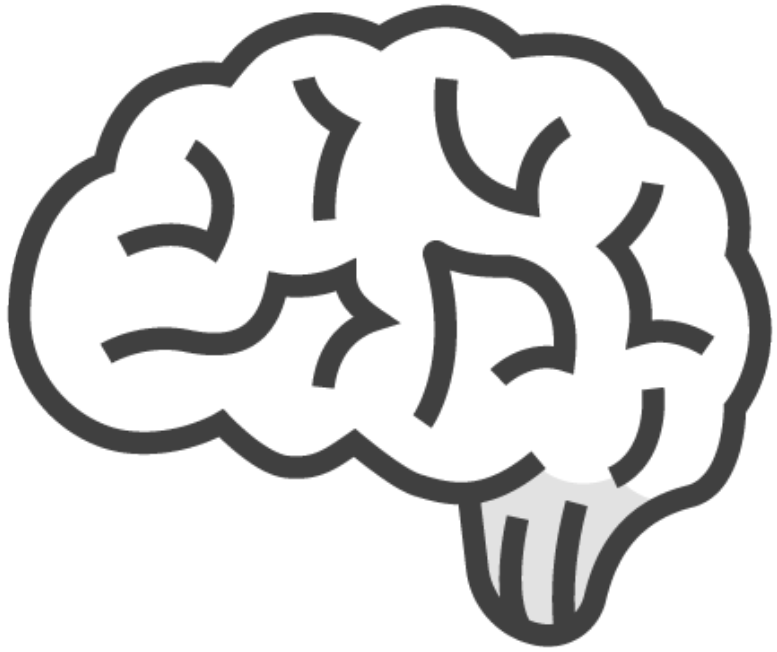
Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

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": "bc4c35c3-3857-4250-9449-155fcf5109ec",
  "title": "The Winds of Winter",
  "description": “Forthcoming 6th novel in A Song of Ice and Fire.",
  "authorId": "76053df4-6687-4353-8937-b45556748abe"

}
```

Issues Without HATEOAS

Intrinsic knowledge of the API contract is required

```
{ "id": "bc4c35c3-3857-4250-9449-155fcf5109ec",
  "title": "The Winds of Winter",
  "description": "Forthcoming 6th novel in A Song of Ice and Fire.",
  "authorId": "76053df4-6687-4353-8937-b45556748abe",
  "numberOfCopiesInLibrary": 10
}
```

Issues Without HATEOAS

Intrinsic knowledge of the API contract is required

```
{ "id": "bc4c35c3-3857-4250-9449-155fcf5109ec",
  "title": "The Winds of Winter",
  "description": "Forthcoming 6th novel in A Song of Ice and Fire.",
  "authorId": "76053df4-6687-4353-8937-b45556748abe",
  "numberOfCopiesInLibrary": 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

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

Supporting HATEOAS

```
{ …
  "numberOfCopiesInLibrary": 10,
  "content": "mature“,
  “links": [
      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": "self",
        "method": "GET"
      },
```

Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": “update-book-full",
        "method": “PUT"
      },
```

Supporting HATEOAS

```
{ …
  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": “update-book-full",
        "method": “PUT"
      },

      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": “update-book-partial",
        "method": “PATCH"
      },
```

Supporting HATEOAS

```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": “delete-book",
        "method": “DELETE"
      }
```

Supporting HATEOAS

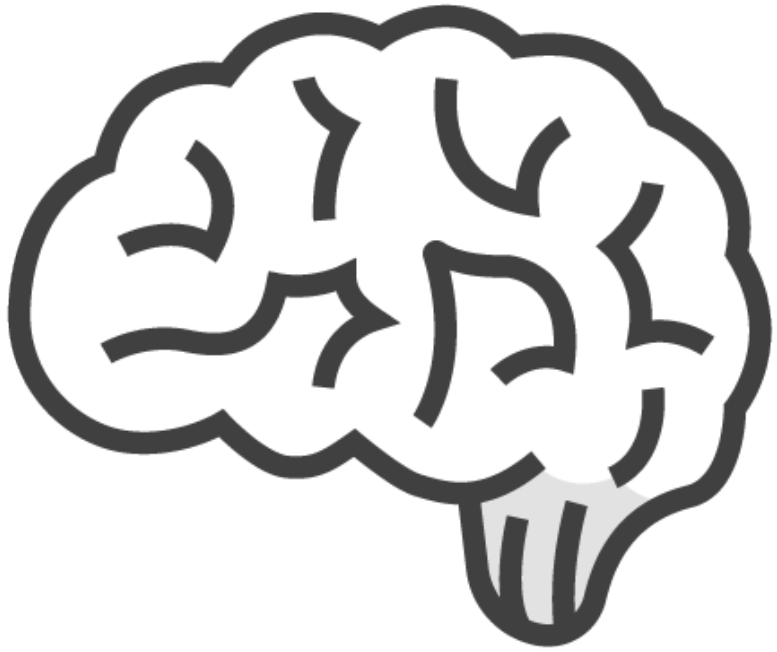
```
{ …

  "links": [ …,
      {
        "href": "http://host/api/authors/{authorId}/books/{bookId},
        "rel": “delete-book",
        "method": “DELETE"
      },

      {
        "href": "http://host/api/bookreservations,
        "rel": “reserve-book",
        "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)

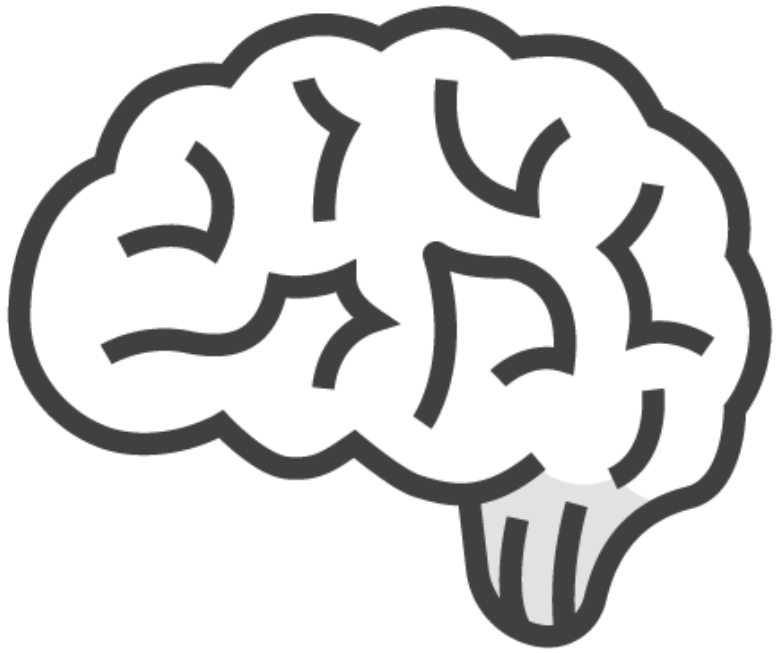
Supporting HATEOAS



This is how the HTTP protocol works: leveraging hypermedia

Links, forms, ... drive application state

Supporting HATEOAS



HTML represents links with the anchor element

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

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

```
{ …

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

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

}
```

Supporting HATEOAS for Collection Resources

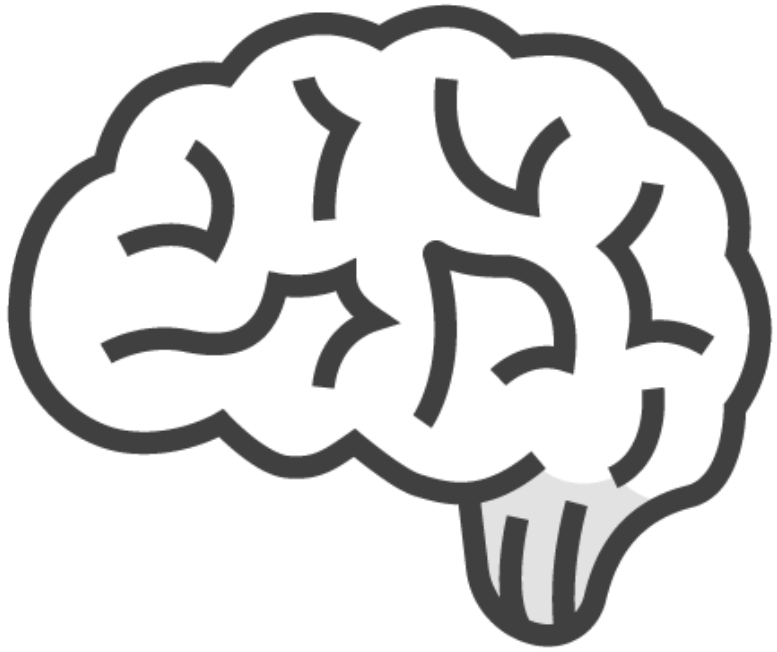
```
{
  "value": [ {author}, { author} ],
  "links": [ … ]
}
```

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 /bookreservations

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

Supporting HATEOAS (Base and Wrapper Class Approach)

Demo

Supporting HATEOAS (Dynamic Approach)

Summary

Hypermedia as the Engine of Application State

- Hypermedia drives how to consume the API
- Hypermedia drives the functionality of the client
- Include links in the response body