Enterprise Programmering 2

Lesson 05: Wrap and Pagination

Dr. Andrea Arcuri

Goals

- Understand the concept of *Wrapped Responses*, and how it helps in logging/debugging of errors

- Understand how to enable *Pagination* with *Links* when dealing with requests retrieving large amounts of data

Wrapped Responses

Errors

- HTTP request can fail due to a 4xx or 5xx error
- But what was the reason?
- How to tell the user that a 400 was due to an invalid query parameter s/he provided?
- Not so great solution: provide a *error message* as a HTML body payload
- Why not so great? Need to marshal payloads in different ways based on status code…
  - ie JSON when OK, and HTML when errors

JSON Wrapped Response

```
{
  "code": 400,
  "status": "ERROR",
  "data": null,
  "message": "Invalid query parameter x"
}
```

```
{
  "code": 200,
  "status": "SUCCESS",
  "data": {foo:4, bar:"a"}
  "message": null
}
```

Wrapping

- Instead of returning a payload directly in the HTTP body of the response, wrap it in a JSON object
- The payload will be in a field called “**data**”
- If there is any error, then “**data**” will be null, with a field “**message**” explaining reason, ie the *error message*
- Can also have fields for the status of the response (eg success vs failure/error)

Benefits

- Error message, if any, is part of the response body, easy to access
- Unmarshaling of HTTP response payload from JSON regardless of success or failure/error
  - ie, success and error responses have the same JSON structure
- Very limited overhead

Standard

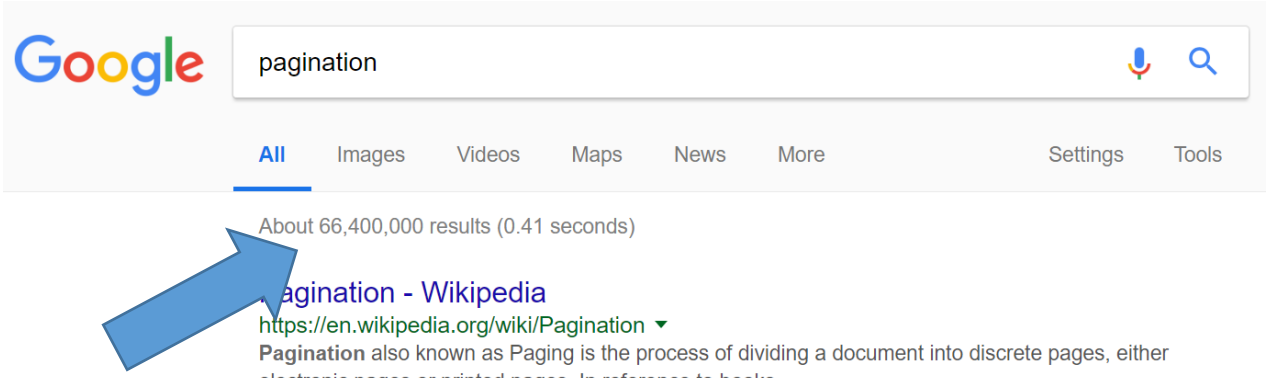
- How to specify which fields to use in a wrapped response?
- This is not part of HTTP, nor something discussed in REST
- There is no “standard”
- Could use your own format for your APIs
- Or use some existing specification like *JSend*
  - *https://labs.omniti.com/labs/jsend*

Pagination

Amount of Data

- Example: **GET /news**
- Return all news in database, marshalling into JSON
- But what if the database has 2 billion news???
- You do not want to return terabytes of data for a single GET…
- It would end up in a easy to exploit Denial-Of-Service (DOS) attack

Searches


Google
pagination
All
Images
Videos
Maps
News
More
Settings
Tools
About 66,400,000 results (0.41 seconds)
agination - Wikipedia
https://en.wikipedia.org/wiki/Pagination
Pagination also known as Paging is the process of dividing a document into discrete pages, either electronic pages or printed pages. In reference to books ...
Pagination in word ... · Pagination in print · Pagination in electronic ...
Bootstrap Pagination - W3Schools
https://www.w3schools.com/bootstrap/bootstrap_pagination.asp
Basic Pagination. If you have a web site with lots of pages, you may wish to add some sort of pagination to each page. A basic pagination in Bootstrap looks like ...
CSS Pagination Examples - W3Schools
https://www.w3schools.com/css/css3_pagination.asp
Learn how to create a responsive pagination using CSS. ... If you have a website with lots of pages, you may wish to add some sort of pagination to each page:.


Pagination | GraphQL
https://graphql.org/learn/pagination/
Pagination. Different pagination models enable different client capabilities. A common use case in GraphQL is traversing the relationship between sets of objects ...
GitHub - vapor-community/pagination: Simple Vapor 3 Pagination
https://github.com/vapor-community/pagination
Mar 7, 2018 - Simple Vapor 3 Pagination. Contribute to vapor-community/pagination development by creating an account on GitHub.
GitHub - react-component/pagination: React Pagination
https://github.com/react-component/pagination
React Pagination. Contribute to react-component/pagination development by creating an account on GitHub.
Searches related to pagination
pagination example
pagination javascript
pagination html
pagination bootstrap
pagination website
pagination php
pagination design
pagination jquery
Gooooooooooogle
Previous
1 2 3 4 5 6 7 8 9 10
Next

Page

- Instead of billions of elements, just return a single *Page*
- A *Page* will contain *n* elements (eg 10 or 20) from the collection
- It will have information on the *previous* and the *next* page
- If you want, can iterate over the whole collection by checking one page at a time, following the *next* links

dto = { “**list**”: […], *//the actual payload*

“**rangeMin**”: 40,

“**rangeMax**”: 49, *//so, 10 element pages*

*“**totalSize**”: 66400000,*

*“**_links**”:[*

*“**next**”: {“href”: “/news?**offset**=50&**limit**=10”},*

*“**self**”: {“href”: “/news?**offset**=40&**limit**=10”},*

*“**previous**” : {“href”: “/news?**offset**=30&**limit**=10”},*

]

}

Offset/Limit

- When dealing with large collections, need a way to specify the boundaries of a *Page*
- Example: *GET /news?**offset**=40&**limit**=10*
- *Offset*: given the collection sorted like an array, this would be the starting index *i*
- *Limit:* starting from the offset, how many elements to return

Links

- To access the next/previous pages, can compute the needed offsets/limits
- Or, we could just provide valid URLs in the JSON responses with "*links*" to those pages
- This is an instance of HATEOAS
  - *Hypermedia as the Engine of Application State*
- Easier to navigate

Standard

- There is no official standard to define pages and links
- In the past, there were some attempts like HAL, but they look like abandoned

Expansion

- A “news” might have a *list* of “comments”
- A “news” might also have a *list* of “*users*” that liked it
- When retrieving a single item, might not want to download as well the hundreds/thousands of other items related to it
- As returning those lists can be very expensive, can have special query parameters to choose if downloaded or not
- Eg.: GET /news?**expand**=NONE (no lists)
- Eg.: GET /news?**expand**=COMMENTS (include comments)

Tradeoff

- Option 1: never return those lists
  - But, then, need further HTTP calls to retrieve those lists if needed
- Option 2: create "*expand*" query parameters to control what returned
  - Good: can return everything needed in a single HTTP request
  - Bad: needs to implement all the needed cases *manually*
- *GraphQL*: a selling point compared to REST is its ability to exactly specify what to return
  - we will see GraphQL later in the course

Git Repository Modules

- *NOTE: most of the explanations will be directly in the code as comments, and not here in the slides*
- **advanced/rest/wrapper**
- **advanced/rest/rest-dto**
- **advanced/rest/pagination**
- Study relevant sections in *RESTful Service Best Practices*
- Study relevant sections in RFC-7230 and RFC-7231