Response Codes

Memi Lavi
www.memilavi.com

Structure of REST API Response

Response Codes

- Notify clients about the status of the request
  - Did it succeed?
  - Did it fail? Why?
  - What kind of error?

Importance of Response Codes

- Most clients check for response code and act accordingly
- Monitoring tools check response codes and report it
- Makes the API easier to use and understand
- Very easy to implement, often overlooked

Response Code Groups

- Five groups
- Each group represent specific response type (success, error, etc.)
- Each group consists of 3 digits status code

Groups

- 1xx – Informational Response — Do Not Use!
- 2xx – Success
- 3xx – Redirection — Low Level
- 4xx – Client Errors
- 5xx – Server Errors

2xx - Success

- 200 – OK
  - The default status code
  - Developers aren't aware of that
  - The request was successful

2xx - Success

- 200 – OK
  - With GET – returns the entity/ies requested
  - With POST – returns the entity/ies created

2xx - Success

- 201 – Created
  - The request has been fulfilled, a new entity has been created
  - Response might contain `Location` header pointing to the entity
  - The expected status code of POST request

2xx - Success

- 202 – Accepted
  - The request has been accepted and is pending processing
  - Processing might not be complete
  - No notification is given when processing completes
  - Usually used for POST, PUT requests

2xx - Success

- 204 – No Content
  - Request was fulfilled but no content was sent back
  - Should NOT include body
  - Used mainly when updating large entity

2xx - Success

- The 204 No Content vs 200 OK dilemma
  - What should be returned when a GET returns no entity?
  - 200 with no body, or 204?
  - No agreed upon answer, better to avoid the 204 and stick to 200

4xx – Client Error

- 400 – Bad Request
  - The client sent a bad request
    - Parameters can't be validated (ie. Negative age)
    - JSON can't be parsed
- Used with all verbs

4xx – Client Error

- 400 – Bad Request
  - Consider using RFC 7807 – Problem Details
  - Standard for returning machine-readable data about the problem

RFC 7807 – Problem Details

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json
Content-Language: en

{
"type": "https://example.net/validation-error",
"title": "Your request parameters didn't validate.",
"invalid-params": [ {
                      "name": "age",
                      "reason": "must be a positive integer"
                    },
                    {
                      "name": "color",
                      "reason": "must be 'green', 'red' or 'blue'"}
                  ]
}
```

https://tools.ietf.org/html/rfc7807

4xx – Client Error

- 401 – Unauthorized
  - The client is not authorized to access the entity
  - Authorization is required
  - Not to be confused with 403 – Forbidden!

4xx – Client Error

- 403 – Forbidden
  - The client is not authorized to access the entity
  - Authorization failed

4xx – Client Error

Authorization flow:

https://avidonline.com

4xx – Client Error

- 404 – Not Found
  - The entity specified in the request was not found
  - Used when a specific entity was looked for, not with query parameter
  - Used with all verbs

5xx – Server Error

- 500 – Internal Server Error
  - Something bad happened while processing the request
  - Client can do nothing about it
- Used with all verbs

5xx – Server Error

- Other 5xx status codes are usually not used, or are low level

Additional Codes

- Many non-standard codes
- Introduced by various companies and organizations
  - ie. Twitter, Apache, Shopify, and more
- WebDAV extends the standard for document editing
  - Adds many status codes

WebDAV Status Example

- 207 – Multi-Status
  - Used for cases where multiple entities are handled, each has its own status
  - ie. Adding a list of 10 entities. Some might fail.

WebDAV Status Example

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<d:multistatus xmlns:d="DAV:">
    <d:response>
        <d:href>/calendars/johndoe/home/132456762153245.ics</d:href>
        <d:propstat>
            <d:prop>
                <d:getetag>"xxxx-xxx"</d:getetag>
            </d:prop>
            <d:status>HTTP/1.1 200 OK</d:status>
        </d:propstat>
    </d:response>
    <d:response>
        <d:href>/calendars/johndoe/home/fancy-caldav-client-1234253678.ics</d:href>
        <d:propstat>
            <d:prop>
                <d:getetag>"5-12"</d:getetag>
            </d:prop>
            <d:status>HTTP/1.1 200 OK</d:status>
        </d:propstat>
    </d:response>
</d:multistatus>
```

https://evertpot.com/http/207-multi-status

Summary

- Return the appropriate status code
- Remember not all clients are familiar with all codes
- Use the most common ones:
  - 200 OK, 201 Created, 204 No Content, 400 Bad Request,

    404 Not Found, 500 Internal Server Error

Summary

Take a look at the resources!