Daniel R. Cender

Grand Canyon University

CST-361: Design Patterns in Java

Charbel Elkhoury

April 14, 2021

**Write-Up**

**Data Transfer Object**

In this activity, the Data Transfer Object is used as an in-between to pass along response metadata. In the case that a request came back empty or invoked an erroring condition, there needed to be a way for API consumers to be alerted appropriately. The request still comes back successful with a 200-code, but the response object code and message now hold valuable information for consumers to use in backing off queries, changing search strategies, or exposing issues in the API service itself over time.

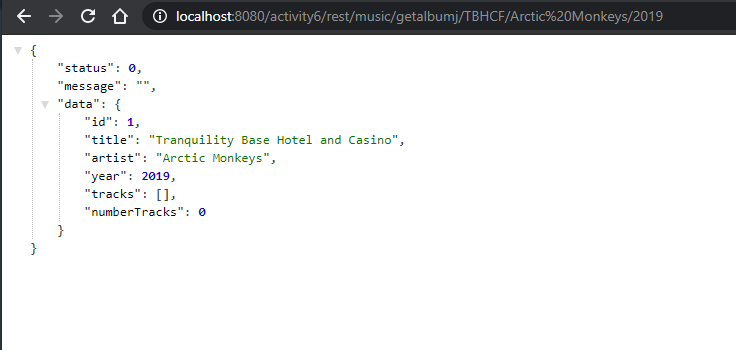
**Facade**

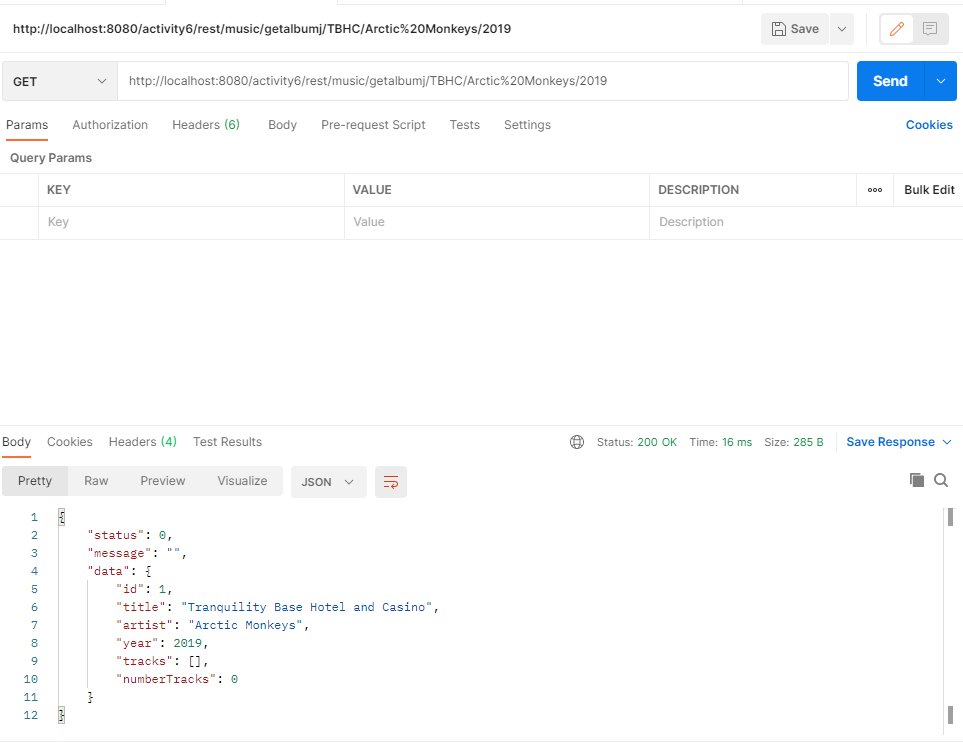
The Facade design pattern is used here to handle the logic of querying for album data and handling error cases. In a sense, the API request route itself is a larger facade, since it encapsulates a lot of specific details related to business logic and data fetching that API consumers simply don’t need to comprehend. The API endpoint provides a pure front for “dumb web clients” or microservices to consume pre-formed data and display as they’d like.

**Deliverables**

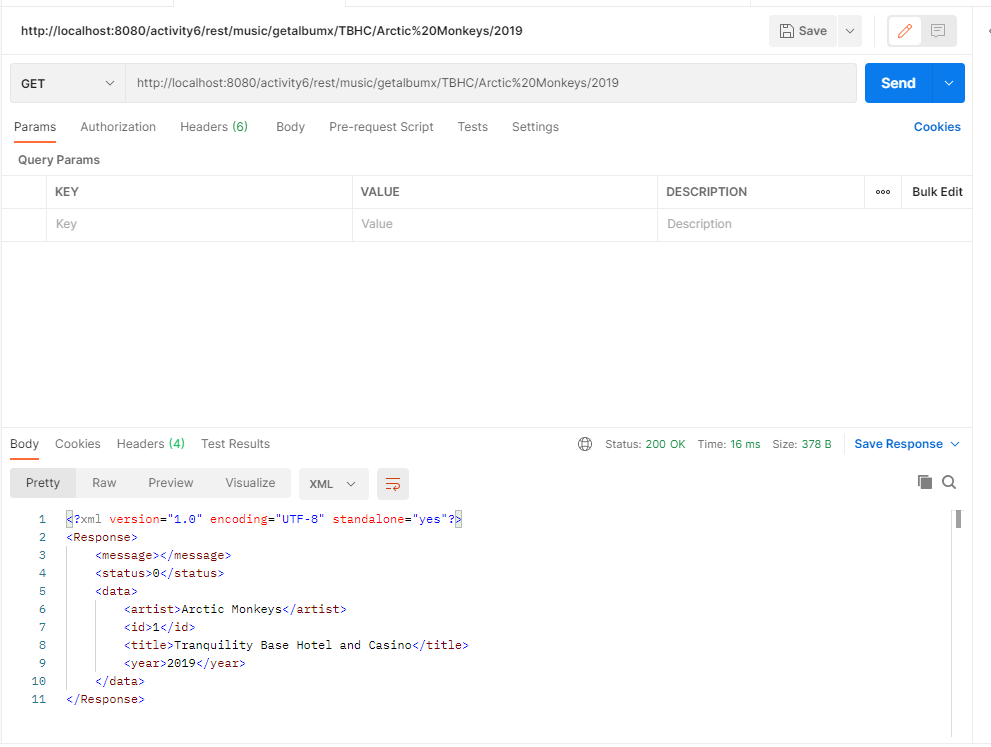
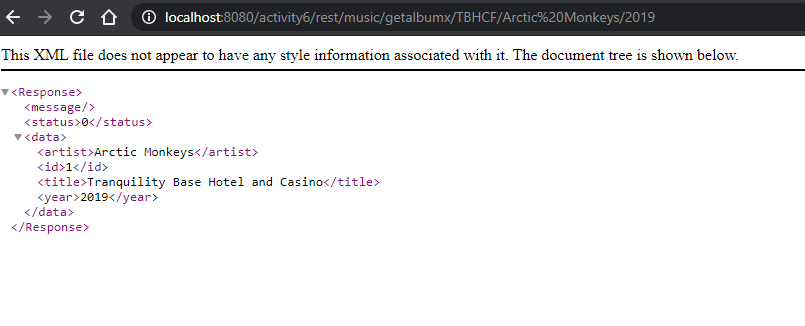
The results of the newly implemented REST API route for searching albums.

A valid album is found (JSON & XML in browser and Postman):

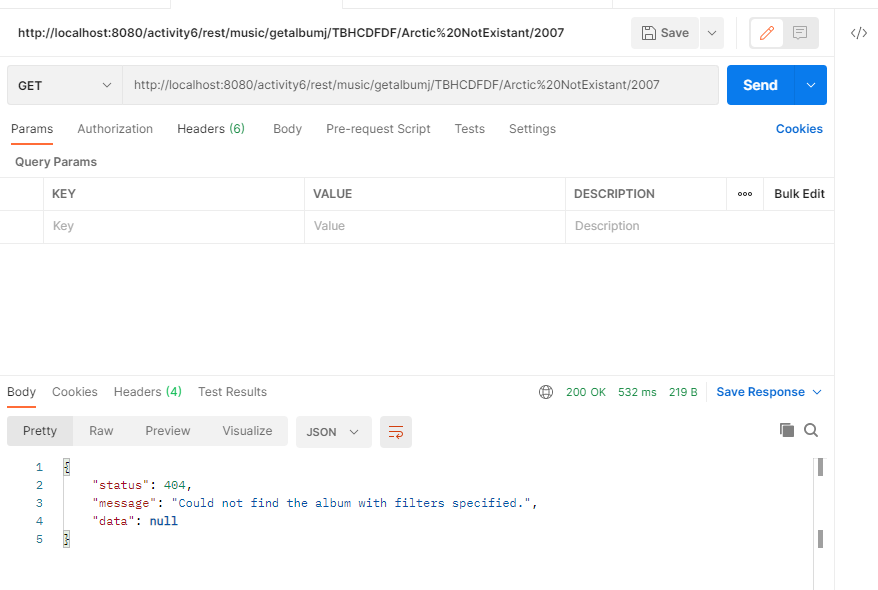
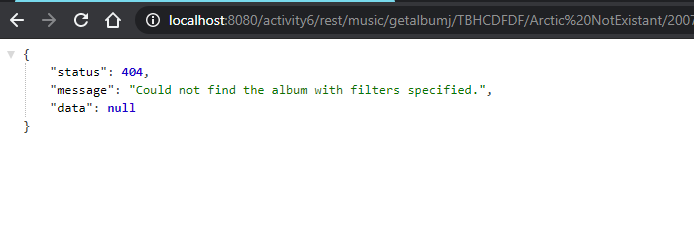




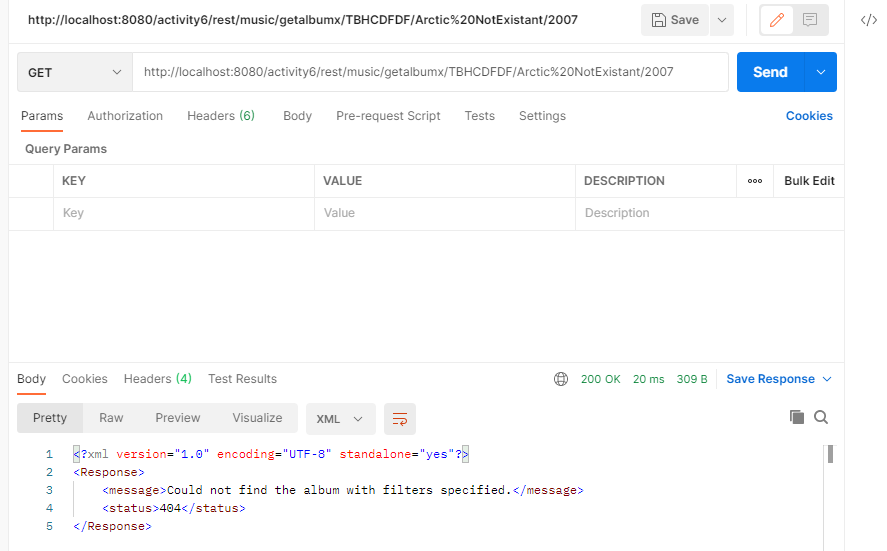
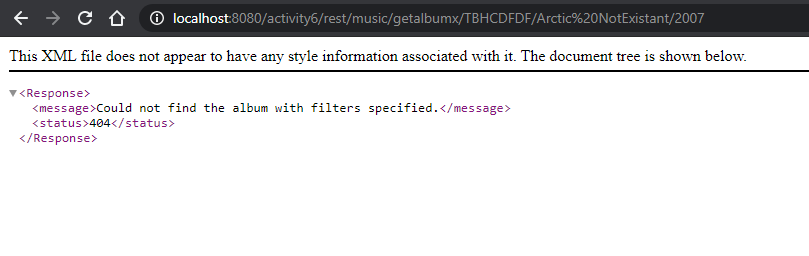
In XML:



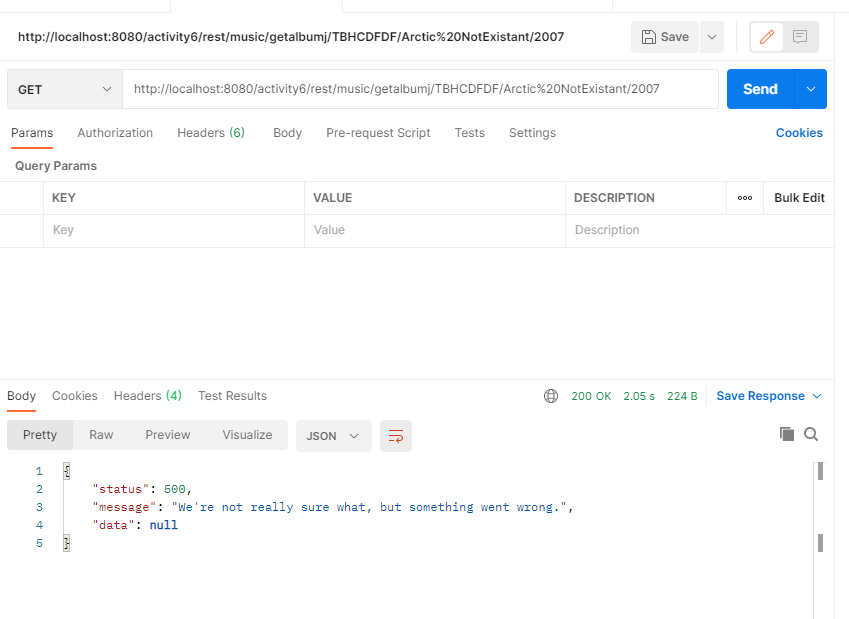
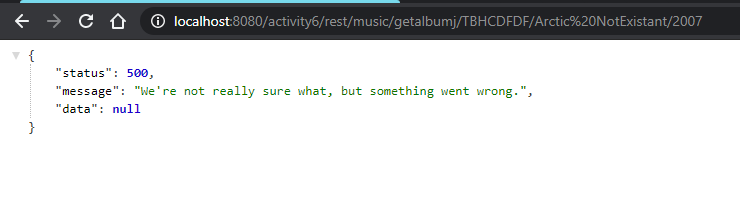
No valid album can be found matching the search parameters (JSON & XML in browser and Postman):



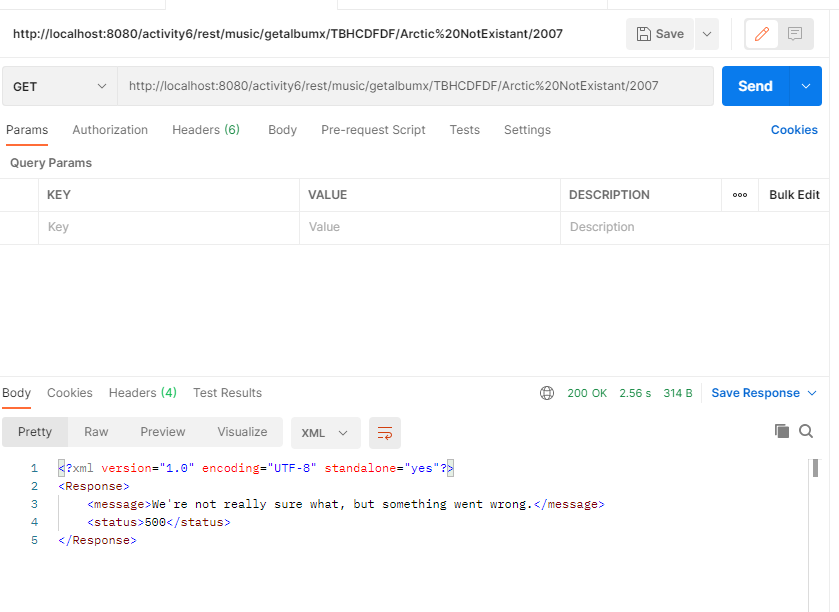
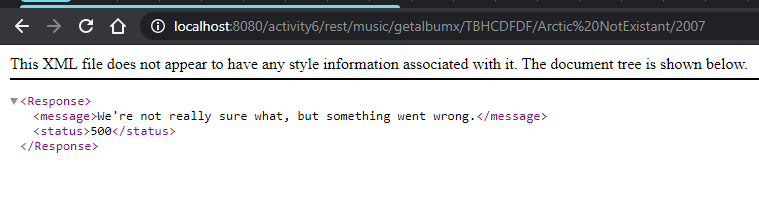
In XML format:



When the MySQL service is shut off locally, the REST API returns a 500 code and message indicating an internal issue (JSON & XML in browser and Postman):



In XML:



**References**