Personal Expense Tracker App

By Ramesh Fadatare (Java Guides)

Expense Tracker Project Overview

The Expense Tracker Application is a comprehensive solution designed to help users manage their finances efficiently by tracking expenses across different categories.

Expense Tracker Project Requirements

1. Category Management
2. Expense Management
3. Exception Handling
4. REST API Documentation

Requirement 1 - Category Management

Category Management feature allows Users to create new categories, update existing ones, retrieve details about a specific category or all categories, and delete categories. This functionality allows for effective categorization of expenses, facilitating better budget management and financial planning.

Build CRUD REST APIs:

1. **Create Category**: Users can add new category.
2. **Get Category**: Users can retrieve details of a specific category.
3. **Get All Categories**: Users can retrieve a lists all available categories.
4. **Update Category**: Users can update the existing category.
5. **Delete Category**: Users can delete the existing category.

Requirement 2 - Expense Management

Expense Management feature allows Users to add new expenses, modify details of existing expenses, view information about a particular expense or all expenses, and remove expenses from the record. This feature is crucial for tracking daily expenses and monitoring financial outflows.

Build CRUD REST APIs:

1. **Create Expense**: Users can add new expense under a specific category.
2. **Get Expense**: Users can retrieve details of a specific expense.
3. **Get All Expenses**: Users can retrieve a lists all available expenses.
4. **Update Expense**: Users can update the existing expense.
5. **Delete Category**: Users can delete the existing expense.

Requirement 3 - Exception Handling

The REST APIs should return the proper error response as shown below:

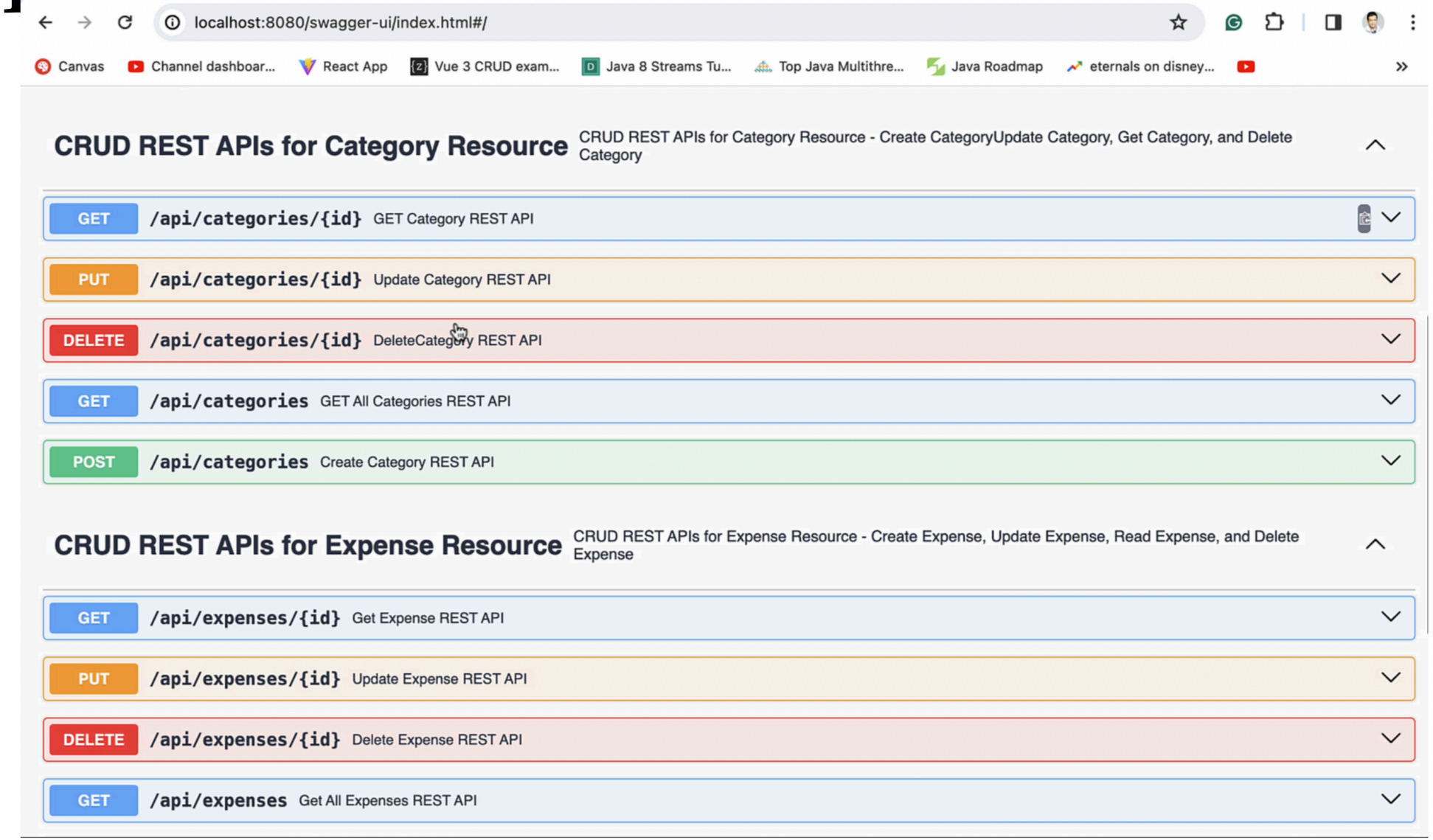
For Category:

```
{
    "timestamp": "2024-02-10T18:07:14.016983",
    "message": "Category not found with id: 5",
    "details": "uri=/api/categories/5",
    "errorCode": "RESOURCE_NOT_FOUND"
}
```

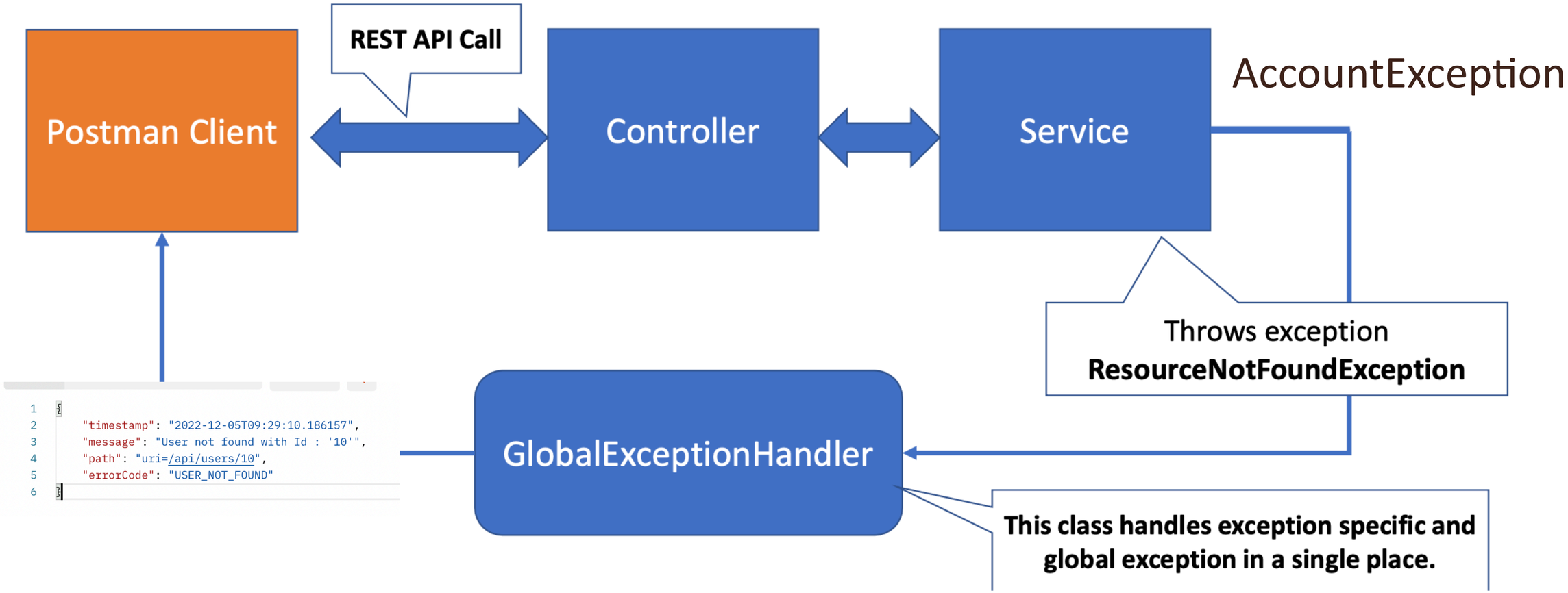
For Expense:

```
{
    "timestamp": "2024-02-10T18:09:48.634167",
    "message": "Expense not found with id: 1",
    "details": "uri=/api/expenses/1",
    "errorCode": "RESOURCE_NOT_FOUND"
}
```

Requirement 4 - REST API Documentation



Spring Boot REST API Exception Handling



Exception Handling in Spring Boot App

1. Handling exceptions in Spring Boot REST APIs is typically done using the **@ControllerAdvice** and **@ExceptionHandler** annotations.

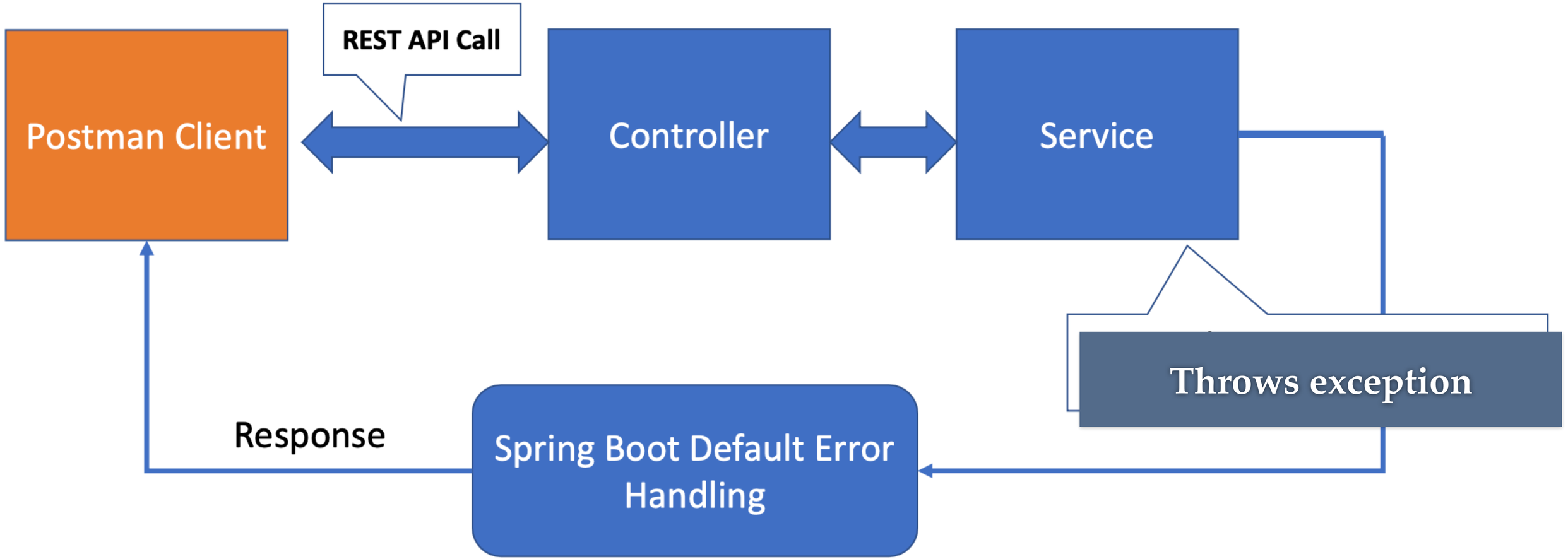
2. **@ControllerAdvice** enables global exception handling for controllers.

3. **@ExceptionHandler** defines methods to handle specific exceptions within a controller or globally with @ControllerAdvice.

Development Steps

1. Create and Use **ResourceNotFoundException** Custom Exception
2. Create ErrorDetails class to hold of the custom error response
3. Create GlobalExceptionHandler class to handle specific and global exceptions

Spring Boot REST API Exception Handling



Spring Boot REST API Documentation Using SpringDoc Open API

By Ramesh Fadatare (Java Guides)

SpringDoc

springdoc-openapi java library helps to automate the generation of API documentation using spring boot projects.

springdoc-openapi java library provides integration between spring-boot and swagger-ui.

Automatically generates documentation in JSON/YAML and HTML format APIs.

This library supports:

- OpenAPI 3
- Spring-boot v3 (Java 17 & Jakarta EE 9)
- JSR-303, specifically for @NotNull, @Min, @Max, and @Size
- Swagger-ui
- OAuth 2

This is a community-based project, not maintained by the Spring Framework Contributors

Development Steps

1. Adding **springdoc-openapi** Maven dependency
2. Defining General API Information (Using Annotations)
3. Customizing Swagger API Documentation with Annotations
4. Customizing Swagger Models Documentation with Annotations