# Backend 2.4_CW Exercises

# understanding REST

## OPTIONS

Tells you what verbs are supported at any end point

You'll see browser doing it as a Preflight.

https://developer.mozilla.org/en-US/docs/Glossary/Preflight_request

## PUT - "Replace & Refresh"

Imagine you have an online store and you want to update a product's image. You replace the old image with a new one, giving the product page a fresh look.

## PATCH - "Tiny Touch-ups"

In your online store, a product's price is incorrect. You don't want to change the entire listing, just the price. You make a small update to fix the mistake without affecting the rest of the product details.

## PUT and PATCH

Are not used as much in industry

People use POST for updates directly

## DELETE - "Vanishing Act"

This doesn't mean you have to delete it from database. Generally, you mark the resource as 'deleted' and filter that data out next time.

Your online inventory includes a product that's out of stock and won't be available again. You remove the product listing, and it disappears from your website as if it was never there.

## In short:

- GET: Ask for data.
- POST: Send new data.
- PUT: Update all data.
- PATCH: Update some data.
- DELETE: Remove data.
- OPTIONS: Ask about methods.

# naming for REST

```
Particular review by user for a movie

/movies/:movieId/reviews/:userId
```

COPY

## understanding

1. Nouns:
    - Use descriptive nouns for endpoints that represent resources.
    - Correct: `/orders` Incorrect: `/order-management`
    - Correct: `/employees` Incorrect: `/employee-list`
    - Correct: `/departments` Incorrect: `/dept`
2. Plurals:
    - Use plural forms for resource names to maintain consistency.
    - This leads to `/products` instead of `/product` and `/customers` instead of `/customer`.
    - Correct: `/tasks` Incorrect: `/task`
    - Correct: `/comments` Incorrect: `/comment-section`
    - Correct: `/articles` Incorrect: `/article-list`
3. Predictable:
    - Make your API endpoints easy to predict and follow logical patterns.
    - This helps users anticipate and remember the structure,
        - like `/products/:id` for a specific product.
        - `/users/:id/orders` - Retrieves orders for a specific user. `user-orders/:id`

        - Correct: `/users/:id/orders/:orderId/items`
        - Incorrect: `/user/:id/order/:orderId/item`
    - `/products/:id/reviews` - Retrieves reviews for a specific product.
4. Consistent:
    - Maintain consistency in endpoint structure and naming conventions throughout your API.
    - Use the same format and style for similar resources,
        - e.g., `/invoices/:id` and `**/invoices/:id/items**`
5. Understandable:
    - Prioritize clarity over brevity.
    - Use clear and readable names such as `/card-number` instead of abbreviations like `/pan-no`.
        - Avoid technical jargons
        - Correct: `/customer-details` Incorrect: `/cust-info`
        - Correct: `/product-reviews` Incorrect: `/prod-comms`
6. Hirearchical:
    - `/customers` then individual product related stuff should be done
        - `/customers/:id`
        - Correct: `/customers/:id/addresses`
        - Incorrect: `/customers-addresses/:customerId`
        - `/customers/:id/addresses/:addressId`

📌 Rule: Think of someone else using your API. And this is not just for web services.

## challenge

How would your API look for getting below data look using the above conventions design?

### ex01: design endpoints for your products

1. Adding a new product
2. Retrieving all products
3. Updating a product
4. Deleting a product
5. Retrieving one product details

## solution

1. Adding a new product: `POST /products`
2. Retrieving all products: `GET /products`
3. Updating a product: `PUT /products/:id`
4. Deleting a product: `DELETE /products/:id`
5. Retrieving one product's details: `GET /products/:id`

## ex02: design endpoints for posts

1. Adding a new post
2. Retrieving all posts
3. Updating a specific post
4. Retrieving all posts by a user
5. Retrieving a particular post by a user

## solution

1. Adding a new post: `POST /posts`
2. Retrieving all posts: `GET /posts`
3. Updating a post: `POST /posts/:postId`
4. Retrieving all posts by a user: `GET /posts/:userId/posts`
5. Retrieving a particular post by a user: `GET /posts/:userId/:postId`

## ex03: design endpoints for customers

1. Adding a new customer
2. Retrieving all customers
3. Updating a customer
4. Deleting a customer
5. Retrieving one specific customer’s all orders
6. Retrieving one specific customer’s particular order details

## solution

1. Adding a new customer: `POST /customers`
2. Retrieving all customers: `GET /customers`
3. Updating a customer: `PUT /customers/:customerId`

4. Deleting a customer: `DELETE /customers/:customerId`
5. Retrieving one specific customer’s all orders: `GET /customers/:customerId/orders`
6. Retrieving one specific customer’s particular order details: `GET /customers/:customerId/orders/:orderId`

# http status code

## understanding

2xx (Success):

These codes indicate that the request was successful and the server has processed it as expected. For example, 200 means "OK," which indicates that the request was successful and the server is sending back the requested data.

3xx (Redirection):

These codes indicate that further action needs to be taken to complete the request. They are often used for situations where a resource has moved or the client should try a different URL. For example, 301 means "Moved Permanently," indicating that the requested resource has been permanently moved to a new location.

4xx (Client Error):

These codes indicate that the client (usually the user's browser or application) has made a mistake or the request cannot be fulfilled due to client-side issues. For example, 404 means "Not Found," indicating that the requested resource could not be found on the server.

5xx (Server Error):

These codes indicate that the server encountered an error or is incapable of performing the request. They typically occur when the server is overwhelmed, misconfigured, or experiencing other technical problems. For example, 500 means "Internal Server Error," indicating that an unexpected condition was encountered on the server.

In essence:

- 2xx: Things went well.
- 3xx: Go somewhere else or take more steps.
- 4xx: You made a mistake or the requested resource can't be found.

- 5xx: The server goofed up or can't handle the request.

## challenge

- Q1. I deleted a resource on the server? 204 No Content The server successfully deleted the resource, no response data sent.
- Q2. The server is telling me that the method I used is not allowed for this resource? 405 Method Not Allowed The requested method isn't allowed for this resource.
- Q3. I requested data using a method that the server doesn't support for this resource? 501 Not Implemented The requested method isn't supported by the server.
- Q4. The server is refusing to process my request because it's too large? 413 Request Entity Too Large The request payload is too big for the server to handle.
- Q5. The resource I'm trying to access requires authentication and I haven't provided any credentials? 401 Unauthorized Authentication is needed to access the resource.
- Q6. The server successfully processed my request, but there's no data to send back in the response? 204 No Content Request processed, no response data included.
- Q7. I made a request, but the server wants me to pay or subscribe to access this resource? 402 Payment Required Payment is needed to access the resource.
- Q8. I requested a resource using a method that the server doesn't recognize? 400 Bad Request The request has invalid syntax or structure.
- Q9. The server cannot handle the request because it's currently overloaded or down for maintenance? 503 Service Unavailable Server is temporarily unable to handle requests.
- Q10. I made a request, but the server wants me to include a specific header in the request? 428 Precondition Required A specific header is necessary for the request.