### **<https://shorturl.at/FPHcw>**

### **What is a REST API?**

**REST API**（**RE**presentational **S**tate **T**ransfer **A**pplication **P**rogramming **I**nterface）是一組用於構建和互動的網路服務的規則和約定。它使用標準的 HTTP 方法來對資源執行操作，這些資源通常以 JSON 或 XML 格式表示。

在後端，API 通常需要操作數據作為其運作的一部分。這些數據操作通常稱為 CRUD。CRUD 代表創建（Create）、讀取（Read）、更新（Update）和刪除（Delete），這四種基本操作。

因為 REST API 調用觸發 CRUD 操作，所以理解它們之間的關係對於 API 開發者和數據工程師來說非常重要。

### **什麼是 CRUD？**

CRUD 操作適用於傳統的關聯數據庫管理系統（如 PostgreSQL 或 SQL Server）以及最近的 NoSQL 數據庫（如 MongoDB 或 DynamoDB）。雖然面向文件的操作也管理信息操作，但 CRUD 通常指的是計算機數據庫中的記錄導向操作。

在關聯數據庫管理系統（RDBMS）中，數據庫表的行稱為記錄，而列稱為屬性或字段。讓我們逐一看一下四個 CRUD 操作。

* **創建（Create）**：創建操作添加一個新的數據庫記錄。這個操作的 SQL 等價是 INSERT。
* **讀取（Read）**：讀取操作根據搜索條件從數據庫表（或集合）中檢索記錄（或文檔或項目）。讀取操作的 SQL 等價是 SELECT。讀取操作可以根據查詢返回記錄和字段的子集。
* **更新（Update）**：更新操作，也就是 SQL 中的 UPDATE，修改數據庫中的現有記錄。更新操作可以影響所有或部分記錄和/或字段。
* **刪除（Delete）**：刪除操作，即 SQL 中的 DELETE，從數據庫中移除一個或多個記錄。

在 NoSQL 數據庫中，與 CRUD 操作相對應的表達式取決於平台、數據結構和語言。例如，在 MongoDB 中：

* 創建操作使用 db.collection.insertOne() 或 db.collection.insertMany()。
* 讀取操作使用 db.collection.find() 或 db.collection.findOne()。
* 更新操作使用 db.collection.updateOne()、db.collection.updateMany() 或 db.collection.replaceOne()。
* 刪除操作使用 db.collection.deleteOne() 或 db.collection.deleteMany()。

通常，API 應用代碼通過調用 RDBMS 中的存儲過程、函數或觸發器來執行 CRUD 操作。有時，API 代碼也可以將動態生成的 SQL 命令傳遞給數據庫引擎。

在 NoSQL 數據庫中，API 應用代碼通過數據庫驅動程序發送命令。例如，Java 應用程序可以調用 HBase 驅動程序庫 - 開源的非關係型分散式資料庫（NoSQL）來向數據庫發送 CRUD 命令。在後端，驅動程序將命令從 Java 轉換並在數據庫上運行。

### **Key Principles of REST:**

REST 是一種用於 API 的架構風格，通常用於分佈式應用程序。遵循 REST 架構原則的 REST（或 RESTful）API 是允許客戶端應用程序和其他 API 通過其 API 端點進行交互的應用程序。訪問 RESTful API 的應用程序通常使用 HTTP 協議方法發送請求。REST 的六個指導約束是：

1. **無狀態性 (Statelessness)**: 每個客戶端對 API 的請求必須包含服務器執行任務所需的所有信息。服務器不會保存有關客戶端或服務器的任何會話狀態信息。對於 API 而言，每個客戶端請求都是新的，與之前的請求分開。
2. **基於資源 (Resource-Based)**: 資源（例如，使用者、帖子、產品）由 URL（統一資源定位符）標識。每個資源可以以多種格式表示，JSON 是最常見的格式。
3. **統一介面 (Uniform Interface)**: REST API 應具有統一的方式來與資源互動，使用標準的 HTTP 方法：  
   * GET: Retrieve a resource or a collection of resources.
   * POST: Create a new resource.
   * PUT: Update an existing resource.
   * PATCH: Partially update an existing resource.
   * DELETE: Remove a resource.
4. **客戶端-伺服器架構 (Client-Server Architecture)**: 客戶端和伺服器是獨立實體。客戶端向伺服器發送請求，伺服器處理請求並返回響應。
5. **可緩存響應 (Cacheable Responses)**: 可緩存約束規定 API 的響應決定客戶端是否被允許緩存該 API 的響應。API 標記響應為可緩存或不可緩存。
6. **分層系統（Layered system）：**分層是指包括提供功能的可選組件。例如，負載平衡、驗證等。這些層對客戶端必須是透明的。每個層中的組件不得能夠看到其工作以外的任何內容。

### 

### **HTTP 方法**

客戶端使用 HTTP 方法來包裝 RESTful API 請求：

1. **GET：**GET 請求要求 API 檢索一個資源，例如數據庫記錄或文件內容，並將其發送給客戶端。
2. **POST：**使用 POST 方法，客戶端將數據發送到 API 服務器，服務器根據提供的數據創建一個資源。
3. **PUT：**當客戶端發送 PUT 請求時，它指向特定資源的 URI，並在請求體中提供數據。API 服務器收到 PUT 請求後，會檢查該資源是否已存在。如果存在，則 API 會用 PUT 請求中包含的數據更新該資源；如果不存在，則 API 會用提供的數據創建該資源。
4. **PATCH：**客戶端發送 PATCH 請求以部分更新現有資源。
5. **DELETE：**客戶端發送 DELETE 請求以刪除現有的資源。

### 

### **What is cURL Command?**

cURL 是一個命令行工具，用於從伺服器傳輸數據或向伺服器發送數據，支持多種協議，包括 **HTTP**、**HTTPS** 和 **FTP** 等。它可以用來向 **API** 發送請求、下載文件和測試網絡服務。cURL 提供了多種選項來自定義請求，例如設定 **headers**、指定請求方法（如 **GET**、**POST**）和處理 **authentication**。

### **示例 REST API 請求**

幾個客戶端工具可以向 API 端點發送請求。一個常見的命令行工具是 cURL。以下是使用 cURL 向虛構 REST API 端點發送不同 HTTP 請求的示例。

我們虛構的 REST API 的端點是 http://www.foobar.com，我們將對該 API 發送不同的 HTTP 請求以創建、讀取、更新或刪除客戶記錄。

在以下代碼片段 (Bash)中，cURL 客戶端要求獲取客戶 ID 為 19 的客戶詳情。這是一個 GET 請求。

curl -v http://www.foobar.com/find\_customer\_record?id=19

接下來，我們使用 POST 請求添加一個客戶記錄。在以下示例命令中，我們將數據封裝在 HTTP 請求的主體中。

curl -X POST -d 'id=10&customer\_name=joe\_bloggs' http://www.foobar.com/add\_customer\_record

在這個示例中，我們指向一個包含記錄數據的文件。

curl -X POST -d @customer\_record.json -H "Content-Type: application/json" http://www.foobar.com/add\_customer\_record

要更新客戶記錄，我們發送以下 PUT 請求。

curl -d 'id=10&client\_name=jane\_bloggs' -X PUT http://www.foobar.com/update\_customer\_record

最後，我們刪除客戶 ID 為 22 的客戶：

curl -X DELETE http://www.foobar.com/delete\_customer\_record?id=22

### **CRUD 與 REST：相似之處和不同之處**

**相似之處：**

* **操作方式相似**：RESTful API 和數據庫 CRUD 操作都涉及創建、讀取、更新和刪除數據。在一個應用程式中，當用戶執行這些操作時，前端會發送請求給 API，然後 API 會將這些請求轉換為數據庫的操作。
* **響應**：不管是 REST 還是 CRUD，系統都會向用戶發送回應。例如，API 會返回狀態碼（如 200 表示成功，404 表示找不到資源），而數據庫也會返回操作成功或失敗的代碼。

**不同之處：**

* **工作層面不同**：CRUD 操作主要是在數據庫內部進行的，而 REST API 是通過 HTTP 協議來進行的。REST API 使客戶端和服務器之間的通信變得簡單和標準化。
* **通訊協議**：REST API 通常通過端口 80 或 443 接受請求，而 CRUD 操作的端口由數據庫設定，例如 SQL Server 通常使用 1433 端口。
* **功能範圍**：CRUD 操作可以在 REST API 內執行，但 REST API 不僅限於 CRUD 操作，它還可以調用其他功能或與其他 API 交互。

總之，CRUD 和 REST 是不同的概念，但它們在數據處理和 API 交互中有著緊密的關聯。

### **RESTful Naming Conventions**

Naming conventions for REST APIs are essential for clarity and consistency. Here are some guidelines to follow:

#### **1. Use Nouns for Resources 使用名詞來表示資源**

* Use clear and descriptive nouns to represent the resources.
* Example:
  + /users (for user resources)
  + /posts (for blog posts)

#### **2. Use Plural Nouns for Collections 對於集合使用複數名詞**

* When referring to a collection of resources, use plural nouns.
* Example:
  + /users (collection of users)
  + /products (collection of products)

#### **3. Hierarchical Structure 層次結構**

* Use a hierarchical structure to represent relationships between resources.
* Example:
  + /users/{userId}/posts (posts belonging to a specific user)
  + /posts/{postId}/comments (comments belonging to a specific post)

#### **4. Use HTTP Methods Appropriately 適當使用 HTTP 方法**

* Use the appropriate HTTP methods to indicate actions:
  + GET for retrieving resources
  + POST for creating resources
  + PUT for updating resources
  + DELETE for removing resources

#### **5. Avoid Using Verbs in Endpoint Names 避免在端點名稱中使用動詞**

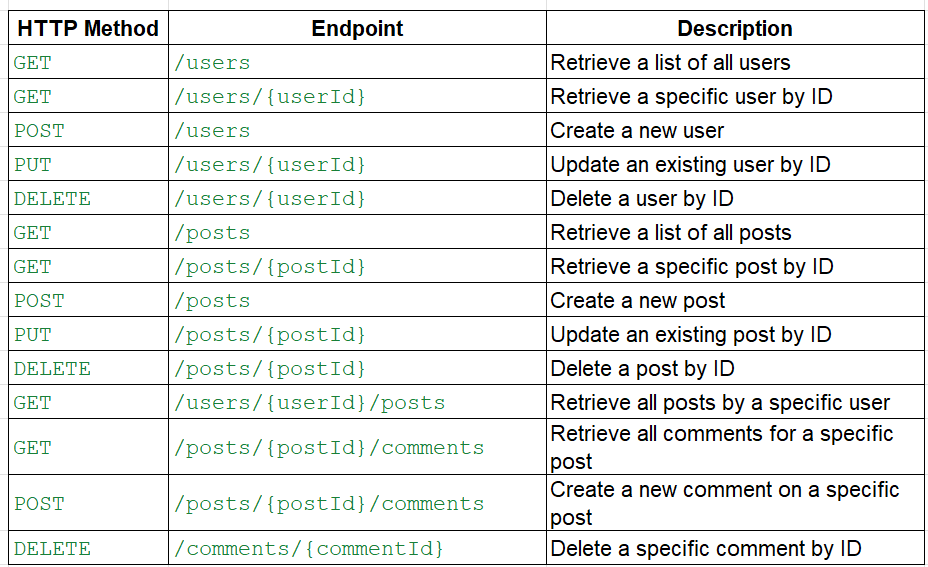
* REST APIs should be resource-oriented, so avoid using verbs in the endpoint names. The HTTP method represents the action.
* Instead of /getUsers, use /users.

#### **6. Use Query Parameters for Filtering and Pagination 使用查詢參數進行過濾和分頁**

* Use query parameters to filter or paginate collections of resources.
* Example:
  + /users?active=true (to retrieve only active users)
  + /posts?page=2&limit=10 (to paginate posts)

### **Example of RESTful Naming**

Here’s a set of example endpoints for a simple blog application using RESTful naming conventions:

****

**Conclusion**

透過遵循 RESTful 命名慣例，您可以創建直觀、一致且易於理解和使用的 API。這種方法有助於構建可擴展和可維護的網絡服務。

### **Exercise 1: Design a REST API for an Online Bookstore**

**Objective:**Create meaningful and consistent REST API endpoint names for an online bookstore that includes various resources such as books, authors, categories, and reviews.

**Instructions:**

**1. Books**

* **CRUD Operations:**
  + **Create**: Add a new book.
  + **Read**:
    - Retrieve all books.
    - Retrieve a specific book by ID.
  + **Update**: Modify an existing book by ID.
  + **Delete**: Remove a book by ID.

**2. Authors**

* **CRUD Operations:**
  + **Create**: Add a new author.
  + **Read**:
    - Retrieve all authors.
    - Retrieve a specific author by ID.
  + **Update**: Modify an existing author by ID.
  + **Delete**: Remove an author by ID.

**3. Categories**

* **CRUD Operations:**
  + **Create**: Add a new category.
  + **Read**:
    - Retrieve all categories.
    - Retrieve a specific category by ID.
  + **Update**: Modify an existing category by ID.
  + **Delete**: Remove a category by ID.

**4. Reviews**

* **CRUD Operations:**
  + **Create**: Add a new review for a specific book.
  + **Read**:
    - Retrieve all reviews for a specific book.
    - Retrieve a specific review by ID.
  + **Update**: Modify an existing review by ID.
  + **Delete**: Remove a review by ID.

**Relationships**

* Each book can have one or more authors.
* Each book belongs to one category.
* Each book can have multiple reviews.

### **Exercise 2: Resources for CRUD Operations in an E-commerce Application**

1. **Products**
   * Represents items available for sale.
   * CRUD Operations:
     + **Create**: Add a new product.
     + **Read**:
       - Retrieve all products.
       - Retrieve a specific product by ID.
       - Query Parameters:
         * Filter by category (e.g., ?categoryId=123)
         * Search by name (e.g., ?search=shirt)
         * Sort by price (e.g., ?sort=price\_asc or ?sort=price\_desc)
         * Pagination (e.g., ?page=2&limit=10)
     + **Update**: Modify an existing product by ID.
     + **Delete**: Remove a product by ID.
2. **Categories**
   * Represents product categories for organization.
   * CRUD Operations:
     + **Create**: Add a new category.
     + **Read**:
       - Retrieve all categories.
       - Retrieve a specific category by ID.
     + **Update**: Modify an existing category by ID.
     + **Delete**: Remove a category by ID.
3. **Customers**
   * Represents users who can purchase products.
   * CRUD Operations:
     + **Create**: Register a new customer.
     + **Read**:
       - Retrieve all customers.
       - Retrieve a specific customer by ID.
     + **Update**: Modify an existing customer by ID.
     + **Delete**: Remove a customer by ID.
4. **Orders**
   * Represents customer orders containing products.
   * CRUD Operations:
     + **Create**: Place a new order.
     + **Read**:
       - Retrieve all orders.
       - Retrieve a specific order by ID.
       - Query Parameters:
         * Filter by customer ID (e.g., ?customerId=456)
         * Filter by order status (e.g., ?status=shipped)
         * Pagination (e.g., ?page=1&limit=5)
     + **Update**: Modify an existing order by ID.
     + **Delete**: Remove an order by ID.
5. **Reviews**
   * Represents customer feedback on products.
   * CRUD Operations:
     + **Create**: Add a new review for a product.
     + **Read**:
       - Retrieve all reviews for a specific product.
       - Retrieve a specific review by ID.
       - Query Parameters:
         * Filter by product ID (e.g., ?productId=789)
         * Sort by rating (e.g., ?sort=rating\_desc)
         * Pagination (e.g., ?page=1&limit=10)
     + **Update**: Modify an existing review by ID.
     + **Delete**: Remove a review by ID.

Advanced Exercises with NodeJs

<https://www.geeksforgeeks.org/rest-api-using-the-express-to-perform-crud-create-read-update-delete/>