MODUL 01

RESTful API - POST

Tujuan Pembelajaran

1. Mahasiswa memahami konsep RESTful API
2. Mahasiswa mampu membuat fungsi POST dengan RESTful API

Materi

1. RESTful API

REST *(Representational State Transfer)* adalah arsitektur API *(Application Programming Interface)* yang menyediakan komunikasi *client-server* untuk aplikasi web melalui protokol HTTP. RESTful API merupakan implementasi dari REST API yang benar-benar mengikuti semua prinsip REST dengan baik. RESTful API beroperasi berdasarkan prinsip CRUD *(Create, Read, Update, Delete)*

2. POST

Salah satu prinsip CRUD yang digunakan RESTful API adalah fungsi *Create* atau biasa dinamakan metode POST. POST digunakan ketika ingin membuat atau menambah data baru ke dalam sistem. POST pada RESTful API akan membuat data baru yang dimasukkan oleh pengguna dan mengirimkannya ke server untuk disimpan.

Deskripsi Tugas

Pada praktikum ini, mahasiswa akan mempraktikkan pembuatan fungsi POST dalam RESTful API. Mahasiswa akan belajar memahami struktur kode program yang paling mendasar dari pembuatan RESTful API dengan fungsi POST. Struktur kode program tersebut memiliki 2 susunan kode, yaitu kode dengan respons sukses dan respons gagal. Struktur kode yang pertama terdiri dari pembuatan fungsi *public* untuk prinsip *Create*, diikuti dengan fungsi *if* yang diperlukan untuk validasi data yang akan dibuat serta diakhiri dengan respons gagal. Struktur kode kedua diawali dengan variabel yang menyimpan data baru dan diikuti dengan respons sukses.

Langkah Praktikum:

1. Deklarasi ***class*** untuk bagian awal dari penulisan kode, tulis seperti contoh di bawah ini

```php
<?php
class CaseCreate {

}
?>
```

2. Buat sebuah variabel ***private*** sebagai properti dari ***class*** dengan nama **`$data`** dan letakkan di dalam ***class***. Variabel data akan menyimpan daftar item yang akan dimasukkan dalam bentuk ***array***.

```php
class CaseCreate {
    private $data = [];
}
```

3. Setelah itu buatlah fungsi ***public*** untuk melakukan ***create*** data. Fungsi ini berada setelah variabel ***private*** yang telah dibuat sebelumnya. Fungsi ini menerima satu parameter **`$item`** yang merupakan ***array*** berisi data item. Ketik seperti contoh berikut

```php
class CaseCreate {
    private $data = [];

    public function create($item) {

    }
}
```

4. Buat sebuah validasi menggunakan ***if*** di dalam fungsi ***public***. Validasi data digunakan untuk mengecek apakah **`$item`** kosong, mengecek apakah **`$item['id']`** dan **`$item['name']`** ada dalam ***array***.

```php
    public function create($item) {
        if (empty($item) || !isset($item['id']) || !isset($item['name'])) {

        }
    }
```

5. Setelah membuat pemilihan, buatlah kembalian dengan **`return`** untuk memberikan respons jika validasi data gagal. Fungsi ini akan mengembalikan status **HTTP 400** ***(Bad Request)*** dengan pesan kesalahan.

```php
    public function create($item) {
        if (empty($item) || !isset($item['id']) || !isset($item['name'])) {
            return [
                'status' => 400,
                'response' => ['message' => 'Invalid Input']
            ];
        }
    }
```

6. Jika validasi sukses maka diperlukan kode program untuk menyimpan data yang telah dimasukkan sebelumnya. Kode ini berfungsi untuk menyimpan **`$item`** ke dalam ***array*** **`$data`** dengan **`id`** sebagai kunci. Tulis kode ini di bawah kode untuk validasi data.

```
12
13          $this->data[$item['id']] = $item;
14      }
```

7. Setelah itu buat kembalian dengan **`return`** untuk validasi data sukses. Fungsi ini akan mengembalikan status **HTTP 201** ***(Created)*** dengan pesan bahwa item berhasil dibuat dan disimpan dalam sistem.

```
13          $this->data[$item['id']] = $item;
14          return [
15              'status' => 201,
16              'response' => ['message' => 'Item Created']
17          ];
```