MODUL 03

RESTful API - PUT

Tujuan Pembelajaran

1. Mahasiswa memahami konsep RESTful API
2. Mahasiswa mampu membuat fungsi PUT 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. PUT

Salah satu prinsip CRUD yang digunakan RESTful API adalah fungsi *PUT* atau biasa dinamakan metode PUT. PUT digunakan ketika ingin mengubah nilai data yang ada dalam sistem. PUT pada RESTful API akan mengambil data yang ditentukan berdasarkan id untuk dilakukan perubahan data dan mengirimkannya kembali ke server untuk disimpan.

Deskripsi Tugas

Pada praktikum ini, mahasiswa akan mempraktikkan pembuatan fungsi PUT dalam RESTful API. Mahasiswa akan belajar memahami struktur kode program yang paling mendasar dari pembuatan RESTful API dengan fungsi PUT. Struktur kode program tersebut memiliki 3 susunan kode, yaitu kode untuk validasi IDE, validasi input dan kode untuk mengubah data. Struktur kode yang pertama terdiri dari pembuatan fungsi *public* untuk prinsip *Update*, diikuti dengan fungsi *if* yang diperlukan untuk validasi data yang akan diubah serta diakhiri dengan respons gagal. Struktur kode kedua diawali dengan pembuatan fungsi *if* sebagai validasi input data yang baru dan diikuti dengan respon gagal. Lalu yang terakhir diawali dengan variabel yang menyimpan perubahan data 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 CaseUpdate {

}
?>
```

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 CaseUpdate {
    private $data = [];
}
```

3. Setelah itu buatlah fungsi ***public*** untuk melakukan ***update*** data. Fungsi ini berada setelah variabel ***private*** yang telah dibuat sebelumnya. Fungsi ini menerima dua parameter yaitu **`$id`** yang merupakan ***id*** milik data item yang ingin diubah dan **`$updatedData`** yang berisi perubahan data yang akan disimpan. Ketik seperti contoh berikut

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

    public function put($id, $updatedData) {

    }
}
```

4. Buat sebuah validasi menggunakan ***if*** di dalam fungsi ***public***. Validasi data digunakan untuk mengecek apakah **`$id`** yang dimasukkan dimiliki oleh data yang tersimpan dalam sistem.

```php
    public function put($id, $updatedData) {
        if (!isset($this->data[$id])) {

        }
    }
```

5. Setelah membuat pemilihan, buatlah kembalian dengan **`return`** untuk memberikan respons jika validasi data gagal. Fungsi ini akan mengembalikan status **HTTP 404** ***(Not Found)*** dengan pesan kesalahan.

```
5       public function put($id, $updatedData) {
6           if (!isset($this->data[$id])) {
7             return [
8               'status' => 404,
9               'response' => ['message' => 'Item Not Found']
10            ];
11          }
12      }
```

6. Jika validasi sukses maka diperlukan validasi data lagi untuk mengecek apakah **`$updatedData`** berisi suatu nilai atau ***null*** atau memiliki nilai dengan variabel yang sesuai dengan ketentuan validasi. Tulis kode ini di bawah kode untuk validasi data.

```
13          if (empty($updatedData) || !isset($updatedData['name'])) {
14
15          }
```

7. Setelah itu buat kembalian dengan **`return`** untuk validasi data gagal. Fungsi ini akan mengembalikan status **HTTP 400** ***(Bad Request)*** dengan pesan kesalahan bahwa data yang dimasukkan tidak sesuai ketentuan.

```
13          if (empty($updatedData) || !isset($updatedData['name'])) {
14            return [
15              'status' => 400,
16              'response' => ['message' => 'Invalid Input']
17            ];
18          }
```

8. Pada bagian bawah validasi data di atas, buat kode program untuk melakukan ***update*** data pada sistem. Tulis seperti di bawah ini.

```
20            $this->data[$id]['name'] = $updatedData['name'];
```

9. Setelah itu buat kembalian dengan **`return`** untuk validasi data sukses. Fungsi ini akan mengembalikan status **HTTP 200** ***(OK)*** dengan pesan bahwa data yang dipilih telah berhasil diperbarui.

```
20          $this->data[$id]['name'] = $updatedData['name'];
21          return [
22            'status' => 200,
23            'response' => ['message' => 'Item Updated']
24          ];
```