

## BERKENALAN LEBIH DEKAT DENGAN SLIM FRAMEWORK

Ada kalanya kita dibebankan tugas untuk membuat REST API services yang mudah diakses, ringan dan tentunya sesuai dengan kebutuhan. Kita akan mulai mencari framework yang sesuai dengan kriteria. Seperti yang kita ketahui, framework PHP baru hampir bermunculan tiap harinya. Tapi hampir semuanya menganut paham MVC seperti pada framework-framework modern pada umumnya. Ada yang *full stack* framework dan ada juga yang micro framework.

*Full stack* framework lebih kompleks dengan membawa banyak library-library umum yang biasanya diperlukan oleh developer, contohnya seperti Yii, CodeIgniter, CakePHP, dan yang lainnya. Sedangkan micro framework hanya menyediakan beberapa fitur standar dari framework. Micro framework digunakan untuk aplikasi yang tidak membutuhkan *extensive caching*, interaksi database yang berat atau keamanan. Contohnya seperti Slim, Silex, Tonic, dan yang lainnya.

Di artikel kali ini, kita akan mencoba untuk berkenalan dengan salah satu micro framework yang cukup *powerfull* untuk digunakan sebagai REST API service, yaitu Slim Framework. Slim Framework terinspirasi dari Sinatra, salah satu micro framework pada bahasa pemrograman ruby. Sesuai dengan namanya, 'slim' yg memang benar-benar 'ramping dan ringkas serta tidak di-bundling dengan library-library selengkap full stack framework. Kelebihannya yg menurut saya paling mencolok, fitur router'nya yang benar-benar membantu saat membuat restfull api utk aplikasi mobile. Dengan mengimplementasikan http method seperti get, post, put, dan delete yang benar-benar keren dan elegan. Ditambah dengan instalasinya yang menggunakan composer, kita dengan mudah menambahkan library dari packagist. Di bawah ini beberapa fitur yang disediakan oleh slim framework dan versi php yang dibutuhkan.

### FEATURES

- Powerful router
  - Standard and custom HTTP methods
  - Route parameters with wildcards and conditions
  - Route redirect, halt, and pass
  - Route middleware
- Template rendering with custom views
- Flash messages
- Secure cookies with AES-256 encryption
- HTTP caching
- Logging with custom log writers
- Error handling and debugging
- Middleware and hook architecture
- Simple configuration

### SYSTEM REQUIREMENTS

PHP 5.3.0 or newer

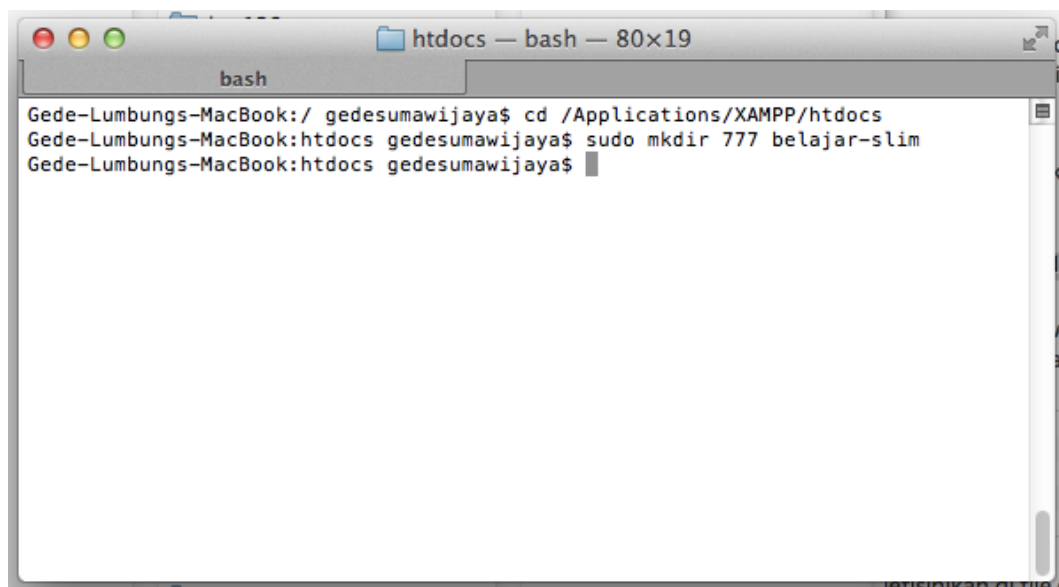
Di artikel ini, kita akan mencoba membahas tentang slim framework mulai dari proses instalasi, membuat restfull API, sampai implementasi rest API untuk aplikasi client.

## 1. Instalasi Slim Framework via Composer

Pada artikel ini, saya menggunakan system operasi mac-osx, rekan-rekan bias menggunakan system operasi unix-based lainnya seperti linux dengan berbagi varian distronya. Saya sangat sarankan menggunakan system operasi unix-based, karena akan mempermudah kita ke depannya.

Tutorial instalasi di website resminya sudah sangat jelas. Kita bisa menggunakan composer maupun instal secara manual. Berhubung saya lagi *kesemsem* sama yang namanya composer, jadi saya pakai tools yang satu ini saja. Untuk penjelasan tentang composer, sudah dijelaskan dengan sangat jelas oleh mas Taufan Aditya di seri e-magazine php versi awal.

- a. Buka terminal/console, masuk ke direktori htdocs. Buat sebuah folder baru dengan nama "belajar-slim"



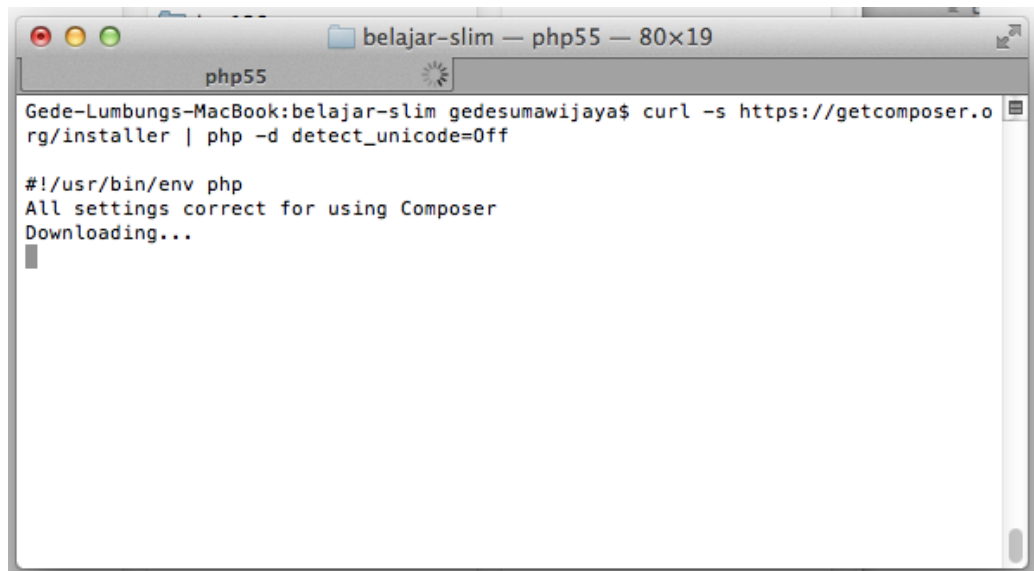
```
htdocs — bash — 80x19
bash
Gede-Lumbungs-MacBook:/ gedesumawijaya$ cd /Applications/XAMPP/htdocs
Gede-Lumbungs-MacBook:htdocs gedesumawijaya$ sudo mkdir 777 belajar-slim
Gede-Lumbungs-MacBook:htdocs gedesumawijaya$
```

- b. Masuk ke direktori "belajar-slim".

```
cd belajar-slim
```

- c. Sekarang kita harus mengunduh file composer terlebih dahulu, dengan menggunakan perintah berikut

```
curl -s https://getcomposer.org/installer | php -d
detect_unicode=Off
```

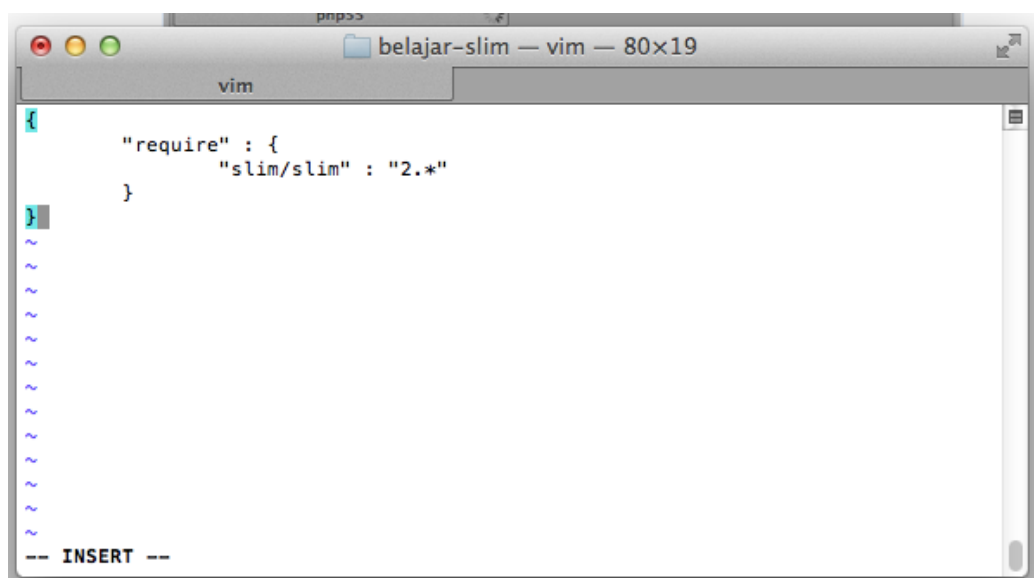


A terminal window titled 'belajar-slim — php55 — 80x19'. The prompt is 'gedesumawijaya\$'. The command entered is 'curl -s https://getcomposer.org/installer | php -d detect\_unicode=0ff'. The output shows the Composer installer script running, displaying 'All settings correct for using Composer' and 'Downloading...'. The cursor is on a new line below the output.

```
gedesumawijaya$ curl -s https://getcomposer.org/installer | php -d detect_unicode=0ff

#!/usr/bin/env php
All settings correct for using Composer
Downloading...
```

- d. Buat sebuah file composer.json dengan menggunakan vim editor dan simpan dengan menekan kombinasi tombol keyboard :w

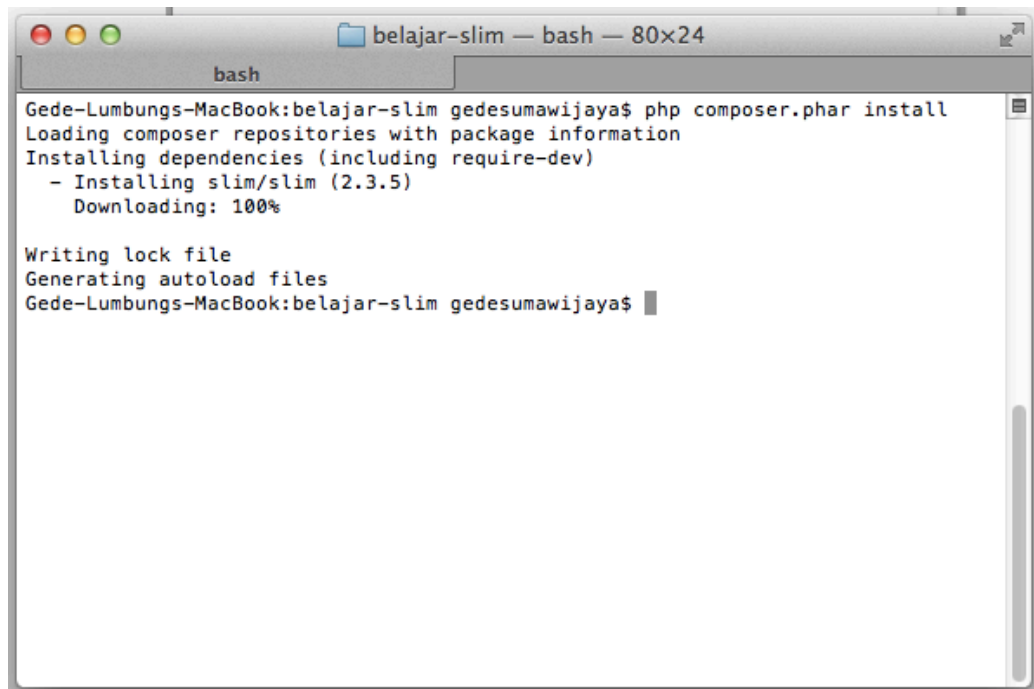


A vim editor window titled 'belajar-slim — vim — 80x19'. The editor is in INSERT mode, as indicated by '-- INSERT --' at the bottom. The file 'composer.json' is being created with the following content: a JSON object with a 'require' key containing an object with 'slim/slim' set to '2.\*'. The cursor is at the end of the first line of the JSON object.

```
{
    "require" : {
        "slim/slim" : "2.*"
    }
}
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

- e. Kita perintahkan composer untuk mengunduh file slim framework yang sudah kita tulis di dalam file composer.json dengan menggunakan perintah berikut via terminal/console

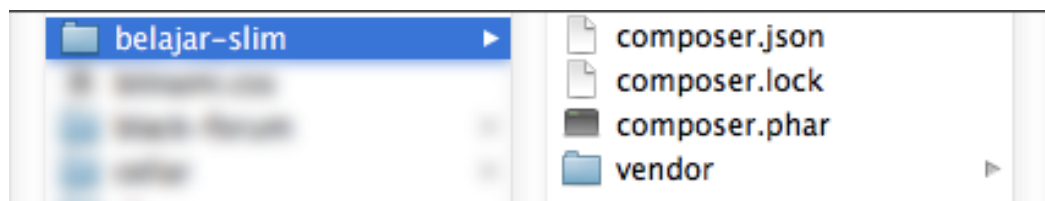
```
php composer.phar install
```



```
gedesumawijaya$ php composer.phar install
Loading composer repositories with package information
Installing dependencies (including require-dev)
- Installing slim/slim (2.3.5)
  Downloading: 100%

Writing lock file
Generating autoload files
gedesumawijaya$
```

- f. Buka file explorer dan masuk ke folder htdocs/belajar-slim, maka akan terdapat file slim framework yang telah selesai diunduh via composer. Di dalam folder vendor/slim/slim, terdapat file index.php dan kode aplikasi akan kita tuliskan di dalam file index.php.



- g. Jika kita langsung mengakses file index.php di atas melalui web browser dengan alamat <http://localhost/belajar-slim/vendor/slim/slim>, maka akan muncul halaman welcome screen seperti di bawah ini :

# Slim

## Welcome to Slim!

Congratulations! Your Slim application is running. If this is your first time using Slim, start with this ["Hello World" Tutorial](#).

### Get Started

1. The application code is in `index.php`
2. Read the [online documentation](#)
3. Follow [@slimphp](#) on Twitter

## 2. Membuat RESTful API Dengan Slim Framework

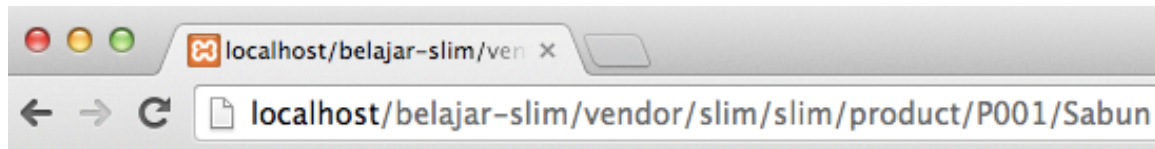
Slim framework terinspirasi dari Sinatra framework sebuah framework untuk bahasa pemrograman ruby ([http://en.wikipedia.org/wiki/Sinatra\\_\(software\)](http://en.wikipedia.org/wiki/Sinatra_(software))),. Gaya penulisan untuk kodenya mirip dengan Sinatra. Yaitu dengan mengimplementasikan http method seperti GET, POST, DELETE dan PUT. Berikut penjelasan masing-masing method yang saya kutip dari <http://restapitutorial.com>.

HTTP Verb	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
GET	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	404 (Not Found), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
POST	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found).
DELETE	404 (Not Found), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Contoh penggunaan method GET pada Slim framework ialah seperti di bawah ini. Buka file index.php, dan ganti dengan kode di bawah ini :

```
<?php
require 'Slim/Slim.php';
\Slim\Slim::registerAutoloader();
$app = new \Slim\Slim();
$app->get('/product/:code/:name', function ($code,$name) {
    echo "<p>Code : $code<p>";
    echo "<p>Product Name : $name<p>";
});
$app->run();
```

Kemudian kita akses via web browser dengan alamat <http://localhost/belajar-slim/vendor/slim/slim/product/P001/Sabun> maka akan muncul tampilan seperti di bawah ini. Terdapat 2 parameter di url tersebut, yaitu kode produk dan nama produk. Parameter ini bisa diubah sesuai kebutuhan kita.



Code : P001

Product Name : Sabun

OK deh, selanjutnya mari kita lanjutkan ke langkah-langkah untuk membuat sebuah RESTFul API sederhana dengan Slim Framework. Disini saya mengambil studi kasus sebuah tabel data customer. Kita buat terlebih dahulu tabelnya seperti di bawah ini :

```
CREATE TABLE IF NOT EXISTS `tbl_customer` (  
  `id_customer` INT( 5 ) NOT NULL AUTO_INCREMENT ,  
  `nama_customer` VARCHAR( 100 ) NOT NULL ,  
  `alamat` TEXT NOT NULL ,  
  `telepon` VARCHAR( 50 ) NOT NULL ,  
  `tempat_lahir` VARCHAR( 100 ) NOT NULL ,  
  `tgl_lahir` VARCHAR( 100 ) NOT NULL ,  
  PRIMARY KEY ( `id_customer` )  
)  
  
CREATE TABLE IF NOT EXISTS `tbl_api_reg` (  
  `id_api_reg` INT( 5 ) NOT NULL AUTO_INCREMENT ,  
  `email` VARCHAR( 50 ) NOT NULL ,  
  `api_key` VARCHAR( 50 ) NOT NULL ,  
  PRIMARY KEY ( `id_api_reg` )  
)
```

Kita sediakan 2 buah tabel, yaitu tabel customer yang berisikan data pelanggan dan tabel api registration yang berisikan API key yang telah terdaftar. Untuk menampilkan data pelanggan, kita harus menyertakan API key yang telah terdaftar di tabel api registration. Yaw mirip seperti kebanyakan website yang menyediakan API untuk para developer luar.

- a. Buka file index.php dengan editor kesayangan rekan-rekan. Kita akan mencoba untuk membuat sebuah RESTFul API yang memerlukan API Key untuk mengakses data di server. API Key kita simpan di tabel tbl\_api\_reg. Hanya API Key yang terdaftar di tabel tersebut saja yang bisa mendapatkan akses data dari tabel tbl\_customer. Pertama, kita memanggil file Slim/Slim.php dan buat sebuah method koneksi ke database.

```

<?php
require 'Slim/Slim.php';

\Slim\Slim::registerAutoloader();

$app = new \Slim\Slim();

function getConnection() {
    $dbhost="127.0.0.1";
    $dbuser="root";
    $dbpass="";
    $dbname="db_customer";
    $dbh = new PDO("mysql:host=$dbhost;dbname=$dbname",
    $dbuser, $dbpass);
    $dbh->setAttribute(PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION);
    return $dbh;
}

```

- b. Untuk autentikasi api key, kita buat sebuah method `validateApiKey($key)` yang nantinya method ini akan kita panggil dengan middleware yang disisipkan melalui routing.

```

function validateApiKey($key) {
    $sql = "select * FROM tbl_api_reg where
    api_key='".$key."'";
    $db = getConnection();
    $sth = $db->prepare($sql);
    $sth->execute();
    return $sth->rowCount();
}



$authKey = function ($route) {
    $app = \Slim\Slim::getInstance();
    $routeParams = $route->getParams();
    if (validateApiKey($routeParams["key"])==0)
    {
        $app->halt(401);
    }
};

```

- c. Selanjutnya, kita buat sebuah route baru dengan nama **customer** yang akan menampilkan seluruh data di dalam tabel `tbl_customer`. Jika diakses tanpa API Key atau API Key tidak valid, maka akan muncul tampilan kosong di browser. Dan jika diakses melalui extension Chrome Advance REST Client, akan muncul status **401 Unauthorized**.

Status	401 Unauthorized ? Loading time: 116 ms
Request headers	User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.36 (Content-Type: text/plain; charset=utf-8 ? Accept: */*

Jika diakses dengan API Key yang valid, maka akan muncul status **200 OK**. Dan data ditampilkan.

Status	200 OK  Loading time: 53 ms
Request headers	User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit Content-Type: text/plain; charset=utf-8  Accept: */*

```
$app->get('/customer/:key/', $authKey, function () use ($app)
{
    $sql = "select * FROM tbl_customer ";
    try {
        $db = getConnection();
        $stmt = $db->query($sql);
        $data = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        $app->response()->header('Content-Type',
            'application/json');
        echo '{"data": ' . json_encode($data) . '}';
    }
    catch(PDOException $e) {
        echo '{"error":{"text":' . $e->getMessage()
        . '}}';
    }
});
```

Jika ingin diakses via terminal/console, bisa menggunakan perintah :

```
curl -i -X GET http://localhost/belajar-
slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd
```

- d. Adakalanya kita hanya ingin mengakses satu baris data saja berdasarkan ID, kita harus membuat routing baru seperti di bawah ini.



```

$app->get('/customer/:key/:id/', $authKey, function ($key,$id)
use ($app) {
    try {
        $sql = "select * FROM tbl_customer where id_customer
='".$id."";
        $db = getConnection();
        $stmt = $db->query($sql);
        $data = $stmt->fetch(PDO::FETCH_OBJ);

        $db = null;
        $app->response()->header('Content-Type',
'application/json');
        echo '{"data": ' . json_encode($data) . '}';

    } catch (Exception $e) {
        $app->response()->status(400);
        $app->response()->header('X-Status-Reason', $e-
->getMessage());
    }
});

```

Contohnya jika ingin mengakses data dengan ID = 1, bisa menggunakan perintah berikut via console :

```

curl -i -X GET http://localhost/belajar-
slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd/1

```

- e. Route dengan method POST kita gunakan untuk menambah data ke dalam tabel. Data dalam format json di push dari client dan ditangkap dengan perintah `$app->request();` oleh RESTful API.

```

$app->post('/customer/:key/', $authKey, function () use ($app) {
    try {
        $request = $app->request();
        $input = json_decode($request->getBody());
        $sql = "INSERT INTO tbl_customer (nama_customer, alamat, telepon,
tempat_lahir, tgl_lahir) VALUES (:nama_customer, :alamat, :telepon,
:tempat_lahir, :tgl_lahir)";

        $db = getConnection();
        $stmt = $db->prepare($sql);
        $stmt->bindParam("nama_customer", $input->nama_customer);
        $stmt->bindParam("alamat", $input->alamat);
        $stmt->bindParam("telepon", $input->telepon);
        $stmt->bindParam("tempat_lahir", $input->tempat_lahir);
        $stmt->bindParam("tgl_lahir", $input->tgl_lahir);

        $stmt->execute();
        $data = $db->lastInsertId();
        $db = null;
        echo json_encode($input);

    } catch (Exception $e) {
        $app->response()->status(400);
        $app->response()->header('X-Status-Reason', $e->getMessage());
    }
});

```

```
curl -i -X POST -H 'Content-Type: application/json' -d
'{"nama_customer": "Dedek", "alamat": "Bandung", "telepon" :
"083856764522" , "tempat_lahir" : "Jakarta" , "tgl_lahir" :
"09/12/1978"}' http://localhost/belajar-
slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd
```

- f. Untuk melakukan update terhadap data, kita bisa menggunakan method PUT.

```
$app->put('/customer/:key:id/', $authKey, function ($key,$id)
use ($app) {
    try {
        $request = $app->request();
        $input = json_decode($request->getBody());
        $sql = "UPDATE tbl_customer set
nama_customer=:nama_customer, alamat=:alamat,
telepon=:telepon, tempat_lahir=:tempat_lahir,
tgl_lahir=:tgl_lahir where id_customer='".$id."'";

        $db = getConnection();
        $stmt = $db->prepare($sql);
        $stmt->bindParam("nama_customer", $input->nama_customer);
        $stmt->bindParam("alamat", $input->alamat);
        $stmt->bindParam("telepon", $input->telepon);
        $stmt->bindParam("tempat_lahir", $input->tempat_lahir);
        $stmt->bindParam("tgl_lahir", $input->tgl_lahir);

        $stmt->execute();
        $db = null;
        echo json_encode($input);

    } catch (Exception $e) {
        $app->response()->status(400);
        $app->response()->header('X-Status-Reason', $e-
->getMessage());
    }
});
```

```
curl -i -X PUT -H 'Content-Type: application/json' -d
'{"nama_customer": "Dedek Irawan", "alamat": "Bandung", "telepon"
: "083856764522" , "tempat_lahir" : "Jakarta" , "tgl_lahir" :
"09/12/1978"}' http://localhost/belajar-
slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd/5
```

- g. Dan jika ingin menghapus data, kita dapat menggunakan method DELETE

```
$app->delete('/customer/:key/:id/', $authKey, function
($key,$id) use ($app) {
    try {
        $sql = "DELETE FROM tbl_customer WHERE
id_customer='".$id."'";
        $db = getConnection();
        $stmt = $db->prepare($sql);
        $stmt->bindParam("id", $id);
        $stmt->execute();
        $db = null;

    } catch (Exception $e) {
        $app->response()->status(400);
        $app->response()->header('X-Status-Reason', $e-
->getMessage());
    }
});
```

```
curl -i -X DELETE http://localhost/belajar-
slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd/5
```

- h. Jangan lupa untuk menambahkan perintah `$app->run();` untuk mengeksekusi semua routing yang sudah kita buat tadi.

Nah, cukup mudah kan untuk membuat sebuah RESTful API dengan Slim Framework,,,?? Sekarang memang lagi keren-keren'nya aplikasi yang dibangun dengan RESTful API, karena dengan begitu target aplikasi di client bisa bermacam-macam. Bisa dibuat ke aplikasi mobile, desktop bahkan web. Sebenarnya pada aplikasi ini bisa kita tambahkan autentikasi tambahan, dimana user yang ingin mengakses data dari REST API harus login terlebih dahulu. Monggo kawan-kawan yang menambahkan. Sebenarnya tadi sudah sempat saya tambahkan, cuma agak kesulitan mencobanya melalui extension chrome Advance Rest Client.

### 3. Implementasi RESTFul API Untuk Aplikasi Client

Setelah membuat RESTFul API, sekarang langkah selanjutnya adalah membuat aplikasi di sisi client. Implementasinya terserah bisa menggunakan bahasa pemrograman apapun, karena hasil output dari RESTFul API ini berupa format JSON yang rata-rata bisa di-parsing oleh masing-masing bahasa pemrograman modern. Untuk contoh implementasinya, saya disini menggunakan html+ajax+jQuery.

a. Buat sebuah file html seperti di bawah ini

```
<!DOCTYPE HTML>
<html>
<head>
<title>CRUD RESTFul API</title>
  <link rel="stylesheet" href="css/styles.css" />
</head>

<body>
  <form>
    <input id="id" name="id" type="hidden" />

    <label>Nama Customer:</label>
    <input type="text" id="nama_customer"
name="nama_customer" required>

    <label>Alamat:</label>
    <input type="text" id="alamat" name="alamat"/>

    <label>Telepon:</label>
    <input type="text" id="telepon" name="telepon"/>

    <label>Tempat Lahir:</label>
    <input type="text" id="tempat_lahir"
name="tempat_lahir"/>

    <label>Tanggal Lahir:</label>
    <input type="text" id="tgl_lahir" name="tgl_lahir"/>

    <button id="SaveBtn">Simpan</button>
    <button id="DeleteBtn">Hapus</button>
  </form>
  <div class="content">
    <button id="AddBtn">Tambah Data</button>
    <ul id="ListData">
      <li>Loading Data...</li>
    </ul>
  </div>
  <script src="js/jquery.min.js"></script>
  <script src="js/script.js"></script>

</body>
</html>
```

- b. Sekarang kita akan membuat sebuah file yang berisikan kode javascript untuk terhubung dengan REST API via ajax yang telah kita buat sebelumnya. Disini saya tidak akan membahas lebih detail tentang ajax.

```
var URL = "http://localhost/belajar-slim/vendor/slim/slim/customer/d2ba5ac651d985a7fad886044d92b5cd";

showAllData();
$('#DeleteBtn').hide();

$('#ListData a').live('click', function() {
    selectData($(this).data('identity'));
});

$('#AddBtn').click(function() {
    newData();
    return false;
});

$('#SaveBtn').click(function() {
    if ($('#id').val() != '')
    {
        updateData();
    }
    else
    {
        addData();
    }
    return false;
});

$('#DeleteBtn').click(function() {
    deleteData();
    return false;
});

function newData() {
    $('#DeleteBtn').hide();
    setDetail({});
}

function showAllData() {
    $.ajax({
        type: 'GET',
        url: URL,
        dataType: "json",
        success: setDataList
    });
}

function selectData(id_customer) {
    $.ajax({
        type: 'GET',
        url: URL + '/' + id_customer,
        dataType: "json",
        success: function(customer){
            $('#DeleteBtn').show();
            console.log('selectData success: ' +
customer.id_customer);
            setDetail(customer);
        }
    });
}
```

```

function addData() {
    $.ajax({
        type: 'POST',
        contentType: 'application/json',
        url: URL,
        dataType: "json",
        data: parseToJson(),
        success: function(data, status, jqXHR){
            $('#DeleteBtn').show();
            $('#id').val(data.id_customer);
            showAllData();
        },
        error: function(jqXHR, status, errorThrown){
            alert('addData error: ' + status);
        }
    });
}

function updateData() {
    $.ajax({
        type: 'PUT',
        contentType: 'application/json',
        url: URL + '/' + $('#id').val(),
        dataType: "json",
        data: parseToJson(),
        success: function(data, status, jqXHR){
            showAllData();
        },
        error: function(jqXHR, status, errorThrown){
            alert('updateData error: ' + status);
        }
    });
}

function deleteData() {
    $.ajax({
        type: 'DELETE',
        url: URL + '/' + $('#id').val(),
        success: function(data, status, jqXHR){
            showAllData();
            setDetail({});
            $('#DeleteBtn').hide();
        },
        error: function(jqXHR, status, errorThrown){
            alert('deleteData error');
        }
    });
}

function setDetail(customer) {
    $('#id').val(customer.id_customer);
    $('#nama_customer').val(customer.nama_customer);
    $('#alamat').val(customer.alamat);
    $('#telepon').val(customer.telepon);
    $('#tempat_lahir').val(customer.tempat_lahir);
    $('#tgl_lahir').val(customer.tgl_lahir);
}

```

```

function setDataList(data) {
    var datalist = data == null ? [] : (data.customer
instanceof Array ? data.customer : [data.customer]);
    $('#ListData li').remove();
    $.each(datalist, function(index, customer) {

        $('#ListData').append('<li><div>'+customer.nama_customer
+'</div><div>'+customer.alamat+'</div><div>'+customer.telepon+
'</div><div>'+customer.tempat_lahir+'</div><div>'+customer.tgl
_lahir+'</div><div><a href="#" data-identity="' +
customer.id_customer + '">Edit</a></div></li>');
    });
}

function parseToJson() {
    var data = JSON.stringify({
        "id_customer": $('#id_customer').val(),
        "nama_customer": $('#nama_customer').val(),
        "alamat": $('#alamat').val(),
        "telepon": $('#telepon').val(),
        "tempat_lahir": $('#tempat_lahir').val(),
        "tgl_lahir": $('#tgl_lahir').val()
    });
    return data;
}

```

- c. Pada URL, jangan lupa menambahkan API Key yang sudah terdaftar di dalam tabel api key. Jika dijalankan via browser, maka hasilnya akan seperti di bawah ini :

← → ↻  ☆ 📧 🌐 🤖 🔄 ☰

**Nama Customer:**

**Alamat:**

**Telepon:**

**Tempat Lahir:**

**Tanggal Lahir:**

Anto	Karangasem	0333467890	Denpasar	04/02/1991	Edit
Budi	Denpasar	0361465676	Denpasar	04/02/1992	Edit
Indah Kalong	Banyuwangi	0333492932	Banyuwangi	12/03/1992	Edit

Masih banyak fitur-fitur dari Slim framework yang bisa diimplementasikan untuk membuat sebuah RESTful API services. Seperti menambahkan autentikasi apakah user sudah login atau belum ketika akan mengakses suatu url yang khusus. OK deh, sekian dulu artikel tentang Slim framework kali ini. Semoga bermanfaat untuk rekan-rekan 😊

### **Download Source Code**

<https://github.com/gedelambung/SlimCRUDClient>

### **Profil Penulis**



**Gede Lumbung**

Email : [gedesumawijaya@gmail.com](mailto:gedesumawijaya@gmail.com)

Website : <http://gedelambung.com>