
	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 1 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

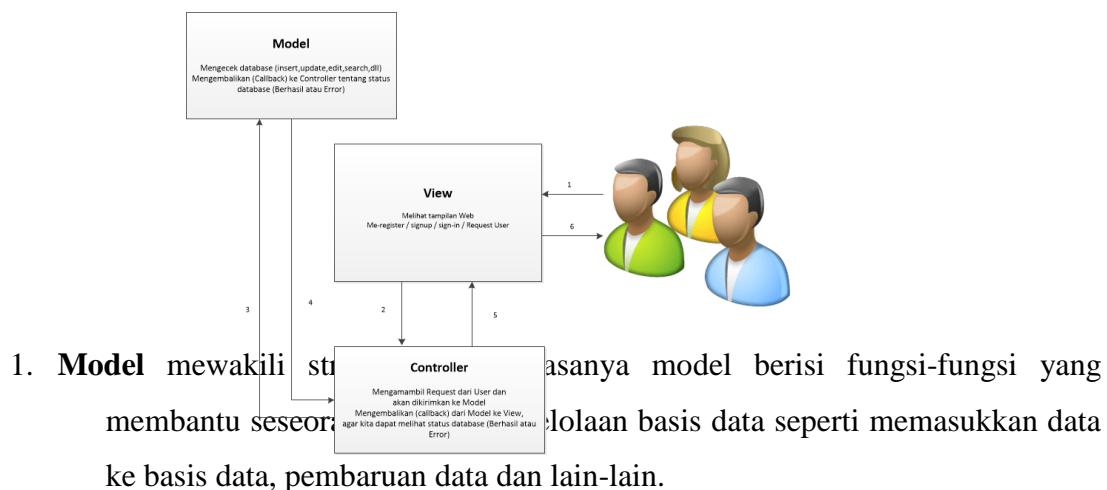
---

<b>Minggu Ke</b>	<b>:</b>	<b>1</b>
<b>Capaian Pembelajaran</b>	<b>:</b>	<b>Pengenalan Framework</b>
<b>Waktu</b>	<b>:</b>	<b>2 x 170 menit</b>
<b>Tempat</b>	<b>:</b>	<b>Laboratorium</b>

---



- |                                    |          |                                     |
|------------------------------------|----------|-------------------------------------|
| <b>1. Sub Capaian Pembelajaran</b> | <b>:</b> | <b>Pengenalan Framework</b>         |
| <b>2. Indikator Kinerja</b>        | <b>:</b> | <b>Mahasiswa mengenal Framework</b> |
| <b>3. Teori</b>                    | <b>:</b> |                                     |

Berikut Konsep Model-View-Controller (MVC)



2. **View** adalah bagian yang mengatur tampilan ke pengguna. Bisa di katakan berupa halaman.
3. **Controller** merupakan bagian yang menjembatani model dan view. Controller berisi perintah- perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman.

- |          |                             |          |  |
|----------|-----------------------------|----------|--|
| <b>4</b> | <b>Bahan dan Alat</b>       | <b>:</b> | Komputer dan Logbook   |
| <b>5</b> | <b>Organisasi</b>           | <b>:</b> | Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah  |
| <b>6</b> | <b>Prosedur Kerja</b>       | <b>:</b> |  |
| <b>7</b> | <b>Tugas dan Pertanyaan</b> | <b>:</b> | Disesuaikan dengan kondisi dan situasi.  |
| <b>8</b> | <b>Pustaka</b>              | <b>:</b> | 1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011. <a href="http://www.koder.web.id">www.koder.web.id</a><br>2. Tutorialspoint. Codeigniter. 2015. <a href="http://www.tutorialspoint.com">www.tutorialspoint.com</a> |
| <b>9</b> | <b>Hasil Praktikum</b>      | <b>:</b> | <b>Laporan Praktikum</b>   |

	<p align="center"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 3 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

**Minggu Ke** : 2  
**Capaian Pembelajaran** : Javascript Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

---

- 
1. **Sub Capaian Pembelajaran** : Pengenalan Node JS
  2. **Indikator Kinerja** : Mahasiswa mampu menginstal dan menggunakan NodeJS.
  3. **Teori** : —
  4. **Bahan dan Alat** : Komputer dan Logbook
  5. **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  6. **Prosedur Kerja** :

Node.js merupakan platform yang sangat banyak dibicarakan oleh web developer saat ini. Node.js sendiri adalah platform berbasis javascript yang berfungsi untuk dieksekusi di sisi server

(*server side*). Selama ini kita mengetahui bahasa untuk *server side* adalah PHP, sekarang ada node.js yang bisa dieksekusi disisi server menggunakan bahasa javascript.

Kenapa harus menggunakan node.js?

Node.js sangat bagus untuk membuat aplikasi berbasis web yang super sibuk. Node.js juga sangat bagus digunakan untuk membuat aplikasi berbasis web yang datanya real-time. Node.js akan membuat aplikasi berbasis web menjadi lebih cepat dan ringan, dikarenakan node.js menggunakan teknik *non-blocking*.

Apa itu teknik *non-blocking*?

Non-blocking merupakan metode yang digunakan oleh nodejs dalam menghandle multiple request. Dengan kata lain, teknik non-blocking ini mengeksekusi proses tanpa harus menunggu proses sebelumnya selesai. Jika dianalogikan seperti sebuah restoran. Dimana pelayan bertugas mencatat pesanan dan menyerahkannya ke koki untuk di masak. Nah, pelayan dapat mencatat pesanan yang lain tanpa harus menunggu pesanan pertama selesai dimasak oleh koki. Jadi kira-kira seperti itulah metode non-blocking yang digunakan oleh node.js.

Tahap Instalasi NodeJS .

### 1. Download node.js di official websitenya yaitu <https://nodejs.org/>

Silahkan download sesuai dengan Operating System (OS) yang Anda gunakan. Biasanya <https://nodejs.org/> akan mengenal otomatis OS yang Anda gunakan, jadi anda bisa langsung klik tombol download.

### 2. Instalasi Node.js

Setelah download silahkan install nodejs di komputer Anda!.

### 3. Cek instalasi

Untuk mengecek apakah instalasi berhasil, silahkan buka **Command Prompt**. Dengan cara Ctrl+R kemudian ketikan cmd, maka akan tampil jendela command prompt-nya. Setelah tampil jendela command prompt, ketikan perintah:

```
1 node -v
```

Lalu enter, jika berhasil maka akan tampil versi dari node.js yang digunakan seperti berikut:

```
1 v7.9.0
```

Setelah itu, cek juga **npm**-nya. Apa itu npm?. Hmm.,npm merupakan **package manager**-nya node.js. Kalau di PHP itu ada namanya Composer sebagai package manager-nya.

Untuk mengecek **npm** silahkan ketikan perintah berikut pada command prompt:

```
1 npm -v
```

Lalu enter, jika berhasil maka akan tampil versi dari **npm** yang digunakan seperti berikut:

```
1 4.2.0
```

Sampai disini, sebenarnya instalasi node.js telah berhasil. Tetapi untuk lebih meyakinkan, mari kita buat sebuah aplikasi sederhana dengan menampilkan “hello world”.

Pertama buat sebuah folder baru dengan nama **nodejs**. Disini saya membuatnya di my documents. Kemudian, buat file javascript dengan nama **app.js** dan letakkan didalam folder **nodejs** yang tadi dibuat. Buka file **app.js** dengan texteditor seperti sublimetext, notepad, nodepad++, atau lainnya. Kemudian ketikkan perintah berikut:

```
1 console.log('Hello World');
```

Lalu save (Ctrl+S). Kemudian kembali ke Command Prompt dan panggil **app.js**. untuk memanggilnya Anda harus berada pada directory **my documents/nodejs/**. Jika belum, change directory-nya dengan perintah berikut:

```
1 cd my documents/nodejs
```

sehingga terlihat seperti ini di command prompt Anda:

```
1 C:\Users\M Fikri\My Documents\nodejs>
```

Nah jika sudah seperti ini. Silahkan jalankan **app.js** dengan perintah berikut:

```
1 C:\Users\M Fikri\My Documents\nodejs>node app.js
```



Jika berhasil maka akan terlihat hasilnya seperti berikut pada command prompt Anda:

```
1 Hello World
```

**7 Tugas dan Pertanyaan** : Buatlah Form untuk Penambahan dan Pengurangan

**8 Pustaka** : 1. Daqiqil, Ibnu. Framework Codeigniter :  
Sebuah Panduan dan Best Practice. 2011.  
www.koder.web.id  
2. Tutorislpoint. Codeigniter. 2015.  
www.tutorialspoint.com

**9 Hasil Praktikum** : **Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 6 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

<b>Minggu Ke</b>	:	3
<b>Capaian Pembelajaran</b>	:	Javascript Framework
<b>Waktu</b>	:	2 x 170 menit
<b>Tempat</b>	:	Laboratorium

---

- |                                    |   |   |
|------------------------------------|---|---|
| 1. <b>Sub Capaian Pembelajaran</b> | : | Basic Looping dan Decision                            |
| 2. <b>Indikator Kinerja</b>        | : | Mahasiswa dapat melakukan proses Looping dan Decision |
| 3. <b>Teori</b>                    | : | —   |

Looping merupakan metode penyelesaian masalah yang digunakan untuk mengenerate data yang banyak sesuai dengan jumlah looping. Looping sangat diperlukan untuk menampilkan data dari database dengan jumlah yang banyak. Looping juga banyak digunakan dalam membuat counter, nomor urut, kode unik, multiple upload, multiple insert, multiple update, multiple delete, memparsing data dari array dan sebagainya.

Decision merupakan proses pemecahan masalah dalam bahasa pemrograman yang memiliki **kondisi** dan **aksi**. Dengan bantuan decision, kita bisa membuat aksi berdasarkan kondisi tertentu. Dalam bahasa pemrograman javascript decision terbagi menjadi dua jenis yaitu **IF** dan **SWITCH**. Keduanya memiliki fungsi yang hampir sama tergantung dari kasus, situasi, dan dari penggunaan yang menurut anda paling mudah.

- 4 **Bahan dan Alat** : Komputer dan Logbook
- 5 **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
- 6 **Prosedur Kerja** :

## **A. DECISION**

Dalam bahasa pemrograman javascript decision terbagi menjadi dua jenis yaitu **IF** dan **SWITCH**. Keduanya memiliki fungsi yang hampir sama tergantung dari kasus, situasi, dan dari penggunaan yang menurut anda paling mudah. Disini saya akan menjabarkan contoh penggunaan dari decision **if** dan **switch**.

### **1. Decision IF**

Decision if cenderung membutuhkan operator pembandingan dalam mendefinisikan kondisi tertentu sehingga membuat decision menjadi sangat sesitif.

Decision IF sangat cocok digunakan untuk kondisi angka (INTEGER), dan kurang cocok untuk kondisi huruf (STRING).

Ok, langsung masuk ke contoh penggunaan. Silahkan buat sebuah file javascript dengan nama **decision\_if.js**. disini saya meletakkannya pada directory **documents/nodejs**. Bagi teman-teman yang mengikuti post saya sebelumnya tentang **pengenalan dan instalasi node.js** pasti sudah tau folder tersebut.

Buka file **decision\_if.js** dengan text editor. Disini penulis menggunakan sublime text sebagai text editor. Tetapi saya menyarankan untuk teman-teman yang ingin belajar node js, sebaiknya menggunakan **IntelliJ IDEA** sebagai text editor, dikarenakan IntelliJ IDEA memberikan kemudahan dalam me-running script yang dibangun menggunakan nodejs.

Ok, setelah dibuka lalu ketikan kode javascript berikut:

```
1   var kondisi="lapar";
2
3   if(kondisi == "lapar"){
4       console.log("Anda harus segera makan!");
5   }else{
6       console.log("Tidak melakukan apa-apa");
7   }
```

Silahkan save dan jalankan script diatas. Jika anda menggunakan text editor **IntelliJ IDEA** tinggal klik kanan pada text editor dan klik **Run decision\_if.js**. tetapi jika anda menggunakan text editor yang lain seperti sublime text, notepad++, atau yang lainnya. Silahkan buka command prompt atau terminal. Kemudian masuk ke directory project dengan perintah berikut:

```
1   cd documents/nodejs
```

Setelah masuk ke directory project ketikan perintah berikut untuk menjalankan file **decision\_if.js**.

```
1   node decision_if.js
```

Maka akan tampil hasilnya sebagai berikut:

```
1    Anda harus segera makan!
```

Jika kondisinya diubah menjadi seperti berikut:

```
1    var kondisi="haus";
2
3    if(kondisi == "lapar"){
4        console.log("Anda harus segera makan!");
5    }else{
6        console.log("Tidak melakukan apa-apa");
7    }
```

Jika dirunning kembali maka akan tampil hasilnya sebagai berikut:

```
1    Tidak melakukan apa-apa
```

Hal ini disebabkan oleh kondisi yang di set tidak sama dengan variable **kondisi**. Oleh sebab itu decision if akan menampilkan kondisi **else**.

## 2. Decision SWITCH

Hampir sama dengan IF. Decision switch memiliki penulisan kode program yang sedikit berbeda. Ok, langsung saja buat file javascript dengan nama **decision\_switch.js** pada folder yang sama dengan sebelumnya. Buka dengan text editor lalu ketikan script berikut:

```
1    var kondisi="lapar";
2
3    switch (kondisi){
4        case "lapar":
5            console.log("Anda harus makan!");
6            break;
7        case "haus":
8            console.log("Anda harus minum!");
9            break;
10       default:
11           console.log("Anda tidak haus dan tidak lapar");
12
13    }
```

Silahkan running script diatas, maka hasilnya akan terlihat seperti berikut:

```
1    Anda harus segera makan!
```

Silahkan ganti nilai pada variabel kondisi untuk melihat perbedaannya.

## **B. LOOPING**

Looping sendiri terbagi menjadi dua yaitu looping **for** dan looping **while**. Disini penulis menggunakan IntelliJ IDEA sebagai IDEA atau text editor. Seperti yang penulis sarankan pada tutorial sebelumnya. IntelliJ IDEA sangat membantu mempermudah kita dalam me-running script yang kita buat dengan node.js. Bagi Anda yang tidak menggunakan IntelliJ IDEA, tidak masalah asalkan tau bagaimana me-running scriptnya.



Disini penulis telah membuat sebuah folder di **Mydocuments** dengan nama **nodejs**. Bagi anda yang mengikuti tutorial sebelumnya, pasti sudah tau folder tersebut. Selanjutnya saya membuat sebuah folder lagi dengan nama **looping** yang terdapat dalam folder **nodejs**. Sehingga struktur folder kita menjadi **mydocuments/nodejs/looping**.

## 1. Looping FOR

Untuk mengetahui cara kerja dari looping for, ada baiknya kita langsung saja pada contoh. Pertama-tama buat sebuah javascript file dengan nama **looping\_for.js** dan pastikan file tersebut diletakan pada folder yang telah dibuat sebelumnya. Sehingga strukturny menjadi seperti ini.

**mydocuments/nodejs/looping/looping\_for.js**

Sangat penting untuk mengetahui struktur folder ini, dikeranakan akan membantu mempermudah anda dalam me-running script yang anda buat. terutama jika anda tidak menggunakan intelliJ IDEA.

Selanjutnya open file **looping\_for.js** dengan texteditor jika anda tidak menggunakan IntelliJ IDEA, dan ketika script berikut:

```
1      var n=10;
2
3      for(var i=1; i<=n; i++){
4          console.log("Looping ke-"+i);
5      }
```

Silahkan running script diatas. Jika anda menggunakan IntelliJ IDEA, tinggal klik kanan pada editor dan klik Run **looping\_for.js**. Maka akan terlihat hasilnya seperti berikut:

```
1      Looping ke-1
2      Looping ke-2
3      Looping ke-3
4      Looping ke-4
5      Looping ke-5
6      Looping ke-6
7      Looping ke-7
8      Looping ke-8
9      Looping ke-9
10     Looping ke-10
```

## 2. Looping WHILE

Untuk mengetahui bagaimana struktur dan cara kerja dari looping while, buat javascript file baru dengan nama **looping\_while.js** dan pastikan file **looping\_while.js** satu folder dengan file **looping\_for.js**.

Buka file **looping\_while.js** dengan text editor bagi anda yang tidak menggunakan IntelliJ IDEA, dan ketika script berikut:

```

1      var i=0;
2      var n=10;
3
4      while(i < n){
5          console.log("Looping ke- "+i);
6          i++;
7      }

```



Silahkan running script diatas. Jika anda menggunakan IntelliJ IDEA, tinggal klik kanan pada editor dan klik Run looping\_while.js. Maka akan terlihat hasilnya seperti berikut:

```

1      Looping ke- 0
2      Looping ke- 1
3      Looping ke- 2
4      Looping ke- 3
5      Looping ke- 4
6      Looping ke- 5
7      Looping ke- 6
8      Looping ke- 7
9      Looping ke- 8
10     Looping ke- 9

```

- 7 Tugas dan Pertanyaan : -**
- 8 Pustaka :**
1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011. [www.koder.web.id](http://www.koder.web.id)
  2. Tutorislpoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)
- 9 Hasil Praktikum : Laporan Praktikum**

	<b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 11 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

<b>Minggu Ke</b>	:	4
<b>Capaian Pembelajaran</b>	:	Javascript Framework
<b>Waktu</b>	:	2 x 170 menit
<b>Tempat</b>	:	Laboratorium

---

- |                                    |   |   |
|------------------------------------|---|---|
| 1. <b>Sub Capaian Pembelajaran</b> | : | Pemanggilan Modul dan Referensi Objek                                 |
| 2. <b>Indikator Kinerja</b>        | : | Mahasiswa dapat melakukan pemanggilan Modul dan referensi Objek       |
| 3. <b>Teori</b>                    | : | —   |
| 4 <b>Bahan dan Alat</b>            | : | Komputer dan Logbook  |
| 5 <b>Organisasi</b>                | : | Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah |
| 6 <b>Prosedur Kerja</b>            | : |   |

### A. PEMANGGILAN MODUL

Module merupakan suatu file, library, atau helper yang dibutuhkan untuk menjalankan suatu fungsi tertentu. Dalam pembuatan suatu aplikasi, kita tidak bisa terlepas dari yang

namanya module, dikarenakan sangat sulit mengembangkan suatu aplikasi hanya dengan menggunakan suatu file.

Untuk memanggil module pada PHP biasanya menggunakan fungsi **include**, **require**, ataupun **require\_once**. Sedangkan pada node.js hanya menggunakan fungsi **require**.

Module terbagi menjadi dua macam yaitu Module bawaan node.js dan module buatan sendiri. Untuk memanggil module bawaan node.js dapat dilakukan dengan cara **require('nama\_module')**. Sedangkan untuk memanggil module buatan sendiri dapat dilakukan dengan cara **require('./nama\_module')**. Jika suatu module diletakkan dalam subfolder, maka dapat di panggil dengan cara **require('./nama\_folder/nama\_module')**.

Pada contoh kali ini saya membuat module sendiri dan memanggilnya pada aplikasi utama.

Untuk memahaminya silahkan ikuti tutorial berikut:

Pertama-tama saya telah membuat sebuah folder dengan nama **nodejs** di documents. Dimana folder ini adalah folder yang akan saya gunakan untuk tutorial kali ini.

#1. Buat sebuah javascript file dengan nama **module.js** di dalam folder **nodejs**. Kemudian ketikan script berikut:

```
1
2   var methods = {};
3
4   methods.function_satu=function(){
5       console.log('Ini adalah function satu');
6   }
7   methods.function_dua=function(){
8       console.log('Ini adalah function dua');
9   }
10  exports.data=methods;
```

Pada script diatas saya membuat sebuah variabel dengan nama methods, dimana variabel methods ini berfungsi untuk menampung object. Dimana objectnya berisi function yaitu **function\_satu** dan **function\_dua**. Agar module dapat digunakan diluar kelas (Public), maka module harus di exports. Pada script diatas saya meng-exports-nya dengan perintah **exports.data=methods**. Dimana **methods** adalah variable dan **data** merupakan nama pemanggilan untuk semua fungsi yang ada pada variable methods dari luar kelas.

Agar lebih mudah dipahami silahkan lanjut ke step dua.

#2. Buat sebuah javascript file lagi dengan nama **app.js** didalam folder yang sama. Dimana file app.js ini akan kita gunakan sebagai file induk dari tutorial kali ini. Kemudian ketikan script berikut pada app.js.

```
1  var respon=require('./module.js');
2
3  respon.data.function_satu();
4  console.log('-----');
5  respon.data.function_dua();
```

Pada script diatas, kita memanggil module.js dengan fungsi require dan menyimpannya dalam variable respon. Dengan begitu, function\_satu dan function\_dua dapat dipanggil melalui app.js.

#3. Jalankan app.js dengan perintah berikut:

```
1  node app.js
```

jika tidak ada error maka akan tampil hasilnya sebagai berikut:

```
1  Ini adalah function satu
2  -----
3  Ini adalah function dua
```

## B. PEMANGGILAN REFERENSI OBJEK

Referensi Object atau Object References merupakan metode yang digunakan oleh node.js untuk handle perubahan pada atribut suatu object. Jika terdapat nilai (*value*) pada atribut suatu object, maka semua nilai object yang terkait juga akan mengalami perubahan. Sederhananya, seperti *primary key* dan *foreign key* pada relational database. Dimana jika terdapat perubahan data pada suatu tabel utama (master), maka akan mengalami perubahan juga pada tabel tamu (transaksi).

#1. Buat sebuah file javascript dengan nama **script.js**, kemudian ketikan kode berikut:

```
1  var object1={
2      "nama" : "Tri Saandhika Jaya",
3      "umur" : 34,
4  }
5
6  console.log(object1);
```

Pada script diatas, penulis membuat sebuah object dalam format JSON, dan disimpan dalam sebuah variabel dengan nama **object1**, dan terdiri dari dua atribut yaitu **nama** dan **umur**. Dimana, atribut nama berisi “Tri Sandhika Jaya” dan atribut umur berisi “34”.

Jika script diatas running, maka akan terlihat hasilnya sebagai berikut:

```
1  { nama: 'Tri Sandhika Jaya', umur: 34 }
```

#2. Kemudian lakukan perubahan pada script diatas menjadi seperti berikut:

```
1
2   var object1={
3       "nama" : "Tri Sandhika Jaya",
4       "umur" : 34,
5   }
6   var object2  = object1;
7   object2.umur = 22;
8   console.log(object1); //hasil object 1
9   console.log("-----");
10  console.log(object2); //hasil object 2
```

Pada script diatas, penulis melakukan perubahan pada atribut umur menjadi 22, dan disimpan pada variable **object2**. Kemudian hasilnya penulis tampilkan dengan perintah:

```
1   console.log(object1); //hasil object 1
2   console.log("-----");
3   console.log(object2); //hasil object 2
```

hasil yang ditampilkan adalah variable **object1** dan **object2**. Dengan garis putus-putus sebagai pembatas. Jika script diatas di running, maka akan terlihat hasilnya sebagai berikut:



```
1   { nama: 'Tri Sandhika Jaya', umur: 34 }
2   -----
3   { nama: 'Tri Sandhika Jaya', umur: 34 }
```

Coba perhatikan hasil diatas, penulis hanya melakukan perubahan pada object2, tetapi value pada object1 juga ikut berubah. Seharusnya, hasilnya seperti berikut:

```
1   { nama: 'Tri Sandhika Jaya', umur: 34 }
2   -----
3   { nama: 'Tri Sandhika Jaya', umur: 22 }
```

Itulah cara kerja object references pada node.js. Node.js akan mengambil nilai terbaru dari suatu object sekaligus merubah nilai sebelumnya.

- 7 Tugas dan Pertanyaan :**
- 8 Pustaka :**
1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
  2. Tutorislpoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)
- 9 Hasil Praktikum : Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 11 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

<b>Minggu Ke</b>	:	5
<b>Capaian Pembelajaran</b>	:	Javascript Framework
<b>Waktu</b>	:	2 x 170 menit
<b>Tempat</b>	:	Laboratorium

---

- |                             |   |   |
|-----------------------------|---|---|
| 1. Sub Capaian Pembelajaran | : | CRUD dengan NodeJS, Bootstrap, MySQL                                  |
| 2. Indikator Kinerja        | : | Mahasiswa dapat melakukan CRUD dengan NodeJS, Bootstrap, MySQL        |
| 3. Teori                    | : | —   |
| 4. Bahan dan Alat           | : | Komputer dan Logbook  |
| 5. Organisasi               | : | Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah |
| 6. Prosedur Kerja           | : |   |

### Step #1. Pendahuluan

Jika Anda belum memiliki bootstrap dan jquery, silahkan download terlebih dahulu di official websitenya [getbootstrap.com](http://getbootstrap.com) dan [jquery.com](http://jquery.com).



## Step #2. Buat Database dan Table

Buat sebuah database baru, disini saya membuat database dengan nama **crud\_db**.

Jika Anda membuat dengan nama yang sama itu lebih baik.

Untuk membuat database dengan MySQL, dapat dilakukan dengan mengeksekusi query berikut:

```
1 CREATE DATABASE crud_db;
```

Perintah SQL diatas akan membuat sebuah database dengan nama **crud\_db**.

Selanjutnya, buat sebuah table di dalam database **crud\_db**.

Disini saya membuat sebuah table dengan nama **product**.

Jika Anda membuat dengan nama yang sama itu lebih baik.

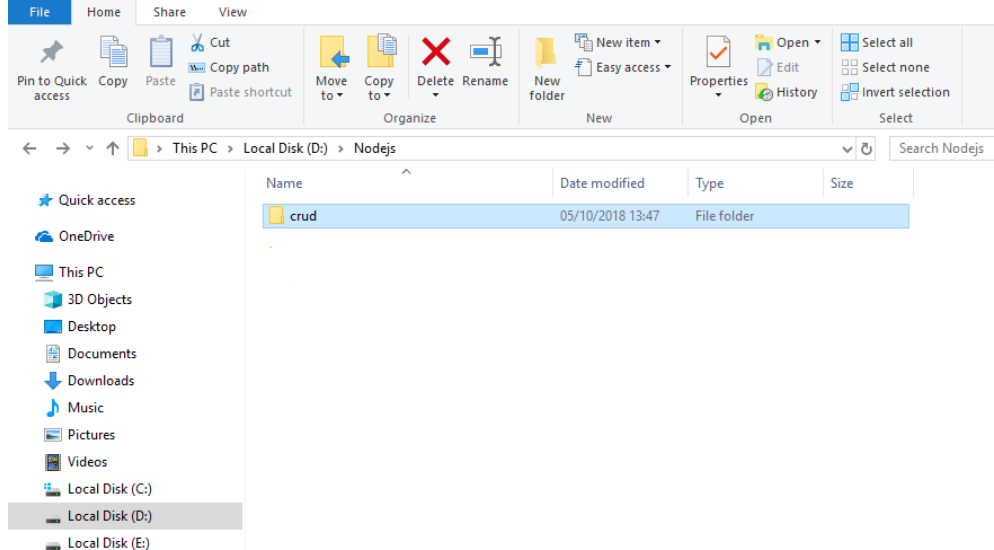
Untuk membuat table **product**, dapat dilakukan dengan mengeksekusi perintah SQL berikut:

```
1 CREATE TABLE product (  
2   product_id INT(11) PRIMARY KEY AUTO_INCREMENT,  
3   product_name VARCHAR(200),  
4   product_price INT(11)  
5 ) ENGINE=INNODB;
```

Perintah SQL diatas akan membuat sebuah table dengan **product**, dengan field **product\_id**, **product\_name**, dan **product\_price**.

## Step #3. Install Dependencies

Sebelum menginstall dependencies, silahkan buat sebuah folder, disini saya membuat sebuah folder dengan nama **crud**.



Dimana folder **crud** merupakan folder project kita pada contoh kali ini.

Mari kita lanjut, kita membutuhkan 4 dependencies yaitu:

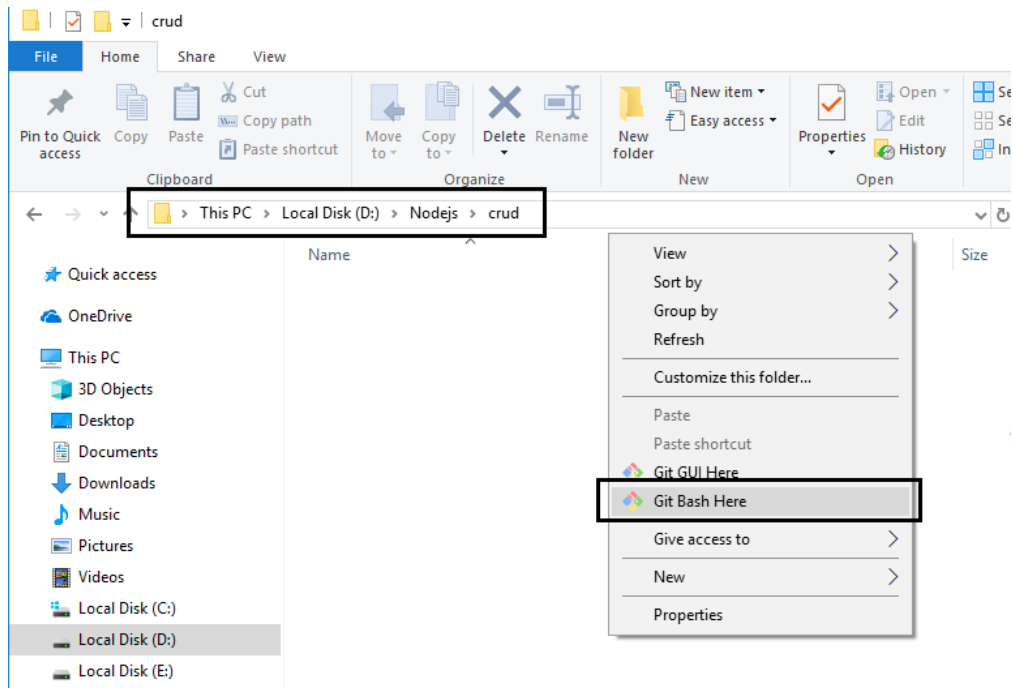
1. **Express** (node.js framework)
2. **MySQL** (driver mysql untuk node.js)
3. **Body-parser** (middleware untuk handle post body request)
4. **Handlebars** (template engine)

Untuk menginstall dependencies pada node.js dapat dilakukan dengan mudah menggunakan NPM (Node Package Manager). Anda dapat menjalankan NPM pada **Terminal** atau **Command Prompt**. Akan tetapi, pada contoh kali ini saya tidak menggunakan Command Prompt, melainkan menggunakan **Git Bash** Terminal.

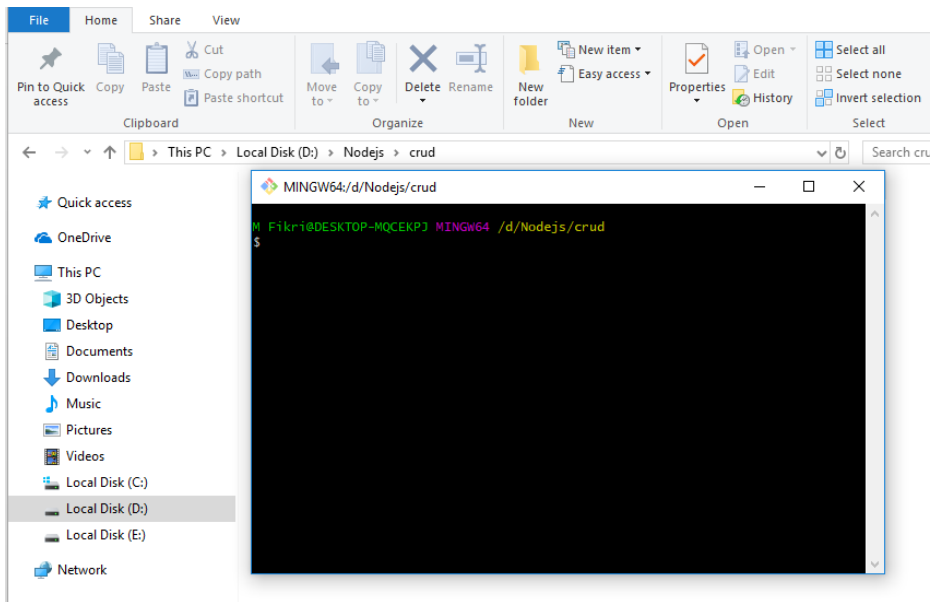
Saya sangat merekomendasikan Anda juga menggunakan Git Bash. Anda dapat mendownload Git Bash pada url berikut:

<https://git-scm.com/downloads>

Silahkan download sesuai dengan platform Anda, kemudian install di komputer Anda. Jika Anda telah menginstall Git Bash, silahkan buka folder **crud** yang telah dibuat sebelumnya. Kemudian klik-kanan dan pilih **Git Bash Here**, seperti gambar berikut:



Maka akan muncul terminal seperti gambar berikut:



Pada terminal diatas, anda dapat menginstall semua dependencies yang Anda perlukan untuk project Anda. Sebelum menginstall dependencies, kita perlu membuat **package.json**. Untuk membuat package.json, anda dapat menjalankan perintah berikut pada terminal.

```
1 npm init
```

Seperti gambar berikut:



Perintah diatas akan membuat sebuah file bernama **package.json** secara otomatis pada project Anda.

Selanjutnya, Install semua dependencies yang dibutuhkan dengan mengetikkan perintah berikut pada terminal:

```
1 npm install --save express mysql body-parser hbs
```

Seperti gambar berikut:



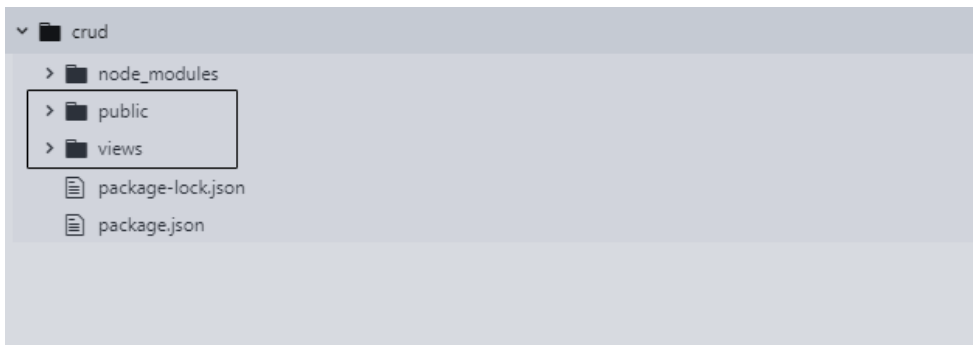
Perintah diatas akan menginstall semua dependencies yang kita butuhkan yaitu: express, mysql, body-parser, dan handlebars.

Jika di buka file **package.json**, maka akan terlihat seperti berikut:

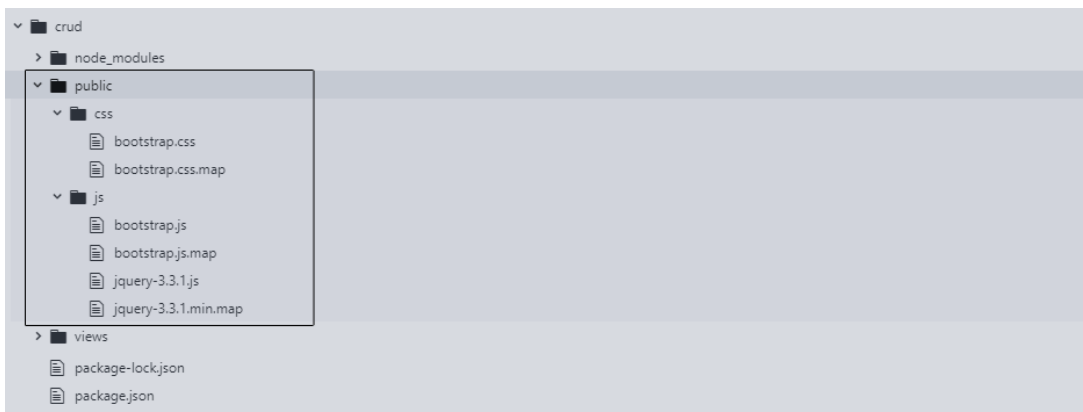
```
1  {
2    "name": "crud",
3    "version": "1.0.0",
4    "description": "Crud Node.js",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "TSJ",
10   "license": "ISC",
11   "dependencies": {
12     "body-parser": "^1.18.3",
13     "express": "^4.16.3",
14     "hbs": "^4.0.1",
15     "mysql": "^2.16.0"
16   }
17 }
```

#### Step #4. Struktur Project

Buka folder **crud** menggunakan text editor, disini saya menggunakan ATOM sebagai text editor. Anda dapat menggunakan Sublime Text, IntelliJ IDEA, ataupun editor lainnya. Kemudian buat folder baru didalam folder **crud**. Disni saya membuat dua folder yaitu folder **public** dan **views**. Seperti gambar berikut:

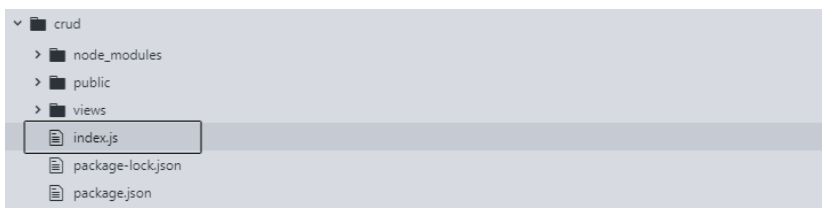


Selanjutnya buat folder **css** dan folder **js** didalam folder **public**. Kemudian copy file **bootstrap.css** yang telah di download sebelumnya kedalam folder **public/css/**. Dan copy file **bootstrap.js** dan **jquery** pada folder **public/js/**. Sehingga terlihat struktur project kita seperti gambar berikut:



### Step #5. Index.js

Buat sebuah file dengan nama **index.js**. Seperti gambar berikut:



Kemudian open **index.js** dan ketikan kode berikut:

```
1 //use path module
2 const path = require('path');
3 //use express module
4 const express = require('express');
5 //use hbs view engine
6 const hbs = require('hbs');
7 //use bodyParser middleware
8 const bodyParser = require('body-parser');
```

```

9    //use mysql database
10   const mysql = require('mysql');
11   const app = express();
12
13   //konfigurasi koneksi
14   const conn = mysql.createConnection({
15     host: 'localhost',
16     user: 'root',
17     password: '',
18     database: 'crud_db'
19   });
20
21   //connect ke database
22   conn.connect((err) =>{
23     if(err) throw err;
24     console.log('Mysql Connected...');
25   });
26
27   //set views file
28   app.set('views',path.join(__dirname,'views'));
29   //set view engine
30   app.set('view engine', 'hbs');
31   app.use(bodyParser.json());
32   app.use(bodyParser.urlencoded({ extended: false }));
33   //set folder public sebagai static folder untuk static file
34   app.use('/assets',express.static(__dirname + '/public'));
35
36   //route untuk homepage
37   app.get('/',(req, res) => {
38     let sql = "SELECT * FROM product";
39     let query = conn.query(sql, (err, results) => {
40       if(err) throw err;
41       res.render('product_view',{
42         results: results
43       });
44     });
45   });
46
47   //route untuk insert data
48   app.post('/save',(req, res) => {
49     let data = {product_name: req.body.product_name, product_price: req.body.product_price};
50     let sql = "INSERT INTO product SET ?";
51     let query = conn.query(sql, data,(err, results) => {
52       if(err) throw err;
53       res.redirect('/');
54     });
55   });
56
57   //route untuk update data
58   app.post('/update',(req, res) => {
59     let sql = "UPDATE product SET product_name='"+req.body.product_name+"',
60       product_price='"+req.body.product_price+"' WHERE product_id='"+req.body.id;
61     let query = conn.query(sql, (err, results) => {
62       if(err) throw err;
63       res.redirect('/');
64     });
65   });
66
67   //route untuk delete data
68   app.post('/delete',(req, res) => {
69     let sql = "DELETE FROM product WHERE product_id='"+req.body.product_id+"'";

```

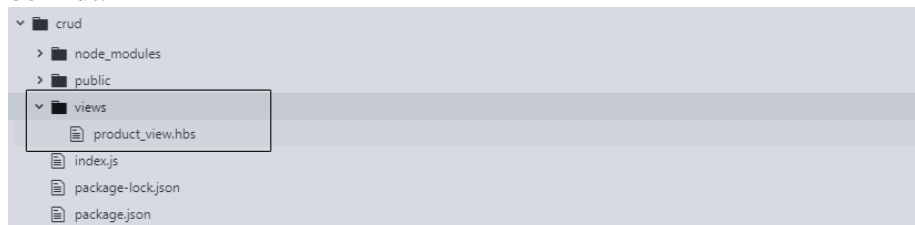
```

70     let query = conn.query(sql, (err, results) => {
71         if(err) throw err;
72         res.redirect('/');
73     });
74 });
75
76 //server listening
77 app.listen(8000, () => {
78     console.log('Server is running at port 8000');
79 });

```

## Step #6. View

Buat sebuah file dengan nama **product\_view.hbs** didalam folder **views**. Seperti gambar berikut:



Kemudian open **product\_view.hbs** dan ketikkan kode berikut:

```

1  <html lang="en">
2  <head>
3      <meta charset="utf-8">
4      <title>CRUD Node.js and Mysql</title>
5      <link href="/assets/css/bootstrap.css" rel="stylesheet" type="text/css"/>
6  </head>
7  <body>
8      <div class="container">
9          <h2>Product List</h2>
10         <button class="btn btn-success" data-toggle="modal" data-target="#myModalAdd">Add New</button>
11         <table class="table table-striped" id="mytable">
12             <thead>
13                 <tr>
14                     <th>Product ID</th>
15                     <th>Product Name</th>
16                     <th>Price</th>
17                     <th>Action</th>
18                 </tr>
19             </thead>
20             <tbody>
21                 {{#each results}}
22                 <tr>
23                     <td>{{ product_id }}</td>
24                     <td>{{ product_name }}</td>
25                     <td>{{ product_price }}</td>
26                     <td>
27                         <a href="javascript:void(0);" class="btn btn-sm btn-info edit" data-id="{{ product_id }}"
28                             data-product_name="{{ product_name }}" data-product_price="{{ product_price }}">Edit</a>
29                         <a href="javascript:void(0);" class="btn btn-sm btn-danger delete"
30                             data-id="{{ product_id }}">Delete</a>
31                     </td>
32                 </tr>
33                 {{/each}}
34             </tbody>
35         </table>
36     </div>
37

```

```

38 <!-- Modal Add Produk-->
39 <form action="/save" method="post">
40   <div class="modal fade" id="myModalAdd" tabindex="-1" role="dialog"
41     aria-labelledby="exampleModalLabel" aria-hidden="true">
42     <div class="modal-dialog" role="document">
43       <div class="modal-content">
44         <div class="modal-header">
45           <h5 class="modal-title" id="exampleModalLabel">Add New Product</h5>
46           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
47             <span aria-hidden="true">&times;</span>
48           </button>
49         </div>
50         <div class="modal-body">
51           <div class="form-group">
52             <input type="text" name="product_name" class="form-control"
53               placeholder="Product Name" required>
54           </div>
55
56           <div class="form-group">
57             <input type="text" name="product_price" class="form-control"
58               placeholder="Price" required>
59           </div>
60         </div>
61         <div class="modal-footer">
62           <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
63           <button type="submit" class="btn btn-primary">Save</button>
64         </div>
65       </div>
66     </div>
67   </div>
68 </form>
69
70 <!-- Modal Update Produk-->
71 <form action="/update" method="post">
72   <div class="modal fade" id="EditModal" tabindex="-1" role="dialog"
73     aria-labelledby="exampleModalLabel" aria-hidden="true">
74     <div class="modal-dialog" role="document">
75       <div class="modal-content">
76         <div class="modal-header">
77           <h5 class="modal-title" id="exampleModalLabel">Edit Product</h5>
78           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
79             <span aria-hidden="true">&times;</span>
80           </button>
81         </div>
82         <div class="modal-body">
83           <div class="form-group">
84             <input type="text" name="product_name" class="form-control product_name"
85               placeholder="Product Name" required>
86           </div>
87
88           <div class="form-group">
89             <input type="text" name="product_price" class="form-control price"
90               placeholder="Price" required>
91           </div>
92         </div>
93         <div class="modal-footer">
94           <input type="hidden" name="id" class="product_id">
95           <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
96           <button type="submit" class="btn btn-primary">Update</button>
97         </div>
98       </div>
99     </div>
100   </div>
101 </form>
102
103 <!-- Modal Delete Produk-->
104 <form id="add-row-form" action="/delete" method="post">
105   <div class="modal fade" id="DeleteModal" tabindex="-1" role="dialog"
106     aria-labelledby="myModalLabel" aria-hidden="true">

```



```

107         <div class="modal-dialog">
108             <div class="modal-content">
109                 <div class="modal-header">
110                     <h5 class="modal-title" id="myModalLabel">Delete Product</h5>
111                     <button type="button" class="close" data-dismiss="modal" aria-label="Close">
112                         <span aria-hidden="true">&times;</span></button>
113                 </div>
114                 <div class="modal-body">
115                     <strong>Anda yakin mau menghapus data ini?</strong>
116                 </div>
117                 <div class="modal-footer">
118                     <input type="hidden" name="product_id"
119                         class="form-control product_id2" required>
120                     <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
121                     <button type="submit" class="btn btn-success">Delete</button>
122                 </div>
123             </div>
124         </div>
125     </div>
126 </form>
127
128<script src="/assets/js/jquery-3.3.1.js"></script>
129<script src="/assets/js/bootstrap.js"></script>
130<script>
131     $(document).ready(function() {
132         //tampilkan data ke modal untuk edit
133         $('#mytable').on('click', '.edit', function() {
134             var product_id = $(this).data('id');
135             var product_name = $(this).data('product_name');
136             var product_price = $(this).data('product_price');
137             $('#EditModal').modal('show');
138             $('.product_name').val(product_name);
139             $('.price').val(product_price);
140             $('.product_id').val(product_id);
141         });
142
143         //tampilkan modal hapus record
144         $('#mytable').on('click', '.delete', function() {
145             var product_id = $(this).data('id');
146             $('#DeleteModal').modal('show');
147             $('.product_id2').val(product_id);
148         });
149     });
150 </script>
151 </body>
152 </html>

```

## Step #7. Testing

Uji coba aplikasi untuk memastikan aplikasi crud yang kita buat berjalan dengan baik. Untuk menjalankan aplikasi, ketikkan perintah berikut pada terminal.

```
1 node index
```

Maka akan tampil di console pesan “Server is running at port 8000” dan “Mysql Connected”.

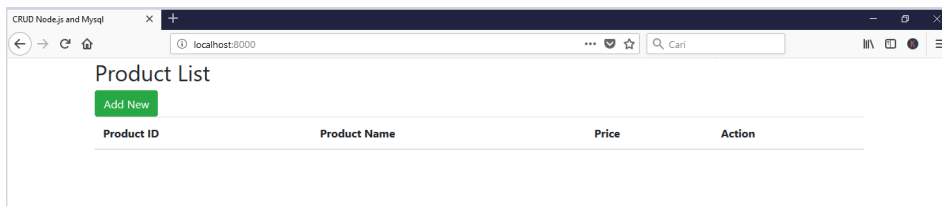
Seperti gambar berikut:

```
MINGW64/d/Nodejs/crud
M Fikri@DESKTOP-MQCEKPJ MINGW64 /d/Nodejs/crud
$ node index
Server is running at port 8000
Mysql Connected...
```

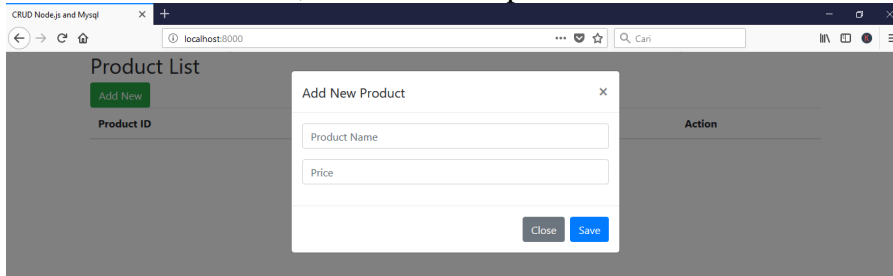
Kemudian buka browser Anda dan ketikkan URL berikut:

<http://localhost:8000/>

Jika berjalan dengan baik, maka akan terlihat seperti gambar berikut:

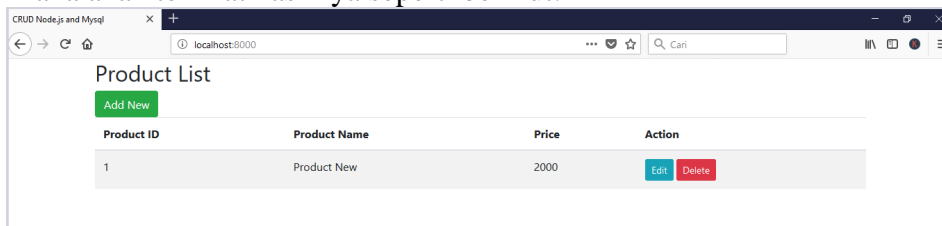


Klik tombol **Add New**, maka akan tampil modal Add New Product seperti berikut:

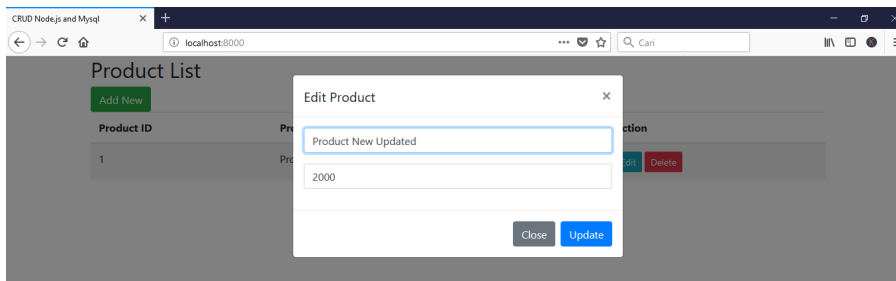


Input Product Name dan Price kemudian Klik tombol Save.

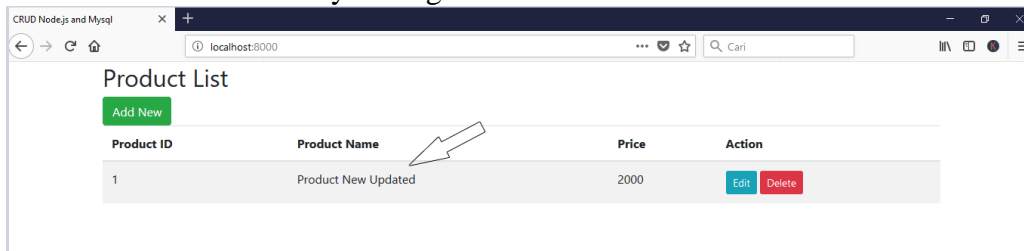
Maka akan terlihat hasilnya seperti berikut:



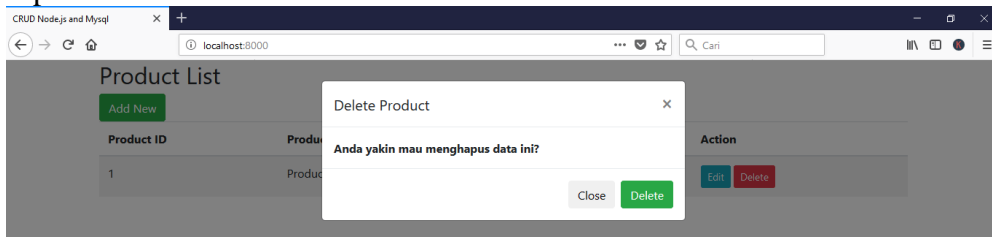
Klik tombol **Edit** untuk mengedit record, maka akan tampil modal Edit Product seperti berikut:



Edit data yang ingin di edit, kemudian klik tombol **Update**.  
Maka akan terlihat hasilnya sebagai berikut:



Untuk menghapus Record, klik tombol **Delete**, maka akan muncul modal konfirmasi seperti berikut:





Kemudian klik tombol **Delete** untuk menghapus record.  
**Selesai.**

**7 Tugas dan Pertanyaan :**

**8 Pustaka :**

1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
2. Tutorislpoin. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)

**9 Hasil Praktikum : Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 27 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

<b>Minggu Ke</b>	<b>:</b>	6
<b>Capaian Pembelajaran</b>	<b>:</b>	Javascript Framework
<b>Waktu</b>	<b>:</b>	2 x 170 menit
<b>Tempat</b>	<b>:</b>	Laboratorium

---

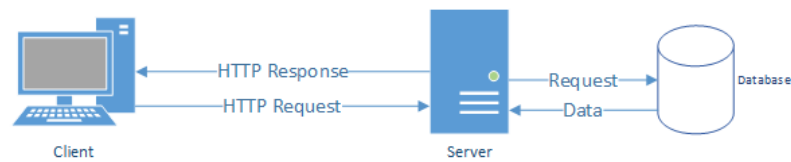
- |                                    |          |                                     |
|------------------------------------|----------|-------------------------------------|
| <b>1. Sub Capaian Pembelajaran</b> | <b>:</b> | Membuat Restful API                 |
| <b>2. Indikator Kinerja</b>        | <b>:</b> | Mahasiswa dapat Membuat Restful API |
| <b>3. Teori</b>                    | <b>:</b> |                                     |

### **Api itu RESTful API?**

RESTful API merupakan implementasi dari API (*Application Programming Interface*). REST (REpresentational State Transfer) adalah suatu arsitektur metode komunikasi yang menggunakan protocol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi. Mungkin terdengar rumit tapi sebenarnya tidak. Tujuannya untuk menjadikan system dengan performa yang baik, cepat, dan mudah untuk dikembangkan (*scale*) terutama dalam pertukaran dan komunikasi data.

### **Kenapa perlu membuat RESTful API?**

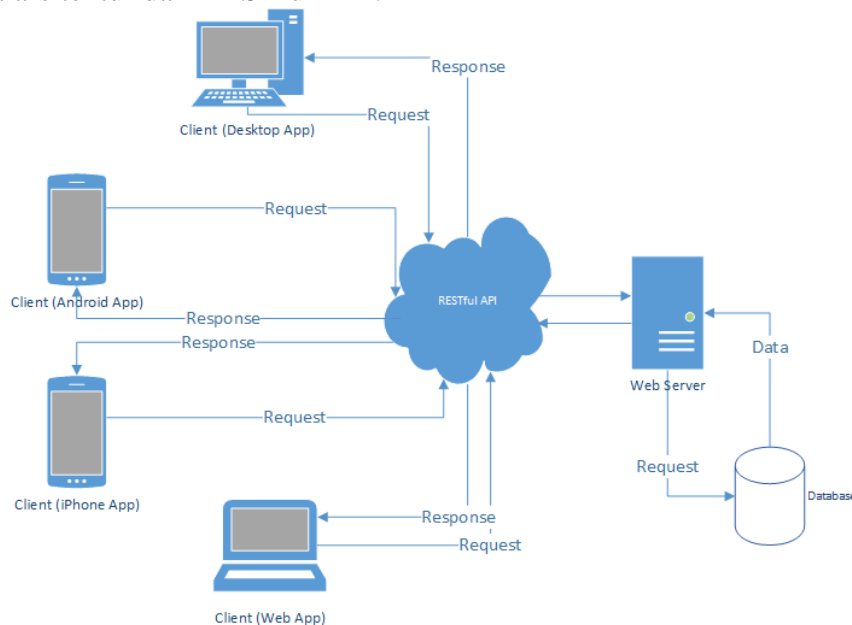
Jika Anda perhatikan arsitektur tradisional komunikasi data antara client dan server terlihat seperti gambar berikut:



Dimana client mengirimkan request ke server melalui HTTP Request, dan server memberikan response melalui HTTP Response. Response yang diberikan oleh server, biasanya berformat HTML. Nah, bayangkan jika Anda harus mengembangkan website Anda ke aplikasi Mobile seperti Android atau iOS. Anda tahu aplikasi Android ataupun iOS tidak membutuhkan HTML sebagai response dari server. Karena Android dan iOS menggunakan bahasa pemrograman yang berbeda dan tidak mengenal HTML. Oleh sebab itu, kita perlu membuat RESTful API.

RESTful API ini akan menjadi jembatan komunikasi data antara client dan server. Sehingga, server tidak lagi mengirimkan HTML sebagai response, melainkan hanya data. Ya, hanya data. Hal inilah yang dapat menghemat bandwidth server. Response dalam bentuk data inilah yang dapat digunakan untuk berbagai macam platform dari aplikasi yang berbeda bahasa pemrograman. Response dalam bentuk data ini, biasanya berformat JSON atau XML. Akan tetapi, yang paling umum digunakan adalah JSON.

Berikut arsitektur dari RESTful API:



Dimana client dapat dari berbagai macam platform, seperti aplikasi web, desktop, ataupun mobile app.

- 4 **Bahan dan Alat** : Komputer dan Logbook
- 5 **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
- 6 **Prosedur Kerja** :

## Step #1. Design RESTful API

Sebelum membuat RESTful API, ada baiknya di kita definisikan dulu EndPoint dari RESTful API yang akan dibuat. EndPoint merupakan routes dari API yang akan kita buat. RESTful API menggunakan HTTP verbs. HTTP verbs yang umum digunakan adalah GET, POST, PUT, dan DELETE. GET untuk mendapatkan data dari server atau lebih dikenal dengan istilah READ, POST untuk meng-CREATE new data, PUT untuk UPDATE data, dan DELETE untuk menghapus data. Atau lebih dikenal dengan istilah **CRUD** (Create Read Update Delete).

Pada materi ini, saya akan sharing bagaimana membuat RESTful API sederhana untuk mengambil data dari server (GET), membuat data baru ke server (POST), mengupdate data ke server (PUT), dan menghapus data ke server (DELETE) dari suatu table di database yaitu table **products**.

Berikut rancangan dari RESTful API yang akan kita buat.

Method	EndPoint	Description
GET	api/products	List of products
GET	api/products/{id}	View a product
POST	api/products	Create new product
PUT	api/products/{id}	Update a product
DELETE	api/products/{id}	Delete a product

## Step #2. Buat Database dan Table

Buat sebuah database baru dengan MySQL, anda dapat menggunakan tools seperti SQLyog, PHPMyAdmin atau sejenisnya. Disini saya membuat database dengan nama **restful\_db**. Jika Anda membuat database dengan nama yang sama itu lebih baik. Untuk membuat database dengan MySQL, dapat dilakukan dengan mengeksekusi query berikut:

```
1 CREATE DATABASE restful_db;
```

Perintah SQL diatas akan membuat sebuah database dengan nama **restful\_db**. Selanjutnya, buat sebuah table di dalam database **restful\_db**. Disini saya membuat sebuah table dengan nama **product**. Jika Anda membuat table dengan nama yang sama itu lebih baik. Untuk membuat table **product**, dapat dilakukan dengan mengeksekusi perintah SQL berikut:

```

1 CREATE TABLE product(
2   product_id INT(11) PRIMARY KEY AUTO_INCREMENT,
3   product_name VARCHAR(200),
4   product_price INT(11)
5 ) ENGINE=INNODB;

```

Selanjutnya, insert beberapa data kedalam table **product** dengan mengeksekusi query berikut:

```

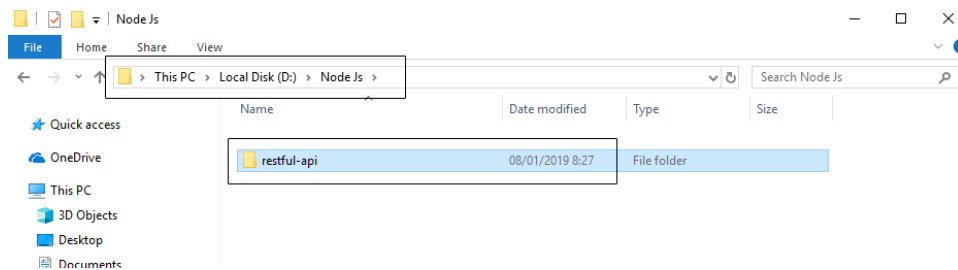
1 INSERT INTO product(product_name,product_price) VALUES
2 ('Product 1','2000'),
3 ('Product 2','5000'),
4 ('Product 3','4000'),
5 ('Product 4','6000'),
6 ('Product 5','7000');

```

Perintah SQL diatas akan menginput 5 data kedalam table **product**.

### Step #3. Install Dependencies

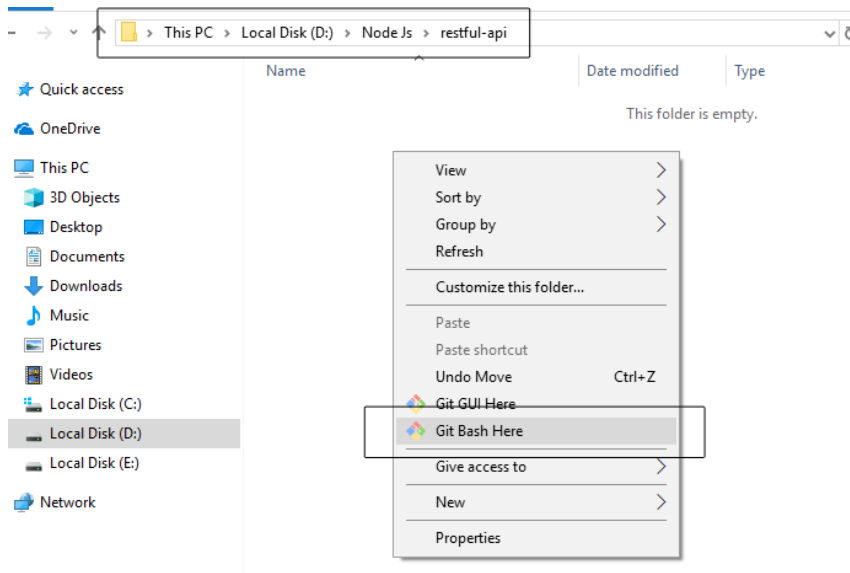
Sebelum menginstall dependencies, silahkan buat sebuah folder, disini saya membuat sebuah folder dengan nama **restful-api**. Perhatikan gambar berikut untuk lebih jelasnya:



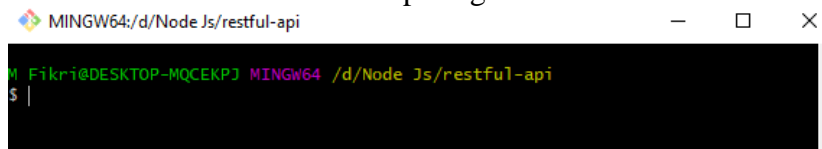
Dimana folder **restful-api** ini akan menjadi folder project kita pada tutorial kali ini. Mari kita lanjut, Kali ini, kita membutuhkan 3 dependencies yaitu:

1. **Express** (node.js framework)
2. **MySQL** (driver mysql untuk node.js)
3. **Body-parser** (middleware untuk handle post body request)

Untuk menginstall dependencies pada node.js dapat dilakukan dengan mudah menggunakan NPM (Node Package Manager). Anda dapat menjalankan NPM pada **Terminal** atau **Command Prompt**. Akan tetapi, pada tutorial kali ini saya tidak menggunakan Command Prompt, melainkan menggunakan **Git Bash** Terminal. Saya sangat merekomendasikan Anda juga menggunakan Git Bash. Anda dapat mendownload Git Bash pada url berikut: <https://git-scm.com/downloads> . Silahkan download sesuai dengan platform Anda, kemudian install di komputer Anda. Jika Anda telah menginstall Git Bash, silahkan buka folder **restful-api** yang telah dibuat sebelumnya. Kemudian klik-kanan dan pilih **Git Bash Here**, seperti gambar berikut:



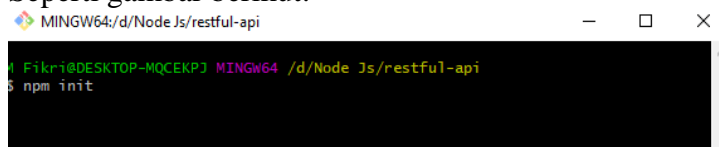
Maka akan muncul terminal seperti gambar berikut:



Pada terminal diatas, anda dapat menginstall semua dependencies yang Anda perlukan untuk project Anda. Sebelum menginstall dependencies, kita perlu membuat **package.json**. Untuk membuat package.json, anda dapat menjalankan perintah berikut pada terminal.

```
1 npm init
```

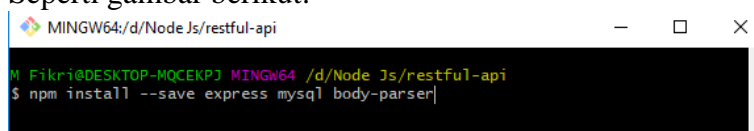
Seperti gambar berikut:



Perintah diatas akan membuat sebuah file bernama **package.json** secara otomatis pada project Anda. Selanjutnya, Install semua dependencies yang dibutuhkan dengan mengetikkan perintah berikut pada terminal:

```
1 npm install --save express mysql body-parser
```

Seperti gambar berikut:





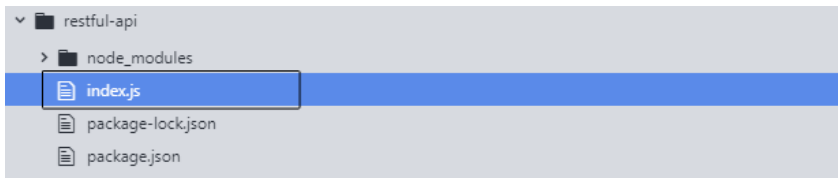
Perintah diatas akan menginstall semua dependencies yang kita butuhkan yaitu: express, mysql, dan body-parser.

Jika di buka file **package.json**, maka akan terlihat seperti berikut:

```
1  {
2    "name": "restful-api",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "M Fikri",
10   "license": "ISC",
11   "dependencies": {
12     "body-parser": "^1.18.3",
13     "express": "^4.16.4",
14     "mysql": "^2.16.0"
15   }
16 }
```

#### Step #4. Buat file Index.js

Buat sebuah file dengan nama **index.js**. Seperti gambar berikut:



Kemudian open **index.js** dan ketikan kode berikut:

```
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const app = express();
4  const mysql = require('mysql');
5
6  // parse application/json
7  app.use(bodyParser.json());
8
9  //create database connection
10 const conn = mysql.createConnection({
11   host: 'localhost',
12   user: 'root',
13   password: '',
14   database: 'restful_db'
15 });
16
17 //connect to database
18 conn.connect((err) =>{
19   if(err) throw err;
20   console.log('Mysql Connected...');
21 });
22
23 //tampilkan semua data product
24 app.get('/api/products', (req, res) => {
25   let sql = "SELECT * FROM product";
```

```

26     let query = conn.query(sql, (err, results) => {
27         if(err) throw err;
28         res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
29     });
30 });
31
32 //tampilkan data product berdasarkan id
33 app.get('/api/products/:id', (req, res) => {
34     let sql = "SELECT * FROM product WHERE product_id="+req.params.id;
35     let query = conn.query(sql, (err, results) => {
36         if(err) throw err;
37         res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
38     });
39 });
40
41 //Tambahkan data product baru
42 app.post('/api/products', (req, res) => {
43     let data = {product_name: req.body.product_name, product_price: req.body.product_price};
44     let sql = "INSERT INTO product SET ?";
45     let query = conn.query(sql, data, (err, results) => {
46         if(err) throw err;
47         res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
48     });
49 });
50
51 //Edit data product berdasarkan id
52 app.put('/api/products/:id', (req, res) => {
53     let sql = "UPDATE product SET product_name='"+req.body.product_name+"',
54         product_price='"+req.body.product_price+"' WHERE product_id="+req.params.id;
55     let query = conn.query(sql, (err, results) => {
56         if(err) throw err;
57         res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
58     });
59 });
60
61 //Delete data product berdasarkan id
62 app.delete('/api/products/:id', (req, res) => {
63     let sql = "DELETE FROM product WHERE product_id="+req.params.id+"";
64     let query = conn.query(sql, (err, results) => {
65         if(err) throw err;
66         res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
67     });
68 });
69
70 //Server listening
71 app.listen(3000, () =>{
72     console.log('Server started on port 3000...');
73 });

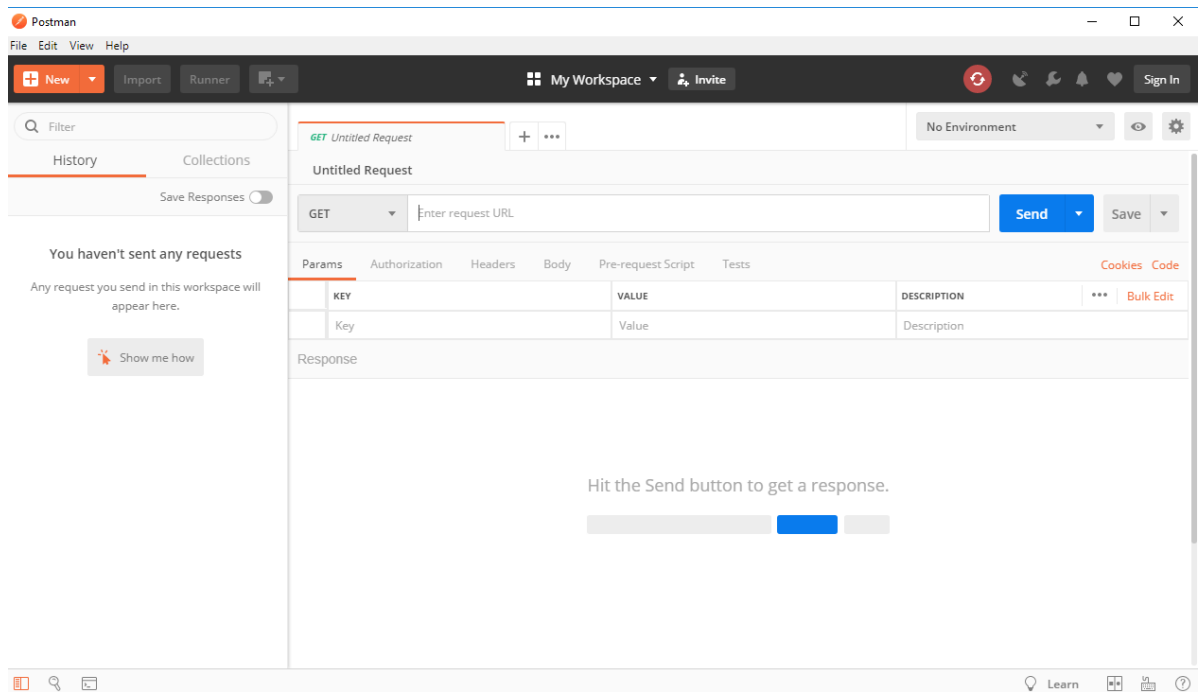
```

## Step #5. Testing

Uji coba API yang telah kita buat untuk memastikan RESTful API berjalan dengan baik. Untuk menguji API, ada banyak tools yang dapat digunakan. Saya menggunakan POSTMAN untuk menguji API yang telah kita buat. Jika Anda juga menggunakan POSTMAN itu lebih baik. Anda dapat mendownload POSTMAN di official websitenya: <https://www.getpostman.com/>

Download dan Install POSTMAN di komputer Anda kemudian open.

Jika POSTMAN telah terbuka, maka akan terlihat seperti gambar berikut:

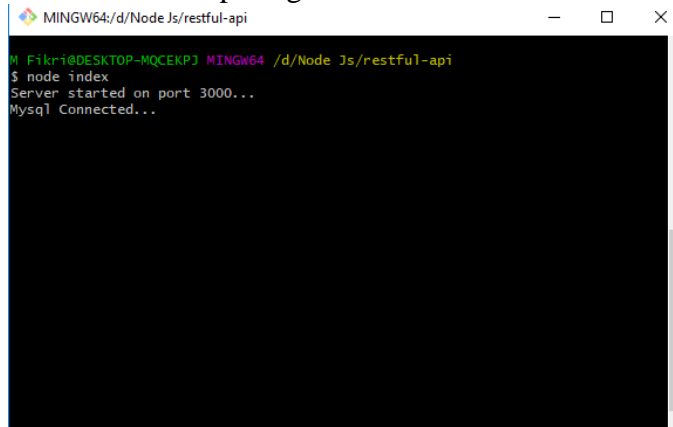


Ok, waktunya pengujian:

Running project dengan mengetikan perintah:

```
1 node index
```

Pada terminal seperti gambar berikut:



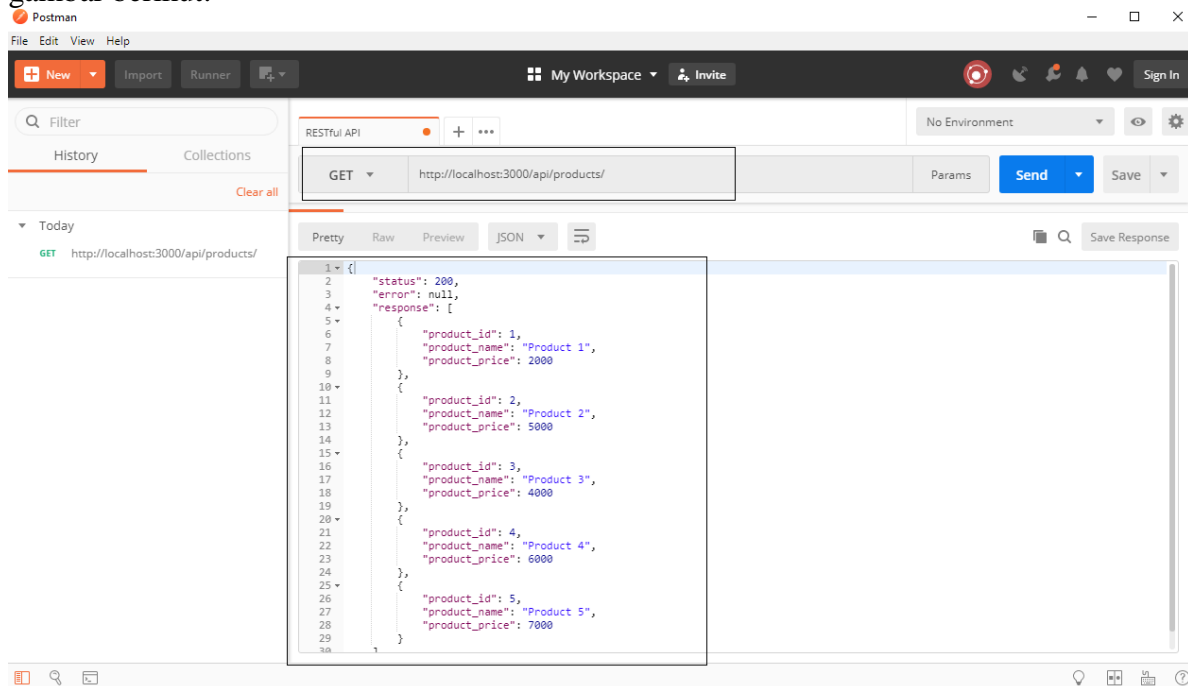
Jika terlihat seperti gambar diatas, berarti berjalan dengan baik.  
Mari kita uji EndPoint-nya satu per satu.

### #1. Get All Product (GET)

Kembali ke POSTMAN, dan ketikan URL berikut pada kolom URL Postman:

```
http://localhost:3000/api/products
```

Pilih method **GET**, kemudian klik tombol **Send**, maka Akan terlihat hasilnya seperti gambar berikut:



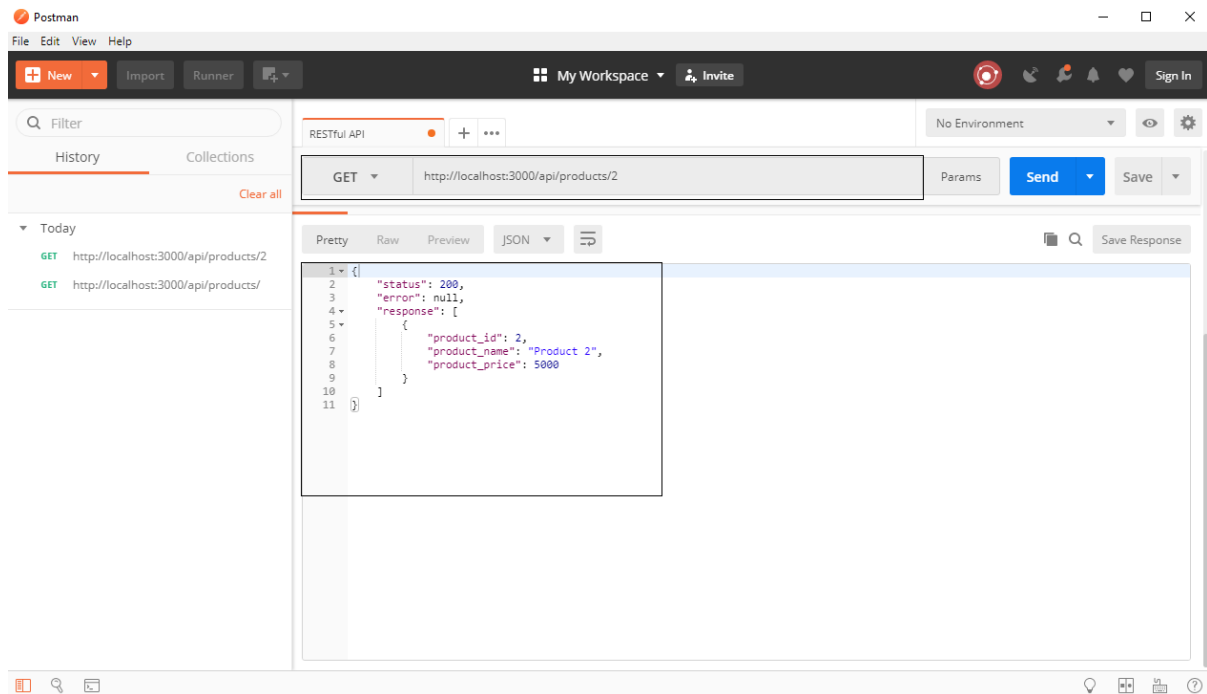
Pada gambar diatas dapat dilihat bahwa EndPoint untuk mendapatkan semua data product berjalan dengan baik.

## #2. Get Single Product (GET)

Ketikan URL berikut pada kolom untuk mendapatkan single product:

`http://localhost:3000/api/products/2`

Pilih dengan method **GET**, kemudian klik tombol **Send**, maka Akan terlihat hasilnya seperti gambar berikut:



Pada gambar diatas dapat terlihat hanya satu data product yang ditampilkan, yaitu product dengan product\_id='2' sesuai dengan parameter pada URL.

### #3. Create New Product (POST)

Ketikan URL berikut pada kolom URL untuk meng-create new product:

**http://localhost:3000/api/products**

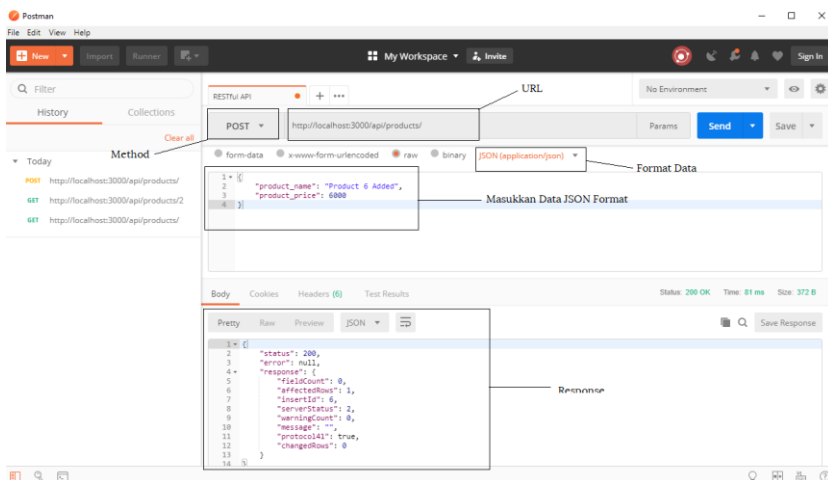
Pilih method **POST**, kemudian masukan data berikut pada kolom JSON(application/json):

```

1 {
2   "product_name": "Product 6 Added",
3   "product_price": 6000
4 }

```

Kemudian klik tombol **Send**, maka Akan terlihat hasilnya seperti gambar berikut:



Jika Anda perhatikan pada bagian response, terdapat “affectedRows”: 1, dan “insertId”: 6.

Itu berarti terdapat satu data yang diinsert ke database dengan product\_id=’6’.

#### #4. Update Product (PUT)

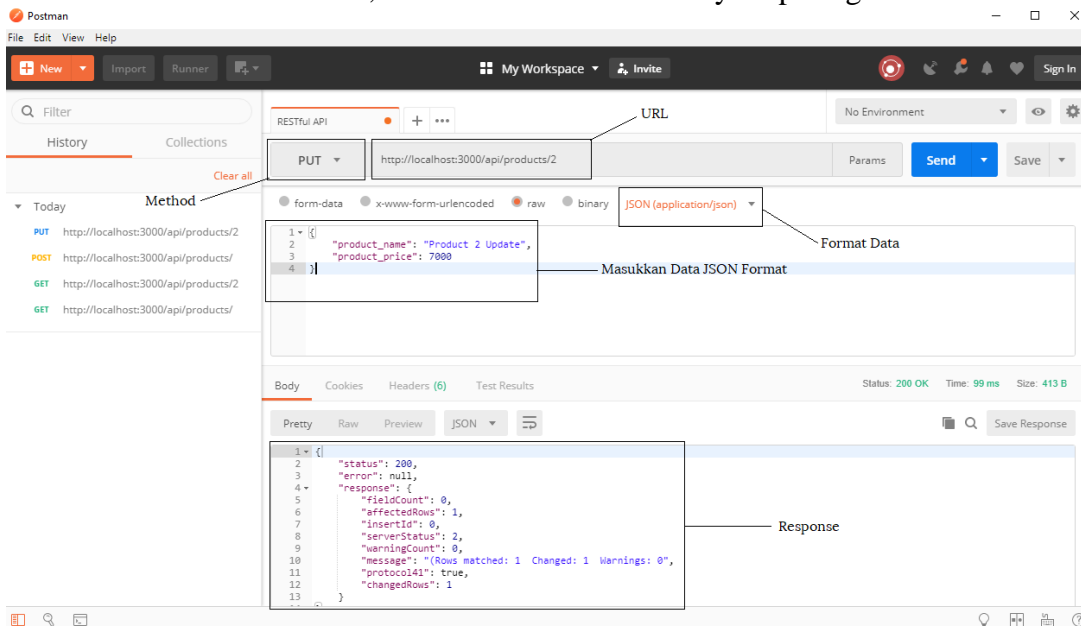
Ketikan URL berikut pada kolom URL untuk meng-update product:

**http://localhost:3000/api/products/2**

Pilih method PUT, kemudian masukan data berikut pada kolom JSON(application/json):

```
1 {
2   "product_name": "Product 2 Update",
3   "product_price": 7000
4 }
```

Kemudian klik tombol Send, maka Akan terlihat hasilnya seperti gambar berikut:



Jika Anda perhatikan pada bagian response, terdapat “affectedRows”: 1, dan “changedRows”: 1.

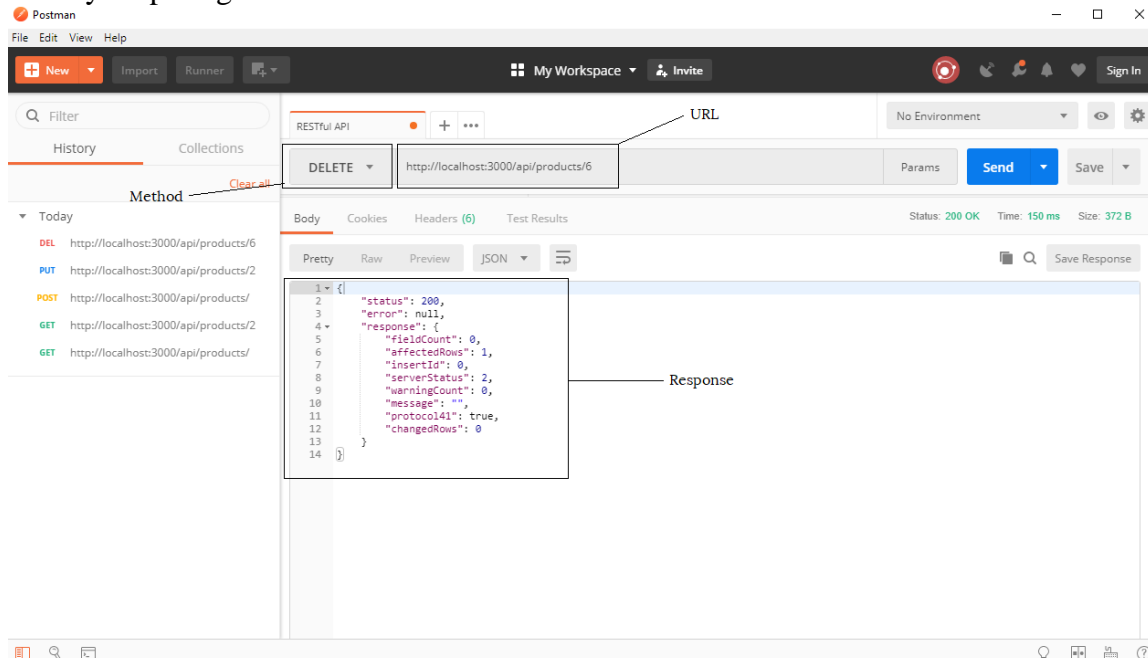
Itu berarti terdapat satu data yang diupdate ke database dengan product\_id='2' sesuai dengan parameter pada URL.

### #5 Delete Product (DELETE)

Ketikan URL berikut pada kolom URL untuk meng-hapus product:

`http://localhost:3000/api/products/6`

Pilih dengan method DELETE, kemudian klik tombol Send, maka Akan terlihat hasilnya seperti gambar berikut:



Jika Anda perhatikan pada bagian response, terdapat “affectedRows”: 1, “insertId”: 0 ,dan “changedRows”: 0.



Itu berarti terdapat satu data yang dihapus (delete) ke database dengan product\_id='6' sesuai dengan parameter pada URL.

**7 Tugas dan Pertanyaan :**

**8 Pustaka :**

1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
2. Tutorislpint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)

**9 Hasil Praktikum : Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 40 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)



---

<b>Minggu Ke</b>	:	7
<b>Capaian Pembelajaran</b>	:	Studi Kasus
<b>Waktu</b>	:	2 x 170 menit
<b>Tempat</b>	:	Laboratorium

---

<b>1. Sub Capaian Pembelajaran</b>	:	Studi Kasus
<b>2. Indikator Kinerja</b>	:	Studi Kasus
<b>3. Teori</b>	:	—
<b>4. Bahan dan Alat</b>	:	Komputer dan Logbook
<b>5. Organisasi</b>	:	Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
<b>6. Prosedur Kerja</b>	:	Kerjakan kasus yang diberikan sesuai dengan contoh teori dan praktik yang sudah diberikan
<b>7. Tugas dan Pertanyaan</b>	:	Buatlah aplikasi untuk manajemen surat.
<b>8. Pustaka</b>	:	1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011. <a href="http://www.koder.web.id">www.koder.web.id</a> 2. Tutorislpoint. Codeigniter. 2015. <a href="http://www.tutorialspoint.com">www.tutorialspoint.com</a>
<b>9. Hasil Praktikum</b>	:	<b>Laporan Praktikum</b>



	<p align="center"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 41 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

**Minggu Ke** : 9  
**Capaian Pembelajaran** : PHP Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

---

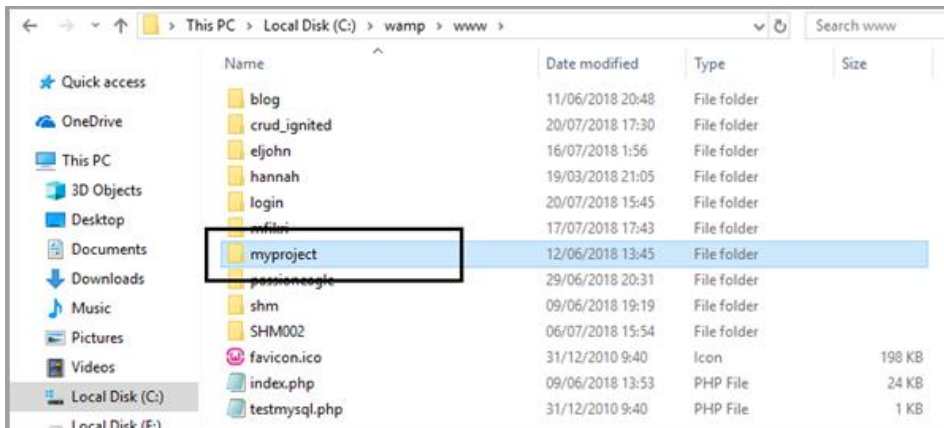
- 
- 1. **Sub Capaian Pembelajaran** : Pengenalan Codeigniter
  - 2. **Indikator Kinerja** : Mahasiswa
  - 3. **Teori** : —
  - 4 **Bahan dan Alat** : Komputer dan Logbook
  - 5 **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  - 6 **Prosedur Kerja** :

Instalasi pada codeigniter ikuti langkah berikut:

1. Pastikan Web Server telah terinstall dan berjalan (*running*) di komputer Anda.
2. Download file codeigniter di situs resminya: [www.codeigniter.com](http://www.codeigniter.com)

3. Extract file **Codeigniter.zip** ke direktori **C:/wamp/www/** (jika Anda menggunakan wampserver). Tetapi, jika Anda menggunakan XAMPP. Extract file **Codeigniter.zip** ke direktori **C:/xampp/htdocs/**.

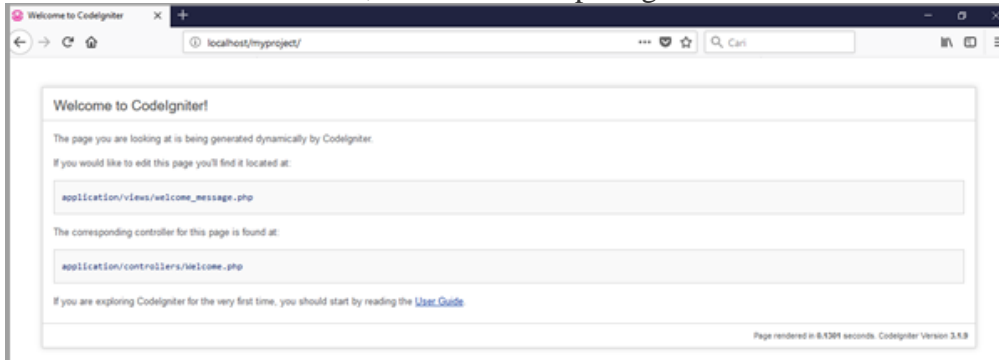
4. Pergi ke folder **c:/wamp/www/** (jika Anda menggunakan WAMP) dan **rename** (ganti nama) file folder codeigniter yang baru di extraxt tadi menjadi nama project Anda. Misalnya, disini saya ganti menjadi “**myproject**”. Sehingga terlihat seperti gambar berikut:



5. Selanjutnya, buka browser Anda. disini saya menggunakan [Mozilla Firefox](#). Kemudian kunjungi URL berikut:

**http://localhost/myproject/**

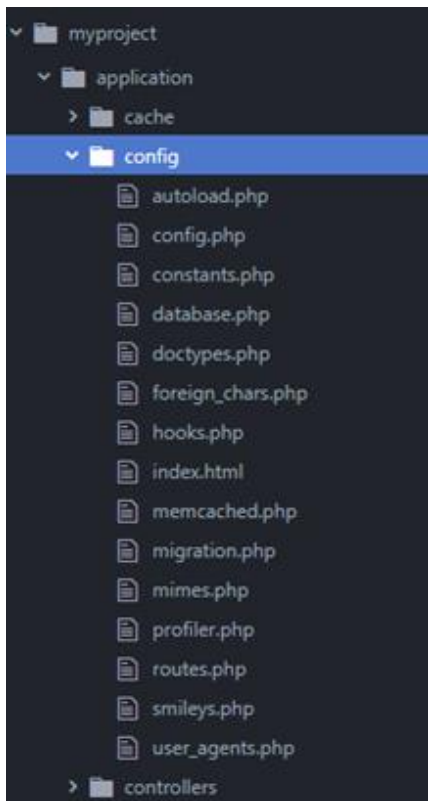
Jika installasi berhasil maka, akan terlihat seperti gambar berikut:



6. Selesai

## Konfigurasi dasar Codeigniter

Dalam memulai codeigniter, ada beberapa konfigurasi dasar yang perlu Anda ketahui. Yaitu **autoload.php**, **config.php**, dan **database.php**. Semua konfigurasi pada codeigniter, terletak pada satu tempat yaitu di dalam folder **application/config**.

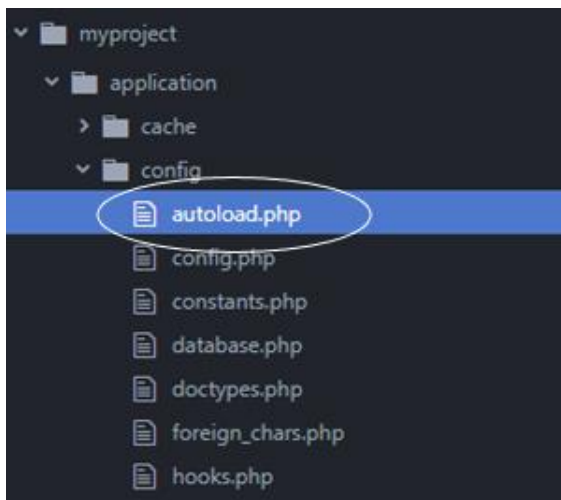


Bagaimana dan apa saja yang perlu di konfigurasi pada file autoload.php, config.php, dan database.php?

Berikut penjelasannya.

### 1. Autoload.php

**Autoload.php**, file ini digunakan untuk mengatur fungsi-fungsi yang akan dimuat otomatis di awal ketika program dijalankan. Untuk melakukan konfigurasi pada file autoload.php, silahkan buka folder: **application/config/autoload.php** seperti berikut gambar berikut:



Ada beberapa hal yang bisa di-load secara otomatis diantaranya: packages, libraries, drivers, helper files, custom config files, language files, dan models. Untuk konfigurasi dasar yang perlu Anda ketahui adalah **libraries** dan **helper files**. Hal ini bertujuan agar beberapa library dan helper tertentu berjalan secara otomatis. Untuk melakukan konfigurasi pada libraries, buka file autoload.php dengan text editor seperti notepad++, sublime text, atau lainnya.

kemudian temukan kode berikut:

```
1 $autoload['libraries'] = array();
```

Atur menjadi seperti berikut:

```
1 $autoload['libraries'] = array('database');
```

Pada kode diatas, artinya kita meload library “database” secara otomatis. Dengan demikian Anda dapat menggunakan fungsi-fungsi database pada codeigniter. Seperti fungsi: **Query Builder Class**. Selanjutnya, untuk melakukan konfigurasi pada helper files, buka file autoload.php dengan text editor. Kemudian temukan kode berikut:

```
1 $autoload['helper'] = array();
```

Atur menjadi seperti berikut:

```
1 $autoload['helper'] = array('url');
```

Pada kode diatas, artinya kita meload helper “url” secara otomatis. Dengan demikian Anda dapat menggunakan fungsi-fungsi url pada codeigniter. Seperti fungsi: **base\_url()**, **site\_url()**, **URI Segment**, dan sebagainya.

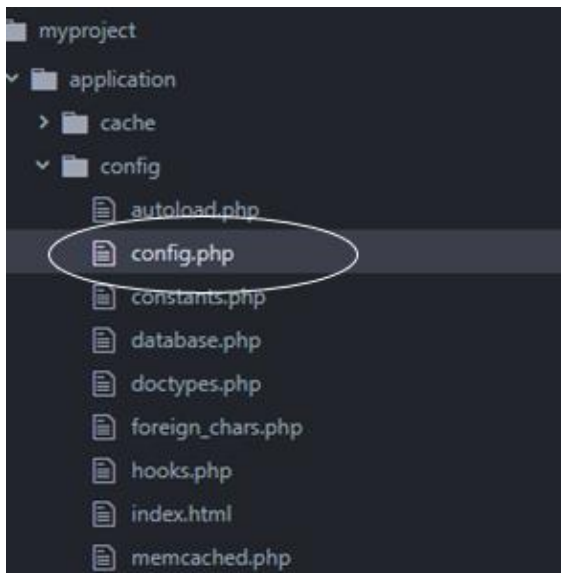
## 2. Config.php

Pada file ini terdapat beberapa konfigurasi yang secara standar sudah terkonfigurasi.

Namun terdapat beberapa konfigurasi yang perlu diperhatikan yaitu:

```
1 $config['base_url']
2 $config['index_page']
3 $config['encryption_key']
```

Untuk konfigurasi dasar, Anda cukup mengetahui konfigurasi **base\_url**. Base\_url merupakan url dasar dari project Anda. Untuk mengkonfigurasi base\_url, buka file config.php dengan text editor.



kemudian temukan kode berikut:

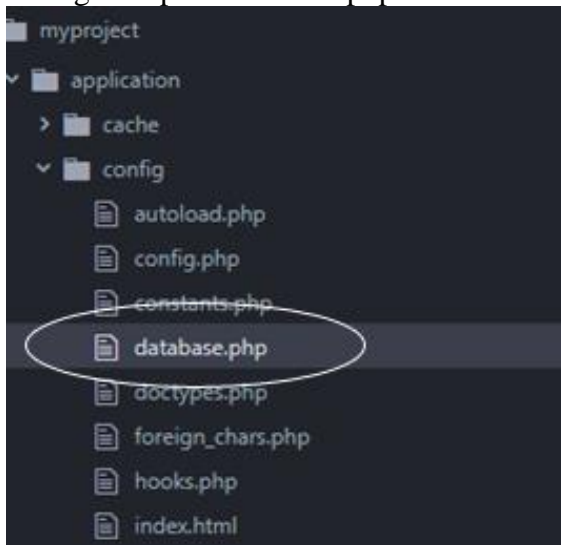
```
1 $config['base_url'] = '';
```

Atur menjadi seperti berikut:

```
1 $config['base_url'] = 'http://localhost/myproject/';
```

### 3. Database.php

Dilihat dari nama filenya maka Anda sudah dapat menangkap apa fungsi dari file ini. File *database.php* digunakan untuk melakukan konfigurasi yang berkaitan dengan konfigurasi database dari *website* yang akan dibuat. Adapun konfigurasi yang perlu diperhatikan yaitu: hostname, username, password, dan database. Untuk melakukan konfigurasi pada *database.php*. Buka file *database.php* dengan text editor.



Kemudian temukan kode berikut:

```

1  $active_group = 'default';
2  $query_builder = TRUE;
3
4  $db['default'] = array(
5      'dsn' => '',
6      'hostname' => 'localhost',
7      'username' => '',
8      'password' => '',
9      'database' => '',
10     'dbdriver' => 'mysqli',
11     'dbprefix' => '',
12     'pconnect' => FALSE,
13     'db_debug' => (ENVIRONMENT !== 'production'),
14     'cache_on' => FALSE,
15     'cachedir' => '',
16     'char_set' => 'utf8',
17     'dbcollat' => 'utf8_general_ci',
18     'swap_pre' => '',
19     'encrypt' => FALSE,
20     'compress' => FALSE,
21     'stricton' => FALSE,
22     'failover' => array(),
23     'save_queries' => TRUE
24 );

```



**Atur menjadi seperti berikut:**

```

1  $active_group = 'default';
2  $query_builder = TRUE;
3
4  $db['default'] = array(
5      'dsn' => '',
6      'hostname' => 'localhost', // Hostname
7      'username' => 'root',      // Username
8      'password' => '',          // password
9      'database' => 'database_name', //database name
10     'dbdriver' => 'mysqli',
11     'dbprefix' => '',
12     'pconnect' => FALSE,
13     'db_debug' => (ENVIRONMENT !== 'production'),
14     'cache_on' => FALSE,
15     'cachedir' => '',
16     'char_set' => 'utf8',
17     'dbcollat' => 'utf8_general_ci',
18     'swap_pre' => '',
19     'encrypt' => FALSE,
20     'compress' => FALSE,
21     'stricton' => FALSE,
22     'failover' => array(),
23     'save_queries' => TRUE
24 );

```

- 7 Tugas dan Pertanyaan :**
- 8 Pustaka :**
1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
  2. Tutorislpoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)
- 9 Hasil Praktikum : Laporan Praktikum**

	<p align="center"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 48 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

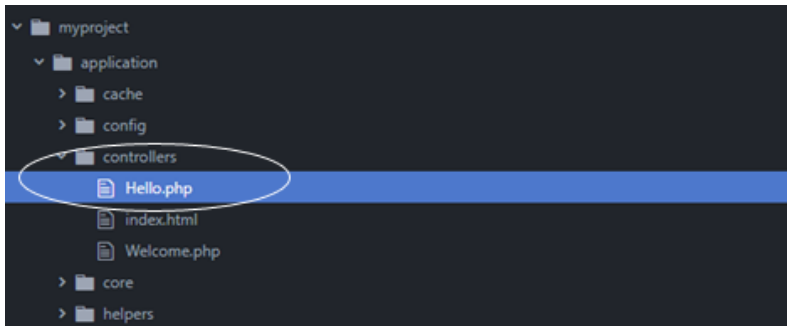
**Minggu Ke** : 10  
**Capaian Pembelajaran** : PHP Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

---

- 
- 1. **Sub Capaian Pembelajaran** : PHP Framework - Controller, URL SEO Friendly, View
  - 2. **Indikator Kinerja** : Mahasiswa
  - 3. **Teori** : —
  - 4 **Bahan dan Alat** : Komputer dan Logbook
  - 5 **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  - 6 **Prosedur Kerja** :

Jika serius dengan codeigniter, Anda harus mengerti bagaimana sebuah controller bekerja. Untuk lebih jelasnya, saya akan sharing kasus sederhana agar Anda dapat memahami bagaimana controller bekerja. Disini saya mengangkat kasus yaitu bagaimana menampilkan text “Hello World” pada browser menggunakan controller. Buat sebuah controller dengan nama **Hello.php** seperti gambar berikut:





Kemudian ketikkan kode berikut:

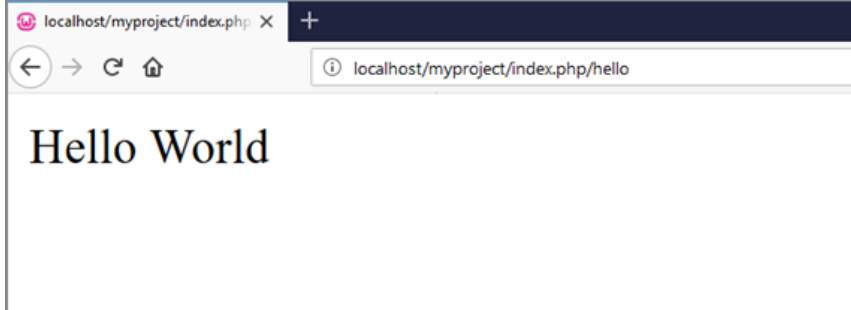
```
1 <?php
2 class Hello extends CI_Controller{
3
4     function index(){
5         echo "Hello World";
6     }
7
8 }
```

note: *Setiap penulisan nama file dan nama class selalu di dahului dengan huruf Capital.*

Setelah itu save dan buka browser Anda, lalu kunjungi url berikut:

**http://localhost/myproject/index.php/hello**

Maka akan akan terlihat text “Hello World” pada browser Anda seperti berikut:



Jika Anda perhatikan dengan seksama, pada dasarnya url pada codeigniter terlihat seperti gambar berikut:

Primary Domain                      Class Name  
 |    |  
 http://www.domain.com/index.php/hello/index  
 |    |    |  
Protocol                                      index page                                      Function Name

Dimana, terdapat protocol, primary domain, index.php, class name, dan function name. Mungkin terdegas rumit, tapi sebenarnya tidak. Untuk lebih jelasnya silahkan tambahkan satu function lagi pada Controller Hello.php. disini saya beri nama “**show**”. Sehingga controller **Hello.php** menjadi seperti berikut:

```

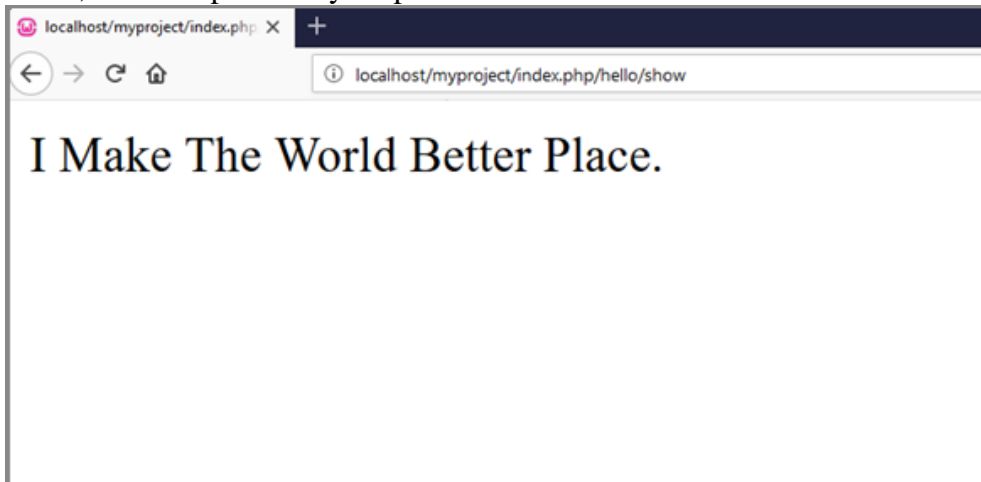
1  <?php
2  class Hello extends CI_Controller{
3
4      function index(){
5          echo "Hello World";
6      }
7
8      function show(){
9          echo "I Make The World Better Place.";
10     }
11
12 }

```

Jika Anda jalankan dengan mengunjungi URL berikut:

**<http://localhost/myproject/index.php/hello/show>**

Maka, akan tampil hasilnya seperti berikut:



## **Menghilangkan index.php pada URL**

Codeigniter merupakan framework php yang mendukung clean URL. Dengan demikian Anda dapat membuat URL yang mudah dibaca dan sekaligus SEO Friendly. Pada URL aplikasi “Hello World” diatas, dapat dilihat bahwa adanya **index.php** pada url yang terlihat mengganggu. Adakah cara untuk menghilangkan index.php dari URL? Tentu saja, Anda dapat menggunakan file **.htaccess** untuk menghilangkannya. Bagaimana membuat file **.htaccess**?

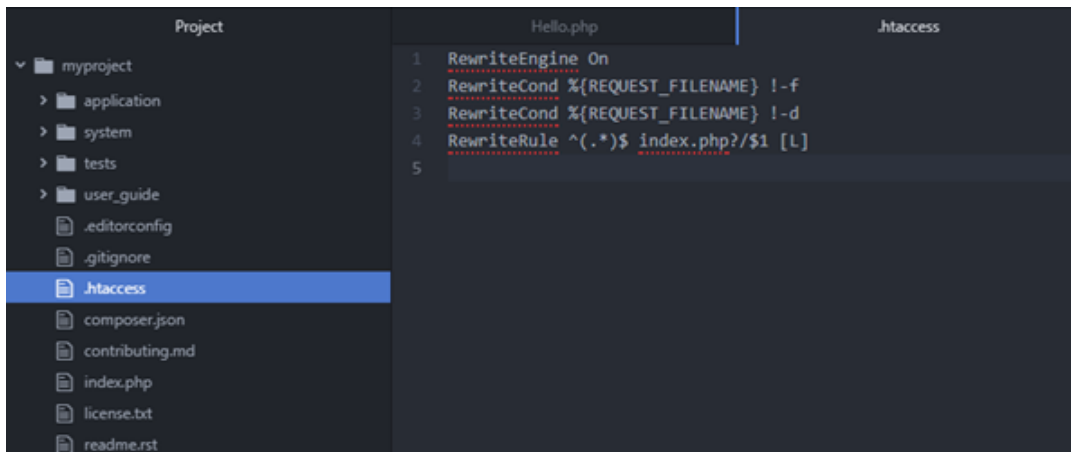
Buat sebuah file dengan nama **.htaccess** pada web root Anda dan ketikan kode berikut:

```

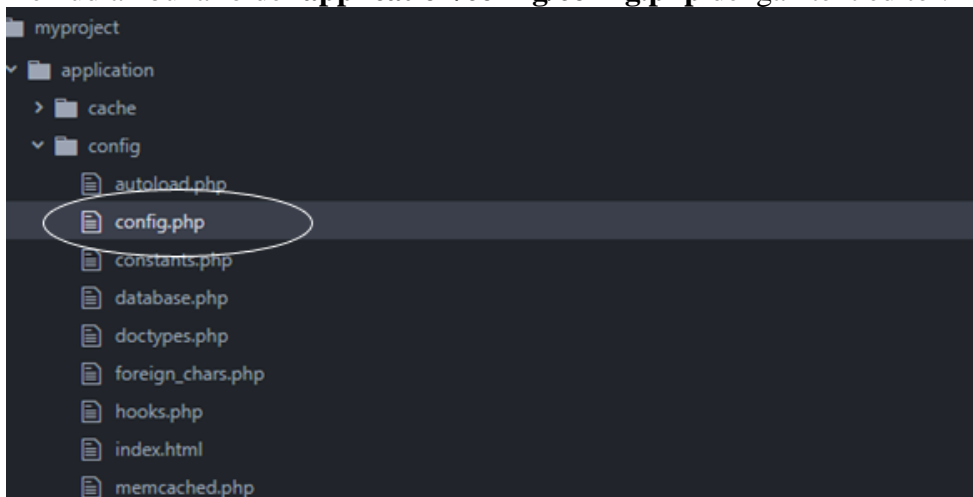
1  RewriteEngine On
2  RewriteCond %{REQUEST_FILENAME} !-f
3  RewriteCond %{REQUEST_FILENAME} !-d
4  RewriteRule ^(.*)$ index.php?/$1 [L]

```

Seperti gambar berikut:



Kemudian buka folder **application/config/config.php** dengan text editor.



Kemudian temukan kode berikut:

```
1 $config['index_page'] = 'index.php';
```

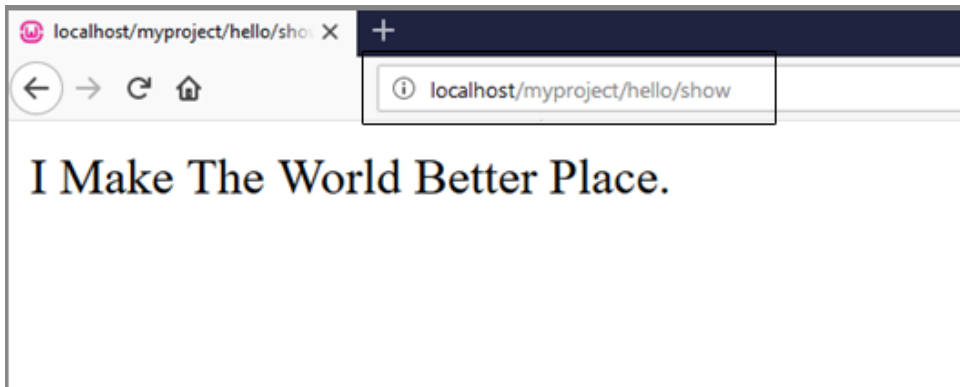
Atur menjadi seperti berikut:

```
1 $config['index_page'] = '';
```

Sekarang silahkan kunjungi url berikut untuk uji coba:

**<http://localhost/myproject/hello/show>**

Maka akan terlihat hasilnya seperti berikut:



Pada gambar diatas, dapat dilihat bahwa URL menjadi lebih rapi dan SEO friendly dengan menghilangkan **index.php** pada URL.

## Controller dan View

Pada kasus sebelumnya, Anda telah mengetahui bagaimana menampilkan text “Hello World” langsung dari controller. Namun, hal tersebut sebaiknya dilakukan di view. Sekarang saya akan menunjukkan bagaimana menampilkan view melalui controller.

Pertama, buat sebuah file pada **application/controller** dengan nama **Blog.php**.

Kemudian ketikkan kode berikut:

```
1  <?php
2  class Blog extends CI_Controller
3  {
4      function __construct()
5      {
6          parent::__construct();
7      }
8
9      function index(){
10         $this->load->view('blog_view');
11     }
12
13 }
```

Kedua, buat sebuah file di **application/views** dengan nama **blog\_view.php**.

Kemudian ketikkan kode berikut:

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title>My Blog</title>
6      </head>
7      <body>
8          <h1>Welcome To My Blog.</h1>
9      </body>
10 </html>
```

Kemudian, buka browser Anda dan akses controller **blog**. Maka akan terlihat hasilnya seperti berikut:



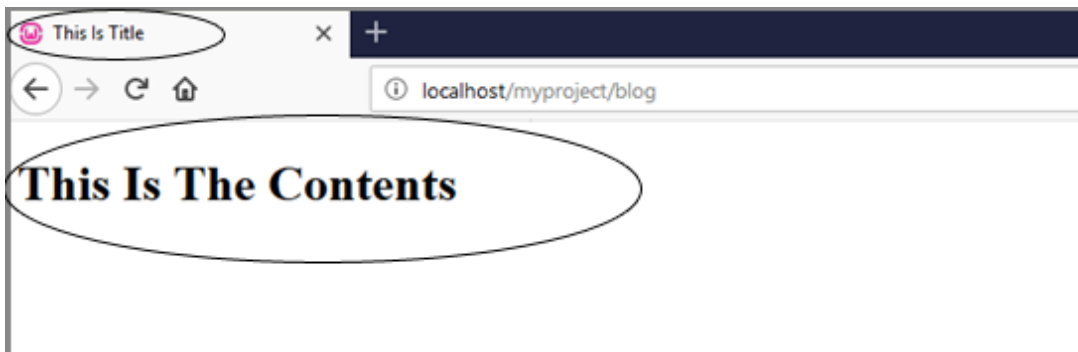
Anda juga dapat mengirimkan parameter ke view melalui controller. Sebagai contoh, silahkan ubah controller **Blog.php** menjadi seperti berikut:

```
1  <?php
2  class Blog extends CI_Controller
3  {
4      function __construct()
5      {
6          parent::__construct();
7      }
8
9      function index(){
10         $data['title']    = "This Is Title";
11         $data['content'] = "This Is The Contents";
12         $this->load->view('blog_view', $data);
13     }
14
15 }
```

Kemudian ubah view **blog\_view.php** menjadi seperti berikut:



```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title><?php echo $title;?></title>
6      </head>
7      <body>
8          <h1><?php echo $content;?></h1>
9      </body>
10 </html>
```

Kemudian, buka browser Anda dan akses kembali controller **blog**. Maka akan terlihat hasilnya seperti berikut:



Saya harap Anda dapat memahami perbedaannya.

- 7 Tugas dan Pertanyaan :**
- 8 Pustaka :**
1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
  2. Tutorislpoin. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)
- 9 Hasil Praktikum : Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 55 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

**Minggu Ke** : 11  
**Capaian Pembelajaran** : PHP Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

---

- 
1. **Sub Capaian Pembelajaran** : Menghubungkan Codeigniter dengan Bootstrap
  2. **Indikator Kinerja** : Mahasiswa
  3. **Teori** : —
  4. **Bahan dan Alat** : Komputer dan Logbook
  5. **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  6. **Prosedur Kerja** :

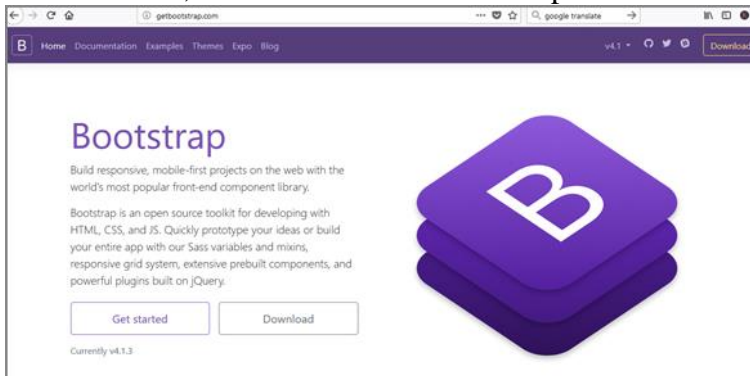
Pada kasus sebelumnya, Anda telah memahami bagaimana memanggil view melalui controller. Sekarang, ada hal yang sangat penting untuk Anda ketahui, yaitu mengkombinasikan codeigniter dengan bootstrap.

Apa itu **BOOTSTRAP**?

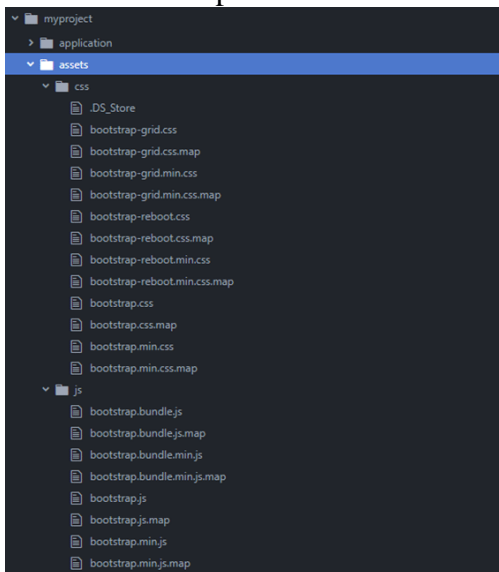
Bootstrap adalah toolkit open source untuk dikembangkan dengan HTML, CSS, dan JS. Dengan kata lain, Bootstrap merupakan framework untuk mempercantik user interface (UI). Bootstrap bersifat responsive. Dengan kata lain, merender dengan baik di berbagai macam perangkat (*platform*) seperti tablet maupun mobile phone.

Bagaimana mengkombinasikan codeigniter dan bootstrap?

Pertama-tama, silahkan download bootstrap di situs resminya **getbootstrap.com**.

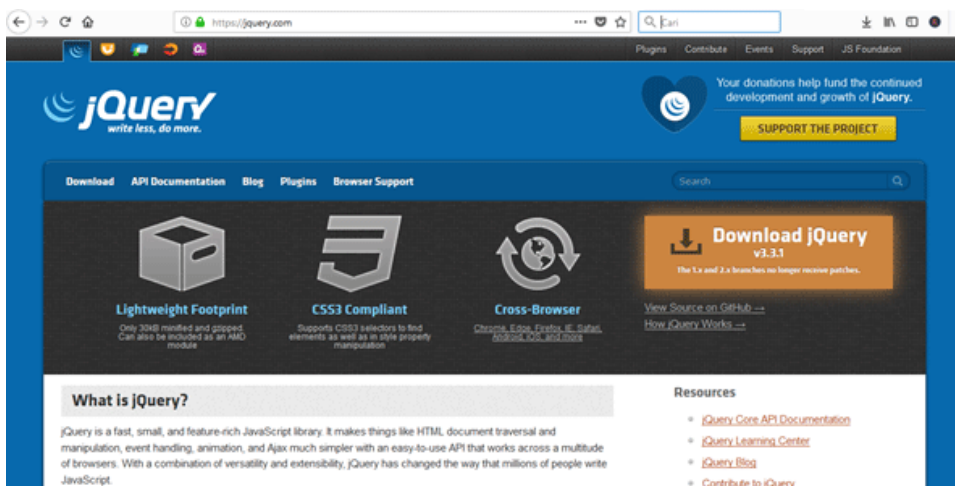


Kedua, buat sebuah folder baru pada project (*webroot*) Anda. Disini saya beri nama folder “**assets**”. Kemudian extract file bootstrap yang telah di download tadi kedalam folder **assets**. Seperti berikut:

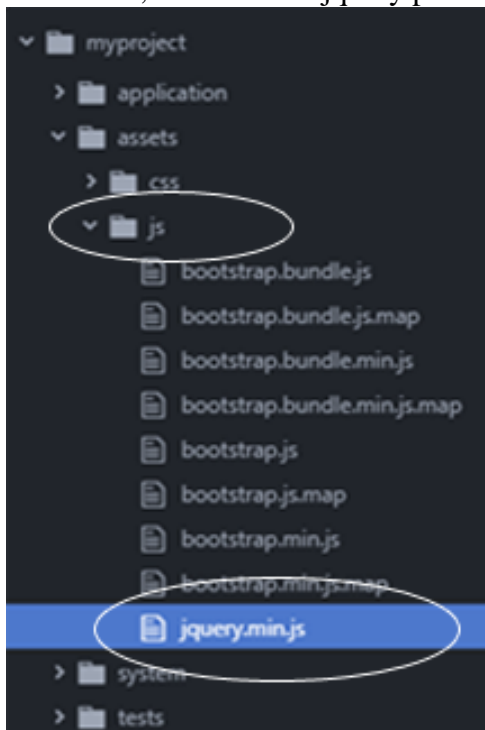


Selain bootstrap, kita juga membutuhkan **jquery** agar javascript pada bootstrap berjalan dengan optimal. Untuk mendownload JQuery, silahkan download di situs resminya **jquery.com**.





Kemudian, letakkan file jquery pada folder **assets/js/** seperti gambar berikut:



Mungkin terlihat sedikit rumit, tapi sebenarnya tidak. Agar Anda dapat memahami seperti apa bootstrap, silahkan edit file view **blog\_view.php** menjadi seperti berikut:

```

1      <!DOCTYPE html>
2      <html lang="en">
3          <head>
4              <meta charset="utf-8">
5              <title><?php echo $title;?></title>
6              <!-- load bootstrap css file -->
7              <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8          </head>
9          <body>
10

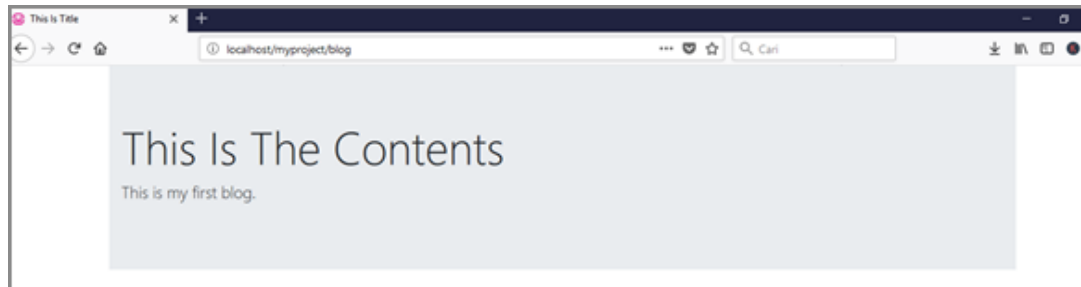
```

```

11     <div class="container">
12         <div class="jumbotron jumbotron-fluid">
13             <div class="container">
14                 <h1 class="display-4"><?php echo $content;?></h1>
15                 <p class="lead">This is my first blog.</p>
16             </div>
17         </div>
18     </div>
19
20     <!-- load jquery js file -->
21     <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
22     <!-- load bootstrap js file -->
23     <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
24 </body>
25 </html>

```

Jika Anda panggil lagi controller **blog** pada browser, maka akan terlihat hasilnya seperti berikut:



Pada gambar diatas, Anda dapat melihat bahwa kita tidak perlu membuat kode **css** untuk memberikan style pada suatu halaman website. Demikian pula, jika Anda membutuhkan table yang cantik. Anda juga tidak perlu mengetikan kode **css** untuk memberikan style pada table tersebut. Melainkan, anda dapat langsung memiliki table yang cantik secara instan. Contoh, silahkan edit lagi file view **blog\_view.php** menjadi seperti berikut:

```

1     <!DOCTYPE html>
2     <html lang="en">
3         <head>
4             <meta charset="utf-8">
5             <title><?php echo $title;?></title>
6             <!-- load bootstrap css file -->
7             <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8         </head>
9         <body>
10
11             <div class="container">
12                 <h1><?php echo $content;?></h1>
13                 <table class="table table-striped">
14                     <thead>
15                         <tr>
16                             <th scope="col">#</th>
17                             <th scope="col">First</th>
18                             <th scope="col">Last</th>
19                             <th scope="col">Handle</th>
20                         </tr>
21                     </thead>
22                     <tbody>
23                         <tr>
24                             <th scope="row">1</th>
25                             <td>Mark</td>
26                             <td>Otto</td>

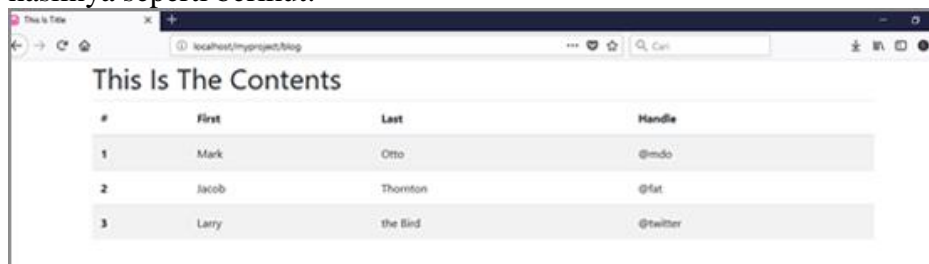
```

```

27         <td>@mdo</td>
28     </tr>
29     <tr>
30         <th scope="row">2</th>
31         <td>Jacob</td>
32         <td>Thornton</td>
33         <td>@fat</td>
34     </tr>
35     <tr>
36         <th scope="row">3</th>
37         <td>Larry</td>
38         <td>the Bird</td>
39         <td>@twitter</td>
40     </tr>
41 </tbody>
42 </table>
43
44 </div>
45
46 <!-- load jquery js file -->
47 <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
48 <!-- load bootstrap js file -->
49 <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
50 </body>
51 </html>

```

Jika Anda jalankan kembali controller **Blog** pada browser, maka Anda akan dapatkan hasilnya seperti berikut:





**7 Tugas dan Pertanyaan :**

**8 Pustaka :**

1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
2. Tutorialspoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)

**9 Hasil Praktikum : Laporan Praktikum**

	<p align="center"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 60 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

**Minggu Ke** : 12  
**Capaian Pembelajaran** : PHP Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

---

- 
- 1. **Sub Capaian Pembelajaran** : Bekerja dengan Database
  - 2. **Indikator Kinerja** : Mahasiswa
  - 3. **Teori** : —
  - 4 **Bahan dan Alat** : Komputer dan Logbook
  - 5 **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  - 6 **Prosedur Kerja** :

Anda akan belajar semua hal yang Anda butuhkan untuk tahu bagaimana berinteraksi dengan database menggunakan codeigniter. Mulai dari Create, Read, Update, dan Delete.

### 1. Persiapan database.

Pertama-tama buat sebuah database. Disini saya membuat sebuah database dengan nama “**pos\_db**”. Jika Anda membuat database dengan nama yang sama, itu lebih baik. Untuk membuat database, anda dapat mengeksekusi query berikut:

```
1 CREATE DATABASE pos_db;
```

Query diatas akan membuat sebuah database dengan nama “**pos\_db**”. Kemudian, buat table “**product**” dengan struktur sebagai berikut:

Field	Type	Null	Key	Default	Extra
product_id	int(11)	NO	PRI	NULL	auto_increment
product_name	varchar(100)	YES		NULL	
product_price	int(11)	YES		NULL	

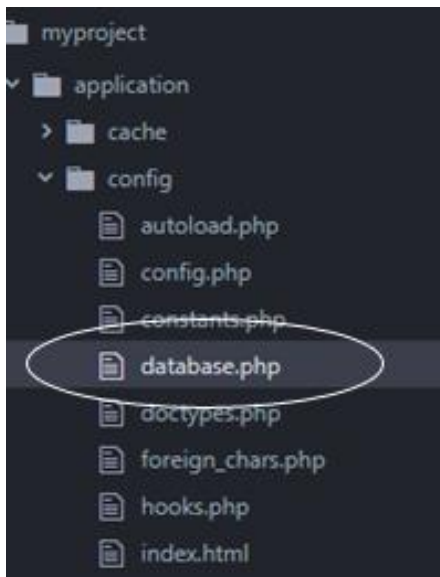
Anda dapat mengeksekusi query berikut untuk menghasilkan table dengan struktur seperti diatas.

```
1 CREATE TABLE product(  
2 product_id INT PRIMARY KEY AUTO_INCREMENT,  
3 product_name VARCHAR(100),  
4 product_price INT  
5 );
```

Kemudian masukan beberapa data kedalam table “**product**” dengan mengeksekusi query berikut:

```
1 INSERT INTO product(product_name,product_price) VALUES  
2 ('Coca Cola','5000'),  
3 ('Teh Botol','3700'),  
4 ('You C 1000','6300'),  
5 ('Ponds Men','18000'),  
6 ('Rexona Men','13000');
```

Langkah selanjutnya adalah mengkoneksikan codeigniter dengan database. Untuk mengkoneksikan codeigniter dengan database sangatlah sederhana, silahkan buka file **application/config/database.php**



Buka file database.php dengan text editor dan temukan kode berikut:

```
1  $active_group = 'default';
2  $query_builder = TRUE;
3
4  $db['default'] = array(
5      'dsn' => '',
6      'hostname' => 'localhost',
7      'username' => '',
8      'password' => '',
9      'database' => '',
10     'dbdriver' => 'mysqli',
11     'dbprefix' => '',
12     'pconnect' => FALSE,
13     'db_debug' => (ENVIRONMENT !== 'production'),
14     'cache_on' => FALSE,
15     'cachedir' => '',
16     'char_set' => 'utf8',
17     'dbcollat' => 'utf8_general_ci',
18     'swap_pre' => '',
19     'encrypt' => FALSE,
20     'compress' => FALSE,
21     'stricton' => FALSE,
22     'failover' => array(),
23     'save_queries' => TRUE
24 );
```

Lalu setting menjadi seperti berikut:

```
1  $active_group = 'default';
2  $query_builder = TRUE;
3
4  $db['default'] = array(
5      'dsn' => '',
6      'hostname' => 'localhost',
7      'username' => 'root',
```

```

8      'password' => '',
9      'database' => 'pos_db',
10     'dbdriver' => 'mysqli',
11     'dbprefix' => '',
12     'pconnect' => FALSE,
13     'db_debug' => (ENVIRONMENT !== 'production'),
14     'cache_on' => FALSE,
15     'cachedir' => '',
16     'char_set' => 'utf8',
17     'dbcollat' => 'utf8_general_ci',
18     'swap_pre' => '',
19     'encrypt' => FALSE,
20     'compress' => FALSE,
21     'stricton' => FALSE,
22     'failover' => array(),
23     'save_queries' => TRUE
24 );

```


Silahkan jalankan lagi project Anda pada browser, jika tidak ada error berarti koneksi ke database berhasil.

**7 Tugas dan Pertanyaan :**

**8 Pustaka :**

1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
2. Tutorialspoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)

**9 Hasil Praktikum : Laporan Praktikum**

	<p style="text-align: center;"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 64 dari 78</b>

## BUKU PANDUAN PRAKTIKUM (BPP)

---

**Minggu Ke** : 13 dan 14  
**Capaian Pembelajaran** : PHP Framework  
**Waktu** : 2 x 170 menit  
**Tempat** : Laboratorium

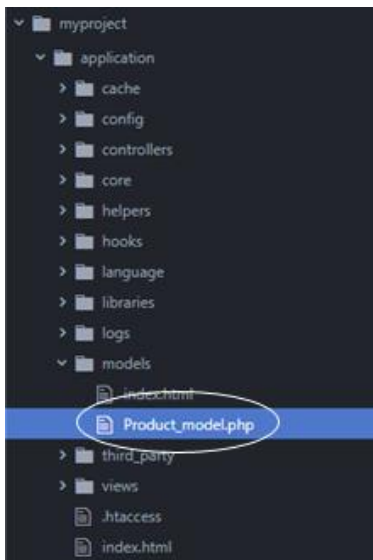
---

- 
- 1. **Sub Capaian Pembelajaran** : CRUD
  - 2. **Indikator Kinerja** : Mahasiswa
  - 3. **Teori** : —
  - 4. **Bahan dan Alat** : Komputer dan Logbook
  - 5. **Organisasi** : Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah
  - 6. **Prosedur Kerja** :
- 

**Menampilkan data dari database ke view (Read).**

1. Buat sebuah file didalam **application/models** dengan nama “**Product\_model.php**”.  
Seperti gambar berikut:



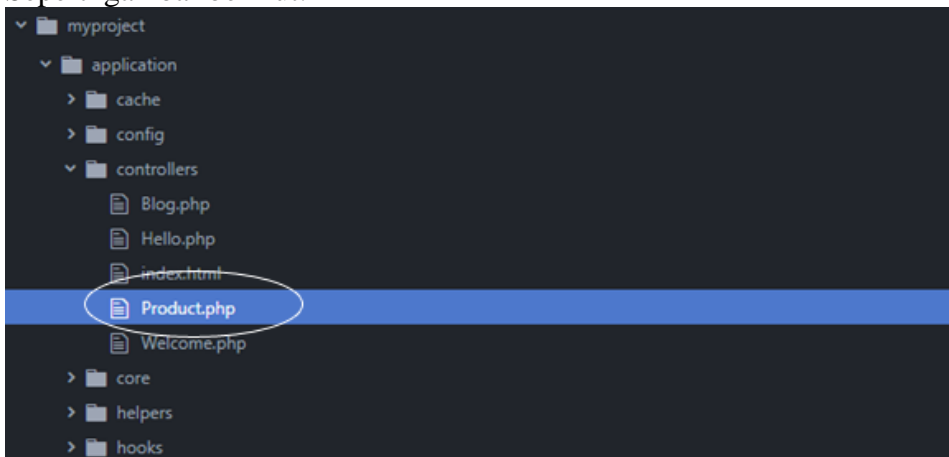


Buka file “Product\_model.php” dengan text editor. Kemudian ketikan kode berikut:

```

1  <?php
2  class Product_model extends CI_Model{
3
4      function get_product(){
5          $result = $this->db->get('product');
6          return $result;
7      }
8
9  }
```

2. Buat sebuah file didalam **application/controllers** dengan nama “**Product.php**”.  
Seperti gambar berikut:



Buka file controller “Product.php” dengan text editor. Kemudian ketikan kode berikut:

```

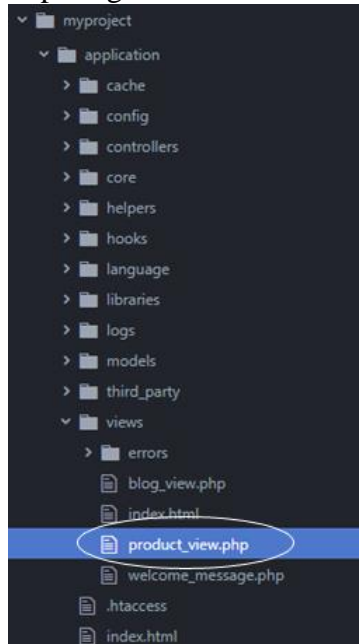
1  <?php
2  class Product extends CI_Controller{
3      function __construct(){
4          parent::__construct();
```

```

5         $this->load->model('product_model');
6     }
7     function index(){
8         $data['product'] = $this->product_model->get_product();
9         $this->load->view('product_view',$data);
10    }
11 }

```

3. Buat sebuah file view dengan nama “**product\_view.php**”.  
Seperti gambar berikut:



kemudian ketikan kode berikut:

```

1     <!DOCTYPE html>
2     <html lang="en">
3         <head>
4             <meta charset="utf-8">
5             <title>Product List</title>
6             <!-- load bootstrap css file -->
7             <link href="php echo base_url('assets/css/bootstrap.min.css'); ?" rel="stylesheet">
8         </head>
9         <body>
10
11             <div class="container">
12                 <h1><center>Product List</center></h1>
13                 <table class="table table-striped">
14                     <thead>
15                         <tr>
16                             <th scope="col">#</th>
17                             <th scope="col">Product Name</th>
18                             <th scope="col">Price</th>
19                         </tr>
20                     </thead>
21                     <?php
22                         $count = 0;
23                         foreach ($product->result() as $row) :
24                             $count++;
25                     <?>

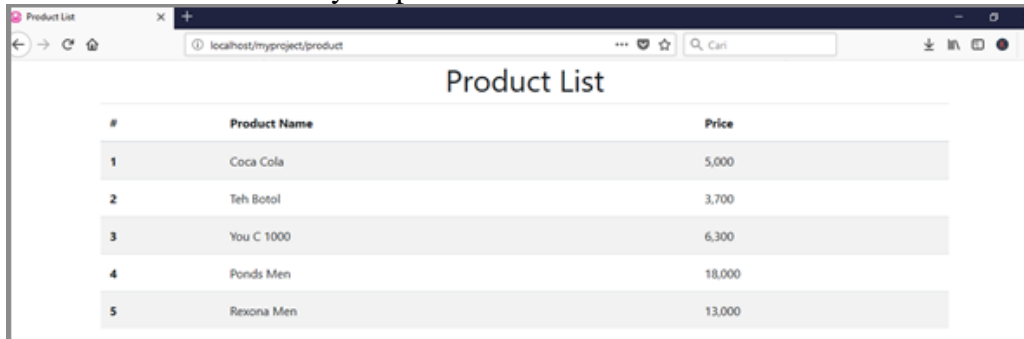
```

```

26         <tr>
27             <th scope="row"><?php echo $count;?></th>
28             <td><?php echo $row->product_name;?></td>
29             <td><?php echo number_format($row->product_price);?></td>
30         </tr>
31     <?php endforeach;?>
32 </tbody>
33 </table>
34
35 </div>
36
37 <!-- load jquery js file -->
38 <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
39 <!-- load bootstrap js file -->
40 <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
41 </body>
42 </html>

```

Kemudian, jalankan controller **“Product”** melalui browser Anda, dengan mengunjungi url berikut: **<http://localhost/myproject/product>**  
Maka akan terlihat hasilnya seperti berikut:



The screenshot shows a web browser window with the address bar displaying 'localhost/myproject/product'. The page title is 'Product List'. Below the title is a table with the following data:

#	Product Name	Price
1	Coca Cola	5,000
2	Teh Botol	3,700
3	You C 1000	6,300
4	Ponds Men	18,000
5	Rexona Men	13,000

## Insert data ke database (Create).

Pada segment kali ini, saya akan menunjukkan kepada Anda bagaimana insert data ke database.

1. Buka file model **“Product\_model.php”**. kemudian tambahkan satu function lagi seperti berikut:

```

1     <?php
2     class Product_model extends CI_Model{
3
4         function get_product() {
5             $result = $this->db->get('product');
6             return $result;
7         }
8         function save($product_name, $product_price) {
9             $data = array(
10                 'product_name' => $product_name,
11                 'product_price' => $product_price
12             );
13             $this->db->insert('product', $data);
14         }
15     }

```

2. Buka file controller “**Product.php**”. kemudian tambahkan beberapa function lagi seperti berikut:

```

1  <?php
2  class Product extends CI_Controller{
3      function __construct(){
4          parent::__construct();
5          $this->load->model('product_model');
6      }
7      function index(){
8          $data['product'] = $this->product_model->get_product();
9          $this->load->view('product_view',$data);
10     }
11     function add_new(){
12         $this->load->view('add_product_view');
13     }
14     function save(){
15         $product_name = $this->input->post('product_name');
16         $product_price = $this->input->post('product_price');
17         $this->product_model->save($product_name,$product_price);
18         redirect('product');
19     }
20 }
```

3. Buat sebuah file view lagi dengan nama “**add\_product\_view.php**”. dengan kode seperti berikut:

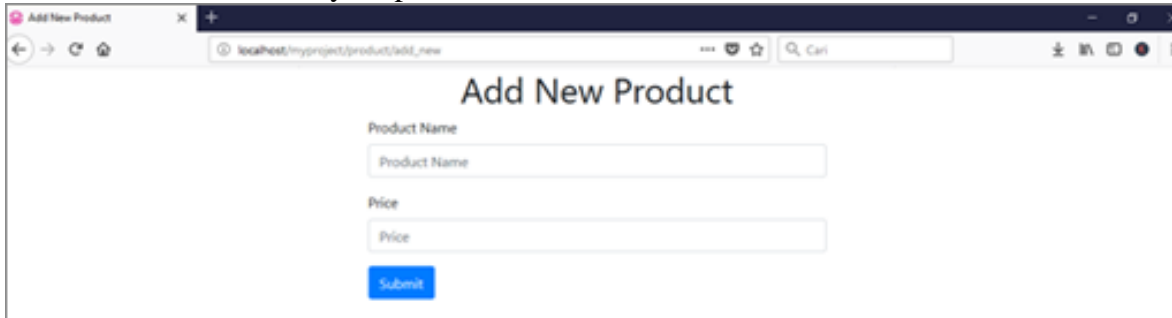
```

1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title>Add New Product</title>
6          <!-- load bootstrap css file -->
7          <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8      </head>
9      <body>
10
11         <div class="container">
12             <h1><center>Add New Product</center></h1>
13             <div class="col-md-6 offset-md-3">
14                 <form action="<?php echo site_url('product/save');?>" method="post">
15                     <div class="form-group">
16                         <label>Product Name</label>
17                         <input type="text" class="form-control" name="product_name" placeholder="Product Name">
18                     </div>
19                     <div class="form-group">
20                         <label>Price</label>
21                         <input type="text" class="form-control" name="product_price" placeholder="Price">
22                     </div>
23                     <button type="submit" class="btn btn-primary">Submit</button>
24                 </form>
25             </div>
26         </div>
27
28         <!-- load jquery js file -->
29         <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
30         <!-- load bootstrap js file -->
31         <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
32     </body>
33 </html>
```

Kemudian, kembali ke browser dan ketikkan url berikut pada browser Anda:

**[http://localhost/myproject/product/add\\_new](http://localhost/myproject/product/add_new)**

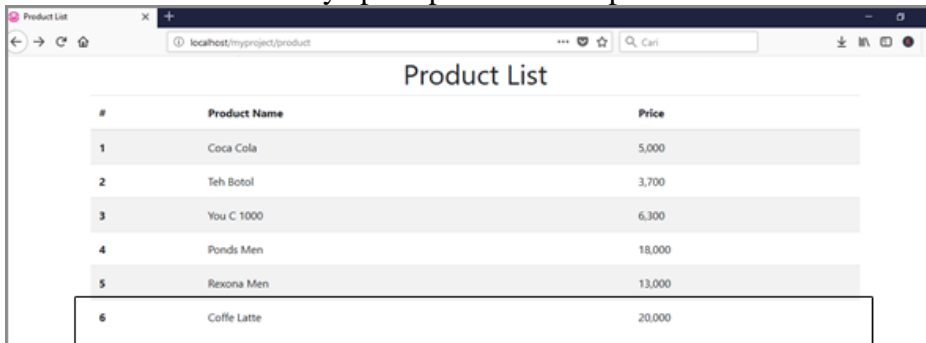
Maka akan terlihat hasilnya seperti berikut:



The screenshot shows a web browser window with the address bar displaying `localhost/myproject/product/add_new`. The page title is "Add New Product". The form contains two text input fields: "Product Name" and "Price". Below the "Price" field is a blue "Submit" button.

Masukkan **product name** dan **price** pada textbox, kemudian klik tombol submit.

Maka akan terlihat datanya pada product list seperti berikut:



The screenshot shows a web browser window with the address bar displaying `localhost/myproject/product`. The page title is "Product List". It displays a table with the following data:

#	Product Name	Price
1	Coca Cola	5,000
2	Teh Botol	3,700
3	You C 1000	6,300
4	Ponds Men	18,000
5	Rexona Men	13,000
6	Coffe Latte	20,000

### Delete data ke database (Delete).

Pada segment kali ini, saya akan menunjukkan kepada Anda bagaimana menghapus (*delete*) data ke database.

1. Buka file view "**product\_view.php**". kemudian ubah menjadi seperti berikut:

```
1      <!DOCTYPE html>
2      <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title>Product List</title>
6          <!-- load bootstrap css file -->
7          <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8      </head>
9      <body>
10
11          <div class="container">
12              <h1><center>Product List</center></h1>
13              <table class="table table-striped">
14                  <thead>
15                      <tr>
16                          <th scope="col">#</th>
17                          <th scope="col">Product Name</th>
18                          <th scope="col">Price</th>
19                          <th width="200">Action</th>
20                      </tr>
21                  </thead>
22                  <?php
23                      $count = 0;
```

```

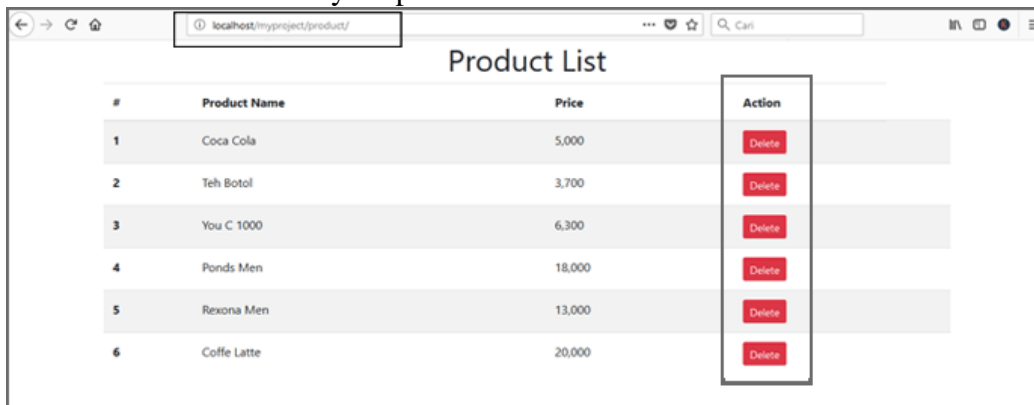
24         foreach ($product->result() as $row) :
25             $count++;
26         ?>
27         <tr>
28             <th scope="row"><?php echo $count;?></th>
29             <td><?php echo $row->product_name;?></td>
30             <td><?php echo number_format($row->product_price);?></td>
31             <td>
32                 <a href="<?php echo site_url('product/delete/'.$row->product_id);?>"
33                     class="btn btn-sm btn-danger">Delete</a>
34             </td>
35         </tr>
36     <?php endforeach;?>
37 </tbody>
38 </table>
39
40 </div>
41
42 <!-- load jquery js file -->
43 <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
44 <!-- load bootstrap js file -->
45 <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
46 </body>
47 </html>

```

Pada file “**product\_view.php**” diatas kita menambahkan satu kolom lagi pada table **product list**. Yaitu kolom **action**. Pada kolom action terdapat tombol **delete**. Sehingga jika Anda jalankan Controller **product**, dengan mengunjungi url berikut:

**<http://localhost/myproject/product/>**

Maka akan terlihat hasilnya seperti berikut:



#	Product Name	Price	Action
1	Coca Cola	5,000	Delete
2	Teh Botol	3,700	Delete
3	You C 1000	6,300	Delete
4	Ponds Men	18,000	Delete
5	Rexona Men	13,000	Delete
6	Coffe Latte	20,000	Delete

## 2. Tambahkan sebuah **function delete** pada controller **Product.php**.

Adapun kodenya sebagai berikut:

```

1 function delete() {
2     $product_id = $this->uri->segment(3);
3     $this->product_model->delete($product_id);
4     redirect('product');
5 }

```

Sehingga terlihat kode lengkap dari controller **Product.php** seperti berikut:

```

1 <?php
2 class Product extends CI_Controller{
3     function __construct() {
4         parent::__construct();

```

```

5         $this->load->model('product_model');
6     }
7     function index() {
8         $data['product'] = $this->product_model->get_product();
9         $this->load->view('product_view', $data);
10    }
11    function add_new() {
12        $this->load->view('add_product_view');
13    }
14    function save() {
15        $product_name = $this->input->post('product_name');
16        $product_price = $this->input->post('product_price');
17        $this->product_model->save($product_name, $product_price);
18        redirect('product');
19    }
20    function delete() {
21        $product_id = $this->uri->segment(3);
22        $this->product_model->delete($product_id);
23        redirect('product');
24    }
25 }

```

### 3. Tambahkan sebuah **function delete** pada mode **Product\_model.php**.

Adapun kodenya sebagai berikut:

```

1    function delete($product_id) {
2        $this->db->where('product_id', $product_id);
3        $this->db->delete('product');
4    }

```

Sehingga terlihat kode lengkap dari model **Product\_model.php** seperti berikut:

```

1    <?php
2    class Product_model extends CI_Model {
3
4        function get_product() {
5            $result = $this->db->get('product');
6            return $result;
7        }
8        function save($product_name, $product_price) {
9            $data = array(
10                'product_name' => $product_name,
11                'product_price' => $product_price
12            );
13            $this->db->insert('product', $data);
14        }
15        function delete($product_id) {
16            $this->db->where('product_id', $product_id);
17            $this->db->delete('product');
18        }
19    }

```

Sekarang kembali ke browser dan kunjungi url berikut:

**<http://localhost/myproject/product>**

Maka akan tampil product list seperti gambar berikut:

#	Product Name	Price	Action
1	Coca Cola	5,000	Delete
2	Teh Botol	3,700	Delete
3	You C 1000	6,300	Delete
4	Ponds Men	18,000	Delete
5	Rexona Men	13,000	Delete
6	Coffe Latte	20,000	Delete

Silahkan klik satu dari tombol **delete** pada kolom **action** untuk menghapus record. Selesai.

## Update data ke database (Update).

Anda telah mengetahui bagaimana menampilkan data (READ) dari database ke view, Meng-insert data ke database (CREATE), dan menghapus data ke database (DELETE). Sekarang waktunya untuk mengetahui bagaimana mengubah data ke database (UPDATE).

1. Buka file view “**product\_view.php**”. kemudian ubah menjadi seperti berikut:

```

1      <!DOCTYPE html>
2      <html lang="en">
3          <head>
4              <meta charset="utf-8">
5              <title>Product List</title>
6              <!-- load bootstrap css file -->
7              <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8          </head>
9          <body>
10
11              <div class="container">
12                  <h1><center>Product List</center></h1>
13                  <table class="table table-striped">
14                      <thead>
15                          <tr>
16                              <th scope="col">#</th>
17                              <th scope="col">Product Name</th>
18                              <th scope="col">Price</th>
19                              <th width="200">Action</th>
20                          </tr>
21                      </thead>
22                      <?php
23                          $count = 0;
24                          foreach ($product->result() as $row) :
25                              $count++;
26                          ?>
27                          <tr>
28                              <th scope="row"><?php echo $count;?></th>
29                              <td><?php echo $row->product_name;?></td>
30                              <td><?php echo number_format($row->product_price);?></td>
31                              <td>
32                                  <a href="<?php echo site_url('product/get_edit/'.$row->product_id);?>"
33                                      class="btn btn-sm btn-info">Update</a>
34                                  <a href="<?php echo site_url('product/delete/'.$row->product_id);?>"
35                                      class="btn btn-sm btn-danger">Delete</a>
36                              </td>

```



```

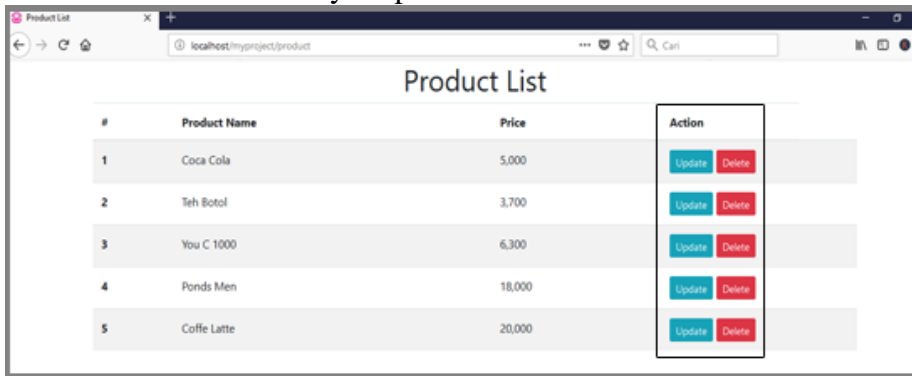
37         </tr>
38         <?php endforeach;?>
39     </tbody>
40 </table>
41
42 </div>
43
44 <!-- load jquery js file -->
45 <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
46 <!-- load bootstrap js file -->
47 <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
48 </body>
49 </html>

```

Pada file “**product\_view.php**” diatas kita menambahkan satu tombol lagi pada kolom **action**. Yaitu tombol **edit**. Sehingga jika Anda jalankan Controller **product**, dengan mengunjungi url berikut:

**<http://localhost/myproject/product/>**

Maka akan terlihat hasilnya seperti berikut:



#	Product Name	Price	Action
1	Coca Cola	5,000	Update Delete
2	Teh Botol	3,700	Update Delete
3	You C 1000	6,300	Update Delete
4	Ponds Men	18,000	Update Delete
5	Coffe Latte	20,000	Update Delete

## 2. Tambahkan sebuah **function get\_edit** pada controller **Product.php**.

Adapun kodenya sebagai berikut:

```

1  function get_edit(){
2      $product_id = $this->uri->segment(3);
3      $result = $this->product_model->get_product_id($product_id);
4      if($result->num_rows() > 0){
5          $i = $result->row_array();
6          $data = array(
7              'product_id'    => $i['product_id'],
8              'product_name'  => $i['product_name'],
9              'product_price' => $i['product_price']
10         );
11         $this->load->view('edit_product_view',$data);
12     }else{
13         echo "Data Was Not Found";
14     }
15 }

```

## 3. Tambahkan sebuah **function get\_product\_id** pada controller **Product\_model.php**.

Adapun kodenya sebagai berikut:

```

1  function get_product_id($product_id){
2      $query = $this->db->get_where('product', array('product_id' => $product_id));

```

```

3         return $query;
4     }

```

4. Buat sebuah view lagi dengan nama **edit\_product\_view.php**. Kemudian ketikkan kode berikut:

```

1     <!DOCTYPE html>
2     <html lang="en">
3         <head>
4             <meta charset="utf-8">
5             <title>Edit Product</title>
6             <!-- load bootstrap css file -->
7             <link href="<?php echo base_url('assets/css/bootstrap.min.css');?>" rel="stylesheet">
8         </head>
9         <body>
10
11             <div class="container">
12                 <h1><center>Edit Product</center></h1>
13                 <div class="col-md-6 offset-md-3">
14                     <form action="<?php echo site_url('product/update');?>" method="post">
15                         <div class="form-group">
16                             <label>Product Name</label>
17                             <input type="text" class="form-control" name="product_name"
18                                 value="<?php echo $product_name;?>" placeholder="Product Name">
19                         </div>
20                         <div class="form-group">
21                             <label>Price</label>
22                             <input type="text" class="form-control" name="product_price"
23                                 value="<?php echo $product_price;?>" placeholder="Price">
24                         </div>
25                         <input type="hidden" name="product_id" value="<?php echo $product_id;?>">
26                         <button type="submit" class="btn btn-primary">Update</button>
27                     </form>
28                 </div>
29             </div>
30
31             <!-- load jquery js file -->
32             <script src="<?php echo base_url('assets/js/jquery.min.js');?>"></script>
33             <!-- load bootstrap js file -->
34             <script src="<?php echo base_url('assets/js/bootstrap.min.js');?>"></script>
35         </body>
36     </html>

```

5. Tambahkan sebuah **function update** pada controller **Product.php**.

Adapun kodenya sebagai berikut:

```

1     function update() {
2         $product_id = $this->input->post('product_id');
3         $product_name = $this->input->post('product_name');
4         $product_price = $this->input->post('product_price');
5         $this->product_model->update($product_id, $product_name, $product_price);
6         redirect('product');
7     }

```

Sehingga terlihat kode lengkap dari controller **Product.php** seperti berikut:

```

1     <?php
2     class Product extends CI_Controller{
3         function __construct(){
4             parent::__construct();
5             $this->load->model('product_model');
6         }
7         function index(){
8             $data['product'] = $this->product_model->get_product();
9             $this->load->view('product_view', $data);

```

```

10     }
11     function add_new(){
12         $this->load->view('add_product_view');
13     }
14     function save(){
15         $product_name = $this->input->post('product_name');
16         $product_price = $this->input->post('product_price');
17         $this->product_model->save($product_name,$product_price);
18         redirect('product');
19     }
20     function delete(){
21         $product_id = $this->uri->segment(3);
22         $this->product_model->delete($product_id);
23         redirect('product');
24     }
25     function get_edit(){
26         $product_id = $this->uri->segment(3);
27         $result = $this->product_model->get_product_id($product_id);
28         if($result->num_rows() > 0){
29             $i = $result->row_array();
30             $data = array(
31                 'product_id' => $i['product_id'],
32                 'product_name' => $i['product_name'],
33                 'product_price' => $i['product_price']
34             );
35             $this->load->view('edit_product_view',$data);
36         }else{
37             echo "Data Was Not Found";
38         }
39     }
40     function update(){
41         $product_id = $this->input->post('product_id');
42         $product_name = $this->input->post('product_name');
43         $product_price = $this->input->post('product_price');
44         $this->product_model->update($product_id,$product_name,$product_price);
45         redirect('product');
46     }
47 }

```

## 6. Tambahkan sebuah function update pada model **Product\_model.php**.

Adapun kodenya sebagai berikut:

```

1     function update($product_id,$product_name,$product_price){
2         $data = array(
3             'product_name' => $product_name,
4             'product_price' => $product_price
5         );
6         $this->db->where('product_id', $product_id);
7         $this->db->update('product', $data);
8     }

```

Sehingga terlihat kode lengkap dari model **Product\_model.php** seperti berikut:

```

1     <?php
2     class Product_model extends CI_Model{
3
4         function get_product(){
5             $result = $this->db->get('product');
6             return $result;
7         }
8         function save($product_name,$product_price){
9             $data = array(
10                 'product_name' => $product_name,
11                 'product_price' => $product_price
12             );
13             $this->db->insert('product',$data);

```

```

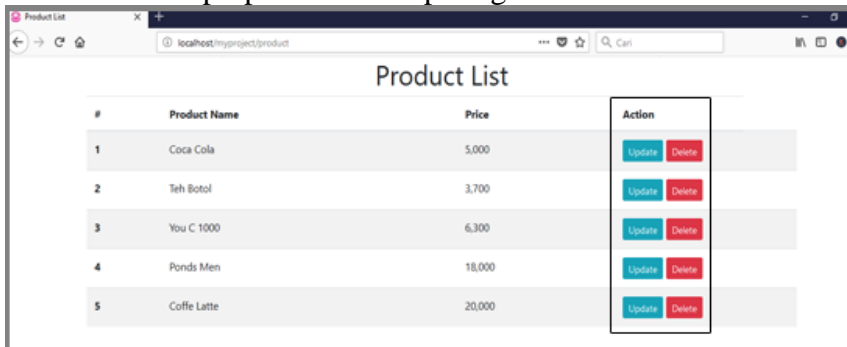
14     }
15     function delete($product_id){
16         $this->db->where('product_id', $product_id);
17         $this->db->delete('product');
18     }
19     function get_product_id($product_id){
20         $query = $this->db->get_where('product', array('product_id' => $product_id));
21         return $query;
22     }
23     function update($product_id,$product_name,$product_price){
24         $data = array(
25             'product_name' => $product_name,
26             'product_price' => $product_price
27         );
28         $this->db->where('product_id', $product_id);
29         $this->db->update('product', $data);
30     }
31 }

```

Sekarang kembali ke browser dan kunjungi url berikut:

**<http://localhost/myproject/product>**

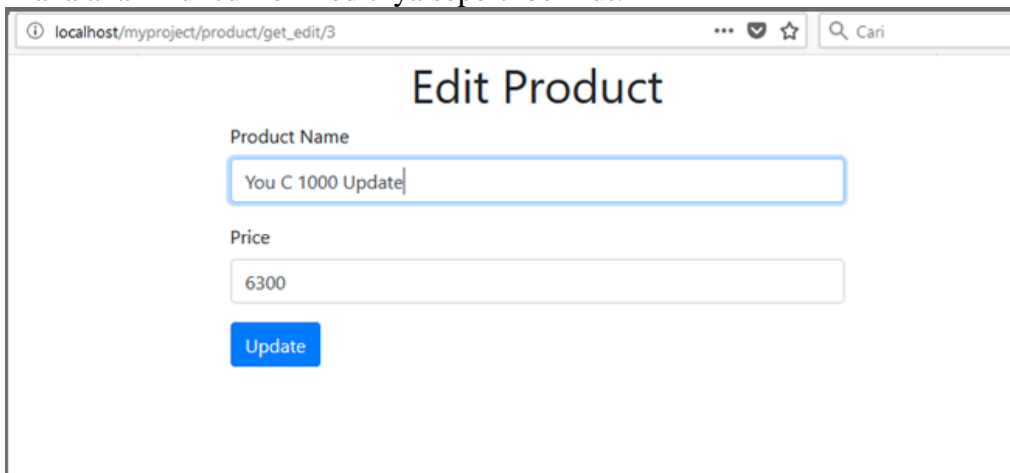
Maka akan tampil product list seperti gambar berikut:



#	Product Name	Price	Action
1	Coca Cola	5,000	Update Delete
2	Teh Botol	3,700	Update Delete
3	You C 1000	6,300	Update Delete
4	Ponds Men	18,000	Update Delete
5	Coffe Latte	20,000	Update Delete

Silahkan klik satu dari tombol **edit** pada kolom **action** untuk mengupdate record.

Maka akan muncul form editnya seperti berikut:

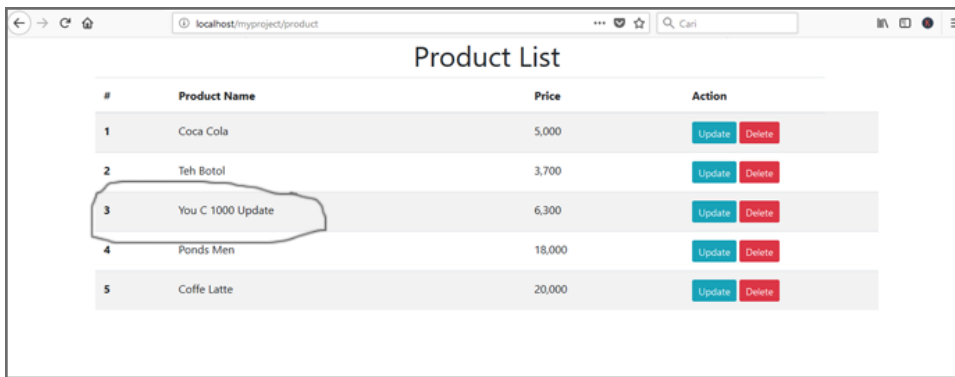


### Edit Product

Product Name

Price

Klik tombol update, maka record akan terupdate.



Product List

#	Product Name	Price	Action
1	Coca Cola	5,000	<a href="#">Update</a> <a href="#">Delete</a>
2	Teh Botol	3,700	<a href="#">Update</a> <a href="#">Delete</a>
3	You C 1000 Update	6,300	<a href="#">Update</a> <a href="#">Delete</a>
4	Ponds Men	18,000	<a href="#">Update</a> <a href="#">Delete</a>
5	Coffe Latte	20,000	<a href="#">Update</a> <a href="#">Delete</a>



Selesai.

**7 Tugas dan Pertanyaan :**

**8 Pustaka :**

1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011.
2. Tutorialspoint. Codeigniter. 2015. [www.tutorialspoint.com](http://www.tutorialspoint.com)

**9 Hasil Praktikum : Laporan Praktikum**

	<p align="center"><b>BUKU PANDUAN PRAKTIKUM POLITEKNIK NEGERI LAMPUNG</b></p>		
<b>Kode : PMI 1412</b>	<b>Tanggal: November 2019</b>	<b>Revisi: 0</b>	<b>Halaman : 78 dari 78</b>

**BUKU PANDUAN PRAKTIKUM  
(BPP)**

---

<b>Minggu Ke</b>	:	15
<b>Capaian Pembelajaran</b>	:	Studi Kasus
<b>Waktu</b>	:	2 x 170 menit
<b>Tempat</b>	:	Laboratorium

---

- 
- |                                    |   |   |
|------------------------------------|---|---|
| <b>1. Sub Capaian Pembelajaran</b> | : | Studi Kasus   |
| <b>2. Indikator Kinerja</b>        | : | Studi Kasus   |
| <b>3. Teori</b>                    | : | —   |
| <b>4. Bahan dan Alat</b>           | : | Komputer dan Logbook  |
| <b>5. Organisasi</b>               | : | Setiap mahasiswa dibimbing oleh dosen dan teknisi pengasuh matakuliah   |
| <b>6. Prosedur Kerja</b>           | : | Kerjakan kasus yang diberikan sesuai dengan contoh teori dan praktik yang sudah diberikan   |
| <b>7. Tugas dan Pertanyaan</b>     | : | Buatlah aplikasi untuk manajemen surat.   |
| <b>8. Pustaka</b>                  | : | 1. Daqiqil, Ibnu. Framework Codeigniter : Sebuah Panduan dan Best Practice. 2011. <a href="http://www.koder.web.id">www.koder.web.id</a><br>2. Tutorislpoint. Codeigniter. 2015. <a href="http://www.tutorialspoint.com">www.tutorialspoint.com</a> |
| <b>9. Hasil Praktikum</b>          | : | <b>Laporan Praktikum</b>  |