MODUL I

Dasar-dasar PHP

Tujuan :

1. Mahasiswa mampu mengetahui perintah dasar PHP
2. Mahasiswa mampu membuat halaman web sederhana dengan PHP

Pada dasar-dasar PHP ini akan dijabarkan tentang penulisan komentar, tipe data, konstanta, variabel dan operator.

Komentar

Komentar adalah bagian dari program yang berfungsi sebagai penjelas atau pemberi keterangan dalam program. Komentar ini tidak akan dieksekusi/dikerjakan oleh interpreter.

Untuk mendefinisikan komentar dipergunakan simbol-simbol karakter berikut :

1. dengan symbol **dobel-slash**(//), biasanya untuk komentar satu baris
   Contoh :

```
<?php
        // nama program : komentar1.php
        // dibuat tanggal : 3 Juni 2004
        echo "Contoh Komentar dengan '//' ";
?>
```

2. diawali dengan symbol **slash-asterik** (/*) dan ditutup dengan **asterik-slash**(*/), biasanya digunakan untuk memberikan komentar lebih dari satu baris.
   Contoh :

```
<?php
        /*
        nama program : komentar2.php
        dibuat tanggal : 3 Juni 2004
        */
        echo "Contoh Komentar dengan '/*' dan '*/'  ";
?>
```

baris komentar tidak ditampilkan di halaman webbrowser karena komentar akan diabaikan oleh interpreter.

Tipe Data

Tipe data dasar PHP terdiri dari
- **integer**, termasuk jenis data bilangan bulat
- **double**, termasuk jenis data bilangan pecahan/desimal
- **string**, termasuk jenis data teks/untaian karakter

Contoh :

```
<?php
        $a=10; //variable $a memiliki tipe data integer
        echo $a;
        $b=22.33; //variable $b memiliki tipe data double
```

```
        echo $b;
        $c=”Skrip PHP”; //variable $c memiliki tipe data string
        echo $c;
?>
```

Variabel

Variabel adalah suatu pengenal dalam program yang berfungsi untuk menyimpan nilai secara sementara dan dapat diubah-ubah nilai.

Untuk mendefinisikan variable, diawali dengan simbol karakter **dollar**(‘**$**’) dan diikuti oleh **nama pengenal**.
$NamaPengenal = nilai;

Adapun aturan dalam menyusun pengenal :

1. tersusun dari karakter huruf, angka dan underscore(_)
2. tidak boleh mengandung spasi
3. karakter pertama nama pengenal harus dari karakter huruf atau underscore.
4. huruf kecil dan besar dibedakan

Dalam PHP, tidak diperlukan pendeklarasian variabel dengan tipe datanya seperti bahasa pemrograman pascal. Setiap variable yang terbentuk dalam program dianggap bertipe variant, dengan kata lain dapat menampung tipe data dengan jenis apapun.

Contoh :

```
<?php
        $info=10; //variable $info menampung bilangan bulat
        echo $info;
        $info=22.33; //variable $info menampung bilangan pecahan
        echo $info;
        $info=”Skrip PHP”; //variable $info menampung data teks/string
        echo $info;
?>
```

Konstanta

Konstanta adalah suatu tetapan nilai dalam program. Konstanta tidak dapat dirubah nilai sewaktu program dijalankan, kalau hal itu dilakukan akan menyebabkan error.
Untuk mendefinisikan konstanta digunakan :

```
        define (NamaPengenal, nilai_konstanta);
```

Contoh :

```
<?php
        // konstanta Judul=”Hitung Luas Lingkaran”
        define (“Judul”, “Hitung Luas Lingkaran”);

        // konstanta PHI=3.14
        define (“PHI”, 3.14);

        echo Judul;
        $r=10;
```

```
    echo "<BR>Jari-jari : $r<BR>\n";

    $luas=PHI * $r * $r;
    echo "Luas Lingkaran = $luas";
?>
```

Operator

Operator adalah suatu symbol yang berfungsi untuk menyusun sebuah ekspresi maupun operasi. Sedangkan yang dioperasikan operator disebut dengan operand. Adapun macam-macam operator yaitu :

1. Operator Aritmetika
   Merupakan symbol-simbol operator untuk melakukan operasi matematis.

Operator	Fungsi	Prioritas
+	Penjumlahan	Ketiga
-	Pengurangan	Ketiga
*	Perkalian	Kedua
/	Pembagian	Kedua
%	Sisa Pembagian	Kedua
++	Penaikan	Pertama
--	Penurunan	Pertama

Contoh :

```
<?php
    $bil1 = 200;
    $bil2 = 33;
    $hasil = $bil1 + $bil2;
    echo "$bil1 + $bil2 = $hasil<BR>\n";
    $hasil = $bil1 - $bil2;
    echo "$bil1 - $bil2 = $hasil<BR>\n";
    $hasil = $bil1 * $bil2;
    echo "$bil1 * $bil2 = $hasil<BR>\n";
    $hasil = $bil1 / $bil2;
    echo "$bil1 / $bil2 = $hasil<BR>\n";
    $hasil = $bil1 % $bil2;
    echo "$bil1 % $bil2 = $hasil<BR>\n";
    $hasil = $bil1++;
    echo "$bil1++ = $hasil<BR>\n";
    $hasil = $bil2--;
    echo "$bil2-- = $hasil<BR>\n";
?>
```

2. Operator Pembandingan
   Merupakan simbol-simbol operator untuk melakukan pembandingan antara dua buah operand. Hasil pembandingan bernilai **satu**(1) jika benar dan bernilai **nol**(0) jika salah.

Operator	Fungsi
<	Lebih kecil
>	Lebih besar
<=	Lebih kecil atau sama dengan
>=	Lebih besar atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

Contoh :

```
<?php
        $bil1 = 100;
        $bil2 = 20;
        $teks1 = “PHP”;
        $teks2 = “php”;
        printf(“%d == %d adalah %d<BR>\n”,$bil1, $bil2, $bil1 == $bil2);
        printf(“%d != %d adalah %d<BR>\n”,$bil1, $bil2, $bil1 != $bil2);
        printf(“%d >= %d adalah %d<BR>\n”,$bil1, $bil2, $bil1 >= $bil2);

        printf(“%s == %s adalah %d<BR>\n”,$teks1, $teks2, $teks1 == $teks2);
        printf(“%s != %s adalah %d<BR>\n”,$teks1, $teks2, $teks1 != $teks2);
?>
```

3. Operator Logika
   Merupakan symbol-simbol operator untuk menyusun kalimat ekspresi/ungkapan logika. Hasil operasi ini akan didapatkan nilai **satu**(1) jika bernilai benar atau **nol**(0) jika bernilai salah.

Operator	Fungsi
AND atau **&&**	Operasi logika and
OR atau ||	Operasi logika or
XOR	Operasi logika eksklusif or
!	Ingkaran/negasi

Untuk lebih jelasnya mengenai penggunaan operator-operator di atas, perhatikan table kebenaran sebagai berikut :

$p	$q	$p and $q	$p or $q	$p xor $q	! ($p and $q)
1	1	1	1	0	0
1	0	0	1	1	1
0	1	0	1	1	1
0	0	0	0	0	1

Contoh :

```
<?php
        $bil1 = 100;
        $bil2 = 20;
```

```
        $teks1 = “PHP”;
        $teks2 = “php”;
        $hasil = ($bil1 <> $bil2) or ($teks1 == $teks2);
        printf(“(%d <> %d) or (%s == %s) adalah %d<BR>\n”,
        $bil1, $bil2, $teks1, $teks2, $hasil);
        $hasil = ! ($teks1 == $teks2);
        printf(“! (%s == %s) adalah %d<BR>\n”,$teks1, $teks2, $hasil);
?>
```

4. Operator String

Dalam PHP juga tersedia operator string, yaitu untuk operasi penggabungan teks. Adapun symbol yang digunakan yaitu berupa karakter **titik/point (.)**.

Contoh :

```
<?php
        $teks1 = “Aku Sedang Belajar”;
        $teks2 = “Pemrograman WEB”;
        $teks3 = “PHP 4”;
        $hasil = $teks1 . $teks2 . $teks3;
        printf(“hasil : %s<BR>\n”,$hasil);
        $hasil = $teks1 .  “ “ . $teks2 .  “ “ . $teks3;
        printf(“hasil : %s<BR>\n”,$hasil);
?>
```

Array

Array adalah variabel khusus, yang dapat menampung lebih dari satu nilai pada suatu waktu.

Jika Anda memiliki daftar item (daftar nama mobil, misalnya), menyimpan mobil dalam variabel tunggal dapat terlihat seperti ini:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

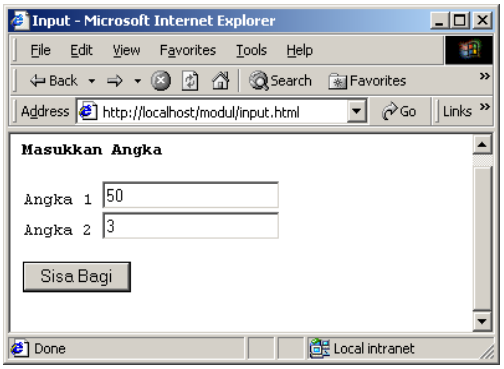
Array dapat menyimpan banyak nilai dengan satu nama, dan Anda dapat mengakses nilai dengan merujuk ke nomor indeks.

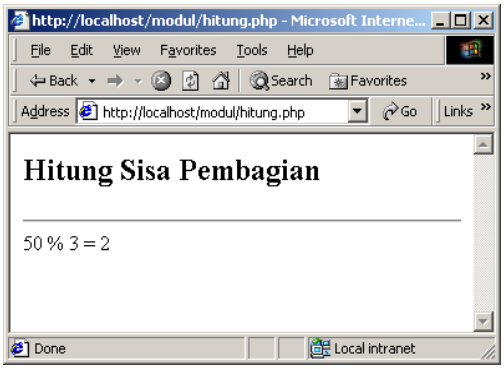
Contoh :

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>

</body>
</html>
```

Tugas :

Buatlah program untuk menghitung sisa pembagian antara **angka1** dengan **angka2**.


Input - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/modul/input.html
Go Links
Masukkan Angka
Angka 1 50
Angka 2 3
Sisa Bagi
Done
Local intranet


http://localhost/modul/hitung.php - Microsoft Interne...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/modul/hitung.php
Go Links
Hitung Sisa Pembagian
50 % 3 = 2
Done
Local intranet

MODUL 2
Koneksi Data PHP dengan MySQL

Tujuan :

1. Mahasiswa mampu melakukan koneksi dari php ke MySQL
2. Mahasiswa mampu melakukan insert data ke dalam database MySQL

Untuk melakukan koneksi ke database dapat menggunakan syntak :

```
mysqli_connect($servername, $username, $password);
```

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Latihan :

1. Buat sebuah database dengan nama : **unjani**
2. BUat script untuk koneksi ke database mysql

   Nama file : koneksi.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "unjani";

// buat koneksi
$conn = mysqli_connect($servername, $username, $password, $dbname);
// cek koneksi
if (!$conn) {
   echo "Connection gagal";
}
else
{
echo "koneksi berhasil";
}
```

```
?>
```

3. Buat table mahasiswa yang berisikan field : nim, nama dan email
4. Buat script untuk menambahkan data ke table mahasiswa
   Nama file : tambahdata.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "unjani";

// buat koneksi
$conn = mysqli_connect($servername, $username, $password, $dbname);
// cek koneksi
if (!$conn) {
   echo "Connection gagal";
}

$sql = "INSERT INTO mahasiswa (nim, nama, email)
VALUES ('4121221', 'Dimas Prikitiw', 'dimas@unjani.com')";

if (mysqli_query($conn, $sql)) {
   echo "data berhasil ditambah";
} else {
   echo "Gagal";
}

mysqli_close($conn);
?>
```

TUGAS

1. **modifikasi program tambah data diatas, dimana inputan nim, nama dan email harus diinputkan dari form. Isian form tadi akan diolah untuk dimasukkan kedalam table mahasiswa**
2. Diketahui sebuah tabel tamu dalam Database MySQL adalah sebagai berikut :

Nama Field	Tipe Data	Panjang	Keterangan
idtamu	int		identitas tamu, sebagai field kunci primer dan AUTO_INCREMENT
nmtamu	varchar	35	untuk mengisikan data nama tamu

email	varchar	40	untuk mengisikan data alamat email tamu

Buatlah form entry data tamu dengan menggunakan pemrograman PHP!

MODUL 3
Koneksi Data PHP dengan MySQL (Tampil data)

Tujuan :

1. Mahasiswa mampu melakukan edit dan delete data pada database menggunakan PHP
2. Mahasiswa mampu melakukan CRUD (Create, Read, Update, Delete) dalam database dengan PHP

Pendahuluan :
Berikut adalah contoh script untuk menampilkan data dari database MySQL dengan menggunakan PHP.

1. Tampil.php

```
<?php
include "koneksi.php";
$sql = "SELECT * FROM mahasiswa";
$result = mysqli_query($conn, $sql);


   while($row = mysqli_fetch_assoc($result)) {
      echo "nim: " . $row["nim"]. " - Nama: " . $row["nama"]. " " .
$row["email"]. "<br>";
   }

mysqli_close($conn);
?>
```

2. Modifikasi source code diatas, sehingga tampilan datanya ada dalam bentuk table seperti tabel dibawah ini :

Data Mahasiswa			
No	**Nim**	**Nama**	**Email**
1	3400012323	Didin	didin@yahoo.com
2	3443334343	Dudin	dudin@yahoo.com

3. Modifikasi lagi script pada no 3 sehingga bentuknya seperti dibawah ini

Data Mahasiswa					
No	**Nim**	**Nama**	**Email**	**Aksi**	
1	3400012323	Didin	didin@yahoo.com	Delete	Edit
2	3443334343	Dudin	dudin@yahoo.com	Delete	Edit

4. Buat sebuah homepage yang menarik, dimana terdapat link ke halaman untuk insert data dan ke halaman tampil data
5. Tampil.php

Data Mahasiswa					
No	**Nim**	**Nama**	**Email**	**Aksi**	
1	3400012323	Didin	didin@yahoo.com	Delete	Edit
2	3443334343	Dudin	dudin@yahoo.com	Delete	Edit

- Jika Tombol delete diklik, maka akan menghapus data baris yang bersangkutan, dan halaman akan langsung kembali ke halaman tampil.php (memakai header location)
- Jika edit di klik maka akan tampil halaman edit data yang berisi form yang sudah terisi data nim, nama dan email. Dan jika tombol update di klik, akan menyimpan hasil perubahan, dan halaman akan diarahkan kembali ke tampil.php

6. Buat sebuah tabel dengan nama tabel **user** dengan field : **username**, **password**
7. Isi data secara manual melalui phpmyadmin untuk mengisi username dan password
8. Buatlah sebuah halaman login untuk memasukkan username dan password. Jika username dan password terdapat pada tabel user, maka buatlah sebuah session dan halaman akan diarahken ke tampil.php. Jika gagal, tampilkan pesan, “login gagal”
9. Modifikasi halaman tampil.php, sehingga hanya bisa diakses jika sudah login.
10. Buat halaman untuk menangani insert data ke tabel user. Halaman ini hanya bisa diakses jika sudah login.

Keterangan :
<a href="edit.php?nim=<?php echo $row['nim']; ?>">Edit : contoh untuk link edit

MODUL 4

Pengenalan Codeigniter dan MVC

Tujuan :

1. Mahasiswa dapat memahami Konsep MVC
2. Mahasiswa dapat menampilkan halaman web menggunakan CI
3. Mahasiswa dapat menampilkan data hasil parsing dari controller ke view.

4.1 Pendahuluan

4.1.1 MVC

Model-View-Controller (MVC) merupakan suatu konsep yang cukup popular dalam pembangunan aplikasi *web*. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu:

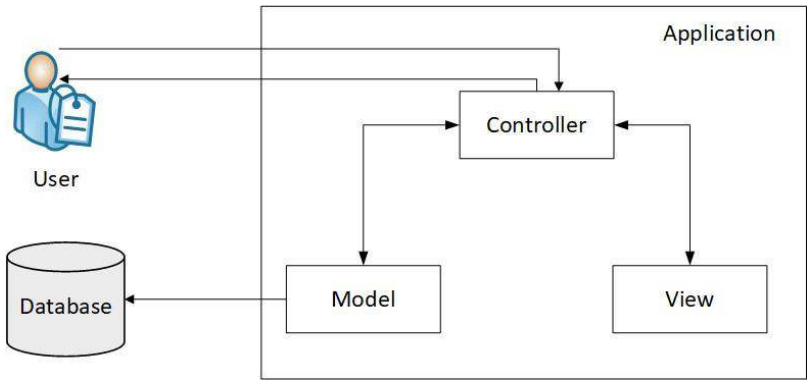
View, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web* bagian ini biasanya berupa *file template* HTML, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian *model*.

Model, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert*, *update*, *delete*, *search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.

Controller, merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Singkat kata ***Model*** untuk mengatur alur *database*, ***View*** untuk menampilkan *web*, sedangkan ***Controller*** untuk mengatur alur kerja antara *Model* dan *View*. Jadi misalnya Anda akan membuat akun *e-mail*. Pertama anda akan melihat tampilan *sign-up / register*, itulah yang disebut dengan *View*. Kemudian Anda mengisi *form username*, *password*, dan lain-lain dan Anda klik tombol *Register*, maka disinilah *View* akan memanggil *Controller* dan *Controller* memanggil *Model*. Adapun tugas *Model* disini untuk mengecek apakah Anda

sudah mengisi sesuai dengan kriteria dan akan dihubungkan dengan *database*. Kemudian *Model* akan mengembalikan ke *Controller* dan *Controller* akan mengembalikan ke *View*. Berikut adalah gambaran konsep MVC yang diterapkan pada Codeigniter.



Cara Kerja MVC

4.1.2 Framework PHP Codeigniter

Framework atau dalam Bahasa Indonesia dapat diartikan sebagai "kerangka kerja" merupakan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang *programmer*, tanpa harus membuat fungsi atau *class* dari awal.

Alasan mengapa menggunakan *Framework*

- Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
- Relatif memudahkan dalam proses *maintenance* karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat *programmer* mengikuti pola standar yang ada).
- Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, *pagination*, *multiple database*, *scaffolding*, pengaturan *session*, *error handling*, dll).
- Lebih bebas dalam pengembangan jika dibandingkan CMS.

CodeIgniter adalah sebuah *web application network* yang bersifat *open source* yang digunakan untuk membangun aplikasi php dinamis. CodeIgniter menjadi sebuah *framework* PHP dengan model MVC (*Model*, *View*, *Controller*) untuk membangun *website* dinamis dengan menggunakan PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi *web*. Selain ringan dan cepat, CodeIgniter juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah

yang menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai *framework* pilihannya. Karena kelebihan-kelebihan yang dimiliki oleh CodeIgniter, pembuat PHP Rasmus Lerdorf memuji CodeIgniter di frOSCon (Agustus 2008) dengan mengatakan bahwa dia menyukai CodeIgniter karena "*it is faster, lighter and the least like a framework*."

CodeIgniter pertamakali dikembangkan pada tahun 2006 oleh Rick Ellis. Dengan logo api yang menyala, CodeIgniter dengan cepat "membakar" semangat para *web developer* untuk mengembangkan *web* dinamis dengan cepat dan mudah menggunakan *framework* PHP yang satu ini.

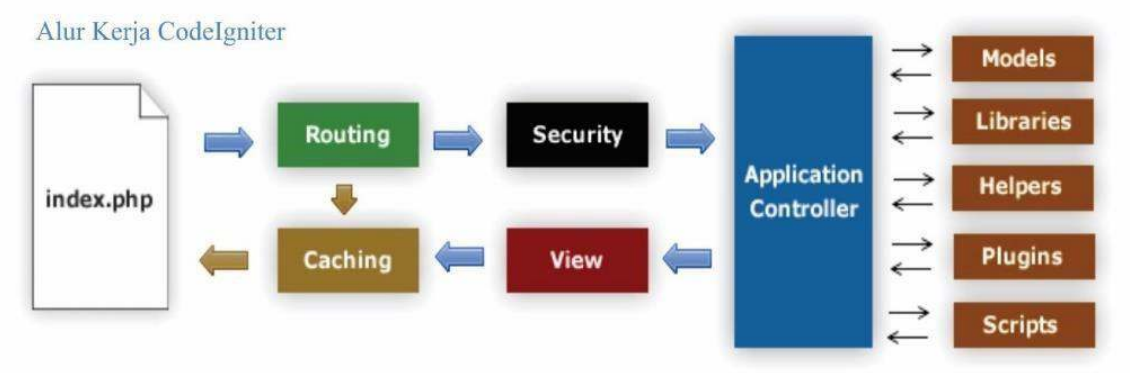
Sebagai web *framework* populer yang menggunakan bahasa pemrograman PHP, Codeigniter mempunyai beberapa keunggulan yaitu:

1. *Free*, karena berada di bawah lisensi open source, kita dapat melakukan apa pun dengan Codeigniter.
2. *Light weight*, sistem inti Codeigniter memerlukan *library* yang sedikit. Sangat berbeda dengan *framework* lainnya yang membutuhkan banyak sumber daya tambahan. *Library* tambahan akan digunakan ketika *request* secara dinamis, membuat sistem yang dibagun menjadi efisien dan cukup cepat.
3. *Fast*, performa yang dimiliki Codeigniter terbukti cepat setelah dibandingkan dengan *framework* lainnya.
4. Menggunakan kaidah MVC, dengan menggunakan *Model-View-Controller*, kita dapat memisahkan bagian *logic* dan *presentation* dari aplikasi yang kita bagun.
5. Menghasilkan URL yang bersih. URL yang dihasilkan oleh Codeigniter bersih dan ramah terhadap *search engine*. Codeigniter menggunakan pendekatan *segment-based* dibandingkan dengan *query string* yang biasa digunakan oleh programmer yang tidak menggunkan *web framework*.
6. *Packs a Punch*, Codeigniter hadir dengan berbagai *library* yang akan membantu tugas-tugas di pengembangan web yang sudah umum dan sering dilakukan, seperti mengakses *database*, mengirim email, validasi data dari form, mengelola *session*, memanipulasi gambar, dan masih banyak lagi.

7. *Extensible*, kita dapat menambahkan *library* atau *helper* yang kita ciptakan sendiri ke dalam Codeigniter. Selain itu, kita dapat juga menambahkan fitur lewat *class extension* atau *system hooks*.
8. *Throughly Documented*, hampir semua fitur, *library*, dan *helper* yang ada di Codeigniter telah terdokumentasi dengan lengkap dan tersusun dengan baik. Dokumentasi cara penggunaannya dapat dilihat di https://www.codeigniter.com/user_guide/index.html.

Berikut adalah istilah yang sering ditemui di Codeigniter.

1. *Model*, *class* PHP yang dirancang untuk bekerja dengan informasi dari *database*.
2. *Controller*, inti aplikasi yang menentukan penanganan HTTP *request*.
3. *View*, halaman web seperti *header*, *footer*, *sidebar*, dan sebagainya yang ditanamkan di halaman *web*. *View* tidak pernah dipanggil secara langsung, tetapi harus dipanggil dari *controller*.
4. *Library*, *class* yang berisi fungsi-fungsi untuk penyelesaian kasus tertentu.
5. *Helper*, pembantu tugas untuk kategori tertentu yang terdiri atas kumpulan fungsi.
6. *Driver*, *library* khusus yang mempunyai *class* induk dan beberapa *class* turunan yang dapat digunakan untuk kasus tertentu.



Alur kerja Codeigniter

a. Index.php: Index.php disini berfungsi sebagai *file* pertama dalam program yang akan dibaca oleh program.
b. *The Router*: *Router* akan memeriksa HTTP *request* untuk menentukan hal apa yang harus dilakukan oleh program.

c. *Cache File*: Apabila dalam program sudah terdapat "*cache file*" maka *file* tersebut akan langsung dikirim ke *browser*. *File cache* inilah yang dapat membuat sebuah *website* dapat di buka dengan lebih cepat. *Cache file* dapat melewati proses yang sebenarnya harus dilakukan oleh program CodeIgniter.

d. *Security*: Sebelum *file controller* di *load* keseluruhan, HTTP *request* dan data yang di-*submit* oleh *user* akan disaring terlebih dahulu melalui fasilitas *security* yang dimiliki oleh CodeIgniter.

e. *Controller*: *Controller* akan membuka *file model*, *core libraries*, *helper* dan semua *resources* yang dibutuhkan dalam program tersebut.

f. *View*: Hal yang terakhir akan dilakukan adalah membaca semua program yang ada dalam *view file* dan mengirimkannya ke *browser* supaya dapat dilihat. Apabila *file view* sudah ada yang di "*cache*" maka *file view* baru yang belum ter-*cache* akan meng-*update file view* yang sudah ada.

4.2 Latihan 1

4.2.1 Instalasi Codeigniter

Langkah pertama – Download Codeigniter dari situs Codeigniter:

```
http://www.codeigniter.com/download
```

Langkah kedua – Unzip folder.

Setelah memindahkan file codeigniter.X..rar ke folder htdocs, Unzip file rar tersebut.

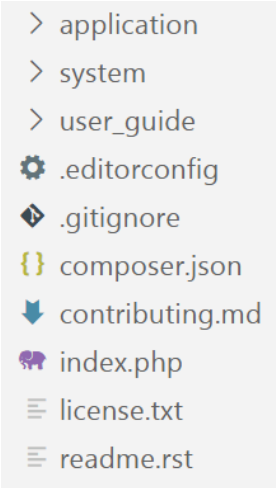
Langkah ketiga – Buka dibrowser dengan alamat url

```
http://localhost/namafoldercodeigniter
```

Jika berhasil maka pada browser akan muncul seperti berikut

4.2.2 Struktur File Codeigniter

Dalam setiap package codeigniter, terdapat code yang tersimpan dalam beberapa folder. Folder dan file yang berada di direktori awal adalah:



application berisi semua kode aplikasi. Di dalam direktori inilah kita akan menulis semua kode aplikasi kita.

system berisi kode-kode inti dari Codeiniter. Jangan mengubah apapun di dalam direktori ini. Jika kita ingin upgrade versi, salah satu caranya adalah me-replace direktori ini dengan yang baru.

user_guide berisi dokumentasi codeigniter berupa html. Kita bisa menghapus direktori ini saat web sudah jadi.

editor_config berisi konfigurasi untuk teks editor.

.gitignore berisi daftar file dan folder yang akan diabaikan oleh Git.

composer.json adalah file yang berisi keterangan project dan keterangan library yang digunakan. File ini dibutuhkan oleh composer.

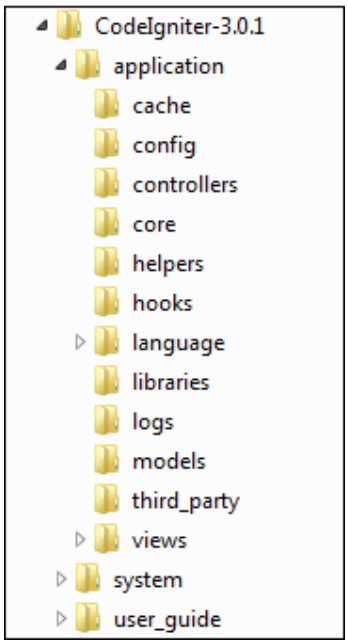
contributing.md adalah file yang berisi penjelasan cara berkontribusi di proyek CI. Kita bisa menghapus file ini, apabila web sudah jadi.

license.txt adalah file yang berisi keterangan lisensi dari CI.

readme.rst sama seperti file _ contributing.md file ini berisi penjelasan dan informasi tentang project CI. Kita juga bisa menghapus file ini saat web sudah selesai.

index.php adalah file utama dari CI. File yang akan dibuka pertamakali saat kita mengakses web.

Lebih detail lagi untuk pada Folder Application, terdapat beberapa sub Folder. Folder yang akan sering digunakan diantaranya adalah:



- **Config** − Folder ini berisi berbagai file untuk mengkonfigurasi aplikasi. Dengan bantuan file config.php, pengguna dapat mengkonfigurasi aplikasi. Menggunakan file database.php, pengguna dapat mengkonfigurasi database aplikasi.
- **Controllers** – Folder ini berisi File Class Controller yang digunakan oleh aplikasi, merupakan folder utama dari aplikasi.
- **Models** – Folder ini berisi File Class Model yang digunakan oleh aplikasi. Umumnya berupa class yang digunakan untuk instruksi menggunakan database.
- **Views** – Folder ini berisi File yang digunakan sebagai representasi tampilan. Umumnya berisi PHP-HTML.
- **Helpers** – Berisi Helper yang digunakan oleh aplikasi.
- **Libraries** – berisi kumpulan library fungsi tambahan yang digunakan oleh aplikasi.
- **Language** – Berisi set Bahasa yang digunakan untuk beberapa file.
- **Cache** − Folder ini berisi semua halaman cache aplikasi. Halaman yang di-cache ini akan meningkatkan kecepatan keseluruhan mengakses halaman.
- **Core** − Berisi basis class dari aplikasi.

4.2.3 Konfigurasi Awal

Hal pertama yang harus dilakukan dalam menggunakan Codeigniter adalah menyesuaikan pengaturan pada file-file pengaturan. File tersebut dapat ditemukan pada folder **application/config**. File yang sering diubah untuk menyesuaikan pengaturan adalah autoload, config, database dan routes. Dalam file konfigurasi tersebut terdapat beberapa variable, baik variable biasa atau berupa array, yang nilainya akan digunakan untuk mengatur jalannya aplikasi.

autoload digunakan untuk mengatur resource apa saja yang akan dijalankan tanpa perlu dipanggil terlebih dahulu dalam controller. Variable yang sering diisi adalah, **library**, **helper**, dan **model**.

config biasanya digunakan untuk konfigurasi umum web/aplikasi. Dalam file ini terdapat variable array yang mempunyai banyak index. Index yang sering diisi adalah, **base_url.** base_url adalah alamat domain utama misalkan http://www.domain.com atau jika localhost, http://localhost/namafolder.

database sesuai namanya file ini digunakan untuk pengaturan dari koneksi ke database meliputi data host sampai ke nama database.

routes digunakan sebagai pointer dari alamat yang bisa diakses dibrowser dan mengarahkannya ke controller yang sesuai. Dalam file ini didefinisikan alamat tersebut, dan controller yang digunakannya. Set awal yang digunakan adalah **default_controller** yang menentukan controller awal yang digunakan saat pertama kali aplikasi dijalankan.

Untuk pertemuan ini, konfigurasi yang dilakukan adalah mengubah base_url pada config.php. Ubah sesuaikan dengan alamat yang digunakan aplikasi di server.

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

4.3 Latihan 2

4.3.1 Membuat Controller

Controller adalah file kelas sederhana. Seperti namanya, itu mengontrol seluruh aplikasi oleh URI.

Pertama, buka folder **Application/Controller**. Anda akan menemukan dua file di sana, index.html dan **Welcome.php**. File-file ini merupakan file bawaan CodeIgniter.

Buat file baru pada directory yang sama, beri nama "**Test.php**". Tulis kode berikut dalam file tersebut:

```
<?php
   class Test extends CI_Controller {
      public function index() {
         echo "Hello World!";
      }
   }
?>
```

Class Test extends kelas built-in yang disebut CI_Controller . Setiap controller yang akan dibuat, harus extends kepada class CI_Controller.

4.3.2 Memanggil Controller

Controller di atas dapat dipanggil oleh URI sebagai berikut –

```
http://www.your-domain.com/index.php/test
```

Perhatikan kata "test " di URI di atas setelah index.php. Ini menunjukkan nama kelas controller. Seperti yang telah kita beri nama controller " Test ", kita menulis " test " setelah index.php. Nama Class harus dimulai dengan huruf besar tetapi kita perlu menulis huruf kecil ketika kita memanggil controller itu dalam URI. Sintaks umum untuk memanggil controller adalah sebagai berikut –

```
http://domain/index.php/nama_controller/nama_method
```

4.3.3 Membuat & Memanggil Method

Mari kita modifikasi kelas di atas dan buat metode lain bernama "hello".

```
<?php
   class Test extends CI_Controller {
      public function index() {
         echo "This is default function. NIM - NAMA";
      }
      public function hello() {
         echo "This is hello function. NIM - NAMA";
      }
   }
?>
```

Kita dapat menjalankan controller di atas dengan tiga cara berikut –

```
http://domain/index.php/test
```

```
http://domain/index.php/test/index
http://domain/index.php/test/hello
```

Pada percobaan pertama, pemanggilan http://domain/index.php/test mempunyai tampilan yang sama dengan http://domain/index.php/test/index. Dalam situasi seperti itu, CodeIgniter memanggil metode standar/default yaitu method " index".

Sedangkan URI ketiga di browser, akan mendapatkan output yang berbeda. Yaitu output dari metode " hello".

- Nama kelas pengontrol harus dimulai dengan huruf besar.
- Pengontrol harus dipanggil dengan huruf kecil.
- Jangan menggunakan nama metode yang sama dengan kelas induk Anda, karena akan mengesampingkan fungsionalitas kelas induk.

4.3.4 Membuat View

View bisa merupakan suatu halaman web yang sederhana atau rumit, yang dapat dipanggil suatu Controller. View tidak bisa dipanggil secara langsung.

Buatlah suatu file Baru dalam folder application/views dengan nama test.php dan salin code berikut:

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>Contoh Halaman View</title>
   </head>
   <body>
      Contoh CodeIgniter View, NIM - NAMA
   </body>
</html>
```

4.3.5 Memanggil View

Untuk menggunakan atau memanggil view pada Controller, dapat menggunakan Syntax berikut:

```
$this->load->view('name');
```

Atau jika dalam file view terdapat dalam subfolder dalam Folder views,

```
$this->load->view('directory-name/name');
```

misalkan application/views/home/test.php akan menjadi,

```
$this->load->view('home/test');
```

Pada Controller test.php, application/controllers/test.php, ubahlah badan dari method index menjadi seperti berikut:

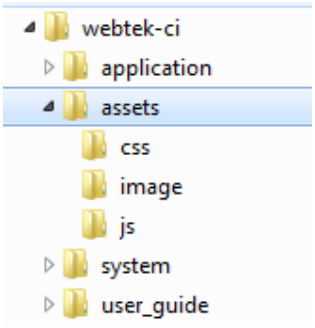
```
<?php
   class Test extends CI_Controller {
      public function index() {
         $this->load->view('test');
      }
   }
?>
```

4.4 Latihan 3

4.4.1 Penggunaan Resource External

Resource dan file lain yang digunakan dalam halaman web seperti gambar, multimedia, css, dan javascript tidak bisa langsung digunakan layaknya penggunaan tanpa framework. Meskipun disimpan dalam satu folder dengan View, resource tersebut tetap harus digunakan menggunakan directory lengkap. Oleh karena itu, biasanya resource external disimpan dalam satu folder yang disimpan pada folder utama codeigniter. Satu level dengan folder Application, System, dll. Nama folder umumnya menggunakan nama "Assets".

Buatlah folder dalam folder codeigniter, beri nama **"Assets"**. Kemudian buat folder di dalamnya, folder **image, css, dan js.** Sehingga direktori file menjadi seperti berikut.



Sesuai dengan nama, folder css akan digunakan untuk menyimpan file css. Folder image akan digunakan untuk menyimpan gambar yang digunakan oleh aplikasi. Dan folder js akan digunakan untuk menyimpan file javascript.

Tambahkan pada folder image, suatu gambar. Misalkan Koala.jpg

Buatlah dalam folder **views (**application/view) file baru, dan beri nama **profile.php**. Kemudian salin code berikut (nama folder ci disesuaikan).

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <title>Profil</title>
</head>
<body>
    <h1>NIM - NAMA</h1>
    <img src="http://localhost/folderci/assets/image/Koala.jpg">
</body>
</html>
```

Masih melanjutkan dari controller Test.php (application/controllers/Test.php), tambahkanlah suatu fungsi profil() yang akan memanggil view yang sudah dibuat di atas. Sehingga code Test.php akan menjadi seperti berikut.

```
<?php
class Test extends CI_Controller {
    /* ..... */
    //Code sebelumnya
    public function profil(){
        $this->load->view("profile");
    }
}
?>
```

Fungsi profil dapat dilihat menggunakan alamat URL berikut (nama folder disesuaikan).

```
http://localhost/folderci/index.php/test/profil
```

Untuk mempermudah dalam penggunaan URL, dapat menggunakan helper “URL” dan fungsi base_url. Dimana fungsi ini akan mengembalikan nilai yang ada pada variable config dengan index base_url pada file config.php.

Panggil Helper URL pada controller, ubah code fungsi profil() pada Test.php sehingga menjadi seperti berikut.

```
<?php
class Test extends CI_Controller {
    /* Code sebelumnya */
    public function profil(){
        $this->load->helper("url");
        $this->load->view("profile");
    }
}
?>
```

Dalam view profile.php, ubah href dari link gambar menjadi seperti berikut.

```
    <img src="<?php echo base_url('assets/image/Koala.jpg');?>">
```

4.5 Latihan 4

4.5.1 Parsing data dari Controller ke View

Buatlah suatu controller baru, buat file dalam folder Controllers (application/controllers) dan beri nama **Calc.php** buatlah dua fungsi, index(), dan hitung().

```
<?php
class Calc extends CI_Controller{
    public function index(){

    }
    public function hitung(){

    }
}
?>
```

Buatlah suatu view baru, buat file dalam folder View (application/views) dan beri nama **calculator_form.php** dan salin code berikut ke file tersebut.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Calculator</title>
</head>
<body>
    <h2>Tambah Bilangan</h2>
    <form action="<?php echo site_url('calc/hitung'); ?>" method="POST">
    <table>
        <tr>
            <td>Bilangan 1</td><td>: <input type="text" name="bil1"></td>
        </tr>
        <tr>
            <td>Bilangan 2</td><td>: <input type="text" name="bil2"</td>
        </tr>
        <tr>
            <td colspan="2">
            <input type="submit" name="submit" value="Tambah"></td>
        </tr>
```

```
    </table>
    <h2>Hasil Tambah</h2>
</form>
</body>
</html>
```

Panggil view calculator_form diatas di fungsi index() pada Controller Calculator. Sehingga menjadi seperti berikut:

```
<?php
class Calc extends CI_Controller{
    public function index(){
        $this->load->helper("url");
        $this->load->view("calculator_form");
    }
/****************** code selanjutnya *****************/
}
?>
```

Penggunaan site_url() pada view calculator form hampir sama dengan base_url() pada latihan sebelumnya. Jika base_url akan mengembalikan nilai base_url pada file config, site_url akan mengembalikan nilai base_url pada file config ditambah index.php sehingga keluarannya adalah **http://localhost/folderci/index.php**. Parameter yang digunakan dalam site_url sama seperti pada base_url akan menambahkan keluaran sesuai parameter yang digunakan. Sehingga

```
site_url('calc/hitung')
```

akan mengembalikan nilai

```
http://localhost/folderci/index.php/calc/hitung
```

Selanjutnya adalah membuat proses hitung pada fungsi hitung(). Tambahkan code berikut pada Calc.php pada fungsi hitung().

```
<?php
class Calc extends CI_Controller{
    /* ---------------code sebelumnya-------------- */
    public function hitung(){
        $x = $this->input->post('bil1');
        $y = $this->input->post('bil2');
        $data['hasil'] = $x+$y;
        $this->load->view("calculator_form",$data);
    }
```

```
}
?>
```

Fungsi hitung akan menangkap data yang dikirim sebelumnya melalui method post, ini dapat dilihat dari code **`$this->input->post()`**, dimana parameter merupakan atribut **name** dari komponen input form pada page sebelumnya.

Data dikirimkan dari Controller ke view melalui suatu **array** atau **object**, yaitu dengan cara memasukan pada parameter kedua pada method **`load->view().`** Pada bagian view sendiri, nilai yang dikirim dari Controller menggunakan variable yang merupakan index dari Array atau object yang dikirimkan dari Controller. Sehingga pada Controller array yang digunakan adalah **`$data['hasil']`** maka pada view nilai tersebut akan didapat melalui **`$hasil`**.

Tambahkan code berikut pada view calculator_form.php.

```
<html lang="en">
<!-- -----------------code sebelumnya --------------- -->
    <h2>Hasil Tambah</h2>
    <?php
    if(isset($hasil)){
        echo $hasil;
    }
    ?>
</form>
</body>
</html>
```

Buka pada browser dengan menggunakan alamat berikut

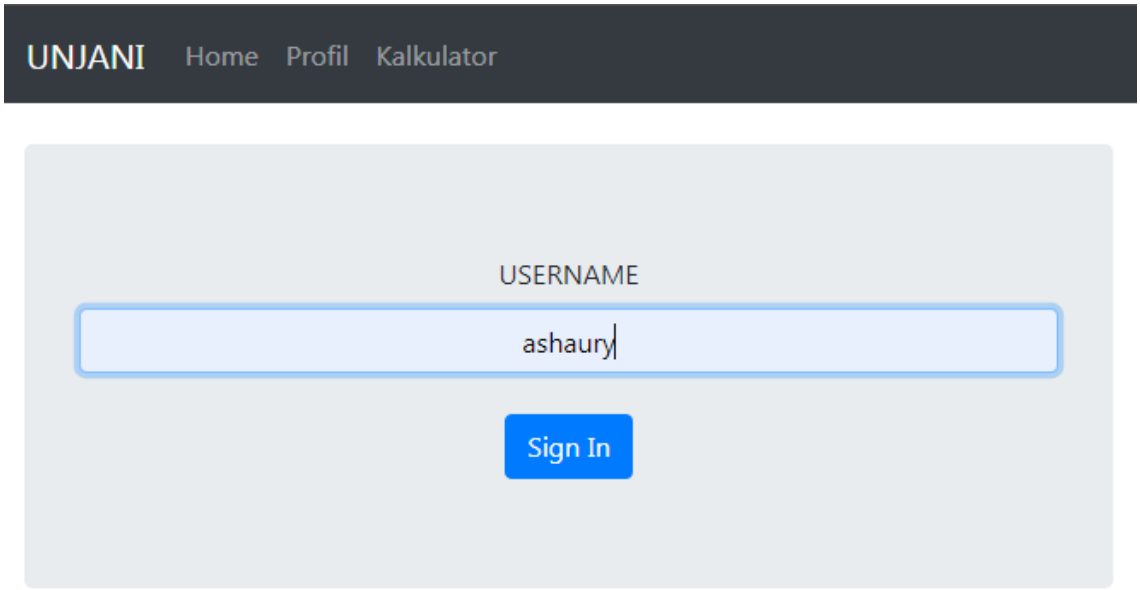
```
http://localhost/webtek-ci/index.php/calc
```

4.6 Latihan 5

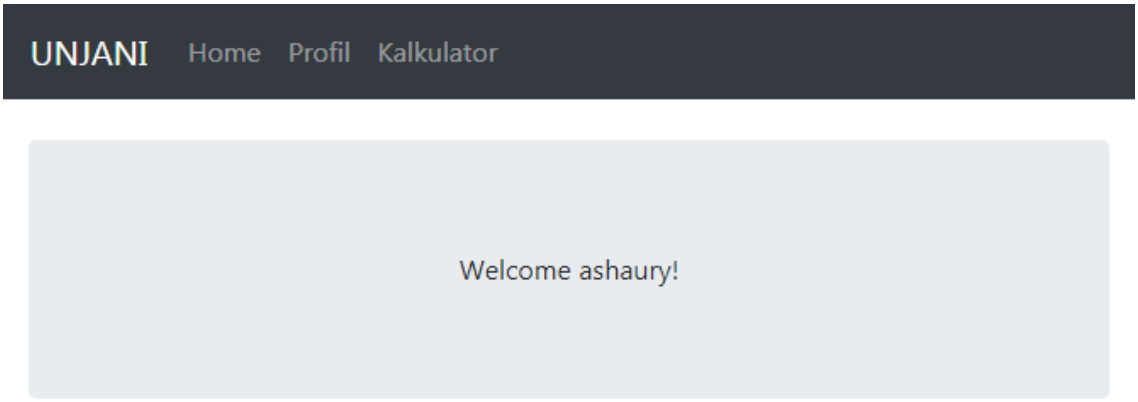
Pada latihan ini, akan digunakan bootstrap sebagai resource css. Download bootstrap pada link berikut:

```
https://github.com/twbs/bootstrap/releases/download/
```

Buatlah controller dan view dengan menggunakan bootstrap, sedemikian rupa sehingga tampilan seperti pada gambar berikut.



Pada Navigasi, **Home** akan mengarahkan ke halaman ini, **Profil** akan mengarahkan pada halaman profil (Latihan 2) dan **Kalkulator** akan mengarahkan ke halaman kalkulator (Latihan 3). Dan ketika tombol Sign In ditekan, maka akan menampilkan sebagai berikut.



4.7 Tugas.

Jawablah pertanyaan-pertanyaan berikut!

1. Jelaskan bagaimana cara kerja Framework PHP Codeigniter!
2. Diketahui suatu aplikasi web menggunakan Codeigniter dan mempunyai Controller Home.php. Jelaskan langkah-langkah supaya Controller tersebut menjadi Controller utama atau yang pertama kali dijalankan dalam aplikasi tersebut!

3. Diketahui suatu web menggunakan codeigniter, mempunyai domain www.webtek.com. Dalam aplikasi tersebut terdapat Controller, Matakuliah.php. Matakuliah.php mempunyai banyak method/fungsi diantaranya, daftar(). Apakah yang harus diisi pada alamat browser untuk memanggil method/fungsi daftar() tersebut?

4. Menggunakan soal nomor 3, gunakan fungsi dalam helper url (site_url atau base_url) untuk men-generate alamat pada soal nomor 3.

5. Diketahui potongan suatu code dalam controller.

```
$data['nama'] = "Herdi Ashaury";
$data['nim'] = 3411101111;
$data['jurusan'] = "IF";
$this->load->view("profil",$data);
```

Lengkapilah code pada view profil.php berikut, menggunakan data yang dikirimkan dari controller supaya nilainya sesuai.

```
<h1>BIODATA</h1>
<p>
Nama : ..... <br>
NIM : ..... <br>
Jurusan : ..... <br>
</p>
```

MODUL 5

Codeigniter Model dan Database

Tujuan

1. Mahasiswa dapat mengimplementasikan Model pada Codeigniter
2. Mahasiswa dapat menggunakan database dalam web aplikasi
3. Mahasiswa dapat menggunakan Library CI dalam menampilkan data dari Database

5.1 Pendahuluan

Model dalam CodeIgniter merupakan kumpulan class PHP yang didesain untuk bekerja dengan data informasi dalam database. Setiap aplikasi yang menggunakan database, pada umumnya mempunyai fungsi untuk menampilkan data, insert, update dan delete data dari database tersebut. Fungsi-fungsi tersebut dipisahkan disimpan secara khusus dalam Model. Model dalam Codeigniter sebetulnya optional, karena tanpa model, controller dapat bekerja dengan database juga. Namun jika seperti itu, pengembangan aplikasi tidak sesuai dengan kaidah MVC yang didukung oleh Codeigniter.

5.1.1 Anatomi Class Model

Model dalam codeigniter disimpan pada alamat direktori **application/models/.** Setiap Class Model harus extends ke class CI_Model. Sama halnya dengan Class pada PHP, nama Class harus sesuai dengan nama file. Contoh suatu class model.

```
class userModel extends CI_Model {


}
```

dan, nama file dari Class tersebut adalah,

```
application/models/userModel.php
```

5.1.2 Pemanggilan Model

Model dapat dipanggil pada suatu controller, dengan cara,

```
$this->load->model('model_name');
```

Setelah class model diload, class tersebut dapat dipanggil menggunakan cara,

```
$this->model_name->method()
```

5.1.3 Autoload Model

Model seringkali digunakan pada beberapa Controller, sehingga akan menyulitkan jika harus diload pada setiap controller yang menggunakannya. Untuk mempermudah penggunaan Model, dapat dikonfigurasi supaya diload otomatis. Buka file **application/config/autoload.php** untuk menambahkan konfigurasi autoload model. Biasanya terdapat pada bagian paling bawah. Tambahkan nama model yang ingin diautoload pada variable array model.

```
$autoload['model'] = array('nama_model1','nama_model2','nama_modeln');
```

5.1.4 Koneksi Database

Setiap model yang diload, tidak otomatis terkoneksi kepada database. Untuk mengkoneksikan aplikasi kedalam suatu database ada beberapa cara yaitu,

a. auto-connect melalui parameter load model.

```
$this->load->model('model_name', '', TRUE);
```

Penggunaan nilai TRUE pada parameter ketiga ini akan mengkoneksikan model yang diload, sesuai pengaturan koneksi default yang ada pada config. File config tersebut adalah pada **application/config/database.php**.

b. Pengaturan koneksi manual menggunakan variabel di parameter ke tiga.

```
$config['hostname'] = 'localhost';
$config['username'] = 'myusername';
$config['password'] = 'mypassword';
$config['database'] = 'mydatabase';
$config['dbdriver'] = 'mysqli';
$config['dbprefix'] = '';
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

$this->load->model('model_name', '', $config);
```

c. Standard Koneksi Codeigniter

Tanpa menggunakan model secara langsung, dapat juga langsung terkoneksi dengan database. Pertama menggunakan auto-connect. Berbeda dengan auto-connect Model, database diautoloadkan pada semua halaman menggunakan konfigurasi **application/config/autoload.php**. Tambahkan kata kunci "database" pada array index library, seperti berikut.

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

Cara kedua adalah memanggil fungsi **`load->database`** secara manual pada fungsi yang menggunakan database. Biasanya dipanggil pada class constructor supaya bisa berjalan disemua fungsi pada class tersebut. Cara untuk memanggil fungsi database.

```
$this->load->database();
```

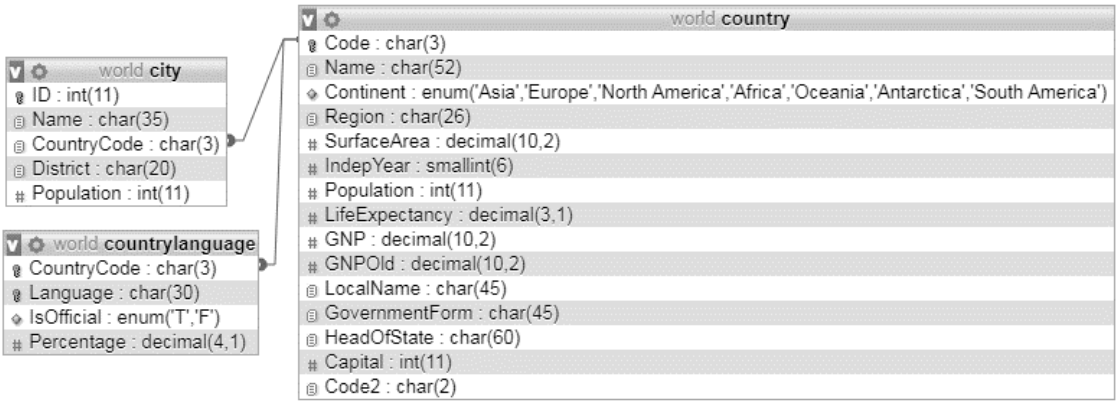
5.2 Latihan 1

5.2.1 Database

Dalam praktikum ini, database yang digunakan adalah dataset World. Database dapat didownload pada alamat url berikut,

```
https://downloads.mysql.com/docs/world.sql.zip
```

Sedangkan skema relasi dari database world adalah sebagai berikut,



Menggunakan phpmyadmin (http://localhost/phpmyadmin) import database tersebut, atau buatlah database berdasarkan skema relasi diatas.

5.2.2 Konfigurasi database

Tambahkan konfigurasi database pada file **application/config/database.php**. Sesuaikan dengan server database masing-masing. Pada umumnya setting untuk localhost sebagai berikut.

```
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'world',
    'dbdriver' => 'mysqli',
```

hostname adalah alamat/host server database.

username adalah username yang mempunyai hak akses ke database

password adalah sandi yang digunakan oleh username untuk masuk ke database

dbdriver adalah driver atau engine yang digunakan oleh codeigniter yang sesuai dengan yang digunakan oleh database. Untuk MySQL versi terbaru, biasanya menggunakan *mysqli* dari pada *mysql* saja.

Konfigurasi lainnya ditambahkan jika diperlukan pengaturan lain yang lebih lanjut dan khusus.

5.2.3 Membuat Model

Buatlah suatu file baru pada direktori **application/models** berikan nama **CountryModel.php**. Model ini akan digunakan untuk menyimpan fungsi-fungsi yang menggunakan tabel country pada database world.

Salin code berikut pada **CountryModel.php**

```
<?php
class CountryModel extends CI_Model{
    function getCountry(){
        return $this->db->get("country");
    }
}
?>
```

Fungsi getCountry akan mengembalikan hasil fungsi db->get(). Fungsi db->get sendiri merupakan salah satu fungsi dari library database, yang merupakan fungsi yang digunakan untuk membentuk suatu query atau disebut query builder. db->get() disi dengan parameter nama tabel, dalam contoh ini yaitu tabel country. Sehingga hasil dari query builder tersebut sama dengan

```
SELECT * FROM COUNTRY.
```

Atau lebih lengkapnya, keluaran dari fungsi tersebut hampir sama dengan hasil dari,

```
mysql_query($dbconnect,"SELECT * FROM country).
```

5.2.4 Membuat Controller

Buatlah suatu file baru pada **application/controllers** dan berikan nama **Country.php.** Controller ini akan digunakan untuk memanggil view dan model untuk **Country**.

Salin code berikut pada **application/controllers/Country.php**

```
<?php
class Country extends CI_Controller{
    public function __construct(){
```

```
        parent::__construct();
        $this->load->model("CountryModel","",TRUE);
    }
    public function index(){
        $data['country'] = $this->CountryModel->getCountry();
        $this->load->view("country",$data);
    }
}
?>
```

Pada controller diatas digunakan fungsi **__construct()** (*underscorenya dua kali*), untuk membuat fungsi yang dijalankan ketika controller tersebut dijalankan. Isi dari construct tersebut adalah, **parent::__construct()** untuk memanggil ulang fungsi construct bawaan dari class parent yaitu class CI_Controller. Dan **load->model()** untuk memanggil dan menggunakan model CountryModel. Dengan cara seperti ini, tidak perlu lagi memanggil model yang dipanggil pada setiap fungsi di class tersebut. Sehingga pada fungsi index, tidak perlu dipanggil ulang.

5.2.5 Membuat View

Buatlah suatu file baru pada application/views dan beri nama country.php. View tersebut akan digunakan untuk menampilkan hasil query yang dilakukan pada model, yaitu daftar negara pada tabel country di database world.

Salin code berikut pada application/views/country.php

```
<html>
<head>
    <title>Daftar Negara</title>
</head>
<body>
    <h1>Daftar Negara</h1>
    <table border="1" align="center">
    <tr>
        <th>Code</th><th>Name</th>
    </tr>
    <?php
    foreach($country->result() as $row){
        echo "<tr>";
        echo "<td>$row->Code</td>";
        echo "<td>$row->Name</td>";
```

```
        echo "</tr>";
    }
    ?>
    </table>
</body>
</html>
```

Variabel **$country** yang merupakan hasil parsing data dari controller, berupa suatu object hasil query dari fungsi **db->get()**. Untuk menampilkan data record dari hasil query tersebut, sebelumnya harus dijalankan fungsi **result()**, agar hasil query disajikan dalam bentuk object. Selanjutnya object tersebut diloop untuk setiap record didalamnya, menggunakan foreach dan tiap baris record tersebut disimpan dalam suatu variabel yang bernama **$row**. Sehingga untuk menampilkan Code dan Name dari tabel country, menggunakan **$row->Code** dan **$row->Name**. Berlaku juga untuk field/kolom tabel country lainnya, misalkan **Continten, Region, Population**, **dll**.

Untuk melihat hasilnya, gunakan alamat url berikut

```
http://localhost/folderci/index.php/country
```

5.3 Latihan 2

Sama seperti latihan 1, pada latihan ini akan menampilkan data dari tabel di database Country. Untuk latihan ini, tabel yang digunakan adalah tabel City dan akan menggunakan library tabel untuk mempermudah tampilan tabel.

5.3.1 Membuat Model

Buatlah suatu file baru pada direktori **application/models** berikan nama **CityModel.php**. Model ini akan digunakan untuk menyimpan fungsi-fungsi yang menggunakan tabel City pada database world.

Salin code berikut pada **CityModel.php**

```
<?php
class CityModel extends CI_Model{
    function getCity(){
        return $this->db->get("city");
    }
}
?>
```

5.3.2 Membuat Controller

Buatlah suatu file baru pada **application/controllers** dan berikan nama **City.php.** Controller ini akan digunakan untuk memanggil view dan model untuk **City**.

Salin code berikut pada **application/controllers/City.php**

```
<?php
class City extends CI_Controller{
    public function index(){
        $this->load->model("CityModel","",TRUE);
        $data[city'] = $this->CountryModel->getCity();
        $this->load->view("city",$data);
    }
}
?>
```

5.3.3 Membuat View

Buatlah suatu file baru pada application/views dan beri nama city.php. View tersebut akan digunakan untuk menampilkan hasil query yang dilakukan pada model, yaitu daftar kota pada tabel city di database world.

Salin code berikut pada application/views/city.php

```
<html>
<head>
    <title>Daftar Kota</title>
</head>
<body>
    <h1>Daftar Kota</h1>
    <?php
    $template = array(
        'table_open' => '<table border="1">'
    );
    $this->table->set_template($template);
    $this->table->set_heading("Nama","Negara","Populasi");
    foreach($city->result() as $r){
        $this->table->add_row($r->Name,$r->CountryCode,$r->Population);
    }
    echo $this->table->generate();
    ?>
</body>
</html>
```

5.4 Latihan 3

Buatlah Model Controller dan View untuk menampilkan data Bahasa dari tabel Language pada database world.

5.5 Latihan 4

Pada latihan ini presentasi tabel hasil dari database akan ditampilkan menggunakan jQuery dan datatables.js. Gunakan pada salah satu view country, city, atau language. Untuk mendownload jQuery gunakan link berikut.

```
https://code.jquery.com/jquery-3.4.1.min.js
```

Selanjutnya menggunakan datatables.js dengan cara download resource pada link berikut

CSS

```
http://cdn.datatables.net/1.10.20/css/jquery.dataTables.min.css
```

Javascript

```
http://cdn.datatables.net/1.10.20/js/jquery.dataTables.min.js
```

Buka file tersebut dalam browser dan Save As, untuk mendownload.

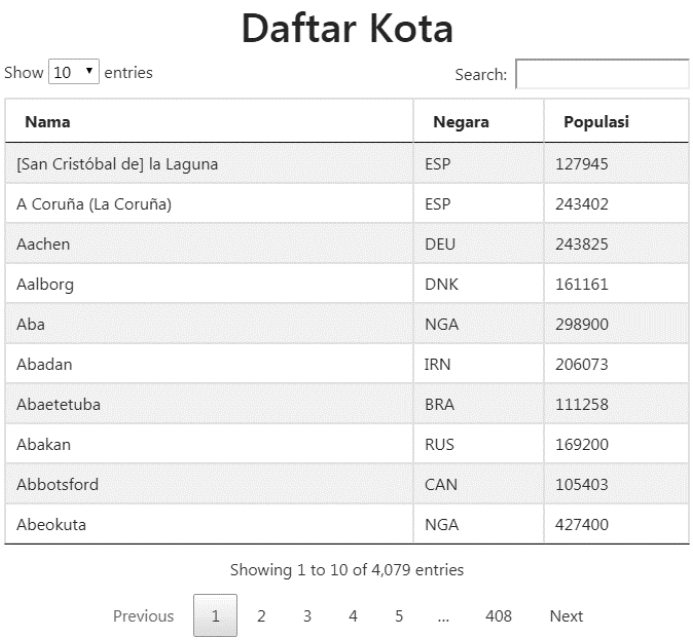
Masukan resource external tersebut pada view, seperti pada Praktikum sebelumnya. Supaya tampilan lebih bagus, gunakan bootstrap. Tambahkan code javascript untuk memanggil fungsi datatables untuk element table yang mempunyai ID myTable dengan code seperti berikut.

```
<script>
        $(document).ready(function() {
            $('#myTable').DataTable();
        });
</script>
```

Selanjutnya tambahkan pada tabel yang digunakan, atribut id=”myTable”. Jika menggunakan bootstrap, tambahkan juga class-classnya.

```
<table id=”myTable” class=”table table-striped table-bordered”>
```

Sehingga tampilan seperti berikut.



5.6 Tugas

Jawablah pertanyaan berikut!

1. Apa yang dimaksud dengan model pada Codeigniter (CI), dan kaitannya dengan Design Pattern MVC?

2. Apa nama Class yang digunakan sebagai Class Parent semua class Model dalam CI?

3. Diketahui suatu Database dengan nama dbAkademik dengan engine oracle, berada di alamat 74.125.24.99 dan port 8080, akun pengguna untuk akses menggunakan username = unjani, dan password = informatika. Buatlah konfigurasi untuk koneksi ke database tersebut.

4. Diketahui suatu table mahasiswa dengan struktur sebagai berikut

NIM	Nama	Jurusan	Peminatan	Email	Telp

Jika `$mhs` merupakan varibel yang dikirmkan dari controller dari suatu fungsi model yang berisi `$this->db->get->("mahasiswa")` lengkapilah potongan code dari view berikut, supaya data mahasiswa dapat muncul dengan baik.

```
foreach($mhs->result() as $data){
echo ……
```

MODUL 6

Query Builder Codeigniter dan CRUD

Tujuan

1. Mahasiswa dapat memahami cara kerja class Query Builder CodeIgniter
2. Mahasiswa dapat mengimplementasikan Fungsi tambah, update, hapus data kedalam database menggunakan Codeigniter

6.1 Pendahuluan

Create, Read, Update dan Delete yang biasa disingkat menjadi CRUD, merupakan berntuk fungsi yang biasa digunakan dalam suatu aplikasi yang menggunakan database. Pada praktikum sebelumnya, dilakukan kegiatan Read data dengan menampilkan hasil query database menggunakan tabel. Untuk praktikum ini, akan dilakukan proses Create, Update dan Delete.

Database yang digunakan untuk praktikum ini, menggunakan dataset World. Bagi yang belum mempunyai database tersebut, buka kembali praktikum sebelumnya.

6.1.1 Query Builder

Dalam praktikum sebelumnya, telah disinggung sedikit tentang Query Builder. Query Builder merupakan suatu Class dalam CodeIgniter yang membantu mempermudah dalam penggunaan database, seperti menampilkan, tambah, update hapus dengan code yang lebih sedikit.

Manfaat utama menggunakan fitur Query Builder adalah memungkinkan membuat basis data aplikasi independen, karena sintaks kueri dihasilkan oleh setiap adapter basis data. Ini juga memungkinkan untuk menghasilkan hasil query yang lebih aman, karena nilai yang masuk telah secara otomatis di escape oleh sistem.

Penggunaan Query Builder akan sulit ketika membutuhkan query yang kompleks misalkan query yang menggunakan join dan filter.

6.1.2 Seleksi Data

- get() → Memilih semua data dari tabel.

```
$query = $this->db->get('produk');
```

- select() → Menentukan kolom apa saja yang ingin diambil dari tabel.

```
$this->db->select('id_produk, nama_produk');
$query = $this->db->get('produk');
```

- select_max(), select_min() → Memilih record dengan nilai terbesar/terkecil.

```
$this->db->select_max('id_produk');
$query = $this->db->get('produk');
```

atau

```
$this->db->select_min('id_produk');
$query = $this->db->get('produk');
```

- select_avg(), select_sum() → Menghitung rata-rata nilai / jumlah nilai dari suatu kolom pada tabel.

Rata-rata

```
$this->db->select_avg('harga);
$query = $this->db->get('produk');
```

Total / Sum

```
$this->db->select_sum('harga);
$query = $this->db->get('produk');
```

- from() → Jika kita ingin menuliskan bagian FROM dari query.

```
$this->db->select('id_produk, nama_produk');
$this->db->from('produk');
$query = $this->db->get();
```

- join() → Melakukan tabel join. Opsi join yang disediakan yaitu inner, outer, right, left, left outer, dan right outer.

```
$this->db->select('*');
$this->db->from('produk');
$this->db->join('user', 'produk.id_produk = user.produk', 'inner');
$query = $this->db->get();
```

- where(), or_where() → Menambahkan fungsi WHERE pada query.

```
$this->db->where('id_produk', $id_produk); // sama dengan
$this->db->where('nama_produk !=', $nama_produk); // tidak sama dengan
$this->db->where('harga <', $harga); // dan kurang dari
$this->db->or_where('id_produk >', $id_produk); // atau lebih dari
```

- where_in(), or_where_in()→ Menambahkan fungsi WHERE nama_kolom IN.

```
$nama_produk = array('Malkist', 'Energen', 'Milo');
$this->db->where_in('nama_produk', $nama_produk); // Atau
$this->db->or_where_in('nama_produk', $nama_produk);
```

- where_not_in(), or_where_not_in() → Menambahkan fungsi WHERE nama_kolom NOT IN.

```
$nama_produk = array('Malkist', 'Energen', 'Milo');
$this->db->where_not_in('nama_produk', $nama_produk); // Atau
$this->db->or_where_not_in('nama_produk', $nama_produk);
```

- like(), or_like() → Mencari data yang mirip.

```
$this->db->like('nama_produk', 'nilai')    // %nilai%
$this->db->like('nama_produk', 'nilai', 'before'); //'%nilai'
$this->db->or_like('nama_produk', 'nilai', 'after'); //'nilai%'
```

- not_like(), or_not_like()→ Mencari data yang tidak mirip.

```
$this->db->not_like('nama_produk', 'nilai');
```

- group_by()→ Menambahkan fungsi GROUP BY ke dalam query.

```
$this->db->group_by('nama_produk');
```

- distinct() → Menambahkan fungsi DISTINCT BY ke dalam query.

```
$this->db->distinct();
$this->db->get('produk');
```

- having(), or_having()→ Menambahkan fungsi HAVING ke dalam query.

```
$this->db->having('id_produk', 13); // Memiliki
$this->db->or_having('id_produk', 6);     // Atau Memiliki
```

- order_by() → Mengurutkan data.

```
$this->db->order_by('nama_produk', 'ASC');       // Kecil ke besar
$this->db->order_by('nama_produk', 'DESC')       // Besar ke kecil
$this->db->order_by('nama_produk', 'RANDOM');   // Acak
```

- limit() → Membatasi banyak baris data yang dihasilkan query.

```
$this->db->limit(10);                    // Limit 10 baris data
```

6.1.3 Insert Data

- insert()→ Memasukkan data ke tabel.

```
$data = array(
'nama_produk' => 'Milo',
```

```
'kategori' => 'Minuman'
);
$this->db->insert('produk', $data);
```

6.1.4 Update Data

- update()→ Mengubah data pada tabel.

```
$data = array(
        'nama_produk' => 'Milo',
        'kategori' => 'Minuman'
);
$this->db->where('id_produk', $id_produk);
$this->db->update('produk', $data);
```

6.1.5 Hapus Data

- delete()→Menghapus data pada tabel.

```
$this->db->where('id_produk', $id_produk);
$this->db->delete('produk');
```

- empty_table()→ Menghapus semua data yang ada pada tabel.

```
$this->db->empty_table('produk');
```

6.1.6 Metode Penulisan Rantai

Pada query builder terdapat metode rantai dimana semua fungsi query builder dapat kita gabungkan ke dalam satu sintak. Berikut contoh penggunaan metode rantai.

```
$query = $this-db->select('nama_produk')
        ->where('kategori', $kategori)
        ->limit(5)
        ->get('produk');
```

6.1.7 Query Helper

Berikut adalah fungsi pada Codeigniter yang dapat kita gunakan untuk mengetahui informasi dari query yang dieksekusi dan infomasi database yang digunakan.

- Mengembalikan banyak data yang terkena efek sebuah query.

```
$this->db->affected_rows()
```

- Mengembalikan string query yang terakhir kali di eksekusi (bukan hasil query).

```
$this-db->last_query()
```

- Mengembalikan banyak data pada sebuah tabel.

```
$this-db->count_all()
echo $this->db->count_all('produk');
```

- Mengembalikan platform database yang digunakan (MySQL, MS SQL, dll).

```
$this-db->platform()
```

- Mengembalikan versi database yang digunakan.

```
$this-db->version()
```

6.2 Latihan 1

Pada latihan ini akan membuat fungsi yang dapat menambahkan data ke dalam database World. Supaya tidak terlalu banyak, gunakan tabel City. Sebelum membuat Controller, model dan view, pastikan konfigurasi database pada application/config/database.php sesuai dengan pengaturan server yang digunakan.

Jika sudah melakukan praktikum pertemuan sebelumnya, seharusnya sudah memiliki Model City yaitu application/models/CityModel.php, Controller City yaitu application/controllers/City.php. Jika belum, buka kembali pertemuan sebelumnya dan buat CityModel.php dan City.php sesuai yang diinstruksikan.

6.2.1 Autoload resources

Karena banyak sekali penggunaan helper URL, sebaiknya helper URL tersebut dijadikan sebagai helper yang autoload pada setiap halaman. Ubah konfigurasi autoload pada application/config/autoload.php dan tambahkan elemen pada array index helper seperti berikut.

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

Kemudian penggunaan model City yang juga banyak sekali digunakan pada Controller City, tetapi tidak diperlukan pada Controller lain. Sehingga lebih tepatnya jika model City di autoload pada Class City saja. Caranya adalah dengan memanggil model tersebut pada fungsi construct class. Tambahkan pada Controller City.php fungsi berikut.

```
public function __construct(){
        parent::__construct();
        $this->load->model("CityModel","",TRUE);
}
```

6.2.2 Membuat Method Form tambah (Controller)

Untuk menambahkan data, dibutuhkan form yang akan digunakan untuk mengisi data yang akan dimasukan kedalam database. Sebuah controller dibutuhkan sebagai alamat untuk akses. Tambahkan dalam controller City.php fungsi tambah() dengan isi code sebagai berikut.

```
public function tambah()
    {
        $this->load->model('CountryModel');
        $data['country'] = $this->CountryModel->getCountry();
        $this->load->view("kota_tambah",$data);
    }
```

Dalam tabel City, terdapat relasi CountryCode dari tabel Country, sehingga dibutuhkan CountryModel untuk memanggil fungsi getCountry(). Daftar CountryCode ini akan menjadi dropdownlist pada view.

6.2.3 Membuat View (Form Tambah)

Untuk menambahkan data, dibutuhkan form yang akan digunakan untuk mengisi data yang akan dimasukan kedalam database. Buatlah view baru, application/views/kota_tambah.php. Salinlah code berikut ke View tersebut.

kota_tambah.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Tambah Kota</title>
    <style>
        label {
            display: inline-block;
            width: 100px;
        }
    </style>
</head>
<body>
    <h1>Tambah Kota</h1>
    <form action="<?php site_url('city/prosestambah'); ?>" method="post">
        <label>Nama</label><input type="text" name="name"><br>
        <label>Code Negara</label><Select name="code">
            <?php
            foreach ($country->result() as $ctr) {
```

```
                echo '<option value="'.$ctr->Code.'">'.$ctr-
>Code.'</option>';
            }
            ?></Select><br>
        <label>District</label><input type="text" name="area"><br>
        <label>Populasi</label><input type="text" name="populasi"><br>
        <input type="submit" value="Tambah">
    </form>
</body>
</html>
```

6.2.4 Membuat Method insert (Model)

Tambahkan dalam CityModel.php fungsi insertCity() yang berisi code sebagai berikut,

```
    function insertCity(){
        $city = array(
            "Name" => $this->input->post("name"),
            "CountryCode" => $this->input->post("code"),
            "District" => $this->input->post("area"),
            "Population" => $this->input->post("populasi")
        );
        return $this->db->insert('City',$city);
    }
```

Penggunaan Query Builder memudahkan dalam pembuatan instruksi untuk tambah data. Salah satu parameter dari fungsi insert yaitu data array yang berisi nama field dan nilai yang akan dimasukan kedalam database. Key array (bagian sebelah kiri =>) harus disamakan dengan nama field tabel yang akan ditambah datanya. Sedangkan Value array (bagian sebelah kanan =>) menggunakan library input, disesuaikan dengan form input yang digunakan pada view. Dalam view tersebut, method form menggunakan POST dan atribut name dari setiap input dijadikan sebagai parameter. Meskipun method insert tidak menghasilkan tampilan data, return dari method ini dapat digunakan sebagai acuan apakah proses penambahan berhasil atau tidak. Perhatikan pada controller proses tambah.

6.2.5 Membuat method proses tambah (Controller)

Tambahkan pada bagian controller, fungsi proses tambah dengan code sebagai berikut.

```
public function prosesTambah(){
        if($this->CityModel->insertCity()){
            redirect(site_url("city"));
        }else{
```

```
            redirect(site_url("city/tambah"));
        }
}
```

Data yang dikirim dari halaman sebelumnya (form tambah), akan dikirimkan ke method controller ini, sesuai dari action yang ada pada atribut form. Pemanggilan method pada model CityModel dilakukan didalam fungsi if supaya dapat dicek apakah proses penambahan berhasil atau tidak.

Retrieving data sebetulnya bisa juga dilakukan didalam controller dan dikirimkan melalui parameter ke method CityModel.

Coba Latihan 1 dengan alamat url berikut:

```
http://localhost/folderci/index.php/city/tambah
```

6.3 Latihan 2

Pada latihan ini akan membuat fungsi yang dapat mengupdate data ke dalam database World. Tabel yang digunakan adalah tabel City. Jika sudah melakukan praktikum pertemuan sebelumnya, seharusnya sudah memiliki View City yaitu application/views/city.php. View city digunakan sebagai halaman awal dalam memilih data mana yang akan diubah. Jika belum ada view city, buka kembali pertemuan sebelumnya dan buat city.php sesuai yang diinstruksikan.

6.3.1 Menambah Aksi pada View City

Untuk mengupadate suatu data dalam database, maka dibutuhkan primary key dari record data tersebut. Tambahkan dalam view application/views/city.php link yang mengirimkan primary key.

Ubah code dalam city.php sehingga seperti berikut,

```
// ------------------ code sebelumnya ------------------------
$this->table->set_heading("Nama", "Negara", "Populasi","Aksi");
foreach ($city->result() as $r) {
    $edit = '<a href="'.site_url("city/update/".$r->ID).'">edit</a>';
    $hapus = '<a href="'.site_url("city/hapus/".$r->ID).'">hapus</a>';
    $this->table->add_row($r->Name, $r->CountryCode, $r->Population,
$edit." ".$hapus);
}
// ------------------code setelahnya ----------------------
```

Alamat yang dihasilkan dari link edit adalah :

http://localhost/folderci/index.php/city/update/ID

dan link hapus,

http://localhost/folderci/index.php/city/hapus/ID

Penambahan satu segmen URI ID pada link diatas, akan digunakan sebagai parsing data ID sebagai acuan data mana yang akan diubah. Segmen URI ID diatas akan menjadi parameter di method update dan hapus di Controller city.

6.3.2 Membuat method Tampil Data berdasarkan ID (Model)

Sebelum data diubah, sebelumnya harus ditampilkan dulu data sebelumnya. Untuk itu, diperlukan suatu method yang dapat mengembalikan data kota berdasarkan primary keynya, yaitu ID.

Tambahkan method berikut pada CityModel.php

```
function getCityById($id){
    $this->db->where("ID",$id);
    return $this->db->get('City');
}
```

6.3.3 Membuat method update (Controller)

Berbeda dengan method biasanya, method update ini menggunakan parameter. Parameter ini didapat dari hasil pemanggilan method pada alamat URL. Dalam URL terdapat satu segmen setelah nama method yang merupakan nilai dari parameter method ini. Tambahkan method update pada Controller City.php, dengan code sebagai berikut.

```
public function update($id){
    $this->load->model('CountryModel');
    $data['country'] = $this->CountryModel->getCountry();
    $data['city'] = $this->CityModel->getCityById($id)->row();
    $this->load->view("kota_update",$data);
}
```

Karena hasil dari query ini diharuskan satu record, maka query result yang digunakan bukan menggunakan method result(), tetapi menggunakan row().

6.3.4 Membuat View Update

Buatlah view baru, application/views/kota_update.php sebagai form edit data kota. Salinlah code berikut pada file tersebut. Supaya lebih cepat, bisa salin dari view kota_tambah dan ubah sesuai instruksi.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Update Kota</title>
    <style>
        label {
            display: inline-block;
            width: 100px;
        }
    </style>
</head>
<body>
<h1>Update Kota</h1>
<form action="<?php echo site_url('City/prosesupdate/'.$city->ID);
?>" method="post">
<label>Nama</label><input type="text" name="name" value="<?php echo $city-
>Name; ?>"><br>
<label>Code Negara</label><Select name="code" value="<?php echo $city-
>CountryCode; ?>">
<?php
foreach ($country->result() as $ctr) {
      $selected = "";
      if($city->CountryCode == $ctr->Code){
      $selected = "selected";
      }
echo '<option value="'.$ctr->Code.'" '.$selected.'>'.$ctr->Code.
'</option>';
}
?>
</Select><br>
<label>District</label><input type="text" name="area" value="
<?php echo $city->District; ?>"><br>
<label>Populasi</label><input type="text" name="populasi" value="
<?php echo $city->Population; ?>"><br>
<input type="submit" value="Update">
</form>
</body>
```

```
</html>
```

Perhatikan action dari form, masih menggunakan parsing data melalui segmen URI. Untuk kasus primary key yang tidak menjadi input, nilai dapat dikirimkan dengan cara segmen URI tersebut. Selain itu juga dapat menggunakan input hidden dan dipanggil menggunakan variabel POST atau GET,

6.3.5 Membuat method update (Model)

Tambahkan dalam CityModel.php fungsi updateCity() yang berisi code sebagai berikut,

```
        function updateCity($id){
        $city = array(
            "Name" => $this->input->post("name"),
            "CountryCode" => $this->input->post("code"),
            "District" => $this->input->post("area"),
            "Population" => $this->input->post("populasi")
        );
        $this->db->where("ID",$id);
        return $this->db->update("City",$city);
    }
```

6.3.6 Membuat method prosesUpdate (Controller)

Tambahkan dalam Controller City.php fungsi prosesUpdate() yang berisi code sebagai berikut,

```
    public function prosesUpdate($id){
        if($this->CityModel->updateCity($id)){
            redirect(site_url("city"));
        }else{
            redirect(site_url("city/update/$id"));
        }
    }
```

Coba latihan 2 menggunakan alamat url berikut:

```
http://localhost/folderci/index.php/City
```

Gunakan tombol edit.

6.4 Latihan 3

Pada latihan ini akan membuat proses hapus data. Hampir sama dengan proses update, dibutuhkan parsing primary key/id data yang akan dihapus. Pada latihan sebelumnya, sudah dilakukan penambahan pada view City.php link yang digunakan untuk menghapus data.

Berbeda dengan update, proses hapus tidak membutuhkan form tampil, hanya proses saja. Sehingga yang dibutuhkan adalah menambahkan method pada model dan Controller saja.

6.4.1 Membuat method hapus (Model)

. Tambahkan method berikut pada Class CityModel.php

```
    function deleteCity($id){
        $this->db->where("ID",$id);
        return $this->db->delete("City");
    }
```

Perlu diperhatikan, jika tidak ingin data terhapus semua, pastikan primary key sudah diset melalui db->where().

6.4.2 Membuat Method hapus (Controller)

Tambahkan method berikut pada Class Controller City.php

```
    public function hapus($id){
        $this->CityModel->deleteCity($id);
        redirect(site_url("city"));
    }
```

Dalam proses penghapusan, lebih baik tidak langsung melakukan proses hapus, tetapi menambahkan javascript sebagai konfirmasi, atau meyakinkan pengguna bahwa data akan hilang.

Untuk mencoba Latihan 3, dapat melalui halaman Daftar Kota di

```
http://localhost/folder-ci/index.php/city
```

6.5 Tugas

1. Buatlah CRUD seperti latihan, untuk tabel Country.

MODUL 7

Library dan Helper CodeIgniter

Tujuan

1. Mahasiswa dapat memahami penggunaan library dan helper codeigniter
2. Mahasiswa dapat mengimplementasikan library dan helper codeIgniter.

7.1 Pendahuluan

Pada praktikum ini akan mempelejari tentang penggunaan Library dan Helper pada CodeIgniter. Salah satu yang sering digunakan adalah Session, Upload, form validation. Salah satu penggunaan session adalah untuk Login dan pengaturan hak akses, sedangkan Upload digunakan untuk membantu proses transfer file ke server, dan Form Validation berfungsi untuk mengecek input. Pada pra

7.1.1 Helper

Helper adalah file-file program pada Codeigniter yang berisi fungsi-fungsi bantuan yang dapat dipakai untuk mengerjakan tugas tertentu. Isi dari helper adalah fungsi-fungsi biasa, bukan sebuah kelas.

Pada praktikum sebelumnya kita sudah menggunakan beberapa helper yaitu url. Codeigniter memiliki banyak helper yang akan dijelaskan manfaat dan cara pemanggilannya. Cara pemanggilan helper ada dua cara, yaitu menyimpannya di file **autoload.php** atau di file **controller/model**. Penggunaan autoload, supaya tidak perlu memanggilnya berulang kali.

Pemanggilan helper di file autoload.php:

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

Pemanggilan helper di file controller:

```
$this->load->helper('nama_helper');
```

Berikut akan dijelaskan manfaat dari setiap helper yang ada di Codeigniter.

- Array Helper, berisi fungsi-fungsi yang berkaitan dengan array.
- Captcha Helper, berisi fungsi-fungsi untuk pembuatan captcha. Captcha adalah cara untuk membedakan apakah user adalah seseorang atau bot. Captcha yang dihasilkan helper ini merupakan captcha dasar berupa gambar yang berisi campuran antara huruf dan angka.

- Cookie Helper, berisi fungsi-fungsi yang berkaitan dengan pembuatan cookie.
- Date Helper, berisi fungsi-fungsi yang berkaitan dengan waktu dan tanggal.
- Directory Helper, berisi fungsi-fungsi yang berkaitan dengan folder.
- Download Helper, untuk membuat force download, yaitu suatu fungsi yang akan memaksa user untuk men-download suatu file ketika link menuju suatu file di-klik oleh user.
- Email Helper, berisi fungsi-fungsi yang berkaitan dengan pengiriman email.
- File Helper, berisi fungsi-fungsi yang berkaitan dengan file, misalnya membaca isi file text, menyimpan file text, dan sebagainya.
- Form Helper, berisi perintah-perintah untuk pembuatan form.
- HTML Helper, berisi fungsi-fungsi yang berkaitan dengan pembuatan elemen-elemen HTML.
- Inflector Helper, berisi fungsi-fungsi yang berkaitan dengan pemformatan untuk string, misalnya membuat bentuk plural dari string “product” menjadi “products”, atau sebalikya. Bisa juga untuk mengubah string ke dalam bentuk camelCase.
- Language Helper, berisi fungsi-fungsi yang berkaitan dengan bahasa.
- Number Helper, berisi fungsi-fungsi yang berkaitan dengan angka.
- Path Helper, berisi fungsi yang berkaitan dengan file path server.
- Security Helper, berisi fungsi yang berkaitan dengan keamanan, misalnya terkait dengan XSS Filtering.
- Smiley Helper, berisi fungsi-fungsi yang membantu mengatur emoticon.
- String Helper, berisi fungsi yang berkaitan dengan string, misalnya untuk menghapus double slash pada alamat URL atau untuk menghasilkan random string dan sebagainya.
- Text Helper, berisi fungsi-fungsi yang berkaitan dengan text. Misalnya untuk membatasi jumlah kata yang akan ditampilkan pada suatu halaman, membatasi jumlah karakter, menjalankan fungsi sensor yang meggantikan daftar kata-kata yang terlarang dengan katakata pengganti lainnya, dan sebagainya.

- Typography Helper, berisi fungsi-fungsi untuk memformat teks agar tepat secara semantik, misalnya mengubah ganti baris menjadi tag
, mengubah suatu karakter ke dalam entitas HTML, dan sebagainya.
- URL Helper, berisi fungsi-fungsi yang berkaitan dengan URL. Misalnya untuk membuat link, mendapatkan base URL, melakukan redirect, dan sebagainya.

7.1.2 Library

Library merupakan class yang berisi fungsi-fungsi untuk penyelesaian kasus tertentu. Untuk menggunakan sebuah library, cukup memanggilnya seperti cara memanggil helper.

Pemanggilan library di file autoload.php:

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

Pemanggilan library di file controller atau file model:

```
$this->load->library('nama_library');
```

Ada banyak library yang dapat digunakan dalam CI lebih lengkap dapat dilihat pada

```
http://localhost/folderci/user_guide/libraries/
```

7.1.3 Session

Session adalah salah satu library yang digunakan untuk menyimpan data pada server. Session menyimpan data ke dalam bentuk variabel global. Variabel ini disimpan pada server dan dapat digunakan oleh semua halaman pada website tempat session dimulai. Codeigniter menyediakan library session yang mempermudah kita dapat mengolah variabel session. Berikut cara penggunaan fungsi yang ada pada library session.

Mengambil nilai variabel

```
$data = $this->session->nama_variabel;
```

// Atau

```
$data = $this->session->userdata('nama_variabel');
```

Menambah variabel session

```
$this->session->set_userdata('nama_variabel', 'nilai_variabel');
```

Menghapus variabel session

```
$this->session->unset_userdata('nama_variabel');
```

Menghapus semua variabel session

```
$this->session->sess_destroy();
```

Penggunaan Flashdata

Flashdata merupakan variabel session yang akan hilang pada saat nilai variabelnya diakses. Variabel flashdata akan terhapus secara otomatis setelah nilai variabelnya diambil / diakses. Cara mengambil nilainya sama seperti variabel session biasa, yang berbeda adalah cara pembuatan variabelnya. Berikut cara membuat variabel flashdata.

```
$this->session->set_flashdata('nama_variabel', 'nilai_variabel');
```

Penggunaan Tempdata

Tempdata merupakan variabel session yang memiki waktu kadaluarsa. Saat waktunya sudah habis, variabel tempdata akan terhapus secara automatis. Berikut cara membuat variabel tempdata.

```
$this->session->set_tempdata('nama_variabel', 'nilai_variabel', 300);
```

// Kadaluarsa dalam waktu 300 detik

7.1.4 Upload

Codeigniter memiliki library upload yang berisi fungsi-fungsi yang dapat kita gunakan dalam mengupload sebuah file. Sebelum memanggil class Upload, diperlukan variabel array config sebagai pengaturan upload tersebut. Contoh config upload

```
$config['upload_path']   = './uploads/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size']      = 100;
$config['max_width']     = 1024;
$config['max_height']    = 768;
```

Lebih lengkap mengenai index Array yang digunakan untuk config upload.

Pilihan	Nilai Default	Opsi Nilai	Deskprisi
upload_**path**	None	None	Path folder tempat unggahan disimpan.
allowed_types	None	None	Jenis file unggahan yang diterima. Dapat berupa array atau string yang dipisahkan pipa (|).
file_name	None	Bebas	Jika di-set, Codeigniter akan mengubah nama file sesuai dengan nilai file_name. Jika tidak, nama file asli yang akan digunakan.

Pilihan	Nilai Default	Opsi Nilai	Deskprisi
file_ext_tolower	FALSE	boolean	Jika di-set TRUE, ekstensi file akan diubah menjadi huruf kecil.
overwrite	FALSE	boolean	Jika di-set TRUE, file dengan nama yang sama akan ditimpa. Jika di-set FALSE, nomor akan ditambahkan ke nama file yang lain ketika nama sama.
max_size	0	None	Ukuran maksimum file yang diterima dalam bentuk KB (kilobytes). Di-set 0 jika tidak dibatasi.
max_width	0	None	Ukuran maksimum lebar gambar yang diterima dalam bentuk pixel. Di-set 0 jika tidak dibatasi.
max_height	0	None	Ukuran maksimum tinggi gambar yang diterima dalam bentuk pixel. Di-set 0 jika tidak dibatasi.
min_width	0	None	Ukuran minimum lebar gambar yang diterima dalam bentuk pixel. Di-set 0 jika tidak dibatasi.
min_height	0	None	Ukuran minimum tinggi gambar yang diterima dalam bentuk pixel. Di-set 0 jika tidak dibatasi.
max_filename	0	None	Panjang maksimum nama file yang diterima. Di-set 0 jika tidak dibatasi.
max_filename_increment	100	None	Jika di-set FALSE, gunakan ini untuk mengatur kenaikan panjang nama file yang ditambahkan ke nama file.
encrypt_name	FALSE	boolean	Jika di-set TRUE, nama file akan dikonversi ke string yang terenkripsi acak.
remove_spaces	TRUE	boolean	Jika di-set TRUE, semua spasi pada nama file akan diubah menjadi underscore.
detect_mime	TRUE	boolean	Jika di-set TRUE, sisi server dari jenis file akan dideteksi untuk menghindari serangan injeksi kode.
mod_mime_fix	TRUE	boolean	Jika di-set TRUE, beberapa ekstensi file akan berakhiran garis bawah untuk menghindari mod_mime Apache.

Setelah ada variabel config, class upload dipanggil dengan cara.

```
$this->load->library('upload', $config);
```

Selanjutnya proses perpindahan file dari local ke server akan dijalankan ketika method do_upload() dijalankan. Yaitu dengan code berikut.

```
$this->upload->do_upload()
```

Method tersebut akan mengembalikan nilai true jika proses perpindahan file berhasil, dan false jika gagal. Jika gagal, informasi mengenai kegagalan proses upload tersebut dapat dilihat menggunakan method display_error(), yaitu dengan code berikut.

```
$this->upload->display_errors()
```

Jika berhasil, data file yang berhasil diupload dapat menggunakan variabel array $this->upload->data(), sebaiknya dimasukan kedalam variabel seperti berikut.

```
$fileupload = $this->upload->data()
```

Variabel array tersebut memiliki banyak data sesuai dengan indexnya. Berikut adalah index dan deskripsinya.

Item	Description
file_name	Name of the file that was uploaded, including the filename extension
file_type	File MIME type identifier
file_path	Absolute server path to the file
full_path	Absolute server path, including the file name
raw_name	File name, without the extension
orig_name	Original file name. This is only useful if you use the encrypted name option.
client_name	File name supplied by the client user agent, but possibly sanitized
file_ext	Filename extension, period included
file_size	File size in kilobytes
is_image	Whether the file is an image or not. 1 = image. 0 = not.
image_width	Image width

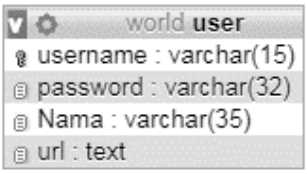
Item	Description
image_height	Image height
image_type	Image type (usually the file name extension without the period)
image_size_str	A string containing the width and height (useful to put into an image tag)

a. Latihan 1

Pada latihan ini akan membuat suatu proses upload menggunakan library upload. Contoh kasus adalah dalam register akun. Database yang digunakan masih world database. Dalam database world tidak ada tabel user, oleh karena itu harus dibuat terlebih dahulu.

7.1.6 Membuat Tabel User

Buatlah tabel user dalam database world dengan spesifikasi seperti berikut



7.1.7 Membuat Controller Form Register (Controller)

Buatlah controller baru di application/controllers dengan nama Home.php, kemudian salin code berikut.

application/controllers/Home.php

```
<?php
class Home extends CI_Controller{
    function register(){
        $this->load->view("register");
    }
}
?>
```

7.1.8 Membuat Form Register (View)

Buatlah view yang berupa suatu form yang digunakan untuk mengisi nilai untuk tabel User. Simpan file tersebut **application/views/register.php.** Salin code berikut pada view tersebut.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <title>Register User</title>
    <link rel="stylesheet" href="<?php echo base_url('assets/css/bootstrap.
css') ?>">
    <style>
        .login-page {
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }
    </style>
</head>
<body class="login-page">
    <div class="card">
        <div class="card-header text-center">Register</div>
        <div class="card-body">
            <form action="<?php echo site_url('home/prosesdaftar') ?>" meth
od="post" enctype="multipart/form-data">
            <label class="label">Username</label>
            <input type="text" class="form-
control" name="username" required>
            <label class="label">Password</label>
            <input type="password" class="form-
control" name="password" required>
            <label>Nama</label>
            <input type="text" class="form-control" name="nama" required>
            <label>Profil Picture</label>
            <input type="file" class="form-control" name="pp" required>
            <br>
            <input type="submit" class="btn btn-primary float-
right" value="Register">
            </form>
        </div>
        <div class="card-footer text-center">
        Sudah Punya Akun?
        <br><a href="<?php site_url('home/') ?>">Sign In</a>
        </div>
    </div>
</body>
</html>
```

Dalam form tersebut terdapat input berupa file, oleh karena itu, perlu ditambahkan atribut enctype=”multipart/form-data” supaya file yang diinputkan bisa ikut dikirimkan ke proses daftar.

7.1.9 Membuat Model User

Supaya data user yang diisi dapat dimasukan kedalam database, maka diperlukan suatu Model untuk menyimpan method simpanUser. Sebetulnya bisa saja ditambahkan pada Model lainnya, tetapi mengikuti kaidah MVC yang baik, sebaiknya Setiap Model dikhususkan untuk satu Entitas saja.

Buatlah Model dalam application/models, UserMode.php. Salin code berikut.

```
<?php
class UserModel extends CI_Model{
    function insertUser($user){
        return $this->db->insert("user",$user);
    }
}
?>
```

Berbeda dengan insert City dan Country, disini input data akan diretrieve di Controller, dan diparsing ke model melalui suatu parameter.

7.1.10 Membuat method Proses Daftar (Controller)

Tambahkan dalam Controller Home.php method prosesdaftar() yang akan digunakan sebagai proses penyimpanan data sekaligus upload gambar. Salin dan tambahkan method berikut.

```
    function prosesDaftar()
    {
        $this->load->model("UserModel","",TRUE);
        $user = array(
            "username" => $this->input->post("username"),
            "password" => $this->input->post("password"),
            "Nama" => $this->input->post("nama")
        );

        $config['upload_path']          = './assets/image';
        $config['allowed_types']        = 'gif|jpg|png';
        // $config['max_size']             = 100;
        // $config['max_width']            = 1024;
```

```
        // $config['max_height']           = 768;
        $this->load->library('upload', $config);
        if (!$this->upload->do_upload('gambar')) {
            echo $this->upload->display_errors();
        } else {
            $upload_data = $this->upload->data();
            $user['url'] = base_url("assets/image/").$upload_data['file_nam
e'];
        }
        if($this->UserModel->insertUser($user)){
            redirect(site_url("home"));
        }else{
            redirect(site_url("home/register"));
        }

    }
```

Untuk mencoba latihan ini, gunakan alamat berikut:

```
http://localhost/folderci/index.php/home/register
```

b. Latihan 2

Pada latihan ini, akan membuat suatu proses login. Proses login sederhananya adalah menyimpan suatu variabel Global yang berinilai status login seseorang. Oleh karena itu, Proses ini memerlukan library session untuk menyimpan variabel Global tersebut.

7.1.11 Autoload Session

Karena library session dan database ini akan diperlukan oleh semua halaman, sehingga perlu diautoloadkan. Tambahkan “session” dan “database” dalam array index libraries pada **application/config/autoload.php**.

```
$autoload['libraries'] = array("session",”database”);
```

7.1.12 Membuat method Index (Controller)

Halaman login akan menjadi halaman utama, sehingga method yang digunakan adalah index(). Method ini nantinya akan memanggil view Login. Tambahkan method berikut ke **application/controllers/Home.php**.

```
    function index(){
        $this->load->view("login");
    }
```

7.1.13 Membuat form Login (View)

Buatlah view yang berupa suatu form yang digunakan untuk mengisi nilai untuk tabel User. Simpan file tersebut **application/views/login.php.** Salin code berikut pada view tersebut.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login</title>
    <link rel="stylesheet" href="<?php echo base_url('assets/css/bootstrap.
css') ?>">
    <style>
        .login-page {
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }
    </style>
</head>
<body class="login-page">
    <div class="card" style="width:300px">
        <div class="card-header text-center">Login</div>
        <div class="card-body">
            <form action="<?php echo site_url('home/proseslogin');?>"
method="post">
            <label class="label">Username</label>
            <input type="text" class="form-
control" name="username" required>
            <label class="label">Password</label>
            <input type="password" class="form-
control" name="password" required>
            <br>
            <input type="submit" class="btn btn-primary float-
right" value="Login">
        </div>
        <div class="card-footer text-center">
        Tidak Punya Akun?
        <br><a href="<?php echo site_url('home/register') ?>">Register</a>
        </div>
    </div>
</body>
```

```
</html>
```

7.1.14 Membuat Halaman Login Berhasil (View)

Buatlah suatu view yang akan digunakan sebagai landing page, jika berhasil login. Berinama **home.php** Salin code berikut.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Home</title>
    <link href="<?php echo base_url('assets/css/ /bootstrap.css')
?>" rel="stylesheet">
</head>
<body>
    <div class="navbar navbar-expand bg-dark navbar-dark">
        <a class="navbar-brand" href="#">UNJANI</a>
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="#">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Profil</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Kalkulator</a>
            </li>
            <li>
                <a class="nav-link"
href="<?php echo site_url('home/logout');?>">Log Out</a>
            </li>
        </ul>
    </div>
    <br>
    <div class="container-fluid">
        <div class="jumbotron text-center">
            Welcome <?php echo $this->session->userdata('username'); ?>!
            <br>
            <img src="<?php echo $this->session->userdata('url');?>"
class="rounded-circle w-25">
        </div>
    </div>
</body>
```

```
</html>
```

Membuat method login (Model)

Untuk melakukan suatu login, username dan password harus diperiksa datanya apakah ada atau sesuai dengan data dalam database. Oleh karena itu perlu suatu method untuk memeriksa ke dalam database. Buatlah method baru dalam UserModel.php, dengan code sebagai berikut.

```
    function login(){
        $username = $this->input->post("username");
        $pass = $this->input->post("password");
        $this->db->where("username",$username);
        $this->db->where("password",$pass);
        return $this->db->get("user");
    }
```

Terdapat dua db->where, secara default ini akan berlaku prinsip AND, sehingga query yang dihasilkan adalah:

```
SELECT * FROM user WHERE username=$username AND password=$pass
```

Membuat method proseslogin (Controller)

Setelah form login diisi, selanjutnya adalah memeriksa username dan password yang dikirimkan dari form login ada dan sesuai dengan data didatabase. Buatlah method baru pada Controller Home.php untuk proses login. Kemudian Salin kode berikut.

```
    function login(){
        $this->load->model("UserModel");
        if($this->UserModel->login()->num_rows>0){
            $session_data = array(
                "login" => true,
                "username" => $this->input->post("username")
            );
            $this->session->set_userdata($session_data);
            redirect(site_url("home"));
        }else{
            $this->session-
>set_flashdata("error","Username atau Password Salah");
            redirect(site_url("home"));
        }
    }
```

Dalam fungsi diatas variabel global session diset pada session->set_userdata(). Menggunakan array asosiatif, bisa langsung digunakan dimana nanti key array akan menjadi index var global dan value akan menjadi nilainya.

Modifikasi method index (Controller)

Karena sudah ada fungsi login, ubah method index sehingga jika sudah login, tidak menampilkan form login lagi, tetapi menampilkan halaman login berhasil. Sehingga method index akan menjadi seperti berikut.

```
    function index(){
        if($this->session->userdata('login')){
            $this->load->view("home");
        }else{
            $this->load->view("login");
        }
    }
```

Modifikasi view form login (View)

Jika login gagal, maka akan menampilkan form login disertai variabel melalui suatu flashdata. Tambahkan suatu tampilan yang menampilkan flashdata tersebut pada view login.php.

```
//----------------------- Code sebelumnya --------------------
<input type="password" class="form-control" name="password" required>
<br>
<?php echo $this->session->userdata("error"); ?>
<br>
<input type="submit" class="btn btn-primary float-right" value="Login">
//----------------------- Code setelahnya ----------------------
```

Set Controller Home sebagai Controller Utama

Supaya halaman utama dari aplikasi yang dibuat adalah halaman login yang berada di controller Home, dapat diatur dalam application/config/routes.php. Ubah default_controller menjadi Home.

```
$route['default_controller'] = Home';
```

Membuat method logout (Controller)

Method logout adalah method yang sangat mudah, cukup menghapus semua session data yang ada pada satu domain yang sama. Pada method ini, dapat menggunakan perintah php native untuk menghapus session yaitu,

```
session_destroy()
```

atau menggunakan fungsi bawaan dari Codeigniter, disertai dengan redirect ke halaman utama. Sehingga fungsi logout adalah sebagai berikut.

```
    function logout(){
        $this->session->sess_destroy();
        redirect(base_url());
    }
```

c. Latihan 3

Perbaiki semua halaman pada controller City, Country, Home, dan Calculator sehingga mengembalikan ke halaman login jika belum login terlebih dahulu.

d. Tugas

Jawablah Pertanyaan berikut!

1. Apa perbedaan helper dan library?
2. Diketahui nama helper Array adalah Array, bagaimana cara memanggil Helper tersebut?
3. Diketahui suatu data yang ingin dimasukan kedalam variabel global session,

   NIM = 3411101134
   NAMA = UNJANI
   JURUSAN = Informatika

   Buatlah code untuk set data tersebut kedalam session!
4. Diketahui dalam suatu database, terdapat tabel user yang terdiri dari username, password, nama, jabatan. Jabatan bisa terdiri dari dua nilai, staff dan admin. Buatlah proses login sehingga jika login masuk seorang staff masuk ke halaman utama staff, dan jika login masuk seorang admin masuk ke halaman utama admin.

MODUL 8
Node.js
(Instalasi dan Konfigurasi)

Tujuan :

- Mahasiswa dapat memahami dan melakukan instalasi Node.js
- Mahasiswa dapat memahami konfigurasi Node.js dalam proses *development web base application*.

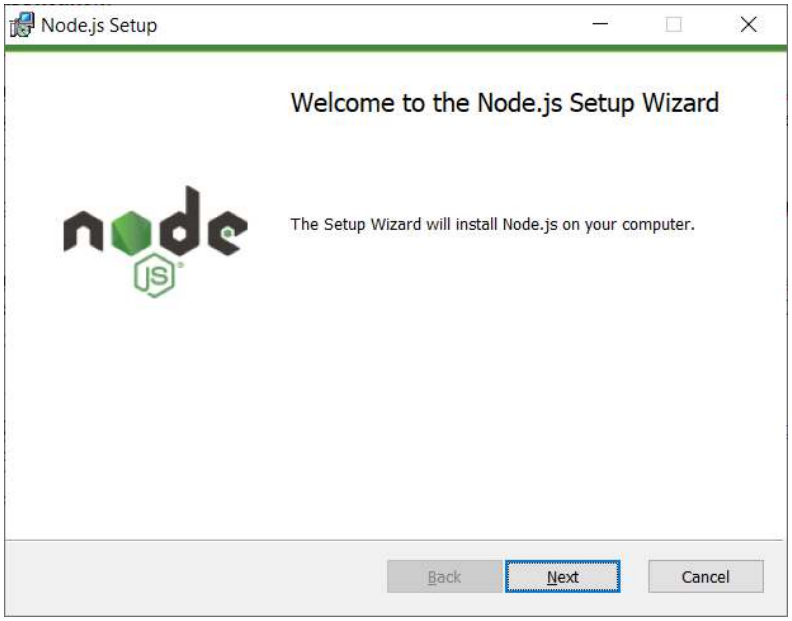
Pendahuluan

Node.js merupakan Javascript environment yang bersifat open-source, cross-platform yang dapat mengeksekusi kode Javascript yang berada di luar browser. Node.js mempersilakan developer menggunakan Javascript untuk menuliskan kode program yang bersifat server-side scripting atau kode program yang ditulis pada sisi server seperti layaknya PHP. Pada server-side scripting akan menjalankan kode yang menghasilkan konten berbentuk dynamic web page sebelum halaman webnya dikirimkan ke browser milik user.

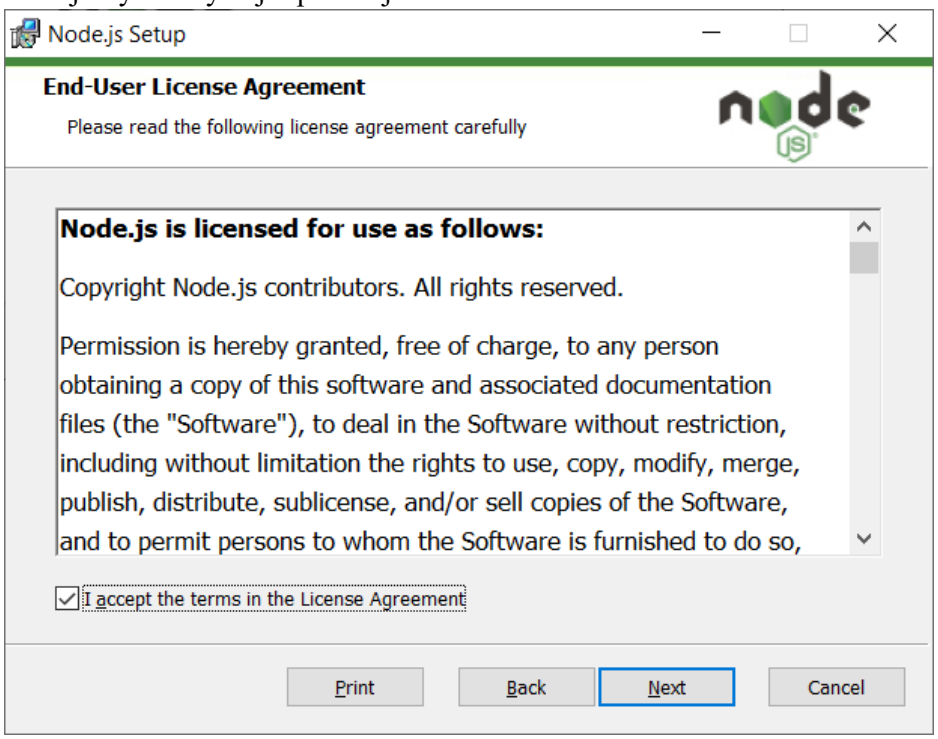
Node.js merepresentasikan paradigma "*Javascript Everywhere*". Hal tersebut menggabungkan seluruh proses development web dalam satu Bahasa pemrograman daripada menggunakan beberapa Bahasa yang berbeda untuk front-end dan back-end.

Instalasi dan *running* Node.js

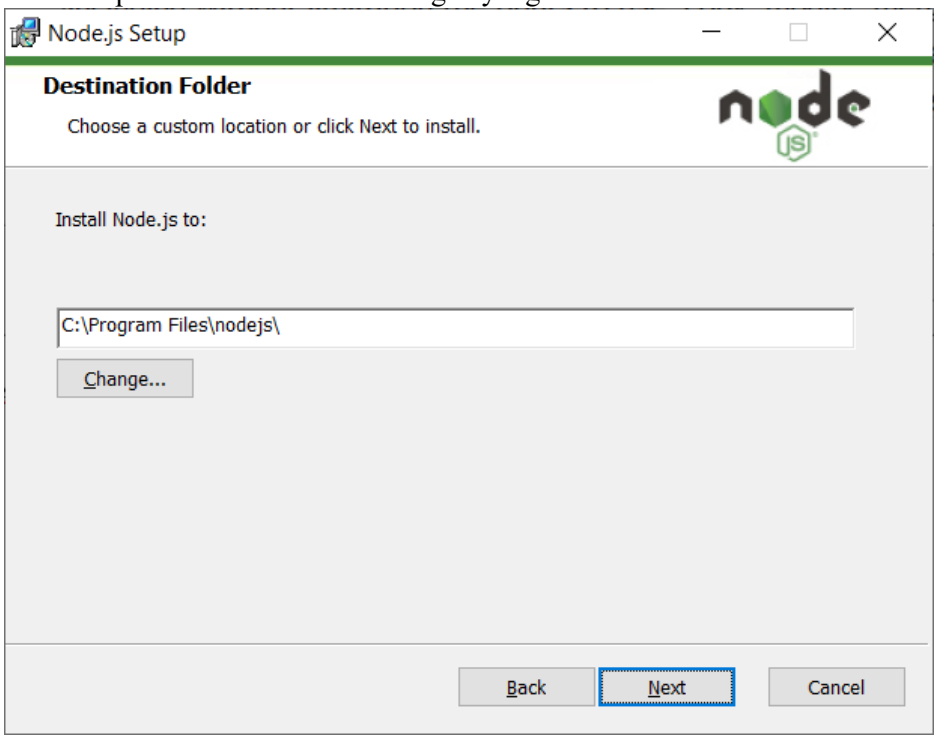
1. Unduh terlebih dahulu installation package Node.js di https://nodejs.org/en/download/. Pilihlah versi LTS untuk Windows atau OS yang sedang digunakan.
2. Lakukanlah proses instalasi node.js seperti yang muncul pada gambar di bawah ini.



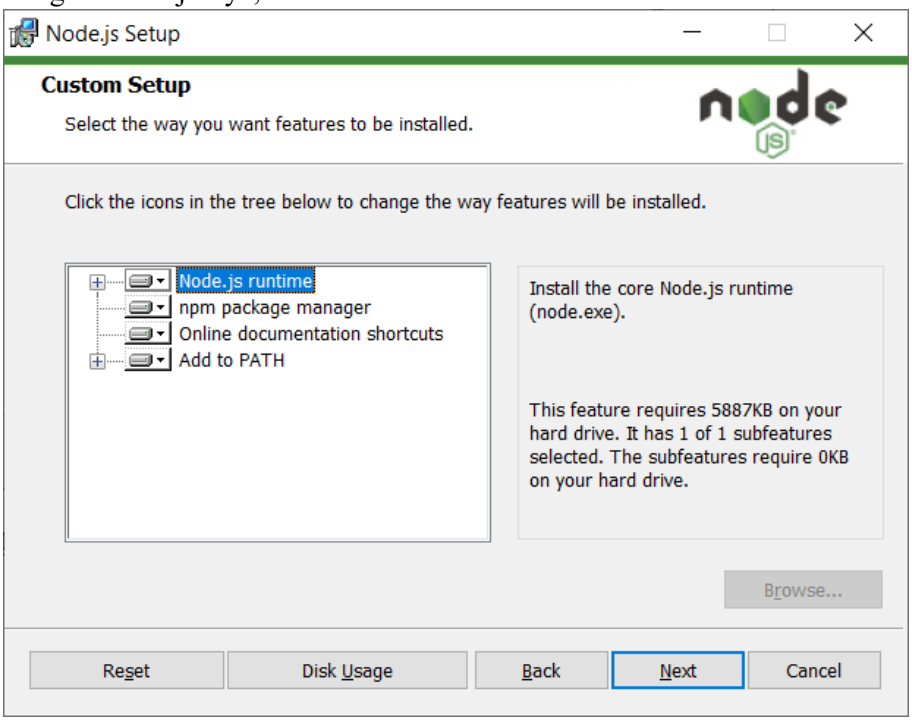
3. Selanjutnya menyetujui persetujuan lisensi.



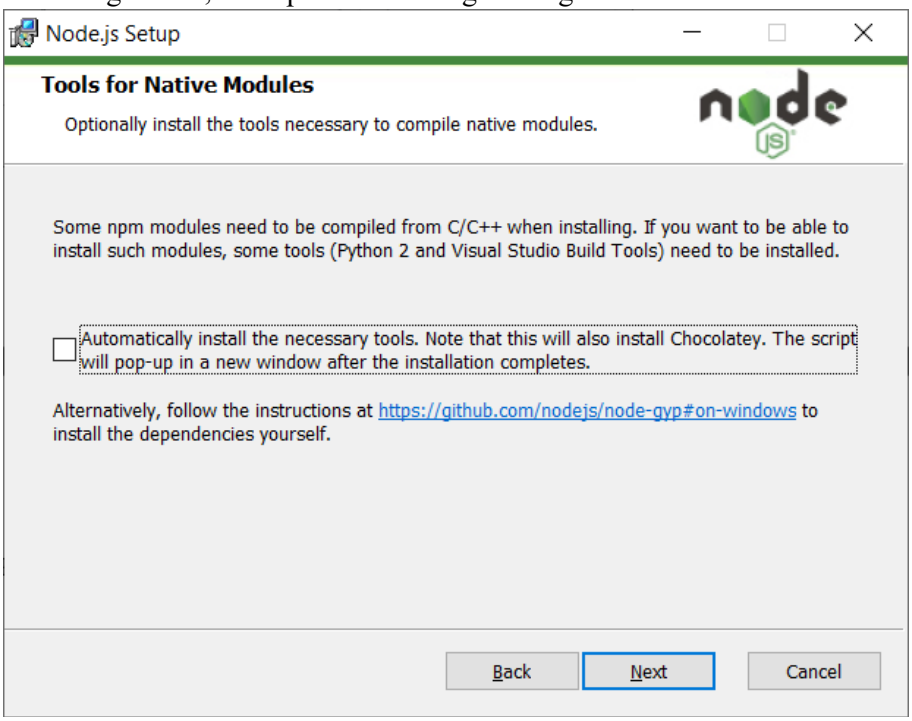
4. Arahkan path instalasasi sesuai dengan yang tertera.



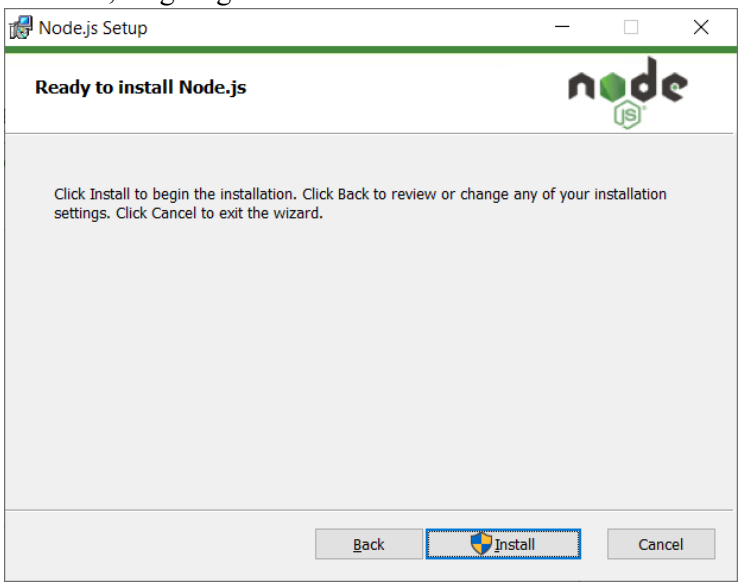
5. Langkah selanjutnya,



6. Pada langkah ini, tidak perlu ceklis bagian tengah.



7. Terakhir, langsung lakukan instalasi.



8. Pada saat proses instalasi Node.js telah selesai, selanjutnya adalah melakukan testing untuk memastikan bahwa Node.js telah terpasang dikomputer. Caranya dengan membuka command prompt dan tuliskan perintah berikut :

```
node -v
```

maka akan muncul tampilan sebagai berikut yang menandakan bahwa Node.js telah terpasang dengan versi 8.11.3.

```
C:\Users\Fatan Kasyidi>node -v
v8.11.3
```

9. Selanjutnya adalah memeriksa keberadaan NPM (Node Package Manager) dengan cara menuliskan perintah ini pada command prompt :

```
npm -v
```

maka akan muncul tampilan sebagai berikut yang menandakan bahwa NPM telah terpasang dengan versi 5.6.0.

```
C:\Users\Fatan Kasyidi>npm -v
5.6.0
```

10. Untuk penggunaan Node.js yang lebih jelas, tulislah kode program di bawah ini dan simpan dalam file dengan nama `app.js`.

`app.js`

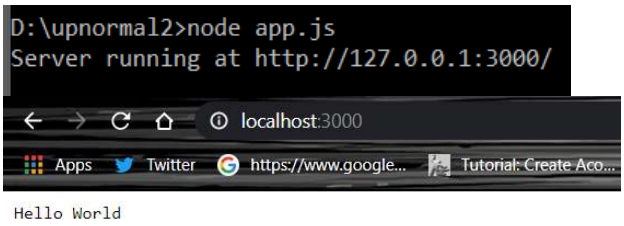
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
 console.log(`Server running at http://${hostname}:${port}/`);
});
```

Dari kode program tersebut menjelaskan bahwa program akan dijalankan pada ip 127.0.0.1 atau kita sering sebut sebagai localhost, port yang digunakan adalah port 3000.

Jalankan file app.js yang telah dibuat pada suatu folder (bebas) dengan mengeksekusi perintah berikut : (ingat, posisikan path command prompt pada folder tempat disimpannya file app.js)

```
node app.js
```

Jika sudah dieksekusi, maka akan keluar tulisan Server running at http://127.0.0.1:3000/ yang artinya hasil eksekusi dapat dibuka pada browser dengan mengakses alamat tersebut. Tampilan hasil eksekusinya adalah sebagai berikut :



Tugas :

- Berikanlah penjelasan terhadap masing-masing baris kode program pada file app.js
- Buatlah laporan modul disertai dengan analisis hasil eksekusinya.

MODUL 9
REST API dengan Node.js

Tujuan :

- Mahasiswa dapat memahami konsep dasar REST API
- Mahasiswa dapat menerapkan REST API menggunakan Node.js dan Express.js sebagai framerworknya

Pendahuluan

Representational State Transfer (REST) merupakan gaya arsitektur *software* yang mendefinisikan kumpulan *constraint* yang digunakan untuk membangun *Web Service*. *Web service* yang merujuk ke arsitektur REST disebut juga *RESTful Web Service*.

RESTful menyediakan interoperabitias antara sistem komputer dan internet. Interoperabilitas merupakan kemampuan saling bekerja sama antara satu sistem dengan sistem lain yang memiliki perbedaan.

Sedangkan *Application Programming Interface* (API) merupakan antarmuka atau prokotol komunikasi antara bagian-bagian yang berbeda di dalam program komputer yang memiliki maksud untuk menyederhanakan proses implementasi dan *maintenance* dari *software*.

Express.js merupakan framework aplikasi berbasis web untuk Node.js. Framework ini memiliki beberapa konsep design pattern yang dapat diterapkan, salah satu yang sudah banyak diketahui adalah konsep MVC (Model View Controller). Express.js akan membantu proses pembangunan REST API yang akan dibuat.

Penerapan REST API

1. Pada folder yang telah dibuat pada modul 8 atau dapat juga membuat folder baru, buatlah file dengan nama `server.js`.
2. Ambil data API berikut ini untuk melakukan passing API ke aplikasi atau program yang dibuat.

```
https://api.chucknorris.io/jokes/random
```

3. Lakukan instalasi express.js melalui npm dengan mengeksekusi perintah berikut :
   ```
   npm i express -save
   ```

4. Lakukan pula instalasi package request, untuk dapat menggunakan http call, perintahnya : https://www.npmjs.com/package/request

   ```
   npm i request --save
   ```

5. Selanjutnya tulislah kode program berikut ini dan simpanlah kode program tersebut pada file server.js.

server.js

```
const express = require('express');
const app = express();
const port = 3000;

const request = require('request');
app.get('/', (req, res) => {
     request('https://api.chucknorris.io/jokes/random', function (er
     ror, response, body) {
             if(error){
                 res.status(400).send('Error');
             }else{
                 res.send(body);
             }
         });
     }); // req = request, res = response
app.post('/', (req, res) => res.send('Halo Post'));
app.listen(port, () => console.log('App Starting'));
```

6. Selanjutnya, eksekusi file tersebut dengan menjalankan baris perintah di bawah ini pada command prompt.

```
nodemon server.js
```

Kemudian periksa hasilnya di localhost:3000

Tugas :

- Analisis *return value* yang muncul berdasarkan alamat API.
- Buatlah laporan modul praktikum beserta analisisnya.