



yiiframework²

Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework

HAFID MUKHLASIN

**Edisi Kedua
Gratis**

Form

Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework

- Edisi Kedua -

Hafid Mukhlasin

Buku lain dari penulis:

Be Fullstack Developer - Laravel: VueJS <https://buku-laravel-vue.com>

Be Fullstack Javascript Developer: ReactJS - ExpressJS <https://bukureact.id>

Edisi Pertama, Februari 2016

Revisi 1, April 2016

Revisi 2, Juli 2016

Revisi 3, Oktober 2016

Edisi Kedua, Januari 2017

Mukhlasin, Hafid.

Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework, Hafid Mukhlasin. – Jakarta: Buku Baik, 2017.

xii, 317 hlm. ; 21 cm.

ISBN 978-602-1018-12-5

1. Pemrograman - Web - Yii Framework. I. Judul II. Mukhlasin, Hafid

Copyright by Hafid Mukhlasin

Dilarang menyalin atau menggandakan buku ini kecuali atas izin tertulis dari penulis buku yaitu Hafid Mukhlasin.

❖ Untuk Allah & Rasul-Nya

❖ Untuk kedua orang tuaku
❖ Untuk istriku tercinta Hari Dwipanjayani
❖ Untuk anak pertamaku Ammar Abdurrahman
❖ Untuk anak keduaku Faqih Abdullah
❖ Untuk anak ketigaku Muhammad Syamil

Tentang Penulis

Hafid Mukhasin, lahir pada tanggal 30 April 1986 di sebuah kota kecil di Jember – Jawa Timur. Mulai tertarik dalam bidang pemrograman komputer sejak duduk di bangku SMA, karenanya kemudian memutuskan untuk kuliah pada jurusan Teknik Informatika Universitas Teknologi Yogyakarta dan berhasil meraih gelar sarjana pada tahun 2008. Penulis melanjutkan studi pasca sarjananya dengan mengambil jurusan Magister Teknologi Informasi Universitas Indonesia, lulus tahun 2017.

Sempat berkarir sebagai pemrogram aplikasi berbasis *web* pada sebuah *software house* di Yogyakarta dan perusahaan periklanan di Surabaya, penulis kemudian memilih bekerja sebagai pengajar dan praktisi IT pada sebuah lembaga pelatihan di Jakarta.

Penulis tertarik mengikuti perkembangan teknologi informasi khususnya yang berkaitan dengan pemrograman komputer. Beberapa teknologi pemrograman yang pernah digeluti antara lain: *web programming* (PHP, Javascript), *desktop programming* (Delphi/Lazarus) serta *mobile programming* (Android). Kompetensi utama penulis adalah pada bidang *web programming*.

Anda dapat menghubungi penulis melalui:

Email : hafidmukhasin@gmail.com

Telp. : +62 81559915720 (Telp/SMS/WA)

Homepage : <https://hafidmukhasin.com>

Kata Pengantar

Puji syukur marilah senantiasa kita panjatkan kehadirat ilahi rabi Allah Subhanahu Wa Ta'ala karena banyaknya limpahan karunia dan nikmatnya sehingga penulis bisa menyelesaikan proyek "Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework" yang sangat berat ini. Walhamdulillah.

Sholawat serta salam semoga tetap tercurahkan kepada suri tauladan kita, Muhammad Shallallahu Alaihi Wasallam yang telah membawa umatnya dari zaman jahiliyah menuju zaman islamiyah.

Terima kasih juga penulis ucapan kepada para guru:

- Misbahul D Munir, yang merupakan guru utama penulis dalam dunia Yii.
- Alexander Makarov (Core developer Yii) atas *framework* Yii, diskusi serta *review* terhadap beberapa artikel penulis.
- Petra Novandi Barus (CTO Urbanindo), atas diskusi dan masukannya selama ini tentang Yii.
- Peter Jack Kambey (*Software Developer* Agung Podomoro), atas motivasinya dan bantuannya untuk penyelenggaraan *event-event* Yii.
- Iskandar Soesman (*Software Architect* Kumparan), atas diskusinya tentang studi kasus yang diangkat dalam buku ini.

Di samping itu juga kepada para senior dan member group Yii Framework Indonesia maupun Internasional yang telah membantu penulis dalam pengembangan Yii selama ini maupun dalam penyusunan buku ini.

Semua gading retak, demikian juga dengan buku ini, banyak kekurangan disana sini, kontradiksi dan mungkin kerancuan yang semua disebabkan karena keterbatasan pengetahuan & wawasan penulis. Oleh karena itu, kritik saran yang membangun tentu sangat penulis harapkan demi perbaikan pada langkah selanjutnya.

Jakarta 31 Januari 2017

Penulis

Daftar Isi

Tentang Penulis	iv
Kata Pengantar	v
Daftar Isi	vi
Gambaran Umum Buku	xi
A. 12	
A. 1. Error! Bookmark not defined.	
A. 2. 12	
A. 3. 13	
B. 13	
C. 13	
Persiapan Lingkungan Kerja	1
A. 1	
A. 1. 1	
A. 2. 1	
A. 3. 1	
A. 4. 1	
A. 5. 1	
A. 6. 2	
B. 2	
C. 5	
D. 6	
D. 1. 6	
D. 2. 9	
D. 3. 9	
D. 4. 9	
E. 10	
Bekerja dengan Composer	11
A. 11	
A. 1. 11	
A. 2. 11	
A. 3. 12	
B. 12	
B. 1. 12	
B. 2. 14	
B. 3. 14	
C. 15	
C. 1. 15	
C. 2. 15	
C. 3. 15	
C. 4. 16	
D. 17	
D. 1. 17	
D. 2. 19	
D. 3. 20	
E. 20	
Pengenalan Yii Framework	22
A. 21	
A. 1. 21	
A. 2. 21	

- A. 3. 22
- A. 4. 22
- A. 5. 24
- A. 6. 25
- B. 25
 - B. 1. 25
 - B. 2. 28
 - B. 3. 29
- C. 32
 - C. 1. 32
 - C. 2. 33
- D. 34
 - D. 1. 34
 - D. 2. 37
 - D. 3. 35
 - D. 4. **Error! Bookmark not defined.**
 - D. 5. 36
- E. 37

Model View Controller 39

- A. 38
- B. 40
 - B. 1. 40
 - B. 2. 41
- C. 41
- D. 43
 - D. 1. 43
 - D. 2. 44
- E. 44
- F. 49
- G. 50
- H. 52

Bekerja dengan Basis Data 52

- A. 53
- B. 53
 - B. 1. 53
 - B. 2. 54
 - B. 3. 54
- C. 56
 - C. 1. 56
 - C. 2. 56
- D. 58
 - D. 1. 58
 - D. 2. 62
 - D. 3. 63
- E. 63
 - E. 1. 64
 - E. 2. 66
 - E. 3. 66
 - E. 4. 68
 - E. 5. 69
- F. 72
 - F. 1. 72
 - F. 2. 72
 - F. 3. 74
- G. 79

Gii: Code Generator 75

- A. 81
- B. 82

<i>B. 1.</i>	82	
<i>B. 2.</i>	87	
<i>B. 3.</i>	88	
<i>C.</i>	91	
<i>C. 1.</i>	93	
<i>C. 2.</i>	93	
<i>D.</i>	98	
Ajax dan Pjax		91
<i>A.</i>	99	
<i>A. 1.</i>	99	
<i>A. 2.</i>	99	
<i>A. 3.</i>	99	
<i>B.</i>	102	
<i>B. 1.</i>	102	
<i>B. 2.</i>	102	
<i>B. 3.</i>	102	
<i>C.</i>	106	
<i>C. 1.</i>	106	
<i>C. 2.</i>	107	
<i>C. 3.</i>	107	
<i>D.</i>	117	
Layout dan Module Aplikasi		111
<i>A.</i>	119	
<i>A. 1.</i>	119	
<i>A. 2.</i>	120	
<i>B.</i>	122	
<i>B. 1.</i>	122	
<i>B. 2.</i>	123	
<i>B. 3.</i>	125	
<i>C.</i>	126	
Bekerja dengan Extension		119
<i>A.</i>	127	
<i>B.</i>	127	
<i>B. 1.</i>	127	
<i>B. 2.</i>	128	
<i>B. 3.</i>	128	
<i>B. 4.</i>	129	
<i>C.</i>	130	
<i>C. 1.</i>	130	
<i>C. 2.</i>	132	
<i>C. 3.</i>	133	
<i>C. 4.</i>	137	
<i>D.</i>	139	
Code Testing		133
<i>A.</i>	141	
<i>A. 1.</i>	141	
<i>A. 2.</i>	141	
<i>B.</i>	142	
<i>B. 1.</i>	142	
<i>B. 2.</i>	143	
<i>B. 3.</i>	143	
<i>B. 4.</i>	145	
<i>B. 5.</i>	147	
<i>C.</i>	158	
<i>C. 1.</i>	158	
<i>C. 2.</i>	159	
<i>D.</i>	161	

Otentikasi Pengguna	155
A. 163	
A. 1. 163	
A. 2. 163	
A. 3. 164	
A. 4. 166	
A. 5. 166	
A. 6. 167	
B. 167	
B. 1. 167	
B. 2. 171	
B. 3. 174	
C. 177	
C. 1. 177	
C. 2. 177	
C. 3. 178	
C. 4. 178	
C. 5. 189	
C. 6. 191	
C. 7. 200	
D. 205	
Otorisasi Pengguna	199
A. 207	
B. 207	
B. 1. 207	
B. 2. 207	
B. 3. 208	
B. 4. 209	
B. 5. 210	
B. 6. 212	
C. 217	
C. 1. 217	
C. 2. 218	
C. 3. 219	
C. 4. 219	
C. 5. 219	
D. 225	
Unggah, Impor & Pelaporan	219
A. 227	
A. 1. 227	
A. 2. 227	
A. 3. 228	
A. 4. 232	
A. 5. 234	
B. 235	
C. 238	
C. 1. 238	
C. 2. 240	
C. 3. 241	
D. 242	
D. 1. 243	
D. 2. 249	
E. 252	
Web Service	246
A. 253	
A. 1. 253	
A. 2. 253	
A. 3. 253	

- B. 254
 - B. 1.* 254
 - B. 2.* 254
 - B. 3.* 255
 - B. 4.* 268
 - B. 5.* 269
- C. 275
 - C. 1.* 275
 - C. 2.* 277
- D. 282
 - D. 1.* 282
 - D. 2.* 283
- E. 284

Tips dan Trik**278**

- A. 285
- B. 286
- C. 293
- D. 295
- E. 296
 - E. 1.* 296
 - E. 2.* 296
 - E. 3.* 297
 - E. 4.* 297
 - E. 5.* 297
- F. 297
 - F. 1.* 297
 - F. 2.* 298
 - F. 3.* 300
- G. 302
 - G. 1.* 302
 - G. 2.* 307
 - G. 3.* 308
 - G. 4.* 308
 - G. 5.* 309
- H. 309
 - H. 1.* 309
 - H. 2.* 309
 - H. 3.* 310
 - H. 4.* 310
 - H. 5.* 310
 - H. 6.* 311
- I. 311

Deploying Aplikasi**306**

- A. 312
 - A. 1.* 312
 - A. 2.* 312
- B. 313
 - B. 1.* 313
 - B. 2.* 318
- C. 328

Daftar Pustaka**322**

Gambaran Umum Buku

Buku yang sedang anda baca ini merupakan buku edisi kedua yang berjudul “Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework”. Secara umum topik yang disampaikan pada buku ini masih sama dengan edisi pertama yaitu fokus membahas pengembangan aplikasi berbasis web menggunakan sebuah *framework* PHP yang cukup *powerful* dan memiliki waktu pengembangan yang sangat cepat, yaitu Yii Framework versi 2, mulai dari awal (instalasi *framework*) sampai dengan publikasi aplikasi (*deployment*)

Untuk selanjutnya, penulis hanya akan menggunakan kata “Yii” saja, namun yang dimaksud adalah Yii *Framework* versi 2.

Penulis berusaha menyajikan materi secara sistematis disertai dengan topik-topik lain yang membantu pemahaman anda akan topik utama.

Belajar sebuah bahasa pemrograman atau *framework* tentu sangat menyenangkan apabila menggunakan studi kasus. Oleh karena itu, sebenarnya penulis ingin mengangkat studi kasus yang sudah sama-sama kita fahami alurnya yaitu toko daring. Adapun karena keterbatasan waktu, sehingga studi kasus tersebut belum tuntas. Namun meskipun demikian, materi yang diberikan pada buku ini sudah lebih dari cukup bagi anda untuk membangun sebuah aplikasi toko daring bahkan lebih dari itu.

Kode sumber dan berbagai bahan yang dibutuhkan pada panduan dalam buku ini bisa anda unduh melalui URL

<https://github.com/hscstudio/yii2-book-id>

selain itu juga penulis gunakan sebagai tempat untuk berdiskusi tentang materi pada buku ini.

A. Teknologi yang Digunakan

Teknologi yang digunakan pada panduan buku ini antara lain.

A. 1. Bahasa Pemrograman

PHP

PHP telah berevolusi dari bahasa pemrograman untuk tujuan yang sederhana yaitu mencatat pengunjung blog menjadi bahasa pemrograman yang banyak digunakan hingga level *enterprise*. Hal itu dimulai dari dukungan terhadap konsep-konsep pemrograman berorientasi objek atau *object oriented programming* (OOP), pembuatan standardisasi kode, pembaharuan *engine* PHP yaitu versi 7 dengan karakteristik utama adalah kecepatan, hingga dukungan teknologi-teknologi lain seperti Git, dan Composer. Untuk lebih jelasnya silakan kunjungi situs web resminya di <http://www.php.net>

Javascript

Bahasa pemrograman yang berjalan di sisi peramban (*browser*) sebagai pelengkap dalam pembuatan aplikasi berbasis web. Secara *default*, Yii menggunakan *library* JQuery.

A. 2. Framework

Yii Framework

Yii merupakan salah satu *framework* PHP yang cukup *powerfull* dan *professional* untuk digunakan dalam mengembangkan berbagai aplikasi berbasis web baik berskala kecil maupun besar. Situs web resminya bisa kita jumpai pada alamat URL <http://www.yiiframework.com>

Penjelasan lebih detail tentang Yii akan dibahas pada bab selanjutnya.

Twitter Bootstrap

Twitter Bootstrap merupakan *framework user interface* yang sangat populer digunakan sebagai dasar dari aplikasi-aplikasi berbasis web. Secara *default*, Yii telah menjadikannya sebagai “*theme*” pada *widget-widget*-nya.

∞ <http://www.getbootstrap.com>

A. 3. Basis Data

MySQL merupakan sistem manajemen basis data yang sangat populer, dan cocok disandingkan dengan PHP. *Tools* ini gratis untuk digunakan pada proyek yang bukan komersial.

Kita juga bisa menggunakan versi komunitas dari MySQL yaitu Mariadb, yang merupakan pengembangan dari MySQL, setelah MySQL diakuisisi oleh Oracle.

∞ <http://www.mysql.com>

B. Untuk Siapa Buku ini?

Buku ini cocok untuk siapapun yang ingin meningkatkan kemampuannya terutama dalam bidang *web programming* menggunakan bahasa pemrograman PHP. Namun untuk dapat mengikuti panduan yang ada dalam buku ini dengan baik, maka anda harus mempunyai pengetahuan yang cukup tentang bahasa pemrograman PHP.

Berikut ini gambarannya:

- Jika anda pernah membuat kode PHP untuk koneksi ke basis data, menampilkan data, dan mengedit data, maka itu modal yang cukup bagi anda untuk mengikuti panduan pada buku ini.
- Jika anda adalah pemula, yang benar-benar baru atau awam terhadap bahasa pemrograman PHP, maka penulis menyarankan kepada anda agar belajar terlebih dahulu tentang dasar-dasar bahasa tersebut.
- Jika anda menguasai pemrograman berbasis objek (OOP) di PHP, atau pernah menggunakan bahasa pemrograman yang kuat OOP-nya seperti Java, atau pernah menggunakan *framework* PHP lain, maka itu adalah nilai tambah dalam mempelajari buku ini.

C. Sistematika Penulisan

Buku ini disusun dengan sistematika penulisan sebagaimana yang bisa anda lihat pada daftar isi. Penulis berusaha membuat urutan yang sedemikian rupa sehingga akan memudahkan anda dalam mengikuti setiap materi dalam buku ini. Penulis menjaga agar kurva belajar anda tidak terlalu menanjak tajam.

Setiap materi yang disampaikan, diawali dengan dasar teori, baru kemudian panduan serta praktik terbaik pada Yii.

Karena merupakan buku pemrograman sehingga anda akan menjumpai banyak sekali kode-kode program. Adapun penulisan kode program pada buku ini mengikuti ketentuan berikut:

- Komentar pada kode program umumnya akan penulis hapus, hal ini dikarenakan terbatasnya *space* buku ini. Namun penulis tetap berprinsip bahwa komentar merupakan bagian yang penting dalam penulisan kode program.
- Kode program seringkali tidak utuh melainkan merupakan potongan kode. Penulis seringkali hanya mengutip bagian penting dari kode. Oleh karena itu, hendaknya anda lebih cermat terkait hal ini.
- Semua kode program yang ada dalam buku ini telah diuji coba pada komputer penulis dan berjalan dengan baik. Apabila mendapati galat, mungkin disebabkan karena ada langkah sebelumnya yang belum anda ikuti.

1

Persiapan Lingkungan Kerja

Kita akan belajar tentang:

- Instalasi *tools-tools* yang dibutuhkan.
- Konfigurasi *tools-tools* yang dibutuhkan.
- Uji coba *tools-tools* yang dibutuhkan.

Ada beberapa *tools* atau *software* yang perlu kita siapkan atau pasang pada komputer kita sebelum mengikuti setiap materi yang ada pada buku ini.

Perhatian:

Untuk dapat mengikuti panduan dalam buku ini dengan baik, maka pastikan anda memiliki koneksi internet yang baik. Hal ini dikarenakan beberapa panduan membutuhkan koneksi internet seperti: Composer dan Git.

A. Tools yang Dibutuhkan

Tools-tools yang dibutuhkan untuk mengikuti panduan pada buku ini yaitu:

A. 1. Web Browser

Karena merupakan aplikasi berbasis web, maka kita perlu *web browser* untuk mengakses aplikasi kita. Ada dua *web browser* yang penulis rekomendasikan yaitu Google Chrome dan Mozilla Firefox, silakan pilih salah satu saja dan pastikan kita menggunakan versi yang terbaru.

A. 2. Web Server

Merupakan *tools* yang terletak di sisi server untuk menangani permintaan dari *client*. Ada dua *web server* yang umum digunakan yaitu Apache dan NginX (dibaca: *engine X*). Meskipun pada saat pengembangan kita sebenarnya tidak harus menggunakan *web server*, karena PHP telah memiliki *built-in web server* sendiri.

A. 3. PHP

Merupakan *tools* untuk menjalankan bahasa pemrograman PHP karena Yii sendiri merupakan *framework* PHP. Adapun versi PHP yang digunakan adalah versi 5.4 keatas, hal ini berkaitan dengan Yii 2.0 yang membutuhkan minimal PHP versi 5.4. Anda juga bisa mencoba PHP versi 7 yang sudah *launching* versi stabilnya dan Yii berjalan dengan baik di PHP 7.

A. 4. Basis Data

Basis data atau sistem manajemen basis data merupakan alat untuk mengelola data dari aplikasi kita. Pada buku ini, penulis akan menggunakan basis data MySQL/MariaDB namun anda juga bisa menggunakan basis data lain misalnya PostgreSQL, Oracle dsb, tentunya dengan beberapa penyesuaian.

A. 5. Code Editor

Merupakan *tools* yang digunakan untuk mengetik kode program yang akan kita buat. Usahakan untuk menggunakan *code editor* yang paling kita suka dan membuat kita nyaman karena dialah yang akan banyak menemani kita dalam pembuatan aplikasi.

Penulis merekomendasikan untuk menggunakan Netbeans (gratis dan mendukung Yii) atau jika itu terasa berat di komputer maka anda bisa menggunakan Visual Studio Code atau Notepad++ yang jauh lebih ringan namun dengan konsekuensi fitur yang dimilikinya tidak selengkap Netbeans. *Core developer* Yii sendiri merekomendasikan PHPStorm, yang merupakan software berbayar namun sangat *powerfull*. Alternatif lain yang bisa digunakan antara lain: Sublime, Aksiide, Atom atau semua PHP *editor* yang lain.

A. 6. “*Kitab Suci*”

Dokumentasi dari Yii yang berisi panduan dan daftar API mutlak diperlukan jika kita ingin serius mendalami *framework* ini. Adapun versi daring dari dokumentasi Yii bisa kita lihat pada tautan di bawah ini

- ∞ <http://www.yiiframework.com/doc-2.0/guide-README.html>

Adapun versi luringnya (html), bisa didapatkan pada tautan berikut

- ∞ <http://stuff.cebe.cc/doc-2.0.tgz>

B. Instalasi Apache, MySQL/MariaDB, dan PHP

Untuk menginstalasi ketiga *tools* di atas yaitu Apache, MySQL/MariaDB dan PHP, kita bisa menggunakan paket instalasi atau stack yang disediakan oleh <https://www.apachefriends.org> yaitu XAMPP

The screenshot shows the Apache Friends website with the XAMPP page highlighted. The page title is "XAMPP Apache + MariaDB + PHP + Perl". On the left, there's a section titled "What is XAMPP?" with text explaining it's a popular PHP development environment. To the right is a large image of the XAMPP logo, which is a stylized '83' inside a square with a play button icon in the center, and the word "XAMPP" below it.

XAMPP merupakan *tools* yang di dalamnya telah berisi Apache, MySQL/MariaDB, PHP dan Perl. XAMPP akan memudahkan kita melakukan instalasi *tools* yang diperlukan untuk pengembangan aplikasi web berbasis PHP & MariaDB. Sehingga kita tidak perlu menginstalasi dan mengkonfigurasi satu persatu pada ketiga tools tersebut melainkan sekaligus dalam satu kali instalasi dan konfigurasi.

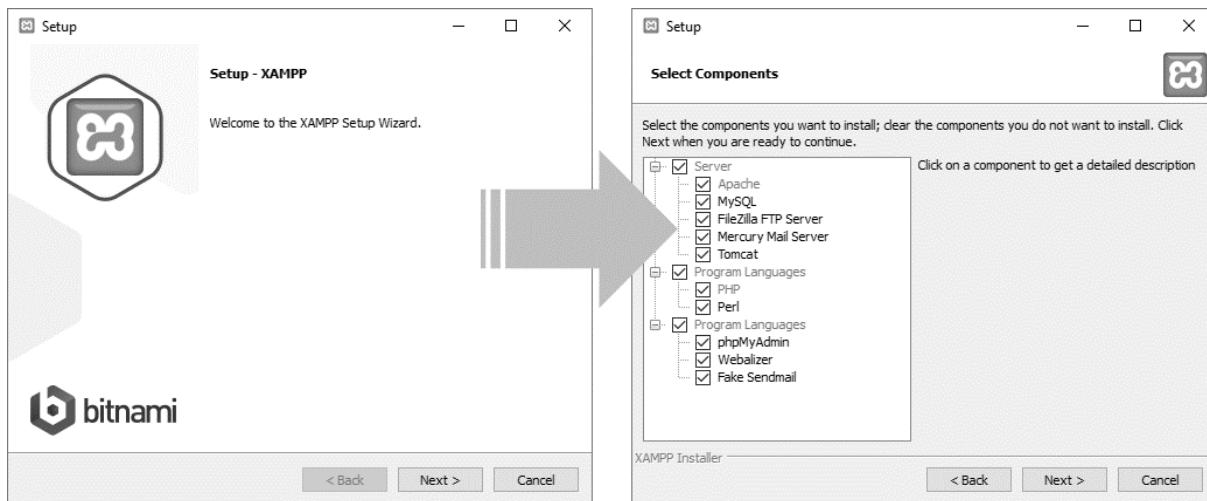
XAMPP menyediakan berbagai versi sistem operasi yaitu Windows, Linux dan MacOS. Selengkapnya silakan buka <https://www.apachefriends.org/download.html>

The screenshot shows the "XAMPP for Windows" download page. It features three download options for different PHP versions:

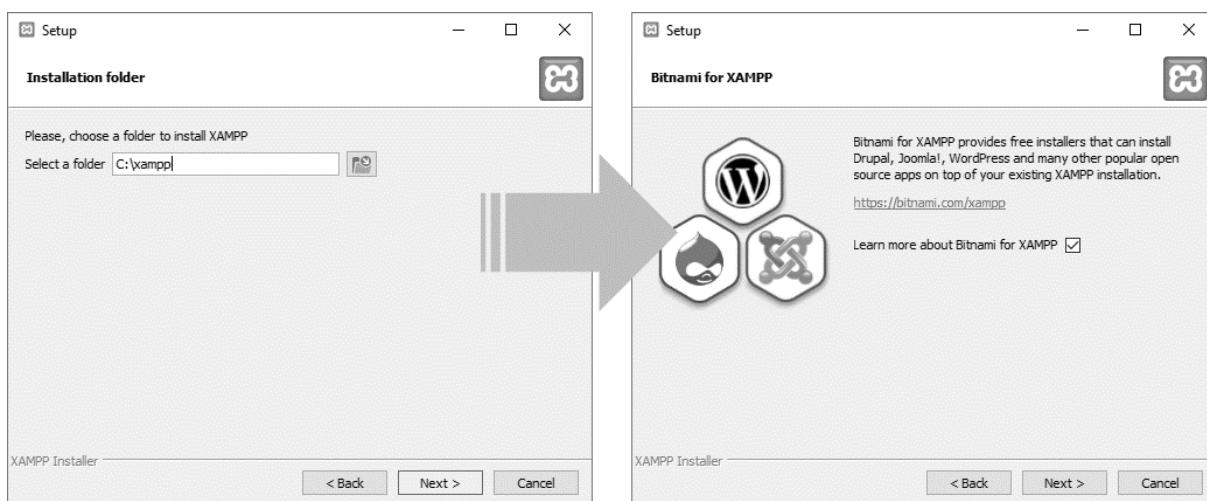
Version	Checksum	Size
5.5.38 / PHP 5.5.38	What's Included? md5 sha1	Download (32 bit) 106 Mb
5.6.28 / PHP 5.6.28	What's Included? md5 sha1	Download (32 bit) 109 Mb
7.0.13 / PHP 7.0.13	What's Included? md5 sha1	Download (32 bit) 119 Mb

Pada buku ini, XAMPP yang digunakan adalah versi Windows karena penulis menggunakan sistem operasi Windows. Adapun untuk sistem operasi lain silakan menyesuaikan. Versi PHP yang digunakan adalah versi 7. Setelah kita mengunduh XAMPP sesuai dengan sistem operasi yang kita gunakan, maka berikut ini panduan instalasinya.

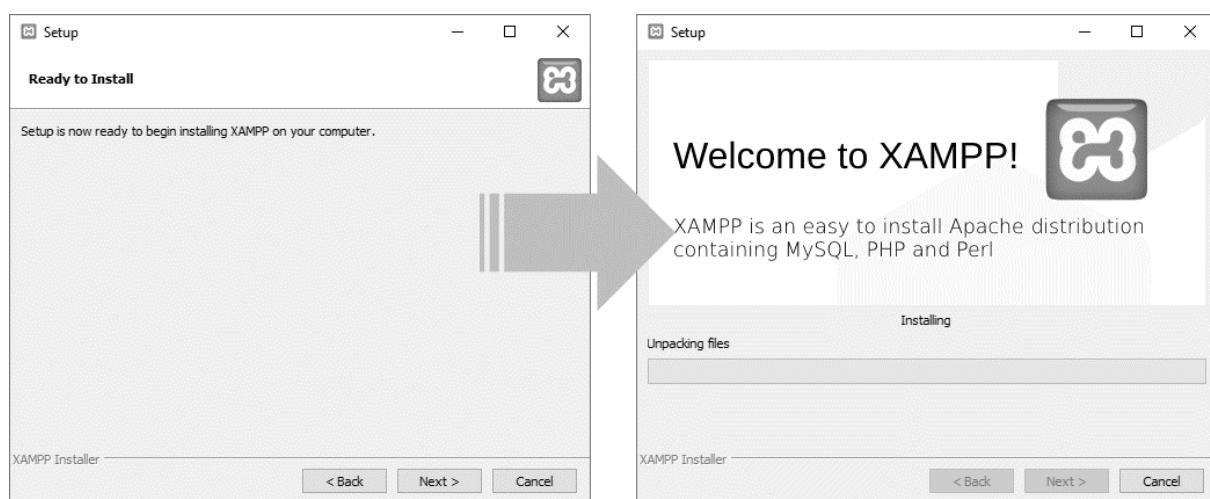
1. Klik dua kali file instalasi XAMPP yang telah diunduh sebelumnya. Pada jendela “Setup XAMPP” klik tombol “Next”, kemudian pilih komponen yang ingin diinstalasi.



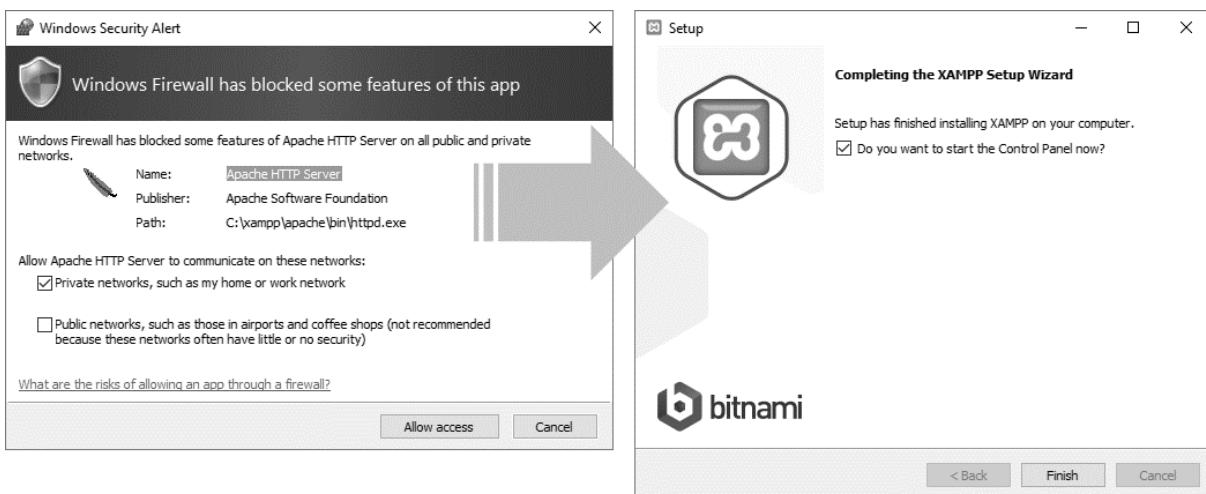
2. Pada jendela “Installation folder”, tentukan lokasi instalasi XAMPP, *default*-nya di C:\xampp



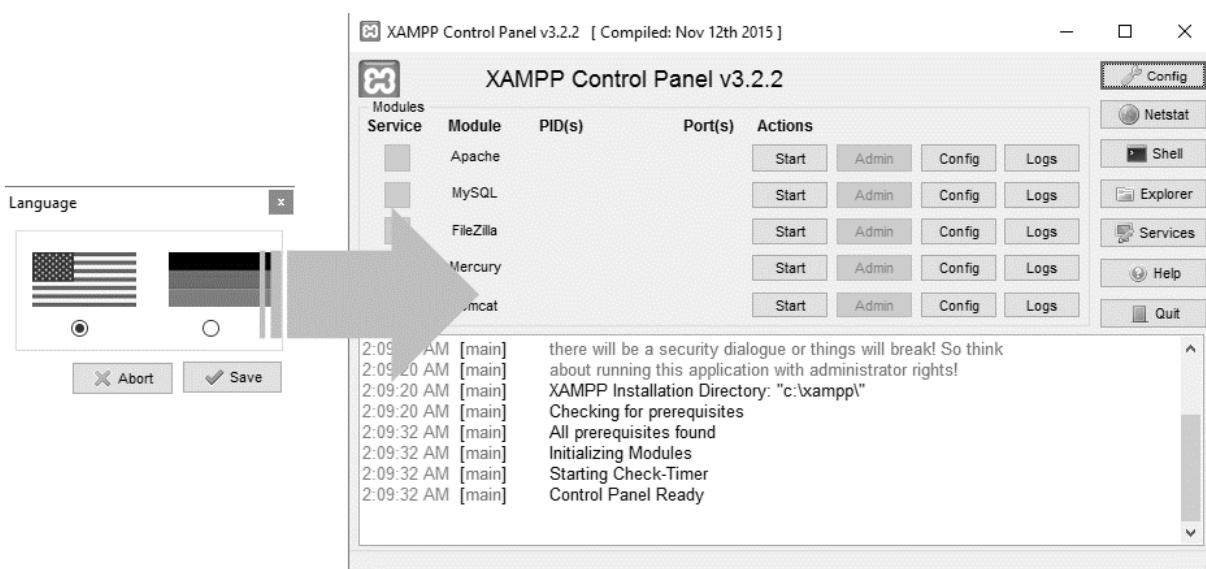
3. Pada jendela “Ready to Install” klik tombol “Next”, kemudian tunggu hingga *progressbar* instalasi selesai



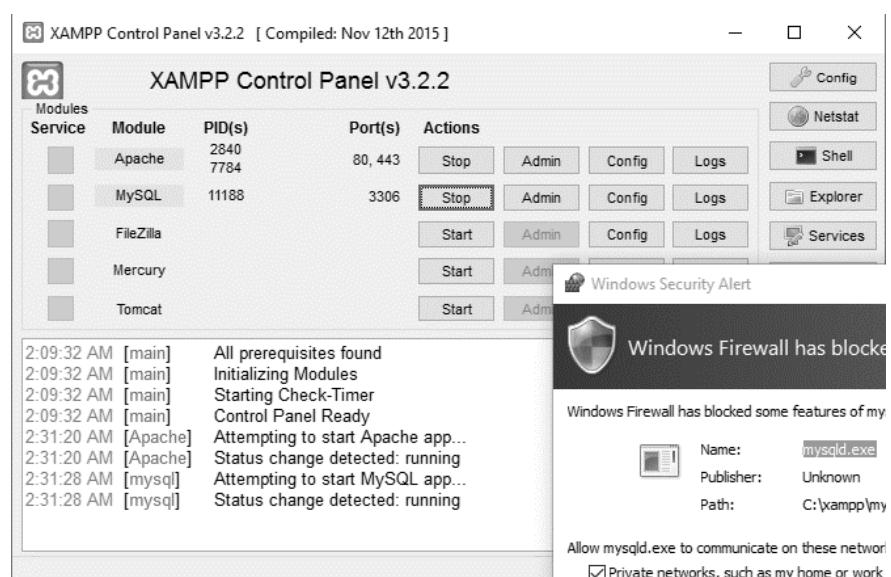
4. Setelah instalasi selesai maka akan muncul jendela “Windows Security Alert” yang berisi permintaan izin untuk menjalankan Apache, klik tombol “Allow access”, kemudian pada jendela berikutnya klik tombol “Finish”



5. Pilih bahasa yang digunakan lalu klik tombol “Save”, maka akan muncul jendela “XAMPP Control Panel”



6. Klik tombol “Start” pada *module* Apache dan MySQL. Klik Allow access jika jendela “Windows Security Alert” muncul



Jika kedua *module* tersebut telah berwarna hijau muda berarti instalasi berhasil.

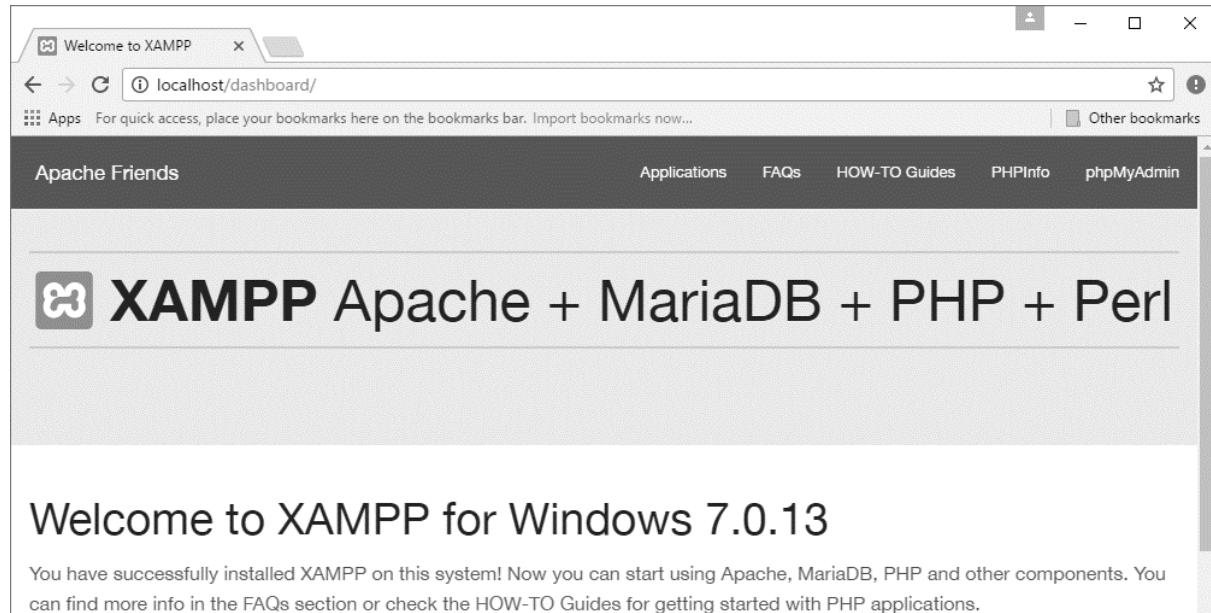
Adapun untuk instalasi XAMPP pada Mac OS X dengan cara mengklik dua kali *DMG image* kemudian ikuti langkah-langkah instalasinya maka ZAMPP akan terinstal pada direktori /Applications/XAMPP.

Catatan:

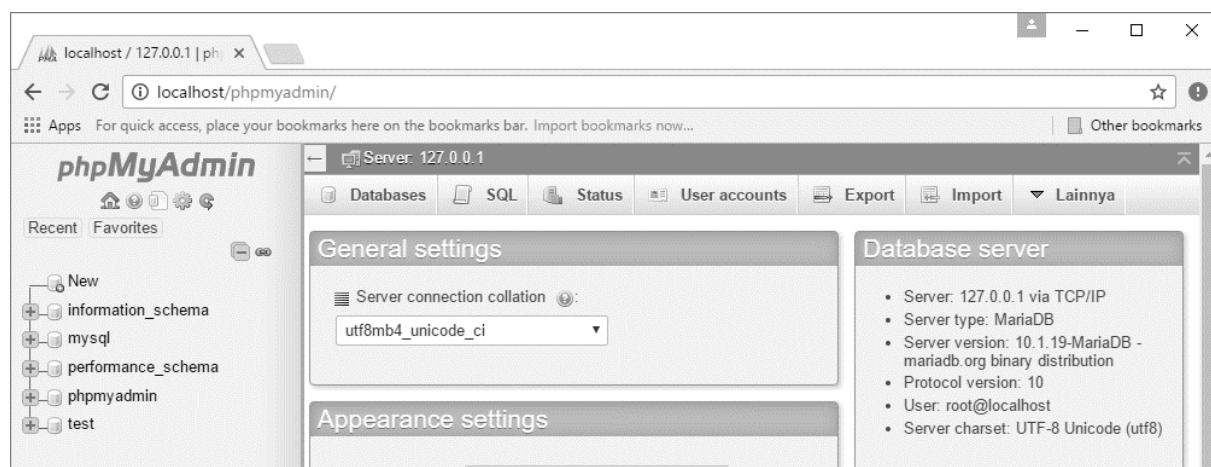
XAMPP dibuat untuk keperluan pengembangan aplikasi pada saat *development* saja, sehingga sangat tidak disarankan bagi kita menggunakan untuk *production*.

C. Uji Coba Instalasi

Untuk menguji apakah ketiga *tools* tersebut yaitu Apache, PHP dan MySQL/MariaDB telah terinstalasi pada komputer kita, adalah dengan membukanya melalui *web browser*. Mari kita buka *web browser* dan akses alamat <http://127.0.0.1> atau <http://localhost>, jika menampilkan informasi tentang XAMPP seperti berikut ini maka berarti instalasi berhasil, namun jika tidak maka kita perlu ulangi langkah sebelumnya.



Selain itu kita juga bisa mencoba mengakses *tools* phpmyadmin yang merupakan paket XAMPP melalui link <http://localhost/phpmyadmin>.



Dua tampilan di atas menunjukkan bahwa XAMPP telah terpasang dengan baik pada komputer kita.

Pada XAMPP, kode aplikasi PHP kita secara *default* diletakkan pada folder yang bernama “*htdocs*” atau biasa disebut *document root* atau *mount point* ketika kita mengakses URL <http://localhost/> (di Windows terletak pada direktori C:\xampp\htdocs), jika tidak diletakkan pada direktori tersebut maka kode kita tidak bisa diakses melalui *web browser*, meskipun bisa saja kita memindahkan pada direktori lain dengan sebelumnya melakukan pengaturan pada file httpd.conf (di Windows terletak di C:\xampp\apache\conf\httpd.conf).

```

245 DocumentRoot "C:/xampp/htdocs"
246 <Directory "C:/xampp/htdocs">
247 #
248 # Possible values for the Options

```

Meskipun bisa, namun untuk saat ini kita tidak perlu mengubah konfigurasi *document root* tersebut. Oleh karena itu, pastikan untuk selanjutnya kita meletakkan kode aplikasi PHP kita pada direktori tersebut.

D. Konfigurasi

Konfigurasi yang akan kita bahas di sini adalah konfigurasi variabel \$PATH untuk PHP. \$PATH adalah variabel global pada sistem operasi yang biasanya digunakan untuk mendefinisikan direktori dari *tools* atau aplikasi tertentu, sehingga untuk mengaksesnya akan lebih mudah, yaitu langsung menyebutkan namanya tanpa perlu menyebutkan direktori dari *tools* tersebut.

Silakan buka *command prompt* pada Windows atau *terminal* pada Linux/Mac OS X.

Perhatian!

Untuk kemudahan, maka selanjutnya, penulis hanya akan menggunakan istilah **CMD** untuk menyebut *command prompt* (di Windows) atau *terminal* (di Linux dan Mac OS X)

Kemudian ketikkan perintah berikut pada CMD.

```
| php -v
```

Perintah tersebut dimaksudkan untuk mengecek atau menampilkan versi PHP yang telah terpasang pada komputer kita.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\hafid>php -v
'php' is not recognized as an internal or external command,
operable program or batch file.

```

Apabila kita mendapati pesan galat (*error*) yang intinya menginformasikan bahwa perintah tersebut tidak dikenali pada CMD maka itu artinya PHP belum didaftarkan pada variabel \$PATH. Oleh karenanya, kita tidak bisa menjalankan PHP dengan menyebut namanya saja, melainkan harus lengkap dengan direktori.

Pada komputer penulis, lokasi *file* php.exe terletak pada direktori C:\xampp\php\php.exe (mungkin berbeda pada komputer anda, silakan menyesuaikan). Oleh karena itu perintahnya menjadi sebagai berikut.

```
| C:\xampp\php\php -v
```

```

C:\WINDOWS\system32\cmd.exe
C:\Users\hafid>C:\xampp\php\php -v
PHP 7.0.13 (cli) (built: Nov 8 2016 13:45:28) ( ZTS )
Copyright (c) 1997-2016 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies

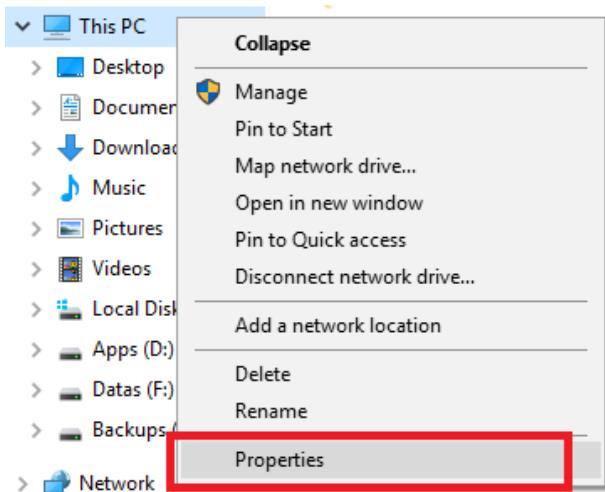
```

Untuk menyederhanakan perintah tersebut, maka pada kasus ini kita akan mendaftarkan PHP pada variabel \$PATH. Cara untuk mendaftarkannya tentu berbeda-beda, tergantung sistem operasi yang digunakan. Oleh karena itu berikut ini langkah-langkahnya berdasarkan sistem operasi yang kita gunakan.

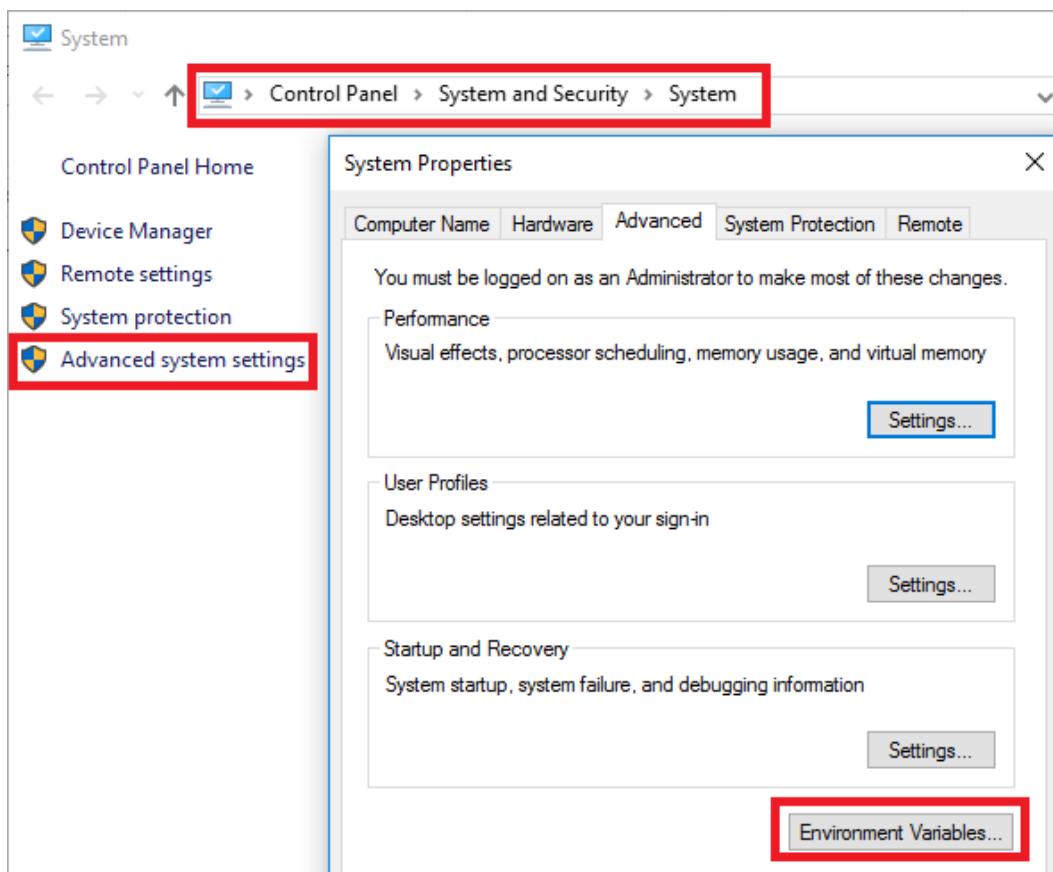
D. 1. Windows

Melalui User Interface

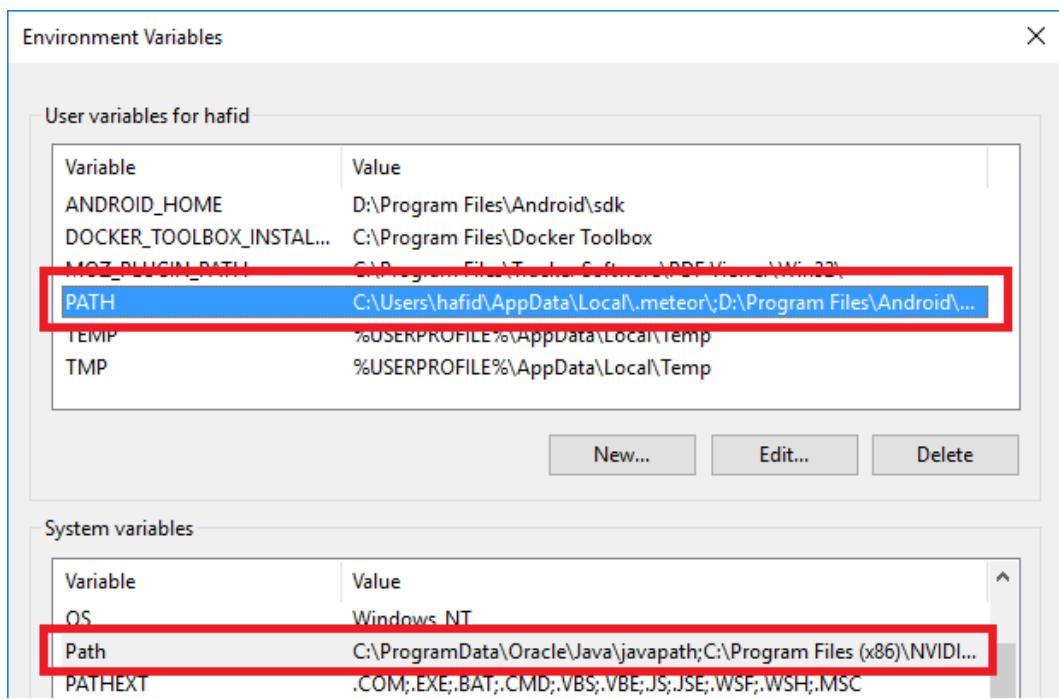
Pada *Windows Explorer*, klik kanan *This PC (My Computer)*, lalu pilih menu *Properties* (atau bisa juga melalui *Control Panel*, pilih *System and Security*, kemudian pilih *System*)



Maka kemudian akan muncul jendela "System". Pilih menu "Advanced system settings" yang berada dipanel sebelah kiri untuk menampilkan jendela "System Properties". Kemudian klik tombol "Environment Variables" untuk menampilkan jendela "Environment Variables".



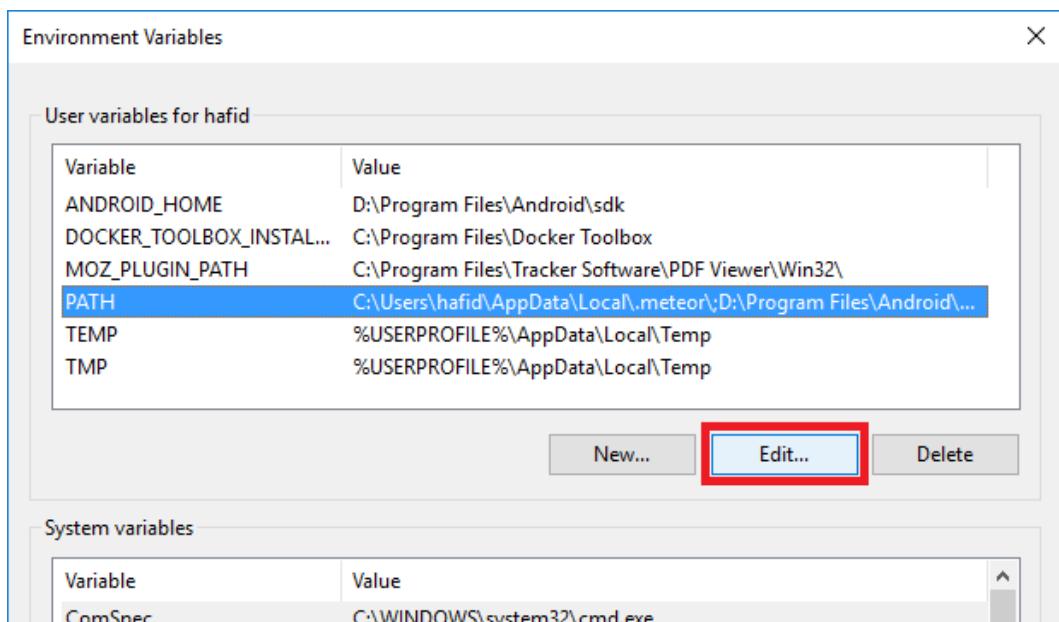
Pada jendela "Environment Variables" inilah kita bisa mengedit variabel \$PATH.



Catatan:

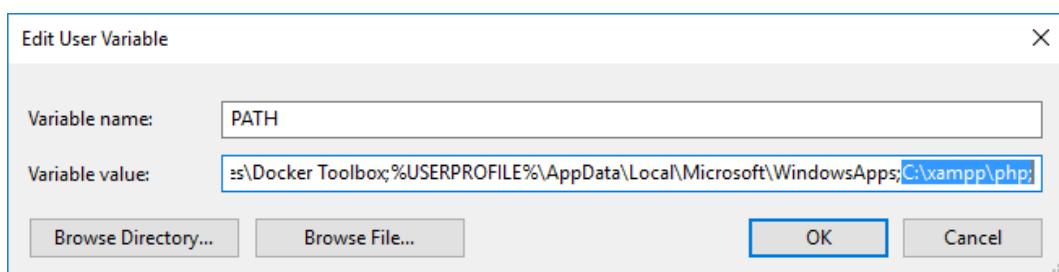
Pilih variable \$PATH yang ada di blok “User Variables” (atas) jika kita ingin mengeset variabel tersebut hanya untuk *current user*, atau “System Variables” (bawah) untuk semua *user* yang ada pada komputer.

Pada contoh ini kita mengeset variabel hanya untuk *current user*. Selanjutnya pada jendela “Environment Variables”, edit variabel \$PATH



Lalu tambahkan lokasi direktori PHP, sebagai berikut.

```
| C:\xampp\php;
```



Ingat:
Gunakan titik koma ; sebagai separator

Melalui Command Prompt

Kita juga bisa mengeset variabel \$PATH melalui CMD dengan menjalankan perintah berikut:

```
| SETX /M PATH "%PATH%;C:\xampp\php;"
```

Catatan

Perintah SETX /M berfungsi untuk mengeset variabel \$PATH secara permanen (meski di-restart tidak akan hilang) untuk semua *user* yang ada pada komputer tersebut. Jika tanpa parameter /M maka variabel tersebut hanya akan ada pada pengguna yang sedang aktif saja. Ada satu perintah lagi yaitu SET (tanpa x) yang berfungsi mengeset variabel juga namun tidak bersifat permanen, yaitu akan hilang apabila *command prompt* di close atau restart.

D. 2. Linux

Asumsinya lokasi *file PHP executable*-nya ada pada direktori /opt/lampp/bin/, maka pada *terminal* kita bisa menjalankan perintah.

```
| export PATH=${PATH}:/opt/lampp/bin
```

Fungsi dari perintah *export* di Linux setara dengan perintah *set* di Windows, dalam hal ini berfungsi menambahkan lokasi direktori *file PHP executable* ke dalam variabel \$PATH. Adapun titik dua : merupakan separator atau jika di Windows menggunakan titik koma ; sebagai separator.

Perintah di atas hanya berlaku pada satu sesi terminal, artinya jika terminal ditutup dan dibuka lagi maka kita perlu menjalankan perintah tersebut lagi. Solusi yang bisa kita lakukan agar pengaturan ini permanen adalah dengan menuliskan perintah tersebut pada *file .bash_profile*, yang mana file ini akan dijalankan pertama kali ketika terminal dibuka. Berikut ini perintah untuk melakukannya.

```
| echo 'export PATH=${PATH}:/opt/lampp/bin' >> ~/.bash_profile
```

D. 3. Mac OS X

Pada Mac OS X, nilai dari variabel \$PATH disimpan dalam *file /etc/paths*. Oleh karena itu untuk mendaftarkan direktori PHP pada variabel \$PATH, cukup dengan menambahkannya pada *file* tersebut.

Dengan asumsi bahwa *file executable PHP* ada di direktori /Applications/XAMPP/xamppfiles/bin/, maka berikut ini perintah yang bisa kita lakukan.

1. Pada terminal, jalankan perintah berikut

```
| sudo nano /etc/paths
```

2. Masukkan kata sandi pada *prompt* yang muncul
3. Masukkan lokasi direktori PHP executable pada baris terakhir
4. Tekan **CTRL-X** untuk keluar
5. Tekan **Y** untuk menyimpan perubahan yang kita lakukan

Lalu sekarang kita coba tes untuk memastikan bahwa PHP sudah terdaftar pada variabel \$PATH.

```
| echo $PATH
```

Seharusnya *terminal* akan menampilkan daftar variabel \$PATH, contohnya sebagai berikut.

```
| /usr/bin:/bin:/usr/sbin:/sbin:/Applications/XAMPP/xamppfiles/bin/
```

Ada cara lain yang bisa kita lakukan untuk mendaftarkan pada variabel \$PATH di Mac OS X yaitu dengan menggunakan cara yang sama dengan yang kita lakukan di Linux. Melalui *terminal*, pada direktori *home* jalankan perintah berikut.

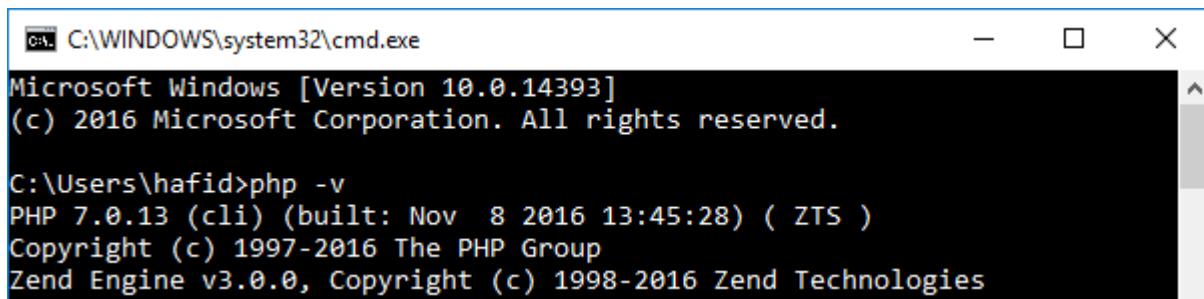
```
| echo 'export PATH="$PATH:/Applications/XAMPP/xamppfiles/bin/"' >> .bash_profile
```

D. 4. Uji Coba

Setelah kita mendaftarkan variabel \$PATH, maka sekarang kita bisa menguji coba apakah konfigurasi kita berhasil atau gagal. Caranya dengan menjalankan perintah berikut pada CMD.

```
| php -v
```

Berikut ini contoh pada komputer penulis.



C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\hafid>php -v
PHP 7.0.13 (cli) (built: Nov 8 2016 13:45:28) ( ZTS )
Copyright (c) 1997-2016 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
```

Mudah sekali bukan?

Perhatian:

Pengaturan variabel \$PATH pada panduan ini mungkin akan sedikit berbeda dengan pengaturan di komputer anda, hal ini berkaitan dengan versi OS yang anda gunakan mungkin berbeda dengan yang digunakan oleh penulis.

E. Kesimpulan

Pada bab ini kita telah belajar tentang instalasi dan konfigurasi *tools-tools* yang dibutuhkan selama mengikuti panduan dalam buku ini. Mungkin ada sedikit perbedaan dengan apa yang disampaikan pada buku ini, bisa jadi karena perbedaan versi *tools* dan OS. Oleh karena itu, penulis menyarankan dalam membaca buku ini maupun buku teknologi yang lain dengan cara memahami konsep atau maksud panduan yang ada dalam buku kemudian menerapkannya pada lingkungan komputer sendiri. Hal ini dikarenakan perkembangan teknologi yang begitu cepat sehingga bisa jadi *tools* yang sama namun beda versi memiliki cara-cara yang berbeda.

Persiapan lingkungan kerja pada bab ini sebenarnya merupakan cara yang paling sederhana. Kita bisa menggunakan *tools* yang menggunakan konsep virtualisasi dan *container* seperti Vagrant dan Docker. Terdapat banyak keunggulan jika kita menggunakan *tools* dengan konsep tersebut, antara lain: instalasinya lebih bersih tanpa banyak “mengotori” komputer kita dengan *tools-tools development*, kemudahan dalam *deployment* aplikasi dan yang pastinya lebih kekinian ☺.

Setelah bab ini kita akan belajar tentang Composer, yaitu *tools* yang sangat penting untuk dikuasai oleh pemrogram aplikasi berbasis PHP saat ini. Hal ini dikarenakan Composer sudah menjadi standard dalam pengembangan aplikasi berbasis PHP.

2

Bekerja dengan Composer

Kita akan belajar tentang:

- Composer
- Instalasi Composer
- Git dan Github
- Cara menggunakan Composer

A. Gambaran Umum Composer

A. 1. Apa itu Composer?



Composer adalah *dependency manager* untuk PHP, yaitu sebuah *tools* yang digunakan untuk memudahkan kita dalam mengelola *library* PHP yang digunakan pada aplikasi kita beserta dependensinya.

Sebagai gambaran, misalnya kita memiliki suatu aplikasi yang membutuhkan *library reporting* untuk meng-generate laporan, di mana *library reporting* tersebut tidak berdiri sendiri yaitu membutuhkan *library* lain sebut saja fpdf (untuk meng-generate dokumen berformat PDF) dan phpxcel (untuk meng-generate dokumen berformat *spreadsheet*). Maka dengan menggunakan composer, kita cukup menjalankan perintah untuk melakukan instalasi *library reporting* saja, selanjutnya composer akan secara otomatis melakukan instalasi *library* lain yang dibutuhkan oleh *library* tersebut.

Untuk lebih jelasnya, kita bisa mengunjungi langsung situs web resminya <http://www.getcomposer.org>

Sebenarnya, Composer bukanlah teknologi atau terobosan baru. Beberapa bahasa pemrograman atau *tools* lain telah menggunakan cara yang relatif sama dengan cara kerja Composer. Jika anda adalah pengguna Linux maka tentu hal ini sudah tidak asing lagi, misalnya pada distro Ubuntu, apa yang dilakukan oleh Composer mirip dengan perintah **apt-get** untuk melakukan instalasi atau memperbarui suatu aplikasi. Di luar PHP, sebut saja NodeJs juga telah menggunakan *tools* serupa untuk instalasi *library*-nya, *tools* itu bernama NPM.

Apakah Mengembangkan Aplikasi berbasis PHP harus menggunakan Composer?

Pengembangan aplikasi berbasis PHP tidaklah mutlak harus menggunakan Composer, hanya saja pada saat ini Composer telah menjadi bagian dari standard pengembangan aplikasi berbasis PHP. Banyak *library* PHP yang memang sejak awal di desain untuk mendukung Composer sehingga tidak ada panduan untuk menginstalasinya secara manual lagi, tentu hal ini akan merepotkan kita jika kita memaksakan diri untuk tidak menggunakannya. Di samping itu, umumnya *framework* PHP modern menggunakan dan mendukung Composer.

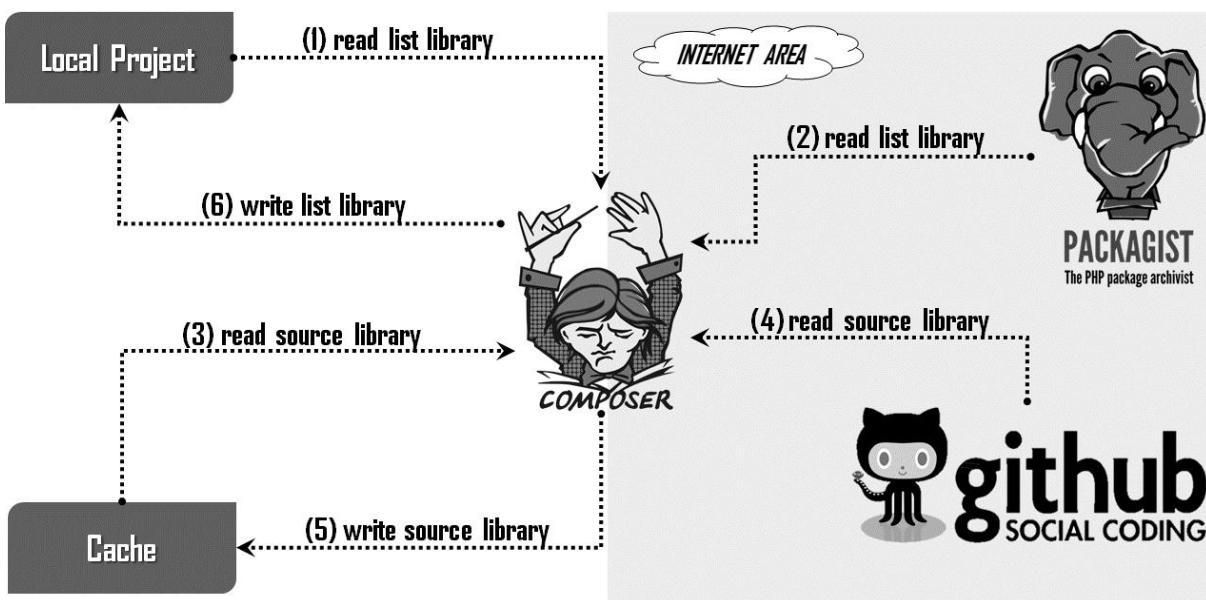
Adapun untuk tujuan pembelajaran Yii pada buku ini, maka Composer mutlak diperlukan.

Catatan:

Composer bisa berjalan dengan baik pada OS: Windows, Mac OS X, dan Linux dan versi minimal PHP adalah 5.3.2+

A. 2. Cara Kerja Composer

Berikut ini gambaran sederhana tentang alur dan cara kerja dari Composer.



Keterangan.

1. Mula-mula Composer membaca daftar *library* yang ingin kita instalasi yang mana daftar tersebut telah kita definiskan terlebih dahulu. Misalnya kita ingin melakukan instalasi *library* Yii Framework versi 2.0.
2. Composer mengecek apakah di lokal komputer (*cache*) telah ada versi yang dimaksud. Jika di lokal komputer kita ada versi tersebut maka Composer akan mengecek lagi apakah *library* dimaksud membutuhkan dependensi dengan *library* lain atau tidak.
3. Jika *library* tersebut membutuhkan *library* lain, maka Composer akan mencoba melakukan instalasi atau menyalin *library* dependensi tersebut. Kemudian setelah itu, Composer akan memasang *library* utamanya ke *current* direktori.
4. Apabila di lokal komputer tidak terdapat *library* dan dependensi dimaksud, maka Composer akan menghubungi server yang dalam hal ini adalah packagist.org (*default*).

Catatan:

Packagist – PHP package archivist – yaitu sebuah server *repository* yang menyimpan daftar URL dari *library - library* PHP seluruh dunia. Adapun kode *library* tersebut disimpan pada *hosting* lain. Sebagai contoh, framework Yii meng-hosting kodennya di <https://github.com>

5. Setelah menemukan paket dimaksud maka kemudian Composer mengunduhnya dan memasangnya pada *current directory*. Di samping itu, Composer juga menyimpan paket tersebut pada *cache* komputer kita. Sehingga jika suatu saat kita akan menginstalnya lagi maka Composer cukup mengambil dari lokal komputer tanpa perlu mengunduh dari server lagi.

Demikianlah kira-kira cara kerja Composer. Oleh karena Composer terhubung ke server internet, maka kita perlu koneksi internet yang memadai untuk bisa menggunakan Composer dengan baik.

A. 3. Mengapa menggunakan Composer?

Sebagaimana yang penulis sebutkan di bagian pertama, bahwa Composer akan memudahkan kita dalam memasang suatu *library* PHP beserta dependensinya. Jadi kata kuncinya adalah memudahkan kita.

Kita sebenarnya bisa saja memasang *library library*-*library* tersebut secara manual satu per satu kemudian mengkonfigurasinya. Namun, selain memakan waktu dan tidak efisien, cara manual ini juga sudah mulai ditinggalkan, yang mana saat ini banyak *library* PHP yang merekomendasikan untuk instalasi menggunakan Composer tanpa ada dokumentasi atau penjelasan tentang bagaimana cara melakukan instalasinya secara manual.

Pada sebuah proyek aplikasi yang besar, dengan banyak *library* yang digunakan, maka memasang dengan cara manual tentu akan sangat merepotkan dan membutuhkan waktu yang lama. Jadi di samping memudahkan, Composer juga akan membuat pekerjaan kita lebih efektif dan efisien.

B. Instalasi Composer

B. 1. Cara Instalasi

Composer merupakan *tools* yang perlu kita instalasi terlebih dahulu pada komputer kita. Ada dua cara instalasi.

Menggunakan Installer (Windows)

Cara ini hanya berlaku jika kita menggunakan OS Windows sekaligus merupakan cara yang paling mudah. Unduh dan jalankan file instalasi Composer-Setup.exe yang bisa kita dapatkan pada tautan berikut.

∞ <https://getcomposer.org/Composer-Setup.exe>

Installer ini akan mengunduh dan melakukan instalasi versi terbaru dari Composer dan kemudian mengatur konfigurasi path-nya sehingga Composer bisa digunakan pada direktori manapun di komputer kita.

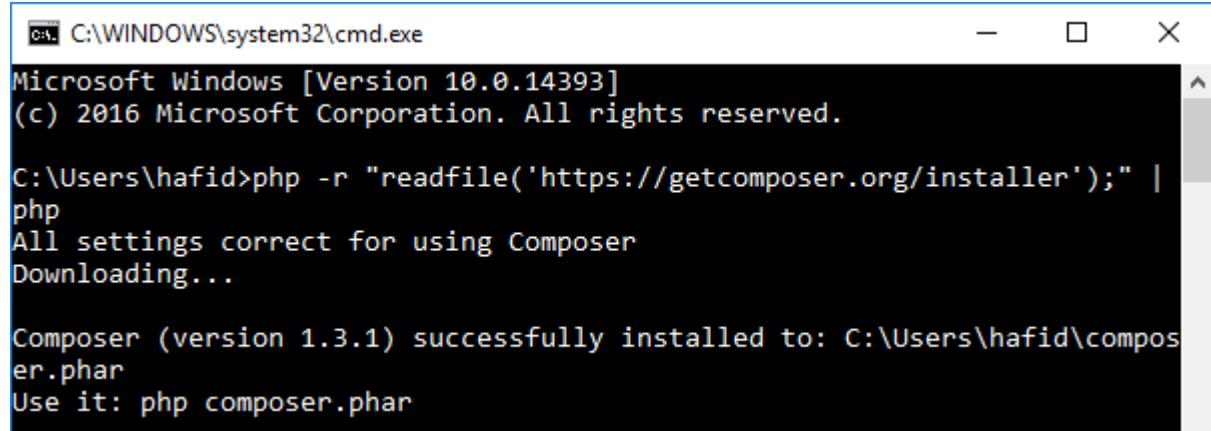
Adapun panduan instalasi ini juga bisa kita jumpai pada tautan berikut

∞ <https://getcomposer.org/doc/00-intro.md#installation-windows>

Menggunakan CMD (Windows)

Melalui CMD jalankan perintah berikut

```
| php -r "readfile('https://getcomposer.org/installer');" | php
```



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\hafid>php -r "readfile('https://getcomposer.org/installer');" | php
All settings correct for using Composer
Downloading...

Composer (version 1.3.1) successfully installed to: C:\Users\hafid\composer.phar
Use it: php composer.phar
```

Catatan:

Jika langkah di atas gagal, maka gunakan URL http (bukan https) atau aktifkan php_openssl.dll di php.ini

Perintah tersebut sebenarnya adalah mengunduh file composer.phar pada alamat URL <https://getcomposer.org/installer>. Pada contoh di atas, penulis menjalankan perintah tersebut pada direktori c:\Users\hafid, maka file composer.phar akan terunduh pada direktori tersebut yaitu di c:\Users\hafid\composer.phar

Setalah menginstalasi Composer (baca: mengunduh file composer.phar) maka kemudia kita bisa menggunakan Composer dengan perintah berikut.

```
| php composer.phar
```



```
C:\WINDOWS\system32\cmd.exe
C:\Users\hafid>php composer.phar
[Progress Bar]
Composer version 1.3.1 2017-01-07 18:08:51
```

Tampilan tersebut menunjukkan bahwa instalasi Composer berhasil.

Perhatian:

Perintah di atas hanya bisa kita jalankan pada direktori dimana file composer.phar berada yaitu c:\Users\hafid, adapun jika kita ingin menggunakannya Composer pada direktori manapun di komputer maka kita perlu mendaftarkannya sebagai variabel \$PATH

Kita bisa mempersingkat perintah eksekusi Composer dengan cara menuliskan kode panjangnya pada file .bat sebagai berikut.

```
| echo @php "%~dp0composer.phar" %*>composer.bat
```

Agar Composer bisa digunakan pada lokasi manapun di komputer maka kita perlu daftarkan direktori dari file composer.bat ini ke dalam variabel \$PATH. Sebagai asumsi, file composer.bat ada pada direktori c:\Users\hafid, maka melalui CMD kita bisa menjalankan perintah berikut.

```
| SETX /M PATH "%PATH%;C:\Users\hafid"
```

Restart CMD terlebih dahulu, maka sekarang kita bisa menggunakan Composer dengan perintah yang lebih pendek.

```
| composer
```

Menggunakan CMD (Linux & Mac OS X)

Melalui CMD, jalankan perintah berikut:

```
| curl -sS https://getcomposer.org/installer | php
```

Jika perintah di atas gagal, maka kita bisa menggunakan cara yang hampir sama dengan cara yang kita lakukan pada instalasi di Windows, yaitu pada CMD dengan menjalankan perintah berikut.

```
| php -r "readfile('https://getcomposer.org/installer');" | php
```

Agar Composer bisa digunakan pada lokasi manapun di komputer maka kita perlu menjalankan perintah berikut.

```
| mv composer.phar /usr/local/bin/composer
```

Perintah di atas adalah perintah untuk memindahkan file composer.phar hasil dari unduh, ke direktori global yaitu /usr/local/bin/composer.

Informasi:

Di OS X Yosemite, secara default, folder /usr tidak ada. Jika kemudian muncul galat "/usr/local/bin/composer: No such file or directory" maka kita perlu membuat folder /usr/local/bin/ secara manual sebelum menjalankan perintah di atas.

```
| mkdir /usr/local/bin
```

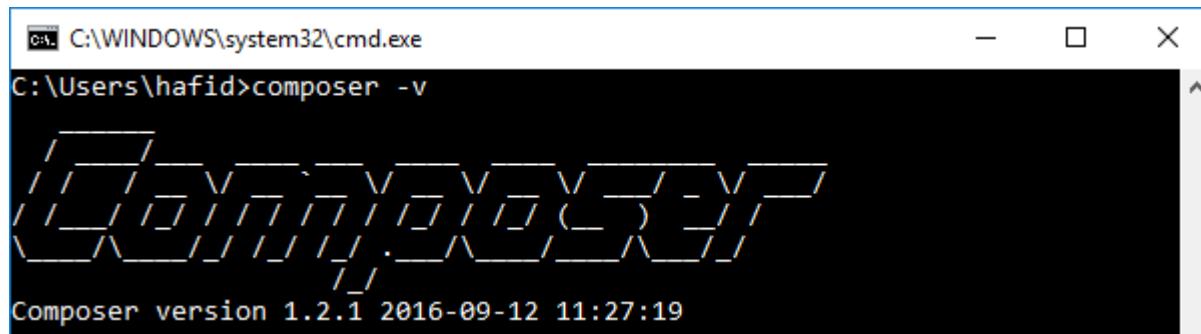
Kemudian, sebelum menggunakan Composer, silakan restart CMD terlebih dahulu.

B. 2. Uji Coba Composer

Untuk memastikan apakah Composer terpasang dengan benar, maka kita perlu mengujinya. Silakan tutup semua CMD, lalu kemudian buka lagi dan jalankan perintah berikut.

```
| composer -v
```

Maka seharusnya kita menjumpai tampilan yang kurang lebih sebagai berikut.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\hafid>composer -v
[decorative characters]
Composer version 1.2.1 2016-09-12 11:27:19
```

Informasi:

Karena telah kita singkat, maka kita cukup menggunakan perintah composer -v bukan lagi php composer.phar -v sehingga kita tidak perlu bingung apabila mendapat panduan di buku atau internet dengan menggunakan cara yang panjang, karena kedua cara itu sama saja.

B. 3. Persiapan Menggunakan Composer

Praktik terbaik sebelum kita menggunakan Composer maka disarankan untuk menjalankan perintah self-update pada Composer untuk memastikan bahwa Composer yang kita gunakan adalah yang terbaru (kadaluarsa setelah 30 hari).

```
| composer self-update
```

```
C:\Users\hafid>composer self-update
Updating to version 1.3.1 (stable channel).
  Downloading: 100%
Use composer self-update --rollback to return to version 1.2.1
```

Setelah berhasil mengupdate Composer maka kemudian kita bisa menggunakan Composer untuk melakukan instalasi atau memperbarui *library*. Untuk memahami lebih dalam lagi tentang topik ini maka kita bisa kita mempelajarinya melalui panduan resminya di <https://getcomposer.org/book.pdf>

C. Persiapan menggunakan Composer

Sebelum mulai menggunakan Composer untuk menginstalasi atau memperbarui *library*, maka ada beberapa hal yang perlu kita persiapkan atau fahami terlebih dahulu.

C. 1. Git & Github

Git merupakan *tools* yang berfungsi untuk mengontrol revisi kode secara terdistribusi dan mengelola kode sumber (*distributed revision control and source code management/SCM*). Dengan menggunakan Git, kita akan lebih mudah dalam mengontrol revisi kode sumber aplikasi kita, apalagi jika kita bekerja secara tim. Pada level mahir, dimana kita telah bisa membuat *extension-extension* Yii, maka *tools* ini wajib untuk diinstallasi pada komputer - <https://git-scm.com>.

Adapun Github - <https://github.com> - merupakan layanan berbasis *web hosting* untuk proyek-proyek pengembangan perangkat lunak yang menggunakan sistem kontrol revisi Git.

Analoginya, jika Git itu merupakan komputer server, maka Github adalah perusahaan penyedia *hosting* atau *cloud hosting*.

C. 2. Mengapa Menggunakan Github.com?

Sebagaimana yang telah penulis jelaskan diawal bahwa kode sumber Yii di-hosting di Github, sehingga ketika proses instalasi, composer akan berkomunikasi dengan Github API. Dalam komunikasi tersebut, kadangkala Github meminta otorisasi *token* atau *username* & kata sandi Github melalui Composer. Oleh karena itu, akun Github diperlukan terutama bagi kita yang akan memrogram menggunakan Yii.

Yii mempunyai akun resmi Github yang bisa kita jumpai pada tautan berikut.

- ∞ <https://github.com/yiisoft/yii2>

Melalui tautan tersebut, kita juga bisa memantau segala perubahan dan diskusi tentang Yii pada setiap detiknya.

C. 3. Membuat Akun Github

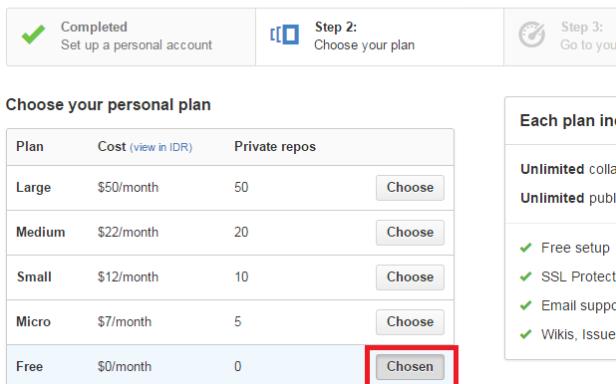
Registrasi akun Github dapat kita lakukan melalui situs resminya yaitu <https://www.github.com>. Kemudian masukkan *username*, *email* dan kata sandi sebagai data registrasi kita.



Setelah itu, kita bisa memilih paket yang ditawarkan oleh Github. Jika kita ingin membuat proyek opensource yang bisa dilihat di download orang lain maka yang kita perlukan cukup paket Free atau gratis. Namun jika proyek kita bukan opensource atau kita tidak ingin kode sumber kita dilihat atau digunakan oleh orang lain maka kita bisa pilih paket lain.

Welcome to GitHub

You've taken your first step into a larger world, [@hafidmukhlasisin](#)



Selanjutnya, kita akan mendapatkan email anda konfirmasi dan verifikasi. Ikuti petunjuk verifikasi akun yang telah dikirim ke email kita tersebut, kemudian setelah itu maka akun Github kita sudah bisa digunakan.

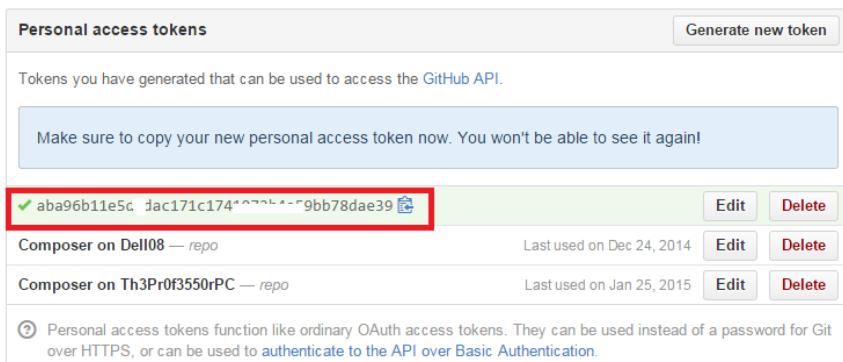
C. 4. Otentikasi Github API

Ketika kita menggunakan Composer untuk melakukan instalasi Yii, maka Composer akan berkomunikasi dengan Github API sebelum mengunduh kode sumbernya. Adakalanya jika penggunaan Github API telah mencapai batas tertentu maka yang terjadi adalah Github melalui Composer akan meminta *token* atau *username* & kata sandi dari akun Github kita untuk keperluan otentifikasi sebelum proses unduh kode sumber dimulai.

Apabila kita tidak menginginkan memasukkan *username* dan kata sandi pada Composer saat instalasi, maka kita bisa membuat *token* secara manual untuk melewati proses otentifikasi tersebut.

Caranya:

- Pada situs web <http://github.com>, buka halaman *setting application*
 - ∞ <https://github.com/settings/applications>
 - Kemudian kita generate OAuth token (*personal access token*), melalui tautan berikut.
 - ∞ <https://github.com/settings/tokens/new>



- Salin *token* yang dihasilkan
 - Kemudian pada CMD, jalankan perintah berikut.
| composer config -q github-oauth.github.com <oauthtoken>

Ganti <oauthtoken> dengan *token* yang dihasilkan. Dengan cara ini, maka kita tidak akan dimintai *username* dan kata sandi lagi ketika menggunakan Composer.

```
C:\xampp\htdocs>composer config -g github-oauth.github.com aba96b11e5dxxxxx  
xxxxxxxxxxsadad
```

D. Cara menggunakan Composer

Untuk mempermudah pemahaman kita tentang cara penggunaan Composer, maka cara terbaik adalah dengan praktik melakukan instalasi *library* PHP dengan menggunakan Composer. Sebagai contoh kasus sekaligus bonus, kita akan mencoba untuk melakukan instalasi sebuah *framework* PHP sederhana yaitu Slim Framework (*a micro framework for PHP*) tentunya dengan menggunakan Composer.

The screenshot shows the official website for Slim Framework at <https://www.slimframework.com>. The page features a large green 'Slim' logo with the subtitle 'a micro framework for PHP'. Below the logo, a tagline reads 'Slim is a PHP micro framework that helps you quickly write simple yet powerful web applications and APIs.' A code snippet in a code block shows the beginning of a PHP file using PSR-7 interfaces:

```
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;
```

Kita bisa mendapatkan info lebih lengkap tentang *framework* ini melalui situs resminya <http://www.slimframework.com>.

D. 1. Instalasi Aplikasi menggunakan Composer

Cara instalasi *framework* Slim dengan menggunakan Composer sangat mudah karena Slim telah menyediakan *skeleton* yang merupakan kode dasar aplikasi atau Yii menyebutnya sebagai *application template*.

Pertama, pastikan kita berada pada direktori *web root* (htdocs), kemudian jalankan perintah berikut.

```
| composer create-project slim/slim-skeleton [my-app-name]
```

Ubah my-app-name dengan nama *folder* aplikasi Slim kita misal kita beri nama "slim". Dengan menjalankan perintah tersebut, maka framework Slim akan terinstal pada folder "sim" (C:\xampp\htdocs\slim).

Berikut ini proses instalasinya akan penulis paparkan dengan *step by step* yang terjadi ketika perintah di atas dijalankan.

The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command line shows the user navigating to the htdocs directory and then running the Composer command to create a project named 'slim'. The output shows the progress of the download of required packages from Packagist.org.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\hafid>cd C:\xampp\htdocs

C:\xampp\htdocs>composer create-project slim/slim-skeleton slim
  1/10: http://packagist.org/p/provider-archived$fdf57e4d132d50e9d55f566df276f82a5a38487c4e43643aef7255ec6f5d6cb9.json
  2/10: http://packagist.org/p/provider-2017-01$9baa151f33e77cf5
```

Pada gambar di atas terihat bahwa Composer mulai mengidentifikasi daftar *library* yang diperlukan untuk instalasi Slim dalam bentuk atau format json.

```
//10:      http://packagist.org/p/provider-2014$2a4eb0dec99a791d18
4ca930de93aa7b8799693c7cad90e46cd95ee1d42c27.json
8/10:      http://packagist.org/p/provider-2016-07$fc811c2deb532cf42
eeceaa2c5f5bf0ecae1484b67ee7efc6375d087ecd7f8c74.json
9/10:      http://packagist.org/p/provider-2016-10$75b6ce8ac33f15f13
df2e38a94374646c10e996c6edda1c92fa63d082dc4c0d3.json
10/10:     http://packagist.org/p/provider-2015$710c63a6d20ffcc23710
4829dbc1b3d3552280e2f5f2f6adf4741c4eba0fb1c6.json
    Finished: success: 10, skipped: 0, failure: 0, total: 10
```

Dari proses ini kita ketahui bahwa ada sepuluh daftar library yang diperlukan dan semuanya berhasil diunduh.

```
Finished: success: 0, skipped: 0, failure: 33, total: 33
Package operations: 33 installs, 0 updates, 0 removals
- Installing container-interop/container-interop (1.1.0) Downloading: C
Downloading: 100%
- Installing nikic/fast-route (v1.1.0) Downloading: 100%
- Installing psr/http-message (1.0.1) Downloading: 100%
- Installing pimple/pimple (v3.0.2) Downloading: 100%
- Installing slim/slim (3.7.0) Downloading: 100%
- Installing slim/php-view (2.2.0) Downloading: 100%
```

Setelah daftar *library* (berisi URL *library*) yang diperlukan berhasil diunduh maka Composer melanjutkan dengan mengunduh atau menginstalasi library tersebut satu persatu.

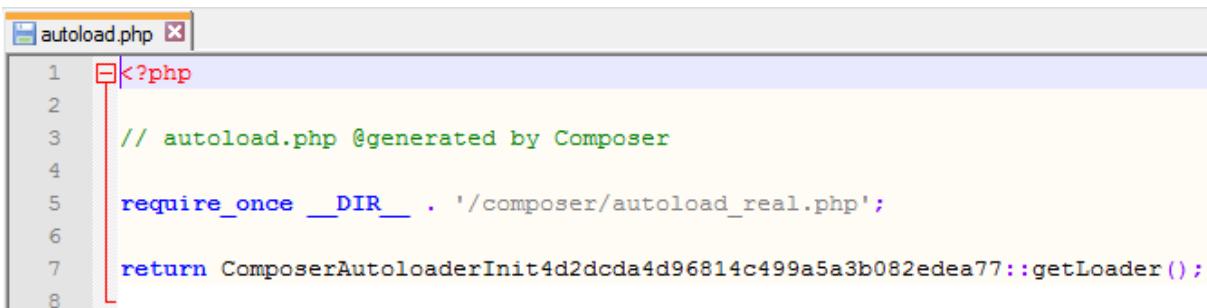
```
sebastian/global-state suggests installing ext-uopz (*)
phpunit/phpunit-mock-objects suggests installing ext-soap (*)
phpunit/php-code-coverage suggests installing ext-xdebug (>=2.4.0)
phpunit/phpunit suggests installing phpunit/php-invoker (~1.1)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating autoload files
```

Terdapat beberapa saran untuk menginstalasi *library* lain namun pada saat ini kita bisa abaikan. Poin yang penting ketika instalasi selesai adalah Composer menulis *lock file*, yaitu file dengan nama composer.lock yang berisi daftar dari semua *library* yang berhasil diinstalasi beserta versi dan URL-nya.

```
8   "packages": [
9     {
10       "name": "container-interop/container-interop",
11       "version": "1.1.0",
12       "source": {
13         "type": "git",
14         "url": "https://github.com/container-interop",
15         "reference": "fc08354828f8fd3245f77a66b9e23a
16       },
17       "dist": {
18         "type": "zip",
19         "url": "https://api.github.com/repos/contain
20         "reference": "fc08354828f8fd3245f77a66b9e23a
21         "shasum": ""
22       },
23     }
24   ]
```

Selanjutnya Composer meng-*generate* file autoload.php yang terletak di *folder vendor*. Fungsi file ini memungkinkan kita meng-*include* *file class* secara otomatis berdasarkan aturan PSR-0 *autoload*, PSR-4 *autoload*, *classmap* dan *files include* biasa.

Berikut ini contoh *file autoload* yang di-*generate* otomatis oleh Composer.

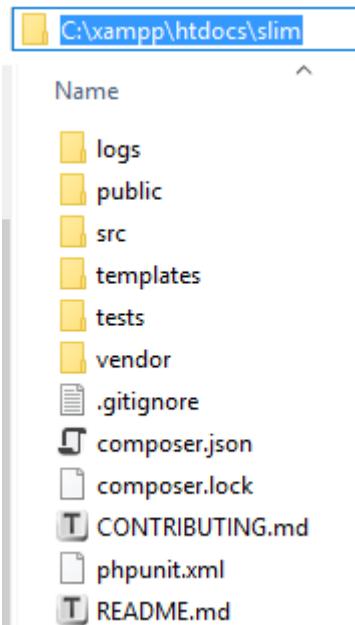


```

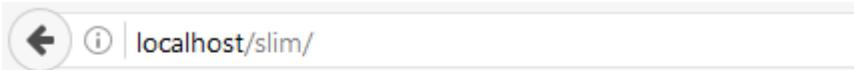
1 <?php
2
3 // autoload.php @generated by Composer
4
5 require_once __DIR__ . '/composer/autoload_real.php';
6
7 return ComposerAutoloaderInit4d2dcda4d96814c499a5a3b082edea77::getLoader();
8

```

Untuk memastikan bahwa instalasi slim berhasil maka mari kita cek di dalam *folder* slim, jika susunan *file/folder* sebagai berikut berarti instalasi berhasil.



Selanjutnya kita bisa akses aplikasi Slim melalui *web browser* dengan alamat URL <http://localhost/slim>



Welcome to Slim!

Selamat, anda berhasil menginstalasi Slim. Mudah sekali bukan? hanya dengan satu baris perintah pada Composer maka framework Slim telah terinstal dan siap di-oprek lebih dalam. Namun, penulis tidak akan membahas lebih jauh lagi karena buku ini bukan membahas *framework* Slim melainkan Yii ☺.

D. 2. Perintah Lain

Ada beberapa perintah lain yang bisa kita gunakan pada Composer, diantaranya:

- *Create project vs Require*

Secara umum ada dua cara yang bisa kita gunakan untuk menginstalasi suatu *library* yaitu menggunakan perintah *create project* dan *require*.

Berikut contoh penggunaan perintah untuk *create project* yaitu untuk menginstalasi *library* slim/slim versi 3 pada ke dalam *folder* slim.

```
| composer create-project slim/slim slim "^3.0"
```

Sedangkan berikut ini contoh penggunaan perintah *require* untuk menginstalasi *library* slim/slim versi 3 ke dalam *project*.

```
| composer require slim/slim "^3.0"
```

Kedua perintah di atas sama-sama akan menginstalasi *library* slim/slim versi 3, namun ada sedikit perbedaan, yaitu:

1. Perintah *create project* akan membuat *folder* baru dan meletakkan *library* hasil unduhan ke *folder* tersebut sedangkan perintah *require* hanya mengunduh dan meletakkan hasil unduhan pada direktori yang sama.
2. Pada perintah *create project*, Composer akan mengunduh *file* composer.json langsung dari *repository*-nya dan biasanya berisi *library* atau *file* pendukung proyek serta melakukan instalasi *library* dan pendukungnya, sedangkan perintah *require* hanya memperbarui *file* composer.json yaitu menambahkan *library* tersebut saja kemudian melakukan instalasi *library* tersebut dan dependensinya.
3. Perintah *create project*, digunakan ketika pertama kali akan membangun proyek aplikasi, sedangkan perintah *require* digunakan hanya ketika ingin menambahkan suatu *library* ke dalam proyek aplikasi.

- *Install*

Perintah ini bisa kita gunakan jika *file* composer.json telah kita buat dan telah diisi dengan *library-library* yang akan kita gunakan. Perintah ini dilakukan pertama kali ketika proyek dibuat.

```
| composer install
```

- *Update*

Perintah ini bisa digunakan jika kita ingin memperbarui *library* misalnya ada penambahan *library*, perubahan versi, dsb. Caranya cukup dengan mengubah *file* composer.json kemudian menjalankan perintah berikut.

```
| composer update
```

- Parameter -vvv

Beberapa informasi tidak ditampilkan secara lengkap pada Composer ketika sedang melakukan instalasi, sehingga kadang-kadang Composer terlihat “*stuck*” padahal sedang ada proses. Karenanya, dengan menambahkan parameter -vvv ini maka akan tampil informasi untuk setiap proses yang dilakukan oleh Composer. Contoh.

```
| composer install -vvv
```

D. 3. Tip dan Trik

Kita bisa meningkatkan performa Composer dengan menggunakan *library* hirak/prestissimo. *Library* ini membantu Composer untuk melakukan proses instalasi suatu *library* secara paralel sehingga performa Composer akan lebih cepat.

```
∞ http://blog.petehouston.com/2016/01/27/improve-composer-performance-with-prestissimo/
```

Berdasarkan *benchmark* di atas maka dapat disimpulkan bahwa dengan *library* ini maka proses instalasi suatu *library* dengan menggunakan Composer bisa tiga sampai empat kali lebih cepat.

```
| composer global require hirak/prestissimo
```

E. Kesimpulan

Sebagai satu-satunya PHP *dependecy manager*, saat ini Composer sudah menjadi bagian yang sangat penting dalam pengembangan aplikasi berbasis PHP, baik karena kemudahan maupun efisiensinya. Oleh karena itulah penulis berinisiatif membuat pembahasan tentang Composer menjadi bab tersendiri. Pastikan anda menguasai topik pada bab ini sebelum melanjutkan untuk mempelajari bab selanjutnya. Jika perlu, pelajari juga dari panduan resmi Composer <https://getcomposer.org/book.pdf>.

Setelah bab ini, kita akan belajar tentang topik utama dari buku ini yaitu *framework* Yii.

3

Pengenalan Yii Framework

Kita akan belajar tentang:

- Kelebihan & kekurangan Yii
- Cara instalasi Yii
- Arsitektur Yii
- Alur kerja Yii
- Konsep MVC

A. Mengenal Yii Framework

A. 1. Apa itu Yii?

Yii, dalam bahasa cina berarti “*simple and evolutionary*”, merupakan sebuah *framework* PHP berbasis komponen, yang memiliki performa mengagumkan. *Framework* ini bisa digunakan untuk membangun aplikasi berbasis web berskala kecil hingga berskala besar secara profesional dan elegan.

Definisi tentang apa itu Yii sebenarnya tergantung pada ekspektasi kita terhadap sebuah *framework* PHP. Namun pada intinya, apapun yang kita inginkan maka *framework* ini adalah pilihan yang tepat, Yes it is!. Untuk informasi lebih lengkap tentang *framework* ini, silakan kunjungi situs web resminya di <http://www.yiiframework.com>.

Yii mulai dikembangkan pada awal bulan Januari 2008 oleh Qiang Xue. Sebelumnya, Qiang merupakan pemrogram *framework* yang cukup populer yaitu Prado. *Framework* Yii sendiri terinspirasi dari banyak *framework* atau *library* lain, seperti Prado, Ruby on Rails, JQuery, Symfony dan Joomla.

Saat ini (2016), Yii telah mencapai versi 2. Sebuah versi yang merupakan lompatan besar dari versi sebelumnya (varian 1.1), bahkan bisa dikatakan bahwa ini adalah *framework* baru yang lebih modern.

The latest version of Yii 2 is 2.0.7, released on Februari 15, 2015. Yii 2.0 is a complete rewrite of Yii on top of PHP 5.4.0. It is aimed to become a state-of-the-art of the new generation of PHP framework. Yii 2.0 is not compatible with 1.1.

Hal ini tidak aneh, karena versi ini sebenarnya merupakan hasil dari tulis ulang secara menyeluruh dari kode sumber versi sebelumnya. Yii versi 2 ini mengusung konsep baru, serta mengadopsi standard dan teknologi-teknologi terbaru yang telah didukung oleh PHP seperti Composer, PSR, Namespace, dsb.

Namun meskipun demikian, Yii masih akan tetap mendukung varian versi 1.1.x untuk memberikan kesempatan bagi para pemrogram aplikasi sebelum mereka migrasi ke versi 2.

A. 2. Siapa yang menggunakan Yii?

Ada banyak situs web atau aplikasi yang menggunakan Yii (versi 2), berikut diantaranya:

- <http://www.humhub.org> => cms social media
- <http://easyiicms.com> => cms weblog
- <http://dotplant.ru> => cms weblog
- <https://luya.io> => cms weblog
- <http://www.salomonvillalobos.com> => cms photo studio
- <http://www.ftzmall.com> => online store
- <http://sklepstolarza.pl> => online store
- <http://travelbuddyafrica.com> => travel
- <http://almundo.com.ar> => ticket

- <http://linxbooks.com/demo> => *accounting*
- <http://www.walle-web.io> => *deployment tools*
- <https://www.logiqids.com> => *educations / e-learning*
- <https://promis.one/site/demo> => *project management*
- <https://quid.pw> => *software for schools*
- <http://www.tvmaze.com> => *TV online*
- <http://yiifeed.com> => *news feed*
- <http://beautyboutiq.ru> => *beauty & health*
- <http://logistics.qwintry.com> => *logistic*
- <http://hosannahighertech.co.tz/forum> => *forum*

Adakah situs web di Indonesia yang menggunakan Yii?

- <http://m.urbanindo.com> (jual beli properti)
- <http://www.rajamobil.com> (jual beli mobil)
- <http://www.chasebrandbali.com> (jual beli pakaian)
- <http://smartgorontalo.net> (*smart city* gorontalo)
- <http://member.grandistanarama.com> (hotel, bagian *membership*)
- <https://pictalogi.com> (*digital publishing*)
- <http://kitabisa.com>

Jika kita perhatikan daftar di atas menunjukkan bahwa Yii bisa digunakan sebagai dasar untuk membuat berbagai macam aplikasi berbasis web. Sepanjang yang penulis ketahui, umumnya Yii digunakan untuk dasar pembuatan aplikasi *back office* atau *internal* perusahaan sehingga tidak banyak tautan lokal yang bisa penulis tampilkan.

A. 3. Mengapa harus Yii?

Jika pertanyaan ini dilontarkan kepada penulis maka, penulis menjawab “tidak harus”. Ya, kita tidak harus menggunakan Yii, karena masih banyak *framework* PHP lain yang juga layak untuk kita coba dan gunakan (janganlah seperti katak dalam tempurung), yang mana diantara mereka memiliki kelebihan masing-masing, misalnya ada yang lebih cepat, lebih efisien, mendukung versi PHP lama, dsb.

Yii seperti umumnya *framework* PHP lainnya, dapat digunakan untuk membangun berbagai jenis aplikasi berbasis web, misalnya: *web portal*, forum, CMS, *e-commerce*, *web service*, dan masih banyak yang lainnya. Di samping itu, Yii juga memiliki fitur-fitur yang umum dimiliki oleh *framework* modern lainnya, diantaranya: MVC (*model – view – controller*), mendukung banyak basis data, ORM, caching, RESTful, *code generator*, dsb.

Dalam pengembangannya, Yii memiliki filosofi tersendiri yaitu: “kode seharusnya ditulis dengan sederhana namun dengan cara elegan, dan performa yang tinggi harus selalu menjadi poin utama”.

A. 4. Kelebihan Yii

Mungkin ini bagian yang banyak dicari oleh pemula yang ingin belajar Yii. Yaitu tentang apa kelebihan dari *framework* ini dibandingkan dengan *framework* lain. Bagi penulis, ini adalah topik yang gampang-gampang susah karena sifatnya sangat-sangat relatif. Pada intinya silakan dicoba dan kita akan menemukan satu persatu kelebihan itu. Namun berdasarkan pengalaman selama ini, penulis akan mencoba menjelaskan tentang kelebihan Yii.

1. Fitur & Panduan yang Lengkap

Secara umum, menurut penulis dan rekan-rekan yang bergelut dengan Yii, salah satu kelebihan *framework* ini adalah Yii memiliki fitur yang sangat lengkap, sehingga apapun yang kita butuhkan dalam suatu pengembangan aplikasi berbasis web, maka semua itu pada umumnya sudah ditangani oleh Yii. Adapun jika masih kurang, maka ada banyak *extension* diluar sana yang bisa kita gunakan atau *library-library* PHP yang dengan mudah bisa kita integrasikan.

Tidak cukup disitu saja, panduan atau *manual* Yii juga sangat lengkap dan jelas. API Yii juga mudah dibaca karena tersusun rapi beserta penjelasannya, bahkan kalau boleh jujur, buku ini maupun buku-buku sejenisnya sebenarnya menjadi tidak perlu jika kita mahir dalam membaca referensi berbahasa Inggris. Hal ini juga diakui oleh pemrogram aplikasi dari *framework* PHP lain. Menarik bukan?

2. Bukan One Man Show

Yii juga bukan lagi *one man show*, memang pada awalnya Yii dibangun oleh satu orang pemrogram *framework* saja yaitu Qiang Xue pada awal tahun 2008. Saat ini, pengembangan Yii dilakukan oleh tim utama, terdiri dari para pemrogram

framework kelas dunia termasuk Qiang sendiri, serta didukung oleh komunitas yang selalu memberikan kontribusi secara aktif.

3. Komunitas Besar dan Aktif

Yii memiliki komunitas yang besar sehingga diharapkan ketika kita menghadapi masalah dalam belajar atau pengembangan aplikasi berbasis Yii akan banyak yang membantu.

- <http://www.yiiframework.com/forum/>, forum resmi Yii memiliki member 178,512 (Des 2015)
- <https://www.facebook.com/groups/yiitalk>, group FB resmi Yii memiliki 27.476 anggota (Des 2015)
- <https://www.facebook.com/groups/yii.indonesia>, group Yii Indonesia memiliki 22.744 anggota (Des 2015)
- <http://stackoverflow.com/search?q=yii>, situs tanya jawab dunia, lebih dari 39,679 pertanyaan (Des 2015)

Menariknya, komunitas yang besar ini sangat aktif, tidak hanya pada pertanyaan atau diskusi yang ringan, namun juga pada pertanyaan atau diskusi dengan topik yang berat. Mari bergabung! dan buktikan.

4. Banyak Extension

Tidak semua fitur dimasukkan ke dalam *core* Yii, jika kita membutuhkan fitur yang spesifik yang tidak ada dalam fitur utama Yii, maka jangan khawatir karena terdapat banyak *extension* yang bisa kita gunakan.

5. Mudah Integrasi dengan Library Lain

Sejak versi 1, Yii terkenal dengan kemudahannya dalam berintegrasi dengan *library* lain, apalagi setelah mendukung Composer. Jadi jangan khawatir, karena *library* PHP apapun pada dasarnya bisa digunakan di Yii.

6. Kode Generator Ajaib

Yii memiliki kode *generator* (Gii) yang berfungsi meng-*generate* kode *template* model, crud, form, dsb. Sehingga akan mempercepat pemrogram aplikasi dalam membangun sebuah aplikasi. Menariknya, *tools* ini bisa dengan mudah dikustomisasi sesuai dengan keinginan kita.

7. Mengadopsi Standard dan Teknologi Terbaru

Setelah versi 2, Yii mengadopsi standard dan teknologi terbaru, sebut saja: PHP 5.4 keatas, Namespace, Composer, PSR, dan Bower. Banyak praktik terbaik dari *framework* lain, cms, atau teknologi web lain juga diadopsi oleh Yii seperti Prado, Symfony, Ruby n Rails, Joomla, dsb. Pemrogram *framework* Yii akan terus menyesuaikan dengan perkembangan teknologi web, dan menerima masukan dari berbagai pihak. Silakan lihat *issue* di Github Yii.

8. Peningkatan Keamanan

Keamanan menjadi faktor penting dalam dunia pemrograman berbasis web, dan pemrogram *framework* Yii menyadari akan hal itu. Yii memiliki beberapa fungsi keamanan *built-in* yang mudah digunakan, antara lain untuk pencegahan SQL *injections*, XSS *attacks*, CSRF *attacks*, cookie *tampering*, dsb.

Pada versi 2.0 ini, Yii juga telah di-*review* oleh dua pakar keamanan *web* yaitu Tom Worster dan Anthony.

9. URL SEO Friendly

Hanya dengan sedikit konfigurasi, kita bisa membuat aplikasi dengan URL yang *SEO friendly*. Fitur ini bernama *pretty URL*, yang pada versi 2.0 ini telah disempurnakan.

10. Built-in Twitter Bootstrap

Siapa yang tidak kenal Twitter Bootstrap (TB), sebuah *framework user interface* besutan *designer* Twitter yang sangat popular di dunia *web design*. Secara *default*, Yii menggunakan TB sebagai dasar dari tampilan webnya. Melalui *widget-widget*-nya, kita akan lebih mudah dalam menerapkan tampilan ala TB, seperti *form*, *tab*, *modal*, dsb.

Pengembang aplikasi juga memiliki pilihan, jika tidak ingin menggunakan TB sebagai dasar dari tampilannya. Hanya dengan sedikit konfigurasi, maka kita bisa menggantinya dengan model tampilan lain.

11. Otentikasi & Otorisasi

Yii secara built-in mendukung fitur otentikasi dan *otorisasi* pengguna, seperti: *login*, *logout*, *cookie-based* dan otentikasi berbasis token, *access control filter* dan *role-based access control* (RBAC). Di samping itu juga mendukung *OpenID*, *OAuth1* dan *OAuth2* protocols.

12. Internasionalization & Localization

Dua fitur ini juga sudah *built-in* dan mudah digunakan. Di samping itu, Yii juga mendukung penerjemahan pesan galat internal ke dalam 26 bahasa dunia, termasuk Bahasa Indonesia.

13. Mendukung Banyak Basis Data Populer

Yii mendukung semua basis data populer, antara lain: MySQL, PostgreSQL, MariaDB, MsSQL, Oracle, dan Mongodb. Di samping itu untuk pengelolaan basis data, Yii juga menyediakan fitur migrasi basis data, Active Record, *query builder*, dan *database access object* (DAO).

14. Mendukung *Template Engine*

Yii mendukung dua *template engine* populer yaitu smarty dan twig, serta sangat memungkinkan bagi pemrogram aplikasi membuat sendiri *extension* yang mendukung *template engine* lain.

15. Mendukung Penuh *Cache*

Yii mendukung berbagai jenis *cache* (APC, Memcache, files, basis data, dsb) pada berbagai level aplikasi, baik dari sisi *client* (*HTTP caching*) maupun server (*fragment caching*, dan *query caching*).

16. Kemudahan Membuat *Web Service*

Implementasi *web service* menjadi sangat mudah, karena mendukung penuh semua fungsi RESTful API.

17. *Application Template*

Yii juga menyediakan *application template* (*basic* dan *advance*) yang akan memudahkan kita dalam memulai men-*develop* aplikasi. Dari sini akan memudahkan kita memahami praktik terbaik dari Yii.

18. Tools Uji Coba Terintegrasi

Secara *built-in* Yii juga menyediakan *tools* untuk uji coba aplikasi yaitu Codeception dan Faker.

A. 5. Kekurangan Yii

Ada kelebihan maka pasti ada kekurangan, demikianlah yang namanya *tools*. Berikut ini akan diulas beberapa kekurangan dari *framework* ini.

1. Tidak *Trendy*

Yii tergolong lambat dalam mengadopsi teknologi baru. Komitmen dari *core developer* adalah Yii hanya akan mengadopsi teknologi baru yang benar-benar diperlukan. Di samping itu, semua *core developer* Yii tidak *fulltime* di Yii melainkan mereka mempunya pekerjaan utama dengan Yii sebagai sampingan.

Yii menerapkan *semantic versioning*, sejak pertama kali *release* sekitar tahun 2008 hingga saat ini Yii masih versi 2.0.x. Artinya perubahan besar selama sembilan tahun hanya terjadi satu kali yaitu ketika naik ke versi 2 yang merupakan *totally rewrite* dari Yii 1. Bandingkan dengan *framework* lain ☺.

Cermat dalam mengadopsi teknologi baru serta menjaga kompatibilitas antar versi, bukankah itu profesional?.

2. Bukan Yang Paling Populer

Awalnya Yii adalah *framework* PHP yang sangat popular, banyak pemrogram aplikasi berbondong-bondong migrasi ke Yii terutama karena performa yang mengagumkan. Namun seiring berjalannya waktu dan perkembangan teknologi, Yii 1.x tidak segera mengadopsi standard dan teknologi baru, justru berusaha untuk tetap kompatibel dengan PHP versi lama, serta menjaga kompatibilitas dan kestabilannya. Padahal saat itu *framework* PHP lain telah mulai melakukannya.

Strategi Yii dengan membagi *framework* menjadi 2 varian versi yaitu 1.x dan 2.x, serta terlalu lamanya pengembangan versi 2.0 yang mengadopsi standard dan teknologi terbaru, membuat banyak pemrogram aplikasi akhirnya migrasi ke *framework* lain.

Meskipun bukan yang paling popular namun komunitas Yii sudah cukup besar.

3. Kurang Promosi

Salah satu penyebab mengapa Yii kurang populer adalah karena kurangnya promosi yang dilakukan pihak internal Yii. Sebaik apapun produk yang kita miliki namun tidak dipromosikan maka bagaimana orang akan tertarik menggunakannya.

Promosi Yii hanya dilakukan secara natural melalui *social media* Facebook dan komunitas.

4. Terlalu Lengkap

Bisa dikatakan bahwa *framework* Yii mengusung semua fitur yang diperlukan bagi pemrogram aplikasi untuk membangun berbagai jenis aplikasi. Namun sebagian pemrogram aplikasi tidak menyukai *tools* yang terlalu lengkap, banyak fitur yang akan mubadzir dan tentunya bisa mempengaruhi performa aplikasi. Namun semua kembali ke preferensi masing-masing.

5. Bukan Yang Paling Cepat

Soal kecepatan Yii berada ditengah, banyak *framework* PHP lain yang memiliki kecepatan jauh melampaui Yii. Namun sebagai *fullstack framework* dengan fitur yang sangat lengkap, performa Yii tergolong bagus. Apalagi jika didukung dengan *environment* yang memadai serta penggunaan *cache* yang tepat dan efisien.

A. 6. Kesimpulan

Yii hanyalah *tools* untuk membantu kita dalam membangun dan mengembangkan aplikasi berbasis web, yang tentunya memiliki kelebihan sekaligus kekurangan sebagai mana yang telah penulis paparkan.

Topik ini bukan untuk diperdebatkan, penulis hanya berusaha menyampaikan sesuai dengan sudut pandang penulis dan beberapa rekan yang telah menggunakan Yii untuk membangun aplikasi. Semoga bisa menambah wawasan anda terhadap *framework* ini. Penulis menyarankan agar anda mencoba beberapa *framework* PHP lain untuk membuktikannya sendiri ☺

Namun sekali lagi, semua tergantung kita. Kelebihan yang dimiliki suatu *framework* menjadi tidak berguna jika kita tidak bisa menggunakanya dengan benar, demikian juga sebaliknya, kekurangan bisa kita minimalisir jika kita memiliki ilmunya. Untuk alasan inilah buku ini hadir ditengah-tengah anda.

B. Instalasi Yii

Sebelum melakukan instalasi Yii, perlu diketahui bahwa Yii menyediakan dua *application template* yaitu kode *template* dasar (*skeleton*) untuk membuat aplikasi berbasis Yii. *Application template* tersebut adalah *basic* dan *advanced*. *Template basic* merupakan aplikasi tunggal yang sudah menyediakan fitur *login* dan *logout* tanpa basis data. Sedangkan *template advance* merupakan dua aplikasi (*frontend* dan *backend*) yang sudah menyediakan fitur *login* dan *logout* menggunakan basis data.

Pada tutorial dalam buku ini, kita akan fokus menggunakan *application template basic* karena *template* ini lebih sederhana dan cocok untuk menjelaskan Yii. Namun, hal ini bukan berarti ketika kita sudah mahir maka kita harus menggunakan *template advanced*, silahkan disesuaikan dengan kebutuhan. Penulis pribadi lebih suka menggunakan *template basic*, beberapa *master* Yii menggunakan templatanya sendiri. Ingat!, ini hanya *application template* yang merupakan contoh saja.

B. 1. Prosedur Instalasi

Ada dua cara instalasi Yii Framework yaitu secara daring menggunakan Composer, dan secara luring melalui *file archive* / zip. Namun sebagaimana yang dijelaskan pada bagian sebelumnya bahwa instalasi dengan Composer akan memudahkan kita serta cara inilah yang direkomendasikan.

Perhatian

Perintah untuk melakukan instalasi mungkin berbeda (terutama pada versi), silakan untuk selalu merujuk panduan pada situs web resminya <http://www.yiiframework.com/download/>

Instalasi secara Luring menggunakan File Archive

Meski tidak disarankan, namun kita bisa juga melakukan instalasi Yii dengan secara manual, yaitu :

1. Unduh kode sumber Yii 2 dalam bentuk *file* terkompresi dari <http://www.yiiframework.com/download/>
2. Ekstrak kode sumber tersebut ke web *root* yaitu *htdocs* atau *www* atau *public_html*. (C:\xampp\htdocs)
3. Edit *file config/web.php* dan masukkan *secret key* yang digunakan untuk konfigurasi cookieValidationKey:

```
| 'cookieValidationKey' => 'masukkan kata kunci rahasia di sini ',
```

Informasi:

Jika kita melakukan instalasi Yii menggunakan Composer, maka langkah di atas tidak perlu dilakukan karena sudah dilakukan oleh Composer

Instalasi secara Daring menggunakan Composer

Instalasi melalui Composer tidak semudah instalasi melalui *archive*, namun kemudahan itu akan kita peroleh nanti. Buka CMD anda, kemudian kita ubah direktori ke *htdocs* (web *root* atau direktori dokumen server kita) yaitu

```
C:\xampp\htdocs\
```

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\hafid>cd C:\xampp\htdocs

C:\xampp\htdocs>
```

Persiapan

Sebelum melakukan instalasi Yii, kita harus melakukan instalasi Composer Asset Plugin, merupakan *tools* yang mengizinkan kita mengontrol Bower dan NPM melalui Composer.

Apa itu Bower? apa itu NPM?

- Bower adalah *asset repository* untuk CSS & JS, mirip dengan Packagist.
- NPM adalah *asset dependency manager* punya NodeJs, mirip dengan Composer.

Dengan menggunakan Composer Asset Plugin maka Composer memiliki kemampuan tambahan layaknya NPM, sehingga Composer kita berubah menjadi web (PHP, CSS, JS, dsb) *dependency manager*.

Berikut ini perintah untuk menginstalnya.

```
| composer global require "fxp/composer-asset-plugin:^1.2.0"
```

```
C:\WINDOWS\system32\cmd.exe - composer global require "fxp/composer-ass... - □ X

C:\xampp\htdocs>composer global require "fxp/composer-asset-plugin:^1.2.0"
"Changed current directory to C:/Users/hafid/AppData/Roaming/Composer
 1/10: http://packagist.org/p/provider-archived$ca376a577f337654
9808ce92aa9ceec7942301a379f542f9cc08eea170167b1d.json
 2/10: http://packagist.org/p/provider-2017-01$115ea82b8ac68d067
cc49a99c0d5e851e3155ebaa3584a45f6684ee9fc1a702c.json
 3/10: http://packagist.org/p/provider-2013$e5c7eceeec5c6c8287e3
93069b3cb2e8e6f1523e168fd0408727a80017cc218c.json
```

Perhatian:

Pada saat membaca panduan ini, mungkin versi Composer Asset Plugin sudah bukan 1.2.0. Oleh karena itu silakan di sesuaikan.

Proses instalasi memakan waktu beberapa menit, tergantung koneksi internet anda. Kita bisa menambahkan parameter – vvv agar detail proses instalasi ditampilkan.

```
| composer global require "fxp/composer-asset-plugin:^1.2.0" -vvv
```

Cara ini bisa kita terapkan pada instalasi atau *update library* lain.

Berikut ini akhir dari proses instalasi Composer Asset Plugin (versi 1.2.2)

```
 9/10: http://packagist.org/p/provider-2015$74315ab012a82422b0ef
d29d3084942c7a50aa50153163541d5d4982a5a10d0c.json
 10/10: http://packagist.org/p/provider-2014$666bfe3eb47f310d9a98
2e9b36032f541bd219d77bcee24ab18603acbafdf548.json
  Finished: success: 10, skipped: 0, failure: 0, total: 10
Loading composer repositories with package information
Updating dependencies (including require-dev)
  Finished: success: 0, skipped: 0, failure: 1, total: 1
Package operations: 0 installs, 1 update, 0 removals
  - Updating fxp/composer-asset-plugin (v1.2.1 => v1.2.2) Downloading: Co
  Downloading: 100%
Writing lock file
Generating autoload files

C:\xampp\htdocs>
```

Instalasi

Cara instalasi Yii melalui Composer cukup mudah, yaitu dengan menggunakan perintah berikut.

```
| composer create-project yiisoft/yii2-app-basic basic 2.0.11
```

Pada saat buku ini ditulis (baca: direvisi), Yii masih versi 2.0.11 (Februari 2017) sehingga mungkin berbeda ketika anda mencobanya, karena itu kita bisa juga melakukan instalasi tanpa menunjuk ke versi tertentu,

```
| composer create-project yiisoft/yii2-app-basic basic
```

Maksud dari perintah di atas adalah melakukan instalasi *application template basic* dari Yii versi stabil dan meletakkannya pada **folder basic**, sehingga kita akan jumpai kodennya pada direktori

```
C:\xampp\htdocs\basic
```

Atau tergantung dimana kita menjalankan perintah tersebut.

```
c:\xampp\htdocs>composer create-project yiisoft/yii2-app-basic basic 2.0.11
  1/10: http://packagist.org/p/provider-archived$32d55f0a701d57183f6b601cad26a744657559fd20190cdfa1bc13770ae516ae.json
  2/10: http://packagist.org/p/provider-2013$5a30b5113f8cb65f41f4c207c080618779da32e77efcb22610a120c383e031b5.json
  3/10: http://packagist.org/p/provider-latest$5b391848d1a82cdb2868892c952935598ba5c730cb3a83031dbc3cb0026b1324.json
  4/10: http://packagist.org/p/provider-2016$3ad53615c881946887ec13dc44b0f5b925760677e6688e317a875e3abff656be.json
```

Proses instalasi memakan waktu sekitar lima sampai dengan sepuluh menit atau tergantung dari koneksi *internet* anda.

Catatan:

Tambahkan parameter `-vvv` setiap kali melakukan instalasi dan memperbarui *library* menggunakan Composer. Supaya informasi tentang apa yang sedang dilakukan oleh Composer bisa kita lihat dengan jelas.

Banyak *library dependent* secara default akan terinstalasi ketika kita menginstalasi *template basic* Yii ini.

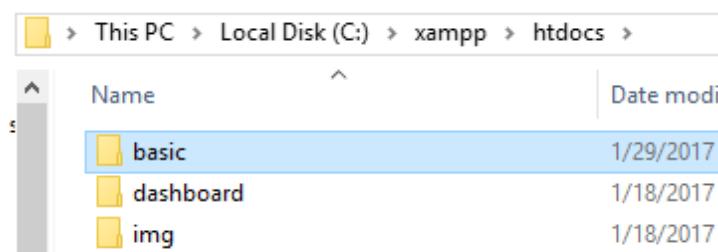
```
2/2: http://packagist.org/p/provider-latest$4ea2b53467cf498b331b079147fcdb715e66147302b1d76bd84ca7e2f3e8f38d.json
  Finished: success: 2, skipped: 0, failure: 0, total: 2
Loading composer repositories with package information
Updating dependencies (including require-dev)
  Finished: success: 0, skipped: 0, failure: 58, total: 58
Package operations: 58 installs, 0 updates, 0 removals
- Installing yiisoft/yii2-composer (2.0.5) Downloading: 100%
- Installing swiftmailer/swiftmailer (v5.4.6) Downloading: Connecting..
  Downloading: 100%
- Installing bower-asset/jquery (2.2.4) Downloading: 100%
- Installing bower-asset/yii2-pjax (v2.0.6) Downloading: 100%
- Installing bower-asset/punycode (v1.3.2) Downloading: 100%
- Installing cebe/markdown (1.1.1) Downloading: 100%
- Installing ezyang/htmlpurifier (v4.8.0) Downloading: 100%
- Installing bower-asset/jquery.inputmask (3.3.4) Downloading: Connecting..
  Downloading: 100%
- Installing yiisoft/yii2 (2.0.11.2) Downloading: 100%
- Installing yiisoft/yii2-swiftmailer (2.0.6) Downloading: Connecting..
  Downloading: 100%
- Installing bower-asset/bootstrap (v3.3.7) Downloading: 100%
- Installing yiisoft/yii2-bootstrap (2.0.6) Downloading: 100%
- Installing yiisoft/yii2-debug (2.0.7) Downloading: 100%
- Installing bower-asset/typeahead.js (v0.11.1) Downloading: Connecting..
  Downloading: 100%
```

Berikut ini akhir dari instalasi.

```
codeception/base suggests installing league/factory-muffin-faker (For Fak
er support in DataFactory module)
codeception/base suggests installing symfony/phpunit-bridge (For phpunit-
bridge support)
Writing lock file
Generating autoload files
> yii\composer\Installer::postCreateProject
chmod('runtime', 0777)...done.
chmod('web/assets', 0777)...done.
chmod('yii', 0755)...done.

c:\xampp\htdocs>_
```

Setelah instalasi selesai, buka *file explorer* lalu masuk pada direktori dimana kita melakukan instalasi Yii. Berikut ini contoh pada komputer penulis.



Jika kita ingin melakukan instalasi Yii terbaru versi *development* (bukan versi stabil) maka tambahkan parameter `stability=dev`

```
| composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-basic basic
```

Versi ini cocok, jika kita sedang dalam tahap pengembangan atau ingin mencoba fitur baru Yii. Sebaiknya tidak digunakan untuk *production*.

B. 2. Verifikasi Kebutuhan Minimal

Untuk memastikan semua fitur Yii berjalan dengan baik, maka kita perlu melihat kebutuhan minimal untuk menjalankan Yii melalui *tools* bawaan yaitu <http://localhost/basic/requirements.php>



Description

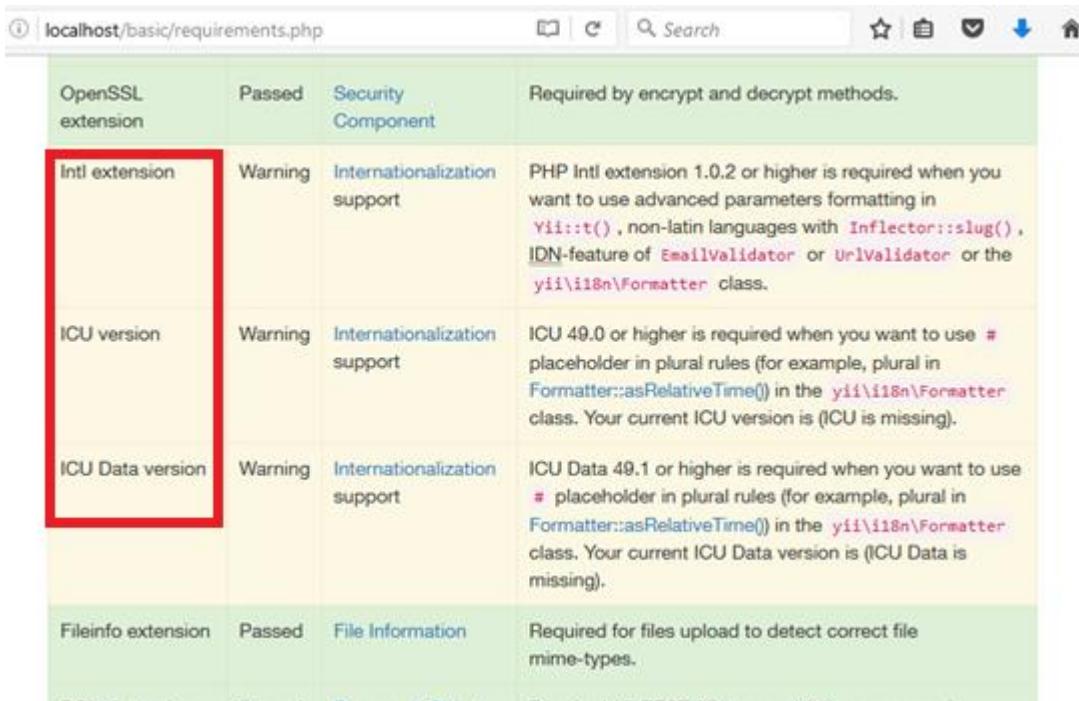
This script checks if your server configuration meets the requirements for running Yii application. It checks if the server is running the right version of PHP, if appropriate PHP extensions have been loaded, and if `php.ini` file settings are correct.

There are two kinds of requirements being checked. Mandatory requirements are those that have to be met to allow Yii to work as expected. There are also some optional requirements being checked which will show you a warning when they do not meet. You can use Yii framework without them but some specific functionality may be not available in this case.

Conclusion

Your server configuration satisfies the minimum requirements by this application.
Please pay attention to the warnings listed below and check if your application will use the corresponding features.

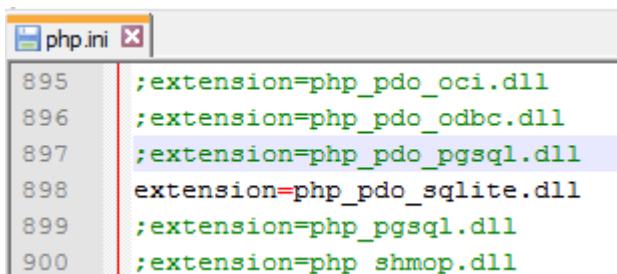
Pastikan semua kebutuhan terpenuhi (*passed* ditandai baris warna hijau) atau minimal hanya muncul *warning*.



OpenSSL extension	Passed	Security Component	Required by encrypt and decrypt methods.
Intl extension	Warning	Internationalization support	PHP Intl extension 1.0.2 or higher is required when you want to use advanced parameters formatting in <code>Yii::t()</code> , non-latin languages with <code>Inflector::slug()</code> , IDN-feature of <code>EmailValidator</code> or <code>UrlValidator</code> or the <code>yii\i18n\Formatter</code> class.
ICU version	Warning	Internationalization support	ICU 49.0 or higher is required when you want to use <code>#</code> placeholder in plural rules (for example, plural in <code>Formatter::asRelativeTime()</code>) in the <code>yii\i18n\Formatter</code> class. Your current ICU version is (ICU is missing).
ICU Data version	Warning	Internationalization support	ICU Data 49.1 or higher is required when you want to use <code>#</code> placeholder in plural rules (for example, plural in <code>Formatter::asRelativeTime()</code>) in the <code>yii\i18n\Formatter</code> class. Your current ICU Data version is (ICU Data is missing).
Fileinfo extension	Passed	File Information	Required for files upload to detect correct file mime-types.

Pada komputer penulis, dengan menggunakan tools XAMPP terbaru, secara *default* terdapat enam *warning* yaitu: Intl extension, versi ICU dan ICU Data, PDO PostgreSQL extension, Memcache extension, ImageMagick PHP extension dan Expose PHP.

Beberapa *warning* bisa diatasi dengan cara mengaktifkan *extension* PHP di php.ini. Sebagai contoh, jika kita ingin menggunakan database PostgreSQL maka kita perlu mengaktifkannya. Caranya, buka php.ini di C:\xampp\php\php.ini menggunakan *text editor*, kemudian cari *extension* dengan nama **php_pdo_pgsql**,



```

895 ;extension=php_pdo_oci.dll
896 ;extension=php_pdo_odbc.dll
897 extension=php_pdo_pgsql.dll
898 extension=php_pdo_sqlite.dll
899 ;extension=php_pgsql.dll
900 ;extension=php_shmop.dll

```

Kemudian, pada baris *extension* tersebut, *uncomment extension* tersebut atau menghilangkan tanda titik koma di depan *extension*-nya. Selanjutnya untuk keamanan maka variabel expose_php perlu diubah menjadi *off*, dan terkait *extension* Intl dan versi ICU makan bisa diselesaikan dengan meng-*uncomment extension* *php_intl*.

Adapun untuk menyelesaikan *warning* memcache dan ImageMagick maka tidak cukup dengan cara sebelumnya, cara menyelesaikan *warning* adalah dengan menginstalasi kedua *library* tersebut. Silakan *googling* untuk menyelesaikan ini, karena terlalu panjang jika dibahas di sini. Namun hal itu hanya bertipe *warning* saja, artinya hanya saran, karena Yii akan tetap bisa berjalan dengan sempurna. Untuk Cache, secara *default*, Yii menggunakan FileCache, dan untuk pemrosesan *image*, jika ImageMagick tidak terinstalasi maka Yii menggunakan *library* GD yang secara *default* telah terinstalasi.

Jika semua konfigurasi sudah dilakukan, maka kita perlu me-*restart web server* melalui XAMPP Control Panel untuk mengaktifkan konfigurasi yang kita lakukan.

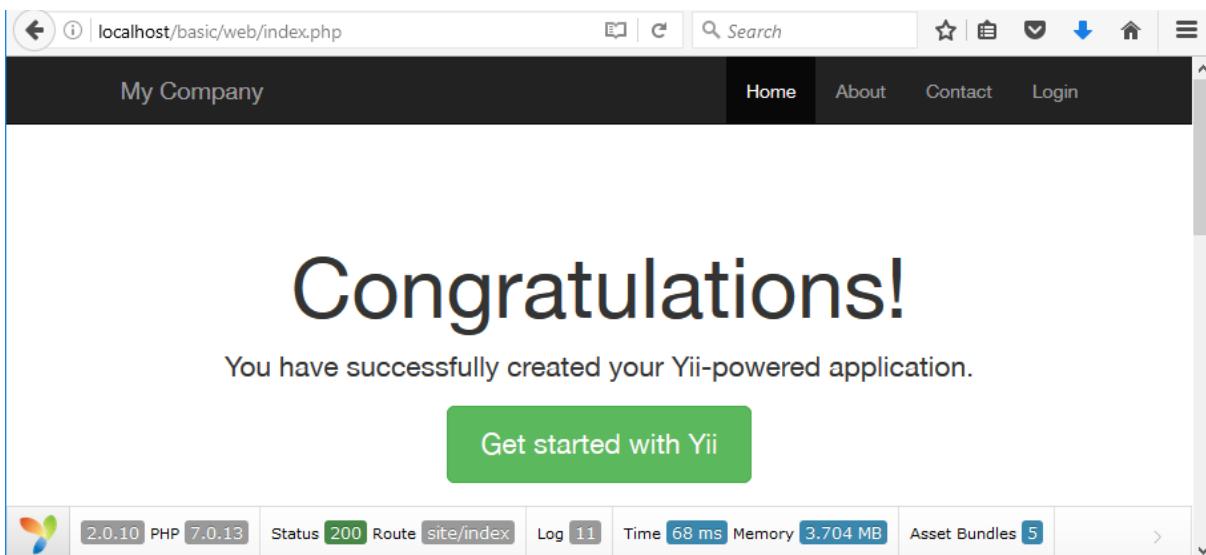
Sekarang mari kita cek lagi dengan mengakses requirements.php.

B. 3. Uji Coba Aplikasi Yii

Setelah instalasi selesai, maka kita bisa menguji apakah instalasi yang telah kita lakukan sebelumnya berhasil atau gagal. Pastikan *web server* anda *running*, kemudian buka *web browser* terbaik anda dan akses URL berikut.

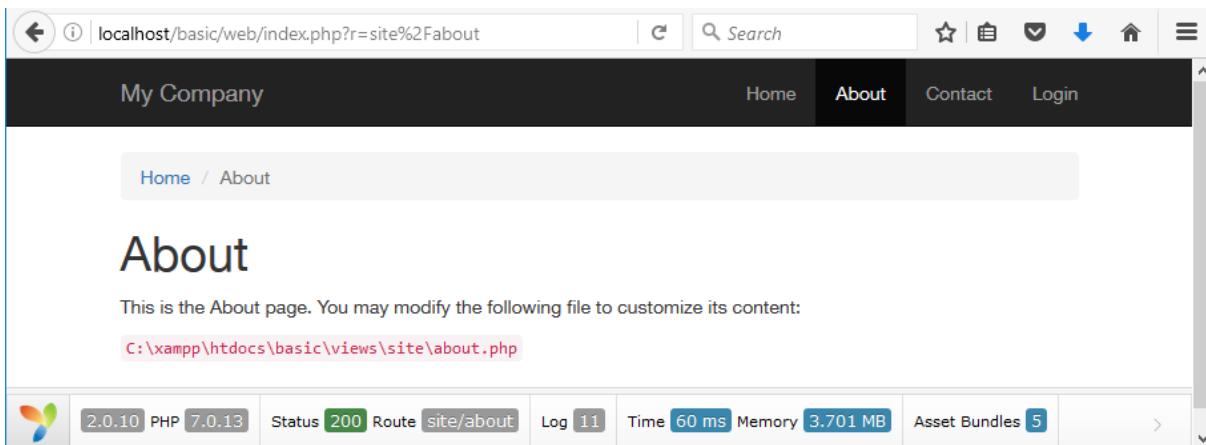
∞ <http://localhost/basic/web/>

Folder web merupakan *mount point default* aplikasi berbasis Yii. Kemudian, jika muncul tampilan sebagai berikut, berarti instalasi kita berhasil.

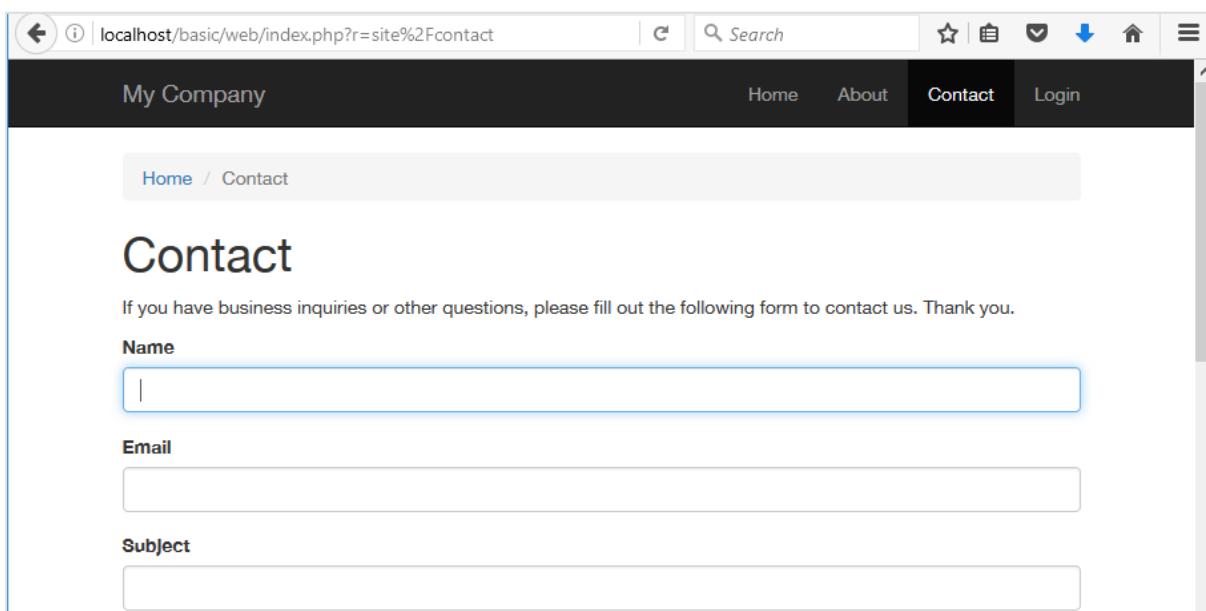


Silakan akses semua halaman melalui menu utama,

Berikut ini adalah halaman About, dimana kita bisa memodifikasi tampilannya dengan mengedit file @app/views/site/about.php



Kemudian berikut ini halaman Contact yang berbentuk formulir dan sudah dilengkapi dengan Captcha sebagai pengaman untuk menghindari spam.



Body

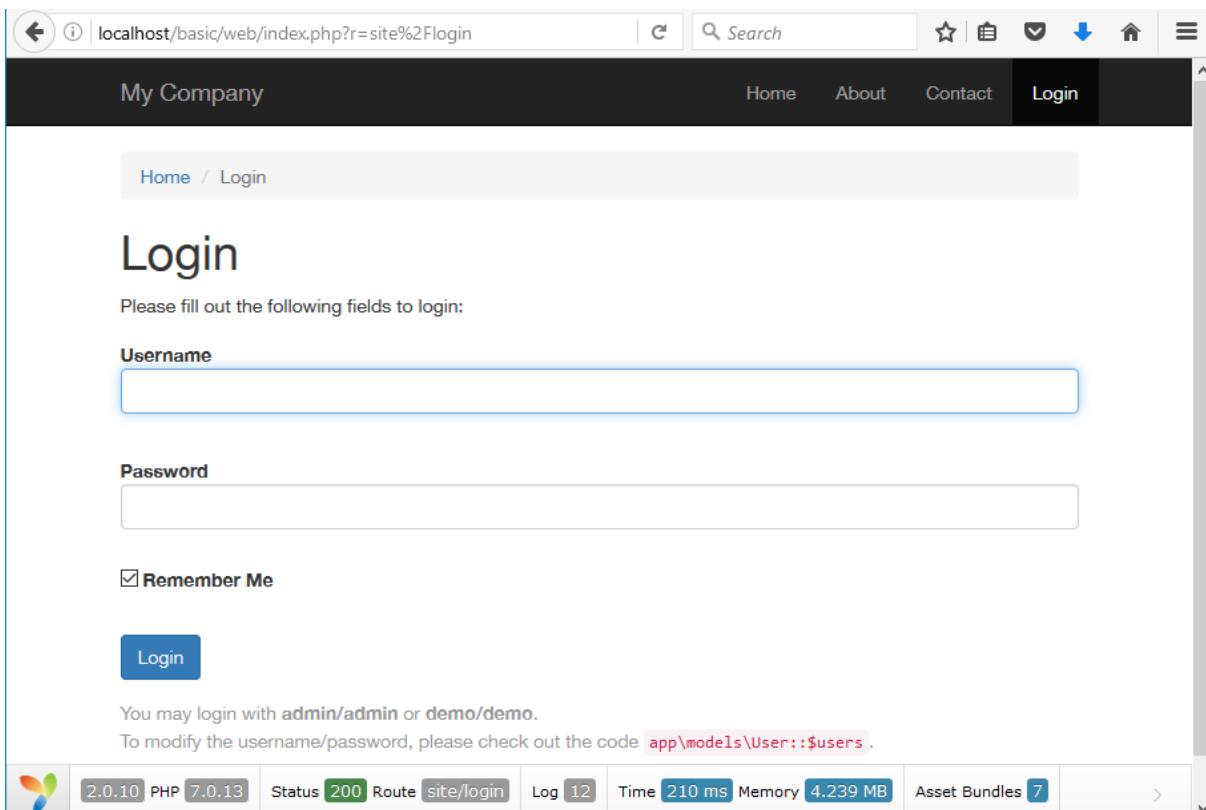
Verification Code

jxiirof

Submit

 2.0.10 PHP 7.0.13 Status 200 Route site/contact Log 12 Time 68 ms Memory 4.348 MB Asset Bundles 8 > ▾

Dan yang terakhir adalah formulir *login*, dimana kita bisa mencoba untuk *login* menggunakan *username/password*: admin/admin dan demo/demo.


 localhost/basic/web/index.php?r=site%2Flogin

My Company

Home About Contact Login

Home / Login

Login

Please fill out the following fields to login:

Username

Password

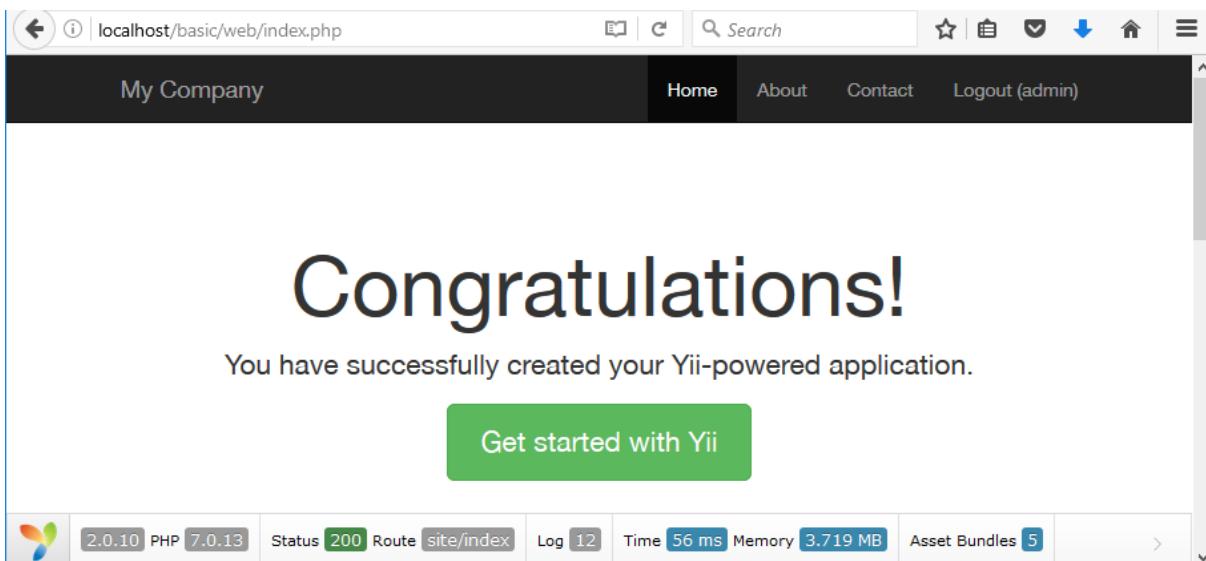
Remember Me

Login

You may login with admin/admin or demo/demo.
To modify the username/password, please check out the code `app\models\User::$users`.


 2.0.10 PHP 7.0.13 Status 200 Route site/login Log 12 Time 210 ms Memory 4.239 MB Asset Bundles 7 > ▾

Setelah *login*, maka akan muncul menu Logout di sebelah kanan atas.



Untuk mengganti *username* / kata sandi, cukup dengan memodifikasi file basic\models\User.php.

```
private static $users = [
    '100' => [
        'id' => '100',
        'username' => 'admin',
        'password' => 'admin',
        'authKey' => 'test100key',
        'accessToken' => '100-token',
    ],
    '101' => [
        'id' => '101',
        'username' => 'demo',
        'password' => 'demo',
        'authKey' => 'test101key',
        'accessToken' => '101-token',
    ],
];
```

Pada *application template* ini data pengguna hanya disimpan secara *hardcode* menggunakan format *array*, jadi belum disimpan ke dalam basis data.

Sebagaimana yang disebutkan di awal bahwa secara *default*, *application template* di Yii menggunakan Twitter Bootstrap (*framework user interface*) untuk menangani tampilannya, sehingga tampilan yang kita lihat di atas merupakan tampilan standard Twitter Bootstrap. Namun demikian, kita pun juga bisa menggunakan *framework user interface* lain atau menggunakan buatan sendiri.

C. Arsitektur Yii

C. 1. Application Template

Sebagaimana yang telah disinggung di muka bahwa *application template* merupakan *skeleton* yang berisi kode-kode dasar yang bisa digunakan untuk pengembangan aplikasi web. Jadi template yang dimaksud disini bukan tentang bagaimana tampilan aplikasi (penulis menyebut hal itu dengan *theme*).

Lantas, apa bedanya antara *application template versus framework*?

Analogi sederhananya adalah, misalnya kita ingin membangun sebuah rumah, maka *application template* adalah pondasi rumah sedangkan *framework* adalah gudang yang berisi bahan bangunan.

Pada konteks Yii, *application template* Yii merupakan pondasi aplikasi yang telah dilengkapi dengan fitur-fitur dasar sebuah aplikasi yaitu: Login, Logout, Contact Form, dsb. Pengorganisasian file pada *application template* merupakan praktik terbaik yang direkomendasikan serta cukup bagus untuk mulai sebuah proyek aplikasi web. Jadi ketika kita melakukan instalasi Yii, maka sebenarnya kita melakukan instalasi dua hal yaitu *core* dari *framework* Yii dan *application template*-nya.

Yii hadir dengan dua pilihan *template* yaitu Basic (yang telah kita instalasi) dan Advanced. Masing-masing memiliki peruntukannya sendiri, namun bagi pemula, direkomendasikan untuk menggunakan *template* Basic. Hal ini dikarenakan *template* ini *simple* yang terdiri dari satu aplikasi, demikian juga dengan buku ini, *base*-nya adalah *template* Basic. Adapun

template Advanced direkomendasikan untuk sebuah proyek aplikasi yang besar, dimana aplikasi *backend* terpisah dengan *frontend*. Fitur *login*-nya juga sudah terkoneksi dengan basis data.

C. 2. Struktur Aplikasi

Berikut ini struktur aplikasi Yii (*template Basic*) yang terdiri dari *folder* dan *file*.

Name	Type	Size
assets	File folder	
commands	File folder	
config	File folder	
controllers	File folder	
mail	File folder	
models	File folder	
runtime	File folder	
tests	File folder	
vendor	File folder	
views	File folder	
web	File folder	
.bowerrc	BOWERC File	1 KB
.gitignore	Text Document	1 KB
codeception.yml	YML File	1 KB
composer.json	JSON File	2 KB
composer.lock	LOCK File	103 KB
LICENSE.md	Markdown Docu...	2 KB
README.md	Markdown Docu...	6 KB
requirements.php	JetBrains PhpStorm	6 KB
yii	File	1 KB
yii.bat	Windows Batch File	1 KB

Berikut ini penjelasan masing-masing.

Nama File/Folder	Keterangan
assets/	berisi definisi <i>assets</i> yang digunakan pada aplikasi (css, js)
commands/	berisi contoh penerapan aplikasi <i>console</i> (CMD)
config/	berisi <i>file</i> konfigurasi aplikasi
console.php	<i>file</i> konfigurasi aplikasi <i>console</i>
web.php	<i>file</i> konfigurasi aplikasi web
db.php	<i>file</i> konfigurasi basis data
controllers/	berisi <i>file class-class controllers</i> aplikasi
mail/	berisi contoh <i>template email</i>
models/	berisi <i>file class-class model</i> aplikasi
runtime/	berisi <i>file-file</i> yang di-generate oleh Yii selama <i>runtime</i> , seperti <i>logs</i> dan <i>cache</i>
tests/	Berisi <i>file-file</i> untuk kebutuhan uji coba aplikasi
vendor/	berisi paket yang diinstal oleh Composer, termasuk <i>core</i> Yii sendiri.
views/	berisi <i>file-file views</i> aplikasi yang dikelompokkan menggunakan <i>folder</i> sesuai dengan nama <i>controller</i> -nya
web/	<i>root</i> dari aplikasi web, atau <i>file</i> yang boleh diakses oleh pengguna

assets/	berisi <i>file-file asset</i> (javascript and css) yang di-publish oleh Yii
index.php	<i>File script entry</i> (atau <i>bootstrap</i>) aplikasi
.bowerrc	<i>file</i> konfigurasi bower
.gitignore	<i>file</i> yang berisi daftar nama <i>file</i> atau <i>folder</i> yang dikecualikan untuk diunggah oleh Git
composer.json	<i>file</i> informasi tentang paket yang digunakan dalam aplikasi (digunakan oleh Composer)
composer.lock	<i>file</i> informasi hasil unduh Composer
LICENSE.md	<i>file</i> informasi lisensi aplikasi kita
README.md	<i>file</i> informasi tentang aplikasi kita
requirements.php	<i>file</i> untuk memeriksa kebutuhan minimal sistem
yii dan yii.bat	<i>file script</i> untuk mengeksekusi perintah-perintah pada <i>console</i>

Secara umum, struktur aplikasi Yii terbagi menjadi dua bagian utama yaitu:

1. Apapun yang terdapat pada *folder basic/web*. *Folder* ini merupakan *web root* yang bisa langsung diakses melalui *HTTP request (web browser)* atau apa yang di awal disebut sebagai *mount point* Yii.
2. Apapun yang terdapat pada *folder* lain selain *basic/web*, seharusnya tidak dapat diakses secara langsung melalui *HTTP*

Sekarang kita seharusnya faham mengapa pada saat uji coba aplikasi, alamat URL yang diakses adalah <http://localhost/basic/web/>. Karena mount [point dari aplikasi Yii berada pada direktori @app\web.

Lantas ada pertanyaan, bagaimana caranya untuk menghilangkan *basic/web* dari URL? Jawabannya adalah ada banyak cara. Bisa melalui konfigurasi Virtual Host di server, bisa juga melalui .htaccess (untuk web server apache), dan melalui pengaturan lokasi *folder* atau *file*. Hal ini akan dibahas pada bagian selanjutnya, sehingga untuk sementara biarkan apa adanya.

D. Alur Kerja Aplikasi

D. 1. Konsep MVC

Secara umum, Yii mengimplementasikan konsep *Model-View-Controller* (MVC) *design pattern*, merupakan konsep populer yang umum digunakan oleh *framework* pemrograman modern yaitu sebuah pola arsitektur perangkat lunak yang membagi aplikasi menjadi tiga bagian yang saling berhubungan, yaitu dengan memisahkan antara data atau informasi, bagaimana data itu ditampilkan, dan aturan bisnis terkait data tersebut.

Jika aplikasi berbasis web ditinjau dari sudut pandang *input* dan *output*, maka yang menjadi *input* adalah *request* dari *user* atau pengunjung web, sedangkan *output* merupakan respon dari server atas permintaan *user* tersebut. Berdasarkan tinjauan ini, maka *controller* berperan sesuai dengan namanya yaitu sebagai pengontrol atas *input* dan *output*.

Request dari *user* bisa berupa data, baik itu data statis dalam bentuk *array* atau *file*, maupun data dinamis dalam bentuk basis data. Apapun itu, setiap data memiliki karakteristik atau struktur atau format. Di Yii, format data menyangkut tipe datanya (*integer*, *string*, *file* dll), validasinya (tidak boleh kosong, minimal 6 karakter, harus angka, dsb) serta hubungan dengan data lain (relasinya). Pengaturan format data tersebut dilakukan oleh *model*. Pada aplikasi yang menggunakan basis data maka model merepresentasikan tabel.

Respon yang diberikan oleh server kepada *user* melalui *web browser* berupa tampilan halaman web atau minimal bagian dari halaman web yang sebenarnya merupakan rangkaian dari elemen-elemen HTML & CSS. Tampilan tersebut bisa ditampilkan kepada *user* dalam bentuk tampilan tabel, gambar, formulir dsb. Komponen MVC yang bertanggung jawab terhadap tampilan ini adalah *view*.

Pada konsep ini, semua *request* atau permintaan dari *user* akan diproses oleh *controller*. *Controller*-lah yang menentukan apakah *request* tersebut perlu direspon dengan sebuah *view* atau dengan *model*-nya sekaligus.

Pemisahan ini akan memudahkan bagi kita dalam mengembangkan aplikasi web, karena konsep ini akan menuntun kita berfikir secara sistematis dan terstruktur. Di samping itu, MVC akan membuat kode program lebih mudah dibaca, karena kode *logic*, tampilan, dan data, ditulis dalam *file* yang terpisah.

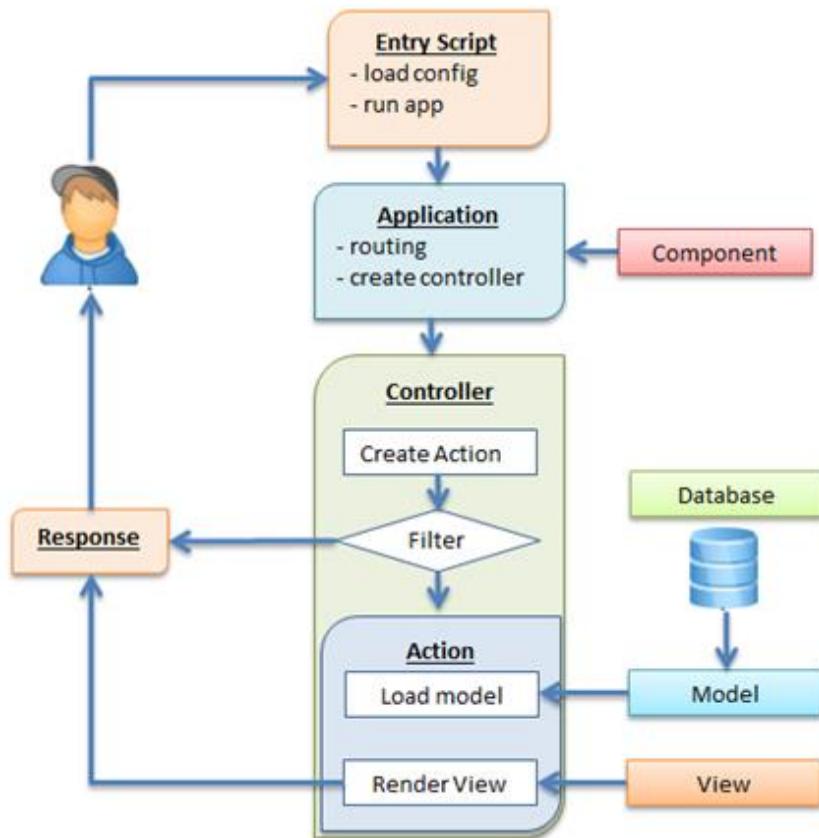
Berdasarkan uraian diatas, dapat disimpulkan bahwa peran dari controller sangat besar yaitu sebagai sentral pengendali *input* dan *output*. Sebuah aplikasi yang dikembangkan menggunakan konsep MVC maka dalam satu siklusnya minimal melibatkan satu *controller*.

D. 2. *MVC & Alur Kerja Aplikasi di Yii*

MVC di Yii juga tercermin pada struktur folder aplikasinya, yaitu terdapat *folder model* (berisi semua *file model*), *views* (berisi semua *file view*), dan *controllers* (berisi semua *file controller*).

Name	Type
assets	File folder
commands	File folder
config	File folder
controllers	File folder
mail	File folder
models	File folder
runtime	File folder
tests	File folder
vendor	File folder
views	File folder
web	File folder

Alur kerja aplikasi yang menggunakan *framework* Yii sama dengan alur kerja pada konsep MVC. Berikut ini gambarannya:



Alur kerja atau siklus aplikasi Yii berawal dari *user* yang mengakses aplikasi Yii melalui *entry script*, maka pada *entry script* tadi akan dibaca konfigurasi aplikasi setelah kemudian berdasarkan konfigurasi tersebut maka aplikasi dijalankan. Aplikasi tersebut memproses *routing* yang diminta oleh *user* melalui URL sehingga diketahui *controller* yang dimaksud *user*, kadangkala proses dalam aplikasi ini membutuhkan komponen tertentu sehingga dipanggilah suatu komponen tersebut. Setelah diketahui *controller*-nya maka diciptakanlah *controller* dimaksud. *Controller* kemudian menciptakan *action* berdasarkan *routing*.

Sebelum *action* benar-benar tercipta maka dilakukan pengecekan atau *filter* terhadap *action* tersebut (misalnya: *filter* otorisasi yaitu apakah *action* tersebut boleh diakses oleh *current user*). Jika tidak lolos *filter* maka *controller* akan

mengembalikan *response* ke *user* dan siklus berakhir, namun jika lolos *filter* maka *action* akan diproses. Pada *action* akan dimuat model (jika menggunakan *model*), demikian juga jika *model* membutuhkan basis data (*active record*) maka akan dilakukan pengaksesan basis data. Selanjutnya pada *action* tersebut akan dilakukan *render* terhadap suatu *view* (jika menggunakan *view*) untuk kemudian *view* tersebut ditampilkan ke *user*, dan siklus berakhir.

Berikut ini beberapa istilah penting.

- *Entry script*

Adalah *file* yang pertama kali diakses oleh aplikasi yaitu *file* index.php yang terdapat pada direktori @app/web/index.php

Catatan:

Ketika penulis menyebutkan “app” maka yang dimaksud adalah direktori dari aplikasi kita, yang pada contoh ini terletak di C:\xampp\htdocs\basic

```
<?php

// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

require(__DIR__ . '/../vendor/autoload.php');
require(__DIR__ . '/../vendor/yiisoft/yii2/Yii.php');

$config = require(__DIR__ . '/../config/web.php');

(new yii\web\Application($config))->run();
```

Pada intinya, *entry script* ini akan memuat *core* Yii, kemudian membaca konfigurasi aplikasi, selanjutnya menciptakan dan menjalankan aplikasi.

- *Default Controller & Action*

Yii mempunyai *default controller* yaitu SiteController yang terletak pada direktori.

@app/controllers/SiteController.php

```
<?php
namespace app\controllers;
use Yii;
...
class SiteController extends Controller
{
    ...
}
```

Adapun *default Action* (*method* atau fungsi) yaitu actionIndex,

```
public function actionIndex()
{
    return $this->render('index');
```

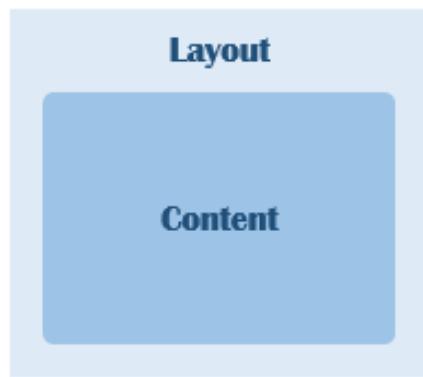
Artinya jika diakses tanpa URL yang spesifik (hanya http://localhost/basic/web) maka *controller* dan *action* tersebut yang akan otomatis dijalankan.

Informasi:

Kita bisa mengubah *default routing* melalui parameter defaultRoute yang bisa kita seting pada konfigurasi, misal: 'defaultRoute' => 'main/index'

D. 3. Default Layout

Layout merupakan tampilan pembungkus dari suatu konten atau halaman. Secara *default*, *layout* ini didefinisikan pada *file* @app/views/layouts/main.php. Isi dari *file layout* ini berisi kode-kode HTML yang menjadi kerangka utama tampilan aplikasi. Berikut ini ilustrasi dari *layout*.



Content pada gambar di atas secara *default* adalah *view* yang sesuai dengan *default controller* dan *action* yaitu @app/views/site/index.php

```
<?php  
/* @var $this yii\web\View */  
$this->title = 'My Yii Application';  
?>  
...
```

Penamaan *folder* dan *file* juga tidak sembarang, melainkan menggunakan aturan yang ditentukan yaitu "site" merupakan nama *controller* (SiteController) sedangkan *index* merupakan nama *action* pada *controller* tersebut yaitu actionIndex(). Pada bagian selanjutnya akan dibahas tentang bagaimana modifikasi dari *layout* ini.

D. 4. Keuntungan Penerapan MVC

Ada banyak keuntungan yang akan kita peroleh sebagai pengembangan aplikasi, jika kita menerapkan konsep MVC ini, diantaranya:

- Struktur aplikasi akan lebih rapi dan mudah difahami terutama untuk proyek aplikasi yang kompleks
- Lebih mudah mengelola ketika terjadi perubahan data, bisnis proses maupun tampilan
- Lebih mudah dalam melacak dan menangani galat.
- Mudah dalam membagi pekerjaan jika proyek aplikasi dikerjakan oleh tim

E. Kesimpulan

Yii merupakan salah satu *framework* PHP yang memiliki fitur lengkap. *Framework* ini tergolong *fullstack* dimana *frontend* dan *backend* juga ditangani. Yii mendukung banyak teknologi dan standard PHP terbaru seperti Composer, PHP 7, PSR, dsb. Yii menerapkan *MVC design pattern* sebagai alur aplikasinya yaitu memisahkan antara format data, bisnis proses dan tampilan.

Pada bab selanjutnya, kita akan belajar lebih dalam tentang bagaimana penerapan konsep MVC pada Yii.

4

Model View Controller

Pada bab ini, kita akan belajar tentang:

- Konsep dasar *routing*
- Implementasi MVC
- Membuat *widget*
- Konfigurasi *Pretty URL*

Setelah belajar tentang konsep dasar MVC, kini saatnya kita mengimplementasikan konsep itu di Yii.

A. Membuat *Hello Word*

Membuat *hello word* adalah cara terbaik untuk mulai belajar suatu bahasa pemrograman atau *framework*, konon jika kita sudah bisa membuatnya maka selanjutnya akan lebih mudah.

Yii mempunyai mekanisme sederhana yang mudah difahami untuk menampilkan teks “*hello word*”. Caranya adalah dengan menambahkan sebuah *action* pada suatu *controller* yang berfungsi untuk mengembalikan teks “*hello word*”. Mari kita praktikkan, pada contoh kali ini kita akan menggunakan *controller* Site (SiteController) yang merupakan *controller default* Yii. Buka file SiteController.php pada lokasi berikut.

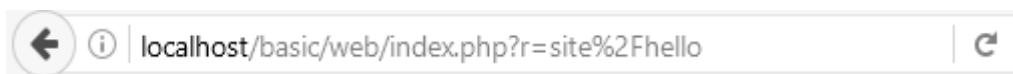
```
@app/controllers/SiteController.php.
```

Selanjutnya kita tambahkan fungsi bernama *actionHello()* pada SiteController, kira-kira sebagai berikut.

```
public function actionHello()
{
    return "Hello Word!";
}
```

Kemudian akses fungsi tersebut melalui *web browser* dengan URL sebagai berikut.

```
∞ http://localhost/basic/web/index.php?r=site/hello
```



Hello Word!

Perhatian:

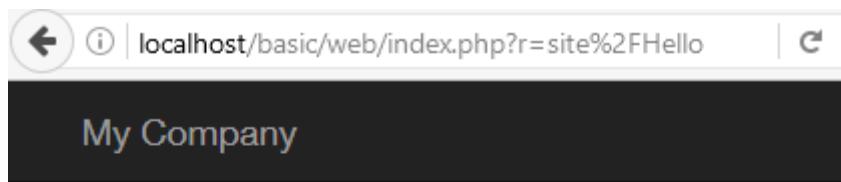
Jangan panik dengan URL yang berantakan ☹, karena *web browser* akan secara otomatis menyandikan (*encode*) karakter-karakter tententu pada alamat URL nya, dalam hal ini karakter / diubah menjadi %2F.

Pada contoh diatas, yang dimaksud dengan URL r=site/hello adalah r merupakan *routing*, site mereferensikan SiteController, dan hello mereferensikan *actionHello*. Perlu diingat bahwa *routing* ini bersifat *case sensitif* jadi *routing* ini membedakan huruf kecil atau besar. Oleh karenanya, r=site berbeda dengan r=Site, site/Hello berbeda dengan site/hello.

Untuk lebih jelaskannya, mari kita coba mengakses *actionHello* dengan URL berikut.

```
∞ http://localhost/basic/web/index.php?r=site/Hello
```

Pada contoh di atas, *routing* “hello” ditulis dengan huruf pertama kapital. Jika kode tersebut dijalankan maka akan muncul galat seperti berikut.



Not Found (#404)

Page not found.

The above error occurred while the Web server was processing

Please contact us if you think this is a server error. Thank you

Mendapatkan tampilan seperti ini, maka anda juga tidak perlu panik ☺. Pastikan bahwa penulisan nama *action* sudah benar atau sesuai dengan contoh di atas, perhatikan juga huruf besar dan kecil karena *case sensitif*. Adapun penjelasan tentang URL tersebut (*routing*) akan dibahas pada bagian selanjutnya. Perhatikan bahwa pada contoh di atas URL yang diakses adalah *Hello* dengan huruf H kapital, padahal seharusnya menggunakan huruf h kecil atau *hello*.

Perhatian:

Pahami pelan-pelan! tidak perlu terburu-buru karena penyebab munculnya kesalahan yang seringkali terjadi adalah karena kurang cermatnya pada aturan *routing* ini.

Baik, selanjutnya kita akan mencoba hal yang lebih menantang lagi, yaitu bagaimana cara mengirimkan parameter dari URL untuk dibaca oleh *controller* dan kemudian ditampilkan? sehingga apa yang ditampilkan bisa dinamis sesuai dengan URL yang diberikan.

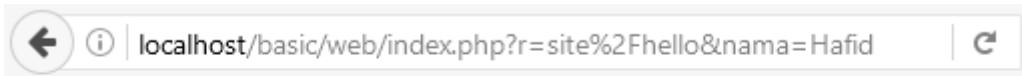
Misalnya, kita akan menambahkan parameter **nama** pada URL

∞ index.php?r=site/hello&nama=Hafid

Dari URL tersebut kita bisa menangkap parameter **nama** dengan cara yang sama seperti PHP biasa yaitu `$_GET`. Perhatikan kode berikut.

```
public function actionHello()
{
    return "Hello ".$_GET['nama'];
}
```

Ya, dengan menggunakan `$_GET['nama']` kita bisa menangkap parameter nama di URL.



Hello Hafid

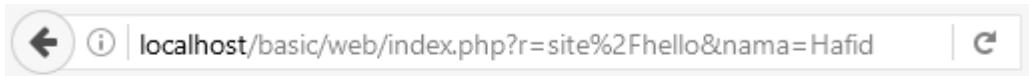
Catatan:

Sebenarnya Yii mempunyai cara tersendiri untuk menangkap variabel yang dilewatkan melalui URL. Jika PHP menggunakan kode `$_GET`, maka Yii menggunakan `Yii::$app->request->get()`. Selengkapnya bisa dibaca pada URL berikut. <http://www.yiiframework.com/doc-2.0/guide-runtime-requests.html>

Di samping cara di atas, kita bisa menggunakan cara yang lebih direkomendasikan oleh Yii, yaitu parameter di URL diubah menjadi variabel PHP biasa, perhatikan contoh kode berikut.

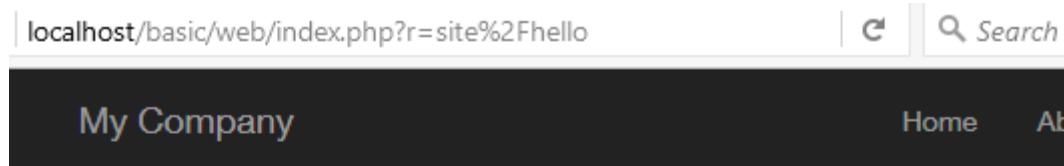
```
public function actionHello($nama)
{
    return "Hello ".$nama;
}
```

Jika kita akses maka hasilnya juga akan sama.



Hello Hafid

Hanya saja, dengan menggunakan kode di atas maka apabila kita mengakses halaman hello tersebut tanpa menyertakan parameter nama, maka akan tampil galat sebagai berikut.



Bad Request (#400)

Missing required parameters: nama

The above error occurred while the Web server was processing your request.

Please contact us if you think this is a server error. Thank you.

Hal ini karena parameter nama tidak ditemukan di URL padahal parameter tersebut dibutuhkan atau *required*. Lihat kode berikut,

```
public function actionHello($nama)
{
```

Oleh karenanya untuk mencegah galat maka kita perlu memberikan *default value* pada variabel tersebut sehingga apabila URL tidak memenuhi parameter yang diharapkan maka tidak muncul galat.

```
public function actionHello($nama="")
{
    return "Hello ".$nama;
}
```

Silakan kode ini diuji coba dengan atau tanpa parameter nama, maka tidak akan muncul galat lagi.

B. Konsep Dasar Routing

Routing adalah pemetaan antara URL dengan konten. Konsep *routing* di Yii sangat-sangat sederhana, karena sifatnya otomatis berdasarkan nama fungsi *action* di *controller* dan tidak perlu kita definisikan secara manual sebagaimana umumnya di *framework PHP* lain.

Routing dimulai dari index.php sebagai *entry script* (anda bisa baca kembali pada bab sebelumnya), dilanjutkan dengan parameter *r* yang merupakan singkatan dari *route*, baru kemudian kita bisa mendefinisikan halaman atau *action* yang ingin kita tuju melalui nilai dari parameter *r* tersebut.

B. 1. Aturan Dasar Routing

Secara umum, aturan-aturan *routing* di Yii mengacu atau tunduk pada aturan *coding standard* (www.php-fig.org). Berikut ini beberapa poin penting:

- *Routing* di Yii sifatnya *case sensitif* atau membedakan huruf besar dan kecil. Secara *default*, routing Yii menggunakan huruf kecil atau *lowercase*.
- *Routing* di Yii otomatis terbentuk sesuai dengan *controller* dan fungsi *action*-nya. Antara *controller* dan fungsi *action* dipisahkan dengan tanda *slash /*.

- Nama fungsi pada *controller* yang akan menjadi *routing* adalah semua fungsi yang diawali dengan kata *action*, contoh : `actionIndex`, `actionCreate`, dsb.
- Jika nama *controller* terdiri dari lebih dari satu kata, misal `HumanResourcesController`, maka *routing* menggunakan *separator dash* (-) untuk memisahkan dua kata tersebut. Yaitu :
 - ∞ `index.php?r=human-resources`

Ingat

Penulisan nama *controller* harus menggunakan format *Studly Caps*

- Demikian juga, jika nama *action* lebih dari satu kata `actionJumlahRoda`, maka *routing*-nya juga akan menggunakan *separator dash* (-) untuk memisahkan dua kata tersebut. Misal pada `MobileController` maka URLnya menjadi sebagai berikut.
 - ∞ `index.php?r=mobile/jumlah-roda`
- Penulisan nama fungsi *action* menggunakan format *camelCase*.
- Secara *default*, ketika kita mengakses aplikasi tanpa menunjuk ke *routing* tertentu maka akan diarahkan ke `SiteController` dan `actionIndex` (`index.php?r=site/index`)

Bagaimana? Sangat logis bukan?

B. 2. Contoh URL

Berikut ini beberapa contoh-contoh URL beserta penjelasannya sesuai dengan aturan dasar routing yang telah dibahas sebelumnya.

∞ `index.php?r=site`

URL di atas akan mengarahkan aplikasi untuk menunjuk ke `SiteController`

∞ `index.php?r=site/index`

URL di atas akan menunjuk ke fungsi `actionIndex` pada `SiteController`

∞ `index.php?r=karyawan/daftar-keluarga`

URL di atas akan menunjuk ke fungsi `actionDaftarKeluarga` pada `KaryawanController`

Jadi kesimpulan yang bisa kita petik dari uraian dan contoh di atas adalah bahwa *routing* di Yii merupakan hubungan antara URL dengan *controller* dan *action*. Secara *default* kita tidak perlu melakukan pengaturan *routing* secara manual, artinya *routing* di Yii bersifat otomatis tanpa perlu didefinisikan secara manual terlebih dahulu. Tentu saja hal ini berbeda dengan umumnya *framework PHP* lain ☺.

Meskipun demikian, Yii juga menyediakan fitur untuk kustomisasi *routing*. Konfigurasinya bisa kita tuliskan di `@app/config/web.php`, yaitu pada bagian *component* tambahkan parameter `urlManager`

```
<?php
...
$config = [
    ...
    'components' => [
        ...
        'urlManager' => [
            'rules' => [
                // tambahkan aturan di sini
            ],
        ],
    ],
]
```

Sederhana bukan? kustomisasi URL belum akan dibahas pada bab ini, *keep calm* yah.

C. Membuat Hello Word Tingkat Lanjut

Kata “Hello word” yang ditampilkan pada bagian sebelumnya hanyalah teks polos, artinya *layout* utama Yii tidak ikut tampil. Oleh karena itu, supaya kata “hello word” tersebut tampil dengan *layout*-nya maka kita perlu mengubah kodingnya. Tidak cukup hanya dengan menggunakan *controller* saja melainkan kita perlu menggunakan *view* juga.

Caranya, buat sebuah fungsi baru pada *controller* misalnya `actionTampil`, lalu gunakan fungsi *render* sebagai berikut.

```
public function actionTampil()
{
```

```
|     return $this->render('hello');
| }
```

Untuk mencoba kode ini, mari kita mengakses *action* tersebut menggunakan URL berikut.

∞ index.php?r=site/tampil

```
View not Found – yii\base\ViewNotFoundException
The view file does not exist: C:\xampp\htdocs\basic\views\site\hello.php

1. in C:\xampp\htdocs\basic\vendor\yiisoft\yii2\base\View.php

218     {
219         $viewFile = Yii::getAlias($viewFile);
```

2.0.10 PHP 7.0.13 Status 500 Route site/tampil Log 22 1 Time 505 ms Memory 3.364 MB

Ops, ternyata terjadi galat. Hal ini disebabkan karena *file view* yang terletak di `@app/views/site/hello.php` tidak ditemukan. Mengapa demikian? Sebab fungsi *render* berfungsi menampilkan *view* sesuai dengan nilai parameter yang didefinisikan.

Pada contoh di atas menggunakan perintah *render* “hello” berarti Yii akan me-*render* *file* bernama `hello.php` pada direktori `@app/views/site/` dimana `site` adalah ID dari *controller* (`SiteController`). Jadi perintah tersebut bukan untuk me-*render* teks “hello”. ☺.

Oleh karena itu, mari kita buat dulu *file* `hello.php`. Sederhana saja, file ini hanya menampilkan teks “Hallo dunia”.

```
<?php
echo "Hallo dunia";
```

Sekarang mari coba kita akses lagi, dan berhasil!

Kita juga bisa mengirimkan variabel dari *controller* ke *view* dengan cara menambahkan parameter pada fungsi *render*. Parameter tersebut dapat didefinisikan dengan menggunakan format *array*.

```
public function actionTampil()
{
    return $this->render('hello', [
        'nama' => 'Hafid Muklasin',
    ]);
}
```

Selanjutnya pada *view* `hello.php` kita bisa menggunakan atau menangkap parameter `nama` yang dikirimkan oleh `actionTampil` sebagai *variabel* biasa.

```
<?php
echo "Hallo ".$nama;
```

Mari kita uji coba.

Done, it's work ☺

Selanjutnya kita bisa mengkombinasikan dengan penambahan parameter di URL sebagaimana yang telah dibahas pada bagian sebelumnya. Misalnya.

```
public function actionTampil($nama="")
{
    return $this->render('hello', [
        'nama' => $nama,
    ]);
}
```

Silahkan dicoba.

D. Tautan Antar Halaman

Sekarang kita akan belajar tentang bagaimana menghubungkan tautan antar halaman yang telah kita buat. Namun sebelumnya kita harus mengetahui terlebih dahulu tentang bagaimana membuat URL di Yii.

D. 1. Membuat URL

Yii menyediakan *class* [[yii\helpers\Url]] untuk menangani pembuatan URL. Berikut ini beberapa contoh cara untuk membuat atau menampilkan URL di Yii. Sebagai latihan, kita bia membuat *action* baru pada SiteController yaitu *actionUrl* serta membuat *file* url.php di direktori view/site.

```
@app/controllers/SiteController.php
```

```
...
public function actionUrl()
{
    return $this->render('url');
}
```

```
@app/views/site/url.php
```

```
<?php
use \yii\helpers\Url;

// URL ke home atau web index
echo Url::home(); echo '<br>';

// URL ke current Controller
echo Url::to(); echo '<br>';

// URL ke current Controller pada Action create
echo Url::to(['create']); echo '<br>';

// URL ke person Controller pada Action index
echo Url::to(['person/index']); echo '<br>';

// URL ke current Controller pada Action index
```

```
// dengan parameter nama yang bernilai Hafid
echo Url::to(['person/index','nama'=>'Hafid']);
```

Fungsi ini bisa berjalan di *controller* dan di *view*.

The screenshot shows a browser window with the URL `localhost/basic/web/index.php?r=site/url`. The page title is "My Company". Below the title, there is a list of URLs:

- `/basic/web/index.php`
- `/basic/web/index.php?r=site/url`
- `/basic/web/index.php?r=site%2Fcreate`
- `/basic/web/index.php?r=person%2Findex`
- `/basic/web/index.php?r=person%2Findex&nama=Hafid`

Kode-kode di atas hanyalah contoh untuk menunjukkan bagaimana cara membuat URL di Yii. Implementasinya akan dijelaskan pada bagian *hyperlink*.

D. 2. Membuat Hyperlink

Hyperlink merupakan tautan yang terdapat pada halaman situs web, dengan tujuan untuk mengarahkan pengunjung web ke suatu target. Adapun target dari suatu tautan bisa berupa halaman lain, bagian dari suatu halaman, *file* unduh, dsb.

Hyperlink dibuat dengan menggunakan *tag* atau kode HTML *anchor*

```
| <a href="URL target">Tekst</a>
```

Kita bisa juga menggabungkan dengan URL di Yii sbb:

```
<?php
use \yii\helpers\Url;
?>
<a href="<?= Url::to(['person/index']) ?>">Data Person</a> <br>
```

Atau untuk lebih elegan, Yii juga telah menyediakan fungsi-fungsi HTML (*helpers*) melalui *class* `[[yii\helpers\Html]]` yang bisa kita gunakan salah satunya untuk meng-generate *tag* HTML *anchor*

```
<?php
use \yii\helpers\Html;

echo Html::a('Example', 'http://www.example.com');
echo "<br>";
echo Html::a('Data Person', ['person/index']);
```

The screenshot shows a browser window with the URL `localhost/basic/web/index.php?r=site/hyperlink`. The page title is "My Company". Below the title, there are three blue hyperlinks:

- Data Person
- Example
- Data Person

Catatan.

Coba buat `actionHyperlink` pada `SiteController` yang me-render *hyperlink*. Kemudian buat *file* `hyperlink.php` pada direktori `@app\views\site\` lalu masukkan kode di atas pada *file* tersebut.

E. Penerapan MVC pada Form

Pada materi yang terdahulu telah dibahas dan dicontohkan tentang penggunaan *controller* dan *view*. Ada satu bagian lagi dalam konsep MVC yang belum lengkap rasanya jika tidak dibahas yaitu model. Sebagaimana telah dijelaskan sebelumnya bahwa model itu berkaitan dengan format data, karenanya penulis memilih satu contoh kasus yang mudah dalam menerapkan fungsi model pada pola MVC yaitu formulir komentar.

Berikut ini contoh tampilan formulir komentar yang akan kita buat dengan menggunakan Yii. Sederhana saja, hanya berisi dua kolom yaitu nama dan pesan.

Nama	Agung Hercules
Pesan	Apa kabar dunia
Simpan	

Untuk membuatnya, maka langkah pertama yang harus kita lakukan adalah dengan mendefinisikan format data yang berkaitan dengan formulir itu pada model.

Sederhana saja, kalau kita lihat sejenak formulir tersebut maka kita bisa menyimpulkan bahwa ada dua data yang diperlukan atau akan di-*input*-kan oleh *user* yaitu data nama dan data pesan. Baik, langsung saja kita membuat *file* model dengan nama Komentar.php pada direktori @app/models/

```
<?php
namespace app\models;
class Komentar extends \yii\base\Model
{
    public $nama;
    public $pesan;

    public function rules()
    {
        return [
            ['nama', 'pesan'], 'required'],
        ];
    }
}
```

Mari kita bahas baris perbaris dari kode model di atas.

Pada baris pertama, terdapat kode *namespace*, apa itu *namespace*? *Namespace* bisa dianalogikan sebagai direktori dari *file* pada aplikasi. *Namespace* diperkenalkan sejak PHP versi 5.3.

Sebagai contoh pada kode di atas,

```
| namespace app\models;
```

Artinya, *file* tersebut terletak pada aplikasi *folder* models. Ingat ini hanya analogi, karena pada dasarnya pendefinisian *namespace* bebas, hanya saja demikianlah praktik terbaiknya ☺

Lalu apa keuntungan menggunakan *namespace*?

Pada PHP, *namespace* digunakan untuk mengatasi dua permasalahan,

1. Nama *class* yang sama

Pada PHP, nama *class* harus unik, maka pada proyek aplikasi yang besar akan sangat sulit mengontrol penamaan *library class* apalagi jika menggunakan *library class* eksternal.

2. Nama *class* yang panjang dan rancu

Seandainya tidak menggunakan *namespace* maka nama *class* bisa dibuat panjang dan bisa jadi tidak terbaca, maka dengan adanya *namespace* pemberian nama *class* akan lebih rapi dan jelas. Di samping itu kita bisa mengaliaskan suatu *class*.

Oke kita kembali ke kode model.

Kita bisa meng-*extends* *class* [\yii\baseModel] untuk membuat sebuah model sederhana. Dimana kita juga bisa menerapkan aturan tentang suatu *field* dengan menggunakan fungsi *rules*. Contoh di atas menunjukkan bahwa *property* nama dan pesan diset sebagai *required*, artinya tidak boleh kosong.

Langkah kedua, kita membuat sebuah fungsi actionKomentar pada *controller* (misal: SiteController).

```
public function actionKomentar()
{
    $model = new \app\models\Komentar();

    return $this->render('komentar', [
        'model' => $model,
    ]);
}
```

Kode di atas di awali dengan menciptakan objek dari model Komentar yang telah kita definiskan sebelumnya.

```
| $model = new \app\models\Komentar();
```

Objek model tersebut kemudian di kirimkan ke *view* komentar menggunakan fungsi *render*. Dengan menggunakan cara ini maka objek model yang dikirimkan ke *view* akan memiliki aturan dari model Komentar.

Langkah ketiga, kita membuat tampilan formulirnya yaitu pada *file*

```
@app/views/site/komentar.php
```

```
<?php
use yii\helpers\Html;
use yii\bootstrap\ActiveForm;
?>
<h1>Komentar</h1>
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'nama') ?>
<?= $form->field($model, 'pesan') ?>
<?= Html::submitButton('Simpan', ['class' => 'btn btn-primary']) ?>
<?php ActiveForm::end(); ?>
```

Kode di atas kalau dalam bahasa HTML biasa, maka akan menjadi seperti berikut ini.

```
<h1>Komentar</h1>
<form action="/basic/web/index.php?r=site/komentar" method="post" role="form">
    <input type="hidden" name="_csrf" value="<?= Html::csrfMetaTags() ?>">
    <div class="form-group field-komentar-nama required">
        <label class="control-label" for="komentar-nama">Nama</label>
        <input type="text" id="komentar-nama" class="form-control" name="Komentar[nama]">
        <p class="help-block help-block-error"></p>
    </div>
    <div class="form-group field-komentar-pesan required">
        <label class="control-label" for="komentar-pesan">Pesan</label>
        <input type="text" id="komentar-pesan" class="form-control" name="Komentar[pesan]">
        <p class="help-block help-block-error"></p>
    </div>
    <button type="submit" class="btn btn-primary">Simpan</button>
</form>
```

Jika anda lebih suka dengan kode yang kedua, tidak masalah ☺

Langkah keempat, waktunya uji coba melalui *web browser*. Mari kita akses menggunakan URL berikut.

```
∞ index.php?r=site/komentar
```

Untuk mengkustomisasi teks *field*, kita bisa tambahkan fungsi *label* pada objek formulir *field* nama

```
| <?= $form->field($model, 'nama')->label('Nama Anda') ?>
```

Formulir ini bukan formulir biasa, melainkan formulir yang dibangun berdasarkan pada model Komentar, dimana pada model tersebut telah didefiniskan data dan aturan bahwa data nama dan pesan tidak boleh kosong. Oleh karena itu, mari

kita uji apakah aturan tersebut telah diterapkan pada formulir tersebut atau tidak. Caranya dengan mengklik tombol Simpan pada formulir berikut tanpa mengisi kolom nama dan pesan. Lihat apa yang terjadi!

The screenshot shows a Yii application's comment creation page. The 'Nama' and 'Pesan' fields are empty and have red borders, indicating they are required. Below each field is a red error message: 'Nama cannot be blank.' and 'Pesan cannot be blank.' respectively. A blue 'Simpan' button is located at the bottom left of the form.

Ternyata setelah tombol simpan diklik, muncul peringatan bahwa properti nama dan pesan tidak boleh kosong. Hal ini berarti aturan yang diterapkan pada model berjalan. Kita juga bisa mengubah aturannya, misalnya hanya properti nama saja yang harus diisi.

```
public function rules()
{
    return [
        ['nama', 'required'],
        ['pesan', 'safe'],
    ];
}
```

Uji coba dan lihat hasilnya ketika kita mengklik tombol Simpan

The screenshot shows the same comment creation page as before, but now both the 'Nama' and 'Pesan' fields are empty and have red borders. The error messages 'Nama cannot be blank.' and 'Pesan cannot be blank.' are still present. The 'Simpan' button is visible at the bottom left.

Oke semua berjalan sesuai yang diharapkan yaitu hanya kolom Nama yang harus diisi. Tapi jangan senang dulu, ini belum selesai . Kecuali validasi, formulir ini belum memproses apapun, artinya ketika formulir di-submit maka kolom isian pengguna akan dibiarkan hambar ☺.

Tugas kita selanjutnya adalah melakukan pemrosesan atas *input* data dari pengguna. Tentunya dengan menambahkannya pada actionKomentar (SiteController).

Perhatikan kode berikut.

```

public function actionKomentar()
{
    $model = new \app\models\Komentar();

    // Jika form di-submit dengan method POST
    if(Yii::$app->request->post()){
        // Kode pemrosesan ketika form di submit
    }
    else{
        return $this->render('komentar', [
            'model' => $model,
        ]);
    }
}

```

Adapun kode lengkapnya sebagai berikut

```

public function actionKomentar()
{
    $model = new \app\models\Komentar();

    // Jika form di-submit dengan method POST
    if(Yii::$app->request->post()){
        $model->load(Yii::$app->request->post());
        if($model->validate()){
            Yii::$app->session->setFlash('success','Terima kasih ');
        }
        else{
            Yii::$app->session->setFlash('error','Maaf, salah!');
        }
        return $this->render('hasil_komentar', [
            'model' => $model,
        ]);
    }
    else{
        return $this->render('komentar', [
            'model' => $model,
        ]);
    }
}

```

Perintah `$model->load(Yii::$app->request->post());` berfungsi untuk membaca semua nilai yang dikirimkan oleh pengguna melalui metode `post` dan disimpan dalam objek model `Komentar`. Method `form` di Yii secara *default* bertipe `post`.

Kemudian `$model->validate()` merupakan perintah untuk mengevaluasi hasil inputan pengguna yang telah disimpan pada objek model, apakah sesuai dengan aturan atau tidak. Validasi ini bersifat *server side*.

Adapun perintah `Yii::$app->session->setFlash('success','Terima kasih ');` dan `Yii::$app->session->setFlash('error','Maaf, salah!');` adalah mekanisme Yii untuk menyimpan pesan sukses atau galat menggunakan `session` yang nantinya bisa ditampilkan pada `view`.

Langkah selanjutnya adalah membuat `view hasil_komentar` yang mana pada `view` tersebut kita tambahkan perintah untuk membaca pesan sukses/galat yang telah diset pada `actionKomentar`.

```
@app/views/site/hasil_komentar.php
```

```

<?php
if (Yii::$app->session->hasFlash('success')){
    echo '<div class="alert alert-success">';
    echo Yii::$app->session->setFlash('success');
    echo '</div>';
    echo '<br>Nama : '.$model->nama;
    echo '<br>Pesanan : '.$model->pesan;
}
else if (Yii::$app->session->hasFlash('error')){
    echo '<div class="alert alert-danger">';
    echo Yii::$app->session->setFlash('error');
    echo '</div>';
}

```

Waktunya uji coba, isi formulir komentar, lalu *submit*.

Komentar

Nama

Pesan

Simpan

Ketika tombol Simpan diklik maka hasilnya sebagai berikut.

Terima kasih

Nama : Hafid
Pesan : Apa kabar

Sukses ☺

Pada kasus ini kita hanya menampilkan data saja, artinya komentar pengguna tidak akan disimpan ke dalam basis data. Adapun pembahasan tentang penyimpanan data ke basis data, akan kita pelajari pada bab-bab selanjutnya.

F. Membuat Widget Sederhana

Widget adalah blok atau bagian dari tampilan yang dapat digunakan kembali. Widget digunakan pada view untuk membuat elemen antar muka yang kompleks namun bisa dikonfigurasi dengan gaya pendekatan berbasis objek. Yii memiliki banyak widget bawaan yang membantu *programmer* untuk membuat bagian dari tampilan aplikasi webnya dengan mudah, diantaranya: Gridview, Detailview, Form, dll.

Pada contoh kode sebelumnya yaitu *file*

```
@app/views/site/hasil_komentar.php
```

Dimana kita melihat bahwa ada mekanisme untuk menampilkan pesan peringatan dari *controller* yang kemungkinan akan kita gunakan pada prosedur-prosedur lain. Oleh karena itu, supaya kita tidak perlu menulis kode tersebut berulang kali maka kita bisa menjadikannya sebagai *class widget*. Hal ini memenuhi konsep DRY atau *Don't Repeat Yourself*.

Caranya:

- Buat *file widget* misalnya Alert.php kemudian letakkan *file* tersebut pada *folders* @app/widgets (sebenarnya tidak harus pada *folder* dengan nama ini alias bebas)
- Kode *file widget* Alert kita, kira-kira sebagai berikut

```
<?php
namespace app\widgets;
use Yii;
class Alert extends \yii\bootstrap\Widget
{
    public function init()
```

```

    {
        parent::init();
        if (Yii::$app->session->hasFlash('success')){
            echo '<div class="alert alert-success">';
            echo Yii::$app->session->getFlash('success');
            echo '</div>';
        }
        else if (Yii::$app->session->hasFlash('error')){
            echo '<div class="alert alert-danger">';
            echo Yii::$app->session->getFlash('error');
            echo '</div>';
        }
    }
}

```

Class widget Alert kita extends dengan *class [[yii\bootstrap\Widget]]* karena *class Alert* kita menggunakan atau bergantung dengan *class-class* CSS Twitter Bootstrap.

Mudah sekali bukan? Lalu bagaimana cara menggunakannya?

Berikut ini cara menggunakan *widget Alert*.

```

use app\widgets\Alert;
echo Alert::widget();

```

Jika kita implementasikan pada *file* sebelumnya yaitu @app/views/site/hasil_komentar.php maka akan menjadi sebagai berikut.

```

<?php
use app\widgets\Alert;
echo Alert::widget();

if (Yii::$app->session->hasFlash('success')){
    echo '<br>Nama : '.$model->nama;
    echo '<br>Pesanan : '.$model->pesan;
}

```

Agar berlaku global, maka kita juga bisa mengimplementasikan *widget Alert* kita pada *layout* utama yaitu pada *file* @app/views/layouts/main.php

```

<div class="container">
    <?= Breadcrumbs::widget([
        'links' => isset($this->params['breadcrumbs']) ? $this-
>params['breadcrumbs'] : [],
    ]) ?>
    <!-- TAMBAHKAN WIDGET ALERT DI SINI INI -->
    <?= \app\widgets\Alert::widget(); ?>
    <?= $content ?>
</div>

```

Sehingga pada *file* @app/views/site/hasil_komentar.php cukup seperti berikut saja:

```

<?php
if (Yii::$app->session->hasFlash('success')){
    echo '<br>Nama : '.$model->nama;
    echo '<br>Pesanan : '.$model->pesan;
}

```

Kemana setelah ini?

Widget merupakan bagian dari tampilan. Adakalanya sebuah *widget* membutuhkan kode CSS atau Javascript tertentu, baik itu eksternal maupun *internal*. Pada *widget Alert* yang telah kita buat di atas kebetulan CSS untuk *alert*-nya, secara *default* sudah *di-include*, sehingga kita tidak perlu memikirkan hal itu. Maksud yang ingin penulis sampaikan di sini adalah bahwa kita juga bisa memasukkan kode-kode CSS dan Javascript pada *widget* buatan kita ini.

Di samping itu kita juga bisa menjadikan *widget* ini sebagai *extension* sehingga bisa digunakan oleh orang lain. Pembahasan tentang hal ini bisa anda jumpai pada bab-bab selanjutnya. *Keep reading*. ☺. Silakan berimajinasi, buat *widget-widget*-mu sendiri.

G. Mempercantik URL

Sebagaimana yang telah kita bahas pada bab awal bahwa, secara *default* URL pada Yii memiliki format sebagai berikut.

∞ index.php?r=controllerID/actionID

Nah, pada bagian ini kita akan mempercantik URL *address* dengan tujuan agar URL *address* menjadi lebih alamiah dan mudah. Caranya, pada bagian *components* di file @app/config/web.php, tambahkan parameter *urlManager*, kemudian masukkan sub parameter *enablePrettyUrl* dengan nilai *true*, lihat kode berikut.

```
'components' => [
    ...
    'urlManager' => [
        'enablePrettyUrl' => true,
    ],
    ...
],
```

Mari kita uji coba, akses salah satu menu di Yii misalnya menu About, lalu perhatikan alamat URL yang dibuat oleh Yii.

Maka format URL *address* berubah menjadi

∞ index.php/controllerID/actionID

Lebih cantik kan? Nah langkah berikutnya kita hilangkan index.php dari URL, dengan cara menambahkan parameter 'showScriptName' dan diset bernilai false,

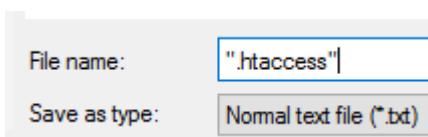
```
'components' => [
    ...
    'urlManager' => [
        'enablePrettyUrl' => true,
        'showScriptName' => false,
    ],
    ...
],
```

Mari kita uji coba dan hasilnya

Dan ops.. ternyata terjadi galat,

Pada *web server* Apache, untuk menyembunyikan *script* harus dibantu dengan konfigurasi tersendiri, bisa melalui httpd.conf atau yang sederhana melalui file .htaccess

Pada OS Windows, untuk membuat file .htaccess kita bisa gunakan Notepad, lalu simpan dengan nama “.htaccess” (sertakan tanda petik dua) maka file yang akan tersimpan adalah .htaccess



Adapun isi dari file .htaccess sebaiknya mengikuti praktik terbaik dari Yii yaitu

```
# use mod_rewrite for pretty URL support
RewriteEngine on
# If a directory or a file exists, use the request directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# Otherwise forward the request to index.php
RewriteRule . index.php
```

Simpan atau letakkan file ini pada @app/web/.htaccess

Kalau sudah, sekarang mari kita coba lagi

Berhasil ☺, sehingga format URL *address* sekarang dan seterusnya menjadi lebih sederhana yaitu

∞ controllerID/actionID

H. Kesimpulan

Banyak sekali yang telah kita pelajari pada bab ini, yaitu tentang implementasi MVC di Yii sekaligus konsep *routing*nya. Untuk melengkapi aplikasi MVC yang telah kita buat maka pada bab selanjutnya kita akan belajar tentang bagaimana menghubungkan Yii dengan basis data, serta melakukan operasi CRUD pada basis data.

5

Bekerja dengan Basis Data

Pada bab ini kita akan belajar tentang:

- Koneksi basis data pada Yii
- Konsep *ActiveRecord*
- CRUD menggunakan basis data
- Migrasi basis data

Basis data merupakan bagian penting pada sebuah aplikasi yang berfungsi untuk menyimpan data. Framework Yii secara *default* mendukung banyak basis data populer seperti MySQL, MariaDB, PostgreSQL, Ms SQL Server, Oracle, dll. Hal ini karena Yii menggunakan teknologi PDO PHP extension dalam hal koneksi datanya.

Pada panduan ini kita akan menggunakan basis data MySQL/MariaDB. Pada praktiknya kita juga bisa menggantinya dengan basis data lain tanpa ada perubahan yang signifikan.

A. Persiapan Basis data

Penulis berasumsi bahwa anda sudah memiliki kemampuan dasar tentang basis data, mengetahui istilah-istilah pada basis data. Namun jika belum maka penulis menyarankan agar anda bisa membaca-baca terlebih dahulu tentang topik dasar basis data.

Sebagai contoh, kita akan membuat basis data bernama “yii2basic”, kemudian buat tabel “employee” pada basis data tersebut dan masukkan beberapa data.



Anda juga bisa mendapatkannya dengan menjalankan perintah *query* berikut.

```

CREATE database `yii2basic`;
use `yii2basic`;
CREATE TABLE IF NOT EXISTS `employee` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `age` int(3) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `employee` VALUES (NULL,'Hafid',29);
INSERT INTO `employee` VALUES (NULL,'Junaidi',35);
INSERT INTO `employee` VALUES (NULL,'Dewi',24);
INSERT INTO `employee` VALUES (NULL,'Agung',51);
    
```

Penulis menggunakan *tools* PHPMyAdmin yang sudah terinstalasi ketika kita melakukan instalasi Xampp. Silakan buka <http://localhost/phpmyadmin> untuk mengeksekusi perintah *query* di atas

B. Konfigurasi Basis Data pada Yii

B. 1. Konfigurasi Global

Pada *template Basic*, konfigurasi global untuk koneksi basis data bisa kita temukan pada file @app/config/db.php

```
<?php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=yii2basic',
    'username' => 'root',
    'password' => '123456',
    'charset' => 'utf8',
];
;
```

Silakan anda sesuaikan nama *database*, *username* dan kata sandi sesuai dengan konfigurasi basis data anda. Yii menggunakan *class* [[yii\db\Connection]] untuk mengatur koneksi basis data.

Konfigurasi yang disimpan pada file db.php ini sebenarnya juga diinclude ke dalam konfigurasi utama yaitu file @app/config/web.php

```
| 'db' => require(__DIR__ . '/db.php'),
```

Sehingga kita bisa memanggil konfigurasi koneksi basis data melalui perintah

```
| Yii::$app->db
```

Koneksi basis data ini berlaku secara *global* pada seluruh bagian aplikasi.

B. 2. Konfigurasi Lokal

Adakalanya pada aplikasi, kita hanya perlu koneksi ke basis data sekali saja atau pada suatu *action* saja, maka dalam kasus ini sebaiknya kita cukupkan menggunakan konfigurasi lokal untuk koneksi basis data.

```
$db = new \yii\db\Connection([
    'dsn' => 'mysql:host=localhost;dbname=yii2basic',
    'username' => 'root',
    'password' => '123456',
    'charset' => 'utf8',
]);
;
```

B. 3. Uji Coba Query

Untuk menguji apakah koneksi basis data kita benar-benar berjalan dengan baik adalah dengan mencoba beberapa *query* untuk mengakses basis data.

Kita bisa menggunakan *class* [[yii\db\Command]] untuk mengeksekusi *query*, *class* ini telah *built* dengan *class* [[yii\db\Connection]]. Sehingga kita bisa melakukannya melalui objek db *connection* yang telah kita buat sebelumnya baik yang lokal maupun yang global.

Query Select

Sebagai contoh, kita buat sebuah fungsi actionQuery() pada SiteController.

```
public function actionQuery()
{
    $db = Yii::$app->db;
    $command = $db->createCommand('SELECT * FROM employee');
    $employees = $command->queryAll();
    //Ekstrak data
    foreach($employees as $employee){
        echo "<br>";
        echo $employee['id']." ";
        echo $employee['name']." ";
        echo "(".$employee['age'].") ";
    }
}
```

Uji coba di *web browser* dengan mengakses URL sebagai berikut

∞ <http://localhost/basic/web/site/query>

Rank	Name	Age
1	Hafid	29
2	Junaidi	35
3	Dewi	24
4	Agung	51

Contoh lain

```
public function actionQuery2()
{
    $db = Yii::$app->db;
    // return a single row
    $employee = $db->createCommand('SELECT * FROM employee WHERE id=1')->queryOne();
    echo $employee['id']." ";
    echo $employee['name']." ";
    echo "($employee['age']).";
    echo "<hr>";
    // return a single column (the first column)
    $names = $db->createCommand('SELECT name FROM employee')
        ->queryColumn();
    print_r($names);
    echo "<hr>";
    // return a scalar
    $count = $db->createCommand('SELECT COUNT(*) FROM employee')
        ->queryScalar();
    echo "Jumlah employee ".$count;
    echo "<hr>";
}
```

Uji coba di *web browser*

1 Hafid (29)

Array ([0] => Hafid [1] => Junaidi [2] => Dewi [3] => Agung)

Jumlah employee 4

Untuk mencegah *SQL Injection*, kita juga bisa mem-*binding* parameter.

```
// Binding Parameter
$employee = $db->createCommand('SELECT * FROM employee WHERE id=:id', ['id'=> 2])->queryOne();
```

Query Non Select

Untuk *Query* yang *non select* maka kita bisa gunakan fungsi `yii\db\Command::execute()`. Perhatikan kode berikut.

```
| $db->createCommand('UPDATE employee SET age=30 WHERE id=1')->execute();
```

fungsi ini mengembalikan jumlah *row* yang berhasil diubah.

```
$row_affected = $db->createCommand('UPDATE employee SET age=30 WHERE id=1')->execute();
echo $row_affected.' row affected';
```

Untuk *query INSERT, UPDATE dan DELETE* kita bisa gunakan fungsi `insert()`, `update()`, `delete()`. Lihat contoh berikut.

```
// INSERT (table name, column values)
$db->createCommand()->insert('employee', [
    'name' => 'Hasan',
    'age' => 30,
])->execute();
// UPDATE (table name, column values, condition)
$db->createCommand()->update('employee', ['status' => 30], 'id = 1')
->execute();
// DELETE (table name, condition)
$db->createCommand()->delete('employee', 'age >30')
->execute();
```

Sedangkan untuk *insert* banyak data sekaligus, Yii menyediakan fungsi `batchInsert()`.

```
// table name, column names, column values
$db->createCommand()->batchInsert('employee',
    ['name', 'age'],
    [
        ['Farhan', 30],
```

```
[ 'Jane', 20],
[ 'Linda', 25],
])->execute();
```

Selengkapnya silakan baca di

∞ <http://www.yiiframework.com/doc-2.0/guide-db-dao.html>

C. Active Record Dasar

Active Record adalah sebuah *class* antar muka yang digunakan untuk mengakses dan memanipulasi data yang disimpan pada basis data. *Class Active Record* bisa diibaratkan sebagai sebuah struktur tabel basis data yang mana terdapat informasi tentang *field* dan tipe datanya. *Class Active Record* juga bisa berisi tentang validasi, serta fungsi-fungsi yang digunakan untuk mengakses dan memanipulasi data.

Konsep *Active Record* di Yii sama dengan konsep *Active Record Pattern* pada umumnya (*Object Relational Model/ORM*). *Class Active Record* pada Yii merupakan model pada konsep MVC, yaitu model yang merepresentasikan tabel basis data.

C. 1. Model Active Record

Berikut ini contoh deklarasi minimal dari model *Active Record* dari tabel *employee* (@app/models/Employee.php).

```
<?php
namespace app\models;
use yii\db\ActiveRecord;

class Employee extends ActiveRecord
{
    public static function tableName()
    {
        return 'employee';
    }
}
```

Class [[yii\db\ActiveRecord]] sebenarnya merupakan *extend* dari *class [[yii\base\Model]]*, sehingga semua fitur *model* di Yii bisa diterapkan juga seperti atribut, *label*, validasi, dsb

C. 2. Implementasi Active Record

Berikut ini contoh penggunaan *class Active Record* untuk *query select* data yaitu dengan menambahkan fungsi *actionActiveRecord* pada SiteController.

```
public function actionActiveRecord()
{
    $employees = \app\models\Employee::find()->all();
    foreach($employees as $employee){
        echo "<br>";
        echo $employee->id." ";
        echo $employee->name." ";
        echo "(".$employee->age.") ";
    }
}
```

Gunakan `use \app\models\Employee;` di awal file, sehingga ketika menggunakan cukup dengan menuliskan Employee saja tanpa Namespace lengkap. Hampir sama dengan *query select* yang telah dibahas pada bagian lalu, namun bedanya adalah nilai kembaliannya berupa objek.

Mari kita uji coba dengan mengakses fungsi tersebut..

∞ <index.php?r=site/active-record>

Rank	Name	Age
1	Hafid	29
2	Junaidi	35
3	Dewi	24
4	Agung	51

Contoh lain

```
// SELECT * FROM `employee` WHERE `id` = 2
$employee = Employee::find()
->where(['id' => 2])
->one();
echo $employee->name;

// return semua employee yang usianya > 25 dan sortir berdasarkan ID
// SELECT * FROM `employee` WHERE `age` > 25 ORDER BY `id`
$employees = Employee::find()
->where(['>', 'age', 25])
->orderBy('id')
->all();
print_r($employees);

// SELECT COUNT(*) FROM `employee`
$count = Employee::find()
->count();
echo $count;
```

Bentuk fungsi `find() ->one()` bisa disingkat dengan `findOne()`, demikian juga dengan bentuk `find() ->all()` bisa disingkat dengan `findAll()`

```
// SELECT * FROM `employee` WHERE `id` = 2
$employee = Employee::findOne(2);

// SELECT * FROM `employee` WHERE `id` IN (1,2,3)
$employees = Employee::findAll([1,2,3]);

// SELECT * FROM `employee` WHERE `age` = 30 LIMIT 1
$employee = Employee::findOne([
    'age' => 30
]);

// SELECT * FROM `employee` WHERE `age` = 30
$employee = Employee::findOne([
    'age' => 30
]);
```

Sedangkan untuk kasus *query insert update dan delete*, perlakunya agak sedikit berbeda. Lihat contoh berikut.

```
// insert a new row of data
// INSERT INTO employee(name,age) VALUES('James',34)
$employee = new Employee();
$employee->name = 'James';
$employee->age = 34;
$employee->save();

// update an existing row of data
// UPDATE employee SET name='James Gordon' WHERE id = 5
$employee = Employee::findOne(5);
$employee->name = 'James Gordon';
$employee->save();

// delete
// DELETE FROM employee WHERE id=5
$employee = Employee::findOne(5);
$employee->delete();
```

Silakan dicoba dengan varian lain, karena pemahaman anda pada bagian ini akan menjadi bekal berharga dalam mengikuti panduan selanjutnya.

Perhatian:

Untuk menjalankan kode di atas, tentu kita harus membuat fungsi di controller sebagaimana yang telah dicontohkan sebelumnya.

D. ActiveRecord Tingkat Lanjutan

D. 1. Validasi

Validasi data di Yii tersentral pada model melalui *method rules*, dan secara *default* mendukung *client* maupun *server side validation*. Artinya hanya dengan mengkonfigurasi sekali pada model maka validasi untuk *client* (melalui Javascript) dan server akan aktif tanpa koding apapun lagi.

```
public function rules()
{
    return [
        ['name', 'age'], 'required'],
    ];
}
```

Berikut ini *class-class validator* yang bisa kita gunakan dan telah tersedia secara *built-in* di Yii

- boolean: yii\validators\BooleanValidator
- captcha: yii\captcha\CaptchaValidator
- compare: yii\validators\CompareValidator
- date: yii\validators\DateValidator
- default: yii\validators\DefaultValueValidator
- double: yii\validators\NumberValidator
- each: yii\validators\EachValidator
- email: yii\validators\EmailValidator
- exist: yii\validators\ExistValidator
- file: yii\validators\FileValidator
- filter: yii\validators\FilterValidator
- image: yii\validators\ImageValidator
- in: yii\validators\RangeValidator
- integer: yii\validators\NumberValidator
- match: yii\validators\RegularExpressionValidator
- required: yii\validators\RequiredValidator
- safe: yii\validators\SafeValidator
- string: yii\validators\StringValidator
- trim: yii\validators\FilterValidator
- unique: yii\validators\UniqueValidator
- url: yii\validators\UrlValidator

Berikut ini contoh deklarasi dari masing-masing *validator*

Boolean

Validator ini mengecek *input* sebagai *boolean* yaitu 1 atau 0

```
[ 'selected', 'boolean',

    // gender = pria atau wanita
    ['gender', 'boolean', 'trueValue' => 'male', 'falseValue' => 'female',
     'strict' => true],
],
```

Captcha

```
[ 'verificationCode', 'captcha',
```

Validator ini biasanya digunakan dengan class [[yii\captcha\CaptchaAction]] dan [[yii\captcha\Captcha]] untuk memastikan bahwa *input-an pengguna* sama dengan *verification code* yang ditampilkan oleh *widget captcha*.

Berikut ini properti yang bisa digunakan:

- caseSensitive: memperhatikan huruf besar dan kecil. *Defaultnya* adalah *false*.
- captchaAction: *routing* yang berkaitan dengan *action captcha* yang me-render *image captcha*. *Defaultnya* adalah 'site/captcha'.
- skipOnEmpty: jika *field captcha* boleh kosong. *Defaultnya* adalah *false*, yang berarti *input captcha* tidak boleh kosong.

Compare

```
[  
    // validates if the value of "password" attribute equals to that of  
    // "password_repeat"  
    ['password', 'compare'],  
  
    // validates if age is greater than or equal to 30  
    ['age', 'compare', 'compareValue' => 30, 'operator' => '>='],  
,]
```

Sebagaimana contoh di atas, *validator* ini memiliki dua fungsi:

- membandingkan nilai dari dua atribut atau *field*
- membandingkan nilai dari atribut dengan nilai tertentu

Berikut ini properti yang bisa kita gunakan:

- compareAttribute: adalah nama atribut yang dibandingkan. Secara *default* jika digunakan untuk membandingkan dua atribut maka atribut yang dibandingkan boleh untuk tidak diisi asalkan nama atributnya diakhiri dengan *_repeat*. Pada contoh di atas, artinya nilai dari atribut *password* dibandingkan dengan atribut *password_repeat*.
- compareValue: jika kita ingin membandingkan atribut dengan nilai tertentu maka kita bisa menggunakan properti ini asalkan properti compareAttribute tidak didefinisikan.
- operator: operator pembanding. *Default*-nya adalah `==`.

Berikut ini *operator* yang didukung:

- `==` cek apakah dua nilai *equal*.
- `==<` cek apakah dua nilai *equal*, dan apakah tipe datanya sama (*mode strict*). Contoh ‘2’ (*string*) beda dengan 2 (*integer*)
- `!=` cek apakah dua nilai TIDAK *equal*.
- `!=<` cek apakah dua nilai TIDAK *equal*, dan *mode strict*.
- `>` cek apakah nilai lebih besar dari compareValue.
- `>=` cek apakah nilai lebih besar sama dengan compareValue.
- `<` cek apakah nilai lebih kecil dari compareValue.
- `<=` cek apakah nilai lebih kecil sama dengan compareValue.

Date

```
[  
    [['from_date', 'to_date'], 'date'],  
,]
```

Validator ini akan mengecek apakah nilai dari input adalah *date*, *time* atau *datetime* sesuai dengan formulir standardnya.

Validator ini juga bisa digunakan untuk mengkonversi nilai *input* ke dalam format UNIX *timestamp* atau yang lainnya dan menyimpannya pada atribut tertentu melalui properti *timestampAttribute*.

Double

```
[  
    // checks if "salary" is a double number  
    ['salary', 'double'],  
,]
```

Validator ini mengecek apakah nilai *input* bertipe *double number* (angka). Hampir sama dengan *validator number*.

Properti yang bisa digunakan:

- max: nilai maksimal input yang diizinkan. Jika nilai ini tidak ditentukan maka validator tidak akan mengecek nilai maksimalnya.
- min: nilai minimal input yang diizinkan. Jika nilai ini tidak ditentukan maka validator tidak akan mengecek nilai minimalnya.

Each

```
[  
    // checks if every category ID is an integer  
    ['categoryIDs', 'each', 'rule' => ['integer']],  
,]
```

Validator ini untuk memvalidasi atribut bertipe *array*. *Validator* ini memvalidasi setiap elemen dari *array* dengan *rule* tertentu. Pada contoh di atas setiap elemen dari atribut *categoryIDs* harus bertipe *integer*.

Properti yang bisa digunakan:

- rule: array dari *rule* validasi misal ['integer','string'].
- allowMessageFromRule: *default*-nya bernilai *true*, artinya akan menampilkan pesan galat sesuai dengan *rule*-nya. Namun jika diset *false* maka akan menampilkan pesan galat sesuai dengan properti *message*.

Catatan:

Jika nilai atribut bukan array maka validasi akan gagal dan menampilkan pesan galat

Email

```
[  
    // checks if "email" is a valid email address  
    ['email', 'email'],  
]
```

Validator ini akan mengecek apakah nilai *input* pengguna merupakan *email* dengan format yang valid atau bukan.

Properti yang bisa kita gunakan:

- allowName: Jika *true* maka kita pengguna boleh menambahkan nama pada alamat *email*, contoh: John Smith <john.smith@example.com>. *Default*-nya *false*.

Exist

```
[  
    // a1 needs to exist in the column represented by the "a1" attribute  
    ['a1', 'exist'],  
    // a1 needs to exist, but its value will use a2 to check for the existence  
    ['a1', 'exist', 'targetAttribute' => 'a2'],  
    // a1 and a2 need to exist together, and they both will receive error message  
    [['a1', 'a2'], 'exist', 'targetAttribute' => ['a1', 'a2']],  
    // a1 and a2 need to exist together, only a1 will receive error message  
    ['a1', 'exist', 'targetAttribute' => ['a1', 'a2']],  
    // a1 needs to exist by checking the existence of both a2 and a3 (using a1 value)  
    ['a1', 'exist', 'targetAttribute' => ['a2', 'a1' => 'a3']],  
    // a1 needs to exist. If a1 is an array, then every element of it must exist.  
    ['a1', 'exist', 'allowArray' => true],  
]
```

Validator ini mengecek apakah nilai input ada pada atribut atau *field* di tabel basis data.

File

```
[  
    // checks if "primaryImage" is an uploaded image file in PNG, JPG or  
    // GIF format.  
    // the file size must be less than 1MB  
    ['primaryImage', 'file', 'extensions' => ['png', 'jpg', 'gif'], 'maxSize' => 1024  
    *1024],  
]
```

Validator ini memvalidasi *file* yang diunggah oleh pengguna. Kita bisa membatasi ekstensi *file* apa saja yang boleh diunggah, berapa maksimal besarnya, dsb.

Filter

```
[  
    // trim "username" and "email" inputs  
    [['username', 'email'], 'filter', 'filter' => 'trim', 'skipOnArray' => true],  
  
    // normalize "phone" input  
    ['phone', 'filter', 'filter' => function ($value) {  
        // normalize phone input here  
        return $value;  
    }],  
]
```

Validator ini bukan memvalidasi data namun hanya menyaring data yang di-*input* oleh pengguna.

```
[['property', 'filter', 'filter' => 'boolval'],  
 ['property', 'filter', 'filter' => 'intval'],
```

Image

Validator ini merupakan *extends* dari *validator file* dengan menambahkan *size* dari *image*

```
[  
    ['foto', 'image', 'extensions' => 'png, jpg',  
     'minWidth' => 100, 'maxWidth' => 1000,  
     'minHeight' => 100, 'maxHeight' => 1000,  
    ],  
]
```

In

Validator ini mengecek apakah *input* pengguna bernilai diantara *range* nilai tertentu

```
[  
    ['level', 'in', 'range' => [1, 2, 3]],  
]
```

Terdapat juga parameter *not* jika ingin memvalidasi *not range*

Integer

Validator ini mengecek apakah *input* pengguna bernilai *integer*.

```
[  
    ['age', 'integer'],  
    ['umur', 'integer', 'min'=>10, 'max'=>60],  
]
```

Match

Validator ini mengecek nilai *input* yang sesuai dengan menggunakan *regular expression*

```
[  
    ['username', 'match', 'pattern' => '/^[a-z]\w*$/i']  
]
```

Tersedia parameter *not* yang berarti *not match*. Default *false*.

Number

Validator ini mengecek nilai input apakah berupa *number* atau *double*

```
[  
    ['salary', 'number'],  
    ['gaji', 'number', 'min'=>1000000, 'max'=>5000000],  
]
```

Kita juga bisa menambahkan nilai minimal dan maksimalnya.

Required

Validator ini untuk memastikan pengguna meng-*input field* tertentu

```
[  
    [['username', 'password'], 'required'],  
]
```

Safe

Validator ini berfungsi untuk menskip validasi, artinya data *input-an* tidak divalidasi. Cocok untuk *field* yang di-*input* oleh sistem.

```
[  
    ['description', 'safe'],  
]
```

String

Validator ini untuk mengecek apakah *input-an* pengguna bertipe *string*

```
[  
    ['username', 'string', 'length' => [4, 24]],  
]
```

Terdapat parameter *min* dan *max* untuk menentukan spesifikasi panjang karakter.

Trim

Validator ini tidak bermaksud untuk memvalidasi data melainkan membuang spasi kosong diawal dan akhir dari data.

```
[  
    [['username', 'email'], 'trim'],  
]
```

Unique

```
[  
    // a1 needs to be unique in the column represented by the "a1" attribute  
    ['a1', 'unique'],  
    // a1 needs to be unique, but column a2 will be used to check the uniqueness of t  
he a1 value  
    ['a1', 'unique', 'targetAttribute' => 'a2'],  
    // a1 and a2 need to be unique together, and they both will receive error message  
    [['a1', 'a2'], 'unique', 'targetAttribute' => ['a1', 'a2']],  
    // a1 and a2 need to be unique together, only a1 will receive error message  
    ['a1', 'unique', 'targetAttribute' => ['a1', 'a2']],  
    // a1 needs to be unique by checking the uniqueness of both a2 and a3 (using a1 v  
alue)  
    ['a1', 'unique', 'targetAttribute' => ['a2', 'a1' => 'a3']],  
]
```

URL

Validator ini untuk memvalidasi format URL.

```
[  
    ['website', 'url', 'defaultScheme' => 'http'],  
]
```

Parameter yang tersedia:

- validSchemes: daftar URI *schemes* yang dianggap valid. *Defaults* ['http', 'https'].
- defaultScheme: *default URI scheme* yang akan ditambahkan diawal jika pengguna meng http atau https.

Kustomisasi Validasi

Untuk kasus dimana tidak terdapat *validator* yang sesuai maka kita bisa membuat *validator* sendiri.

```
['birthday', function($attribute, $params){  
    $lahir = date('Y', strtotime($this->$attribute));  
    $sekarang = date('Y');  
    if(($sekarang-$lahir) < 10){  
        $this->addError($attribute, 'Anda terlalu muda :)')  
    }  
}]
```

Selengkapnya tentang validasi bisa dibaca di

∞ <http://www.yiiframework.com/doc-2.0/guide-tutorial-coreValidators.html>

D. 2. Relasi Tabel

Untuk bekerja dengan data yang berelasi, maka kita perlu deklarasikan relasi antar tabel pada model. Pada contoh ini adalah relasi antara tabel Customer dengan tabel Order, yaitu relasi one to many.

Berikut deklarasi pada model *Customer*.

```
class Customer extends ActiveRecord  
{  
    public function getOrders()  
    {  
        return $this->hasMany(Order::className(), ['customer_id' => 'id']);  
    }  
}
```

Berikut deklarasi pada model *Order*

```
class Order extends ActiveRecord  
{  
    public function getCustomer()  
    {  
        return $this->hasOne(Customer::className(), ['id' => 'customer_id']);  
    }  
}
```

Catatan:

Apabila tabel *Customer* dan *Order* telah direlasikan di level basis data, dan modelnya kita generate menggunakan Gii, maka deklarasi relasi tersebut sudah secara otomatis ter-generate juga, sehingga kita tidak perlu melakukannya secara manual.

Apabila relasi telah dideklarasikan maka akan memudahkan kita untuk mengakses tabel relasi. Perhatikan contoh berikut

```
// SELECT * FROM `customer` WHERE `id` = 123
$customer = Customer::findOne(123);
// SELECT * FROM `orders` WHERE `customer_id` = (SELECT id FROM `customer` WHERE `id` = 123)
$orders = $customer->orders;
```

Contoh di atas, saat data Customer telah di-select maka kita bisa mendapatkan data Order yang pernah dilakukan oleh Customer tersebut dengan cara `$customer->orders`.

Demikian juga sebaliknya, apabila data Order telah didapatkan maka otomatis data Customer yang melakukan Order juga akan didapat. Perhatikan kode berikut.

```
// SELECT * FROM `order` WHERE `id` = 123
$order = Order::findOne(123);
// SELECT * FROM `customer` WHERE `id` = (SELECT customer_id FROM `order` WHERE `id` = 123)
$customer = $order->customer;
```

D. 3. Transaksi Data

Transaksi data atau *data transaction* merupakan fitur pada basis data yang memungkinkan suatu data dikembalikan pada kondisi semula ketika terjadi galat saat proses manipulasi data.

Transaksi data cocok diterapkan apabila kita melakukan beberapa perintah SQL(*create, update, delete*) sekaligus. Hal ini dapat digunakan untuk mencegah terjadinya kerancuan data akibat terjadinya kegagalan saat proses manipulasi data.

Pada Yii, transaksi data bisa diterapkan sebagai berikut.

```
$transaction = Employee::getDb()->beginTransaction();
try {
    $employee = new Employee();
    $employee->name = 'Hasan';
    $employee->age = 22;
    $employee->save();
    // ...other DB operations...
    $transaction->commit();
} catch(\Exception $e) {
    $transaction->rollBack();
    throw $e;
}
```

Setiap transaksi data harus diawali dengan pemanggilan fungsi `beginTransaction()`, kemudian kita gunakan metode `try` dan `catch` yang merupakan metode standard PHP untuk menangkap kemungkinan galat yang terjadi pada proses utama yang berada di bagian kalang `try`. Karenanya setelah proses selesai diakhiri dengan pemanggilan fungsi `commit()` yaitu fungsi yang memerintahkan agar manipulasi data disimpan.

Bagian kalang `catch` adalah kode yang akan dieksekusi ketika terjadi galat pada proses utama. Karenanya dipanggillah fungsi `rollBack` untuk mengembalikan kodisi data seperti semula atau sebelum ada manipulasi data.

E. Membuat CRUD Sederhana

Berbekal pengetahuan tentang MVC dan *Active Record*, kita akan mencoba membuat aplikasi CRUD (*create read update delete*) sederhana. Studi kasus yang akan digunakan adalah membuat CRUD untuk tabel *employee*. Model *employee* telah kita buat sebelumnya, namun karena CRUD ini nantinya berkaitan dengan formulir *input* pengguna maka kita perlu menambahkan fungsi `rules` pada model untuk mengatur *field-field* yang digunakan untuk proses *input* data.

Rules merupakan definisi dari tipe data.

```
<?php
namespace app\models;
use yii\db\ActiveRecord;

class Employee extends ActiveRecord
{
```

```

public static function tableName()
{
    return 'employee';
}

public function rules()
{
    return [
        [['name', 'age'], 'required'],
        [['name'], 'string'],
        [['age'], 'integer'],
    ];
}
}

```

Untuk lebih memudahkan kita, sebaiknya kita buat satu *controller* baru untuk penanganan CRUD tabel *employee* ini yaitu EmployeeController.

```

<?php
namespace app\controllers;
use Yii;
use yii\web\Controller;
use app\models\Employee;
class EmployeeController extends Controller
{
}

```

Fungsi atau *action* pada *controller* ini akan kita jabarkan sebagai berikut:

E. 1. Create

Buat fungsi *actionCreate* untuk me-*render* formulir *create employee*

```

public function actionCreate()
{
    $model = new Employee();
    return $this->render('create', [
        'model' => $model,
    ]);
}

```

Buat *view* formulir untuk *create employee* baru dengan cara membuat *file* baru.

@app/views/employee/create.php

```

<?php
use yii\helpers\Html;
use yii\bootstrap\ActiveForm;
?>
<h1>Employee</h1>
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'name') ?>
<?= $form->field($model, 'age') ?>
<?= Html::submitButton('Simpan', ['class' => 'btn btn-primary']) ?>
<?php ActiveForm::end(); ?>

```

Mari kita uji coba dengan mengakses URL berikut.

∞ <http://localhost/basic/web/employee/create>

The screenshot shows a Yii-based web application. The URL in the browser is `localhost/basic/web/employee/create`. The page title is "Employee". There are two input fields: "Name" and "Age", both currently empty. Below the fields is a blue button labeled "Simpan".

Langkah selanjutnya, mari kita modifikasi fungsi `actionCreate`, dengan menambahkan perintah untuk memproses hasil `input`-an pengguna dan menyimpannya ke dalam basis data.

```
public function actionCreate()
{
    $model = new Employee();
    if(Yii::$app->request->post()){
        $model->load(Yii::$app->request->post());
        if($model->save()){
            Yii::$app->session->setFlash('success','Data berhasil disimpan');
        }
        else{
            Yii::$app->session->setFlash('error','Data gagal disimpan');
        }
        return $this->refresh();
    }
    else{
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}
```

Kode di atas kurang lebih sama dengan kode yang dibahas pada bagian formulir MVC. Fungsi dari kode `return $this->refresh();` adalah untuk me-refresh current page.

Mari kita uji coba formulir ini

∞ <http://localhost/basic/web/employee/create>

The screenshot shows the same Yii application after data has been saved. A green box at the top displays the message "Data berhasil disimpan". The main form still shows the input values "Name: Susanti" and "Age: 32". The "Simpan" button is present at the bottom of the form.

E. 2. Read

Buat fungsi actionIndex untuk menampilkan daftar *employee*

```
public function actionIndex()
{
    $employees = Employee::find()->all();
    return $this->render('index', [
        'employees' => $employees,
    ]);
}
```

Buat view daftar *employee* yang akan ditampilkan dalam bentuk tabel.

```
@app/views/employee/index.php

<h1>Daftar Employee</h1>
<?php
echo "<table class='table table-bordered table-striped'>";
echo "<tr>";
echo "<th>ID</th>";
echo "<th>NAME</th>";
echo "<th>AGE</th>";
echo "</tr>";
foreach($employees as $employee) {
    echo "<tr>";
    echo "<td>".$employee->id."</td>";
    echo "<td>".$employee->name."</td>";
    echo "<td>".$employee->age."</td>";
    echo "</tr>";
}
echo "</table>";
```

Mari kita uji coba lagi.

∞ <http://localhost/basic/web/employee/index>

ID	NAME	AGE
1	Hafid	29
2	Junaidi	35
3	Dewi	24
4	Agung	51

E. 3. Update

Sebelum membuat proses edit data, kita perlu membuat tombol pada masing-masing *row* data tabel *employee* yang menunjuk pada data yang akan diedit. Tombol tersebut merupakan tautan ke sebuah formulir edit yang berisi parameter ID *employee* yang akan diedit.

```
@app/views/employee/index.php
```

```
<?php
use yii\helpers\Html;
?>
<h1>Daftar Employee</h1>
<?php
echo "<table class='table table-bordered table-striped'>";
echo "<tr>";
echo "<th>ID</th>";
```

```

echo "<th>NAME</th>";
echo "<th>AGE</th>";
echo "<th>ACTION</th>";
echo "</tr>";
foreach($employees as $employee) {
    echo "<tr>";
    echo "<td>".$employee->id."</td>";
    echo "<td>".$employee->name."</td>";
    echo "<td>".$employee->age."</td>";
    echo "<td>";
    echo Html::a('<i class="glyphicon glyphicon-pencil"></i>', ['employee/update', 'id'=>$employee->id]);
    echo "</td>";
    echo "</tr>";
}
echo "</table>";

```

“glyphicon glyphicon-pencil” adalah Class CSS untuk menampilkan *icon pencil* sehingga tombol edit tampak lebih cantik ☺, Waktunya uji coba

ID	NAME	AGE	ACTION
1	Hafid	29	
2	Junaidi	35	
3	Dewi	24	
4	Agung	51	

Kemudian buatlah fungsi actionUpdate. Fungsi ini hampir sama dengan fungsi actionCreate, bedanya adalah pada objek modelnya, ada parameter ID yang dikirimkan. Parameter ini menunjukkan ID dari *employee* yang akan diedit, disamping itu modelnya menggunakan fungsi findOne() untuk mendapatkan *employee* berdasarkan ID yang diminta.

```

public function actionUpdate($id)
{
    $model = Employee::findOne($id);

    if(Yii::$app->request->post()){
        $model->load(Yii::$app->request->post());
        if($model->save()){
            Yii::$app->session->setFlash('success', 'Data berhasil disimpan');
        }
        else{
            Yii::$app->session->setFlash('error', 'Data gagal disimpan');
        }
        return $this->refresh();
    }
    else{
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

```

Kemudian buat *view update*, yang isinya sama dengan *view create*

```
@app/views/employee/update.php
```

```
<?php
use yii\helpers\Html;
use yii\bootstrap\ActiveForm;
```

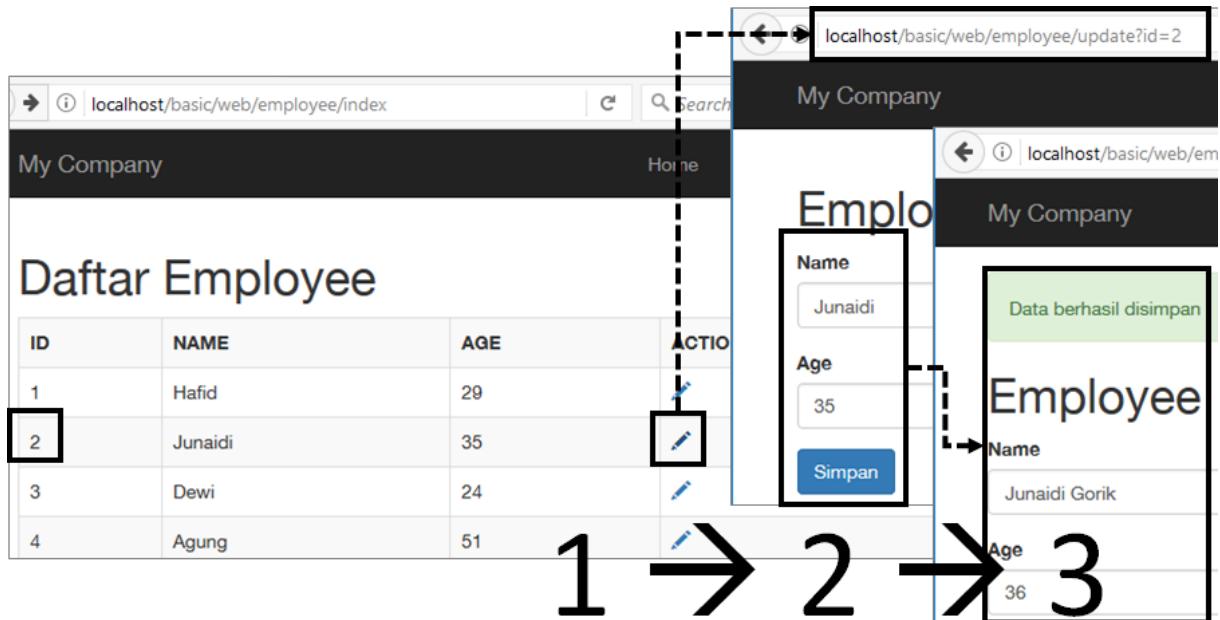
```

?>
<h1>Employee</h1>
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'name') ?>
<?= $form->field($model, 'age') ?>
<?= Html::submitButton('Simpan', ['class' => 'btn btn-primary']) ?>
<?php ActiveForm::end(); ?>

```

Waktunya uji coba, mula-mula buka URL berikut.

∞ <http://localhost/basic/web/employee/index>



E. 4. Delete

Tambahkan tombol *delete* pada tabel daftar *employee* atau *view index*

```

echo Html::a('<i class="glyphicon glyphicon-trash"></i>', ['employee/delete', 'id'=>$employee->id]);

```

ID	NAME	AGE	ACTION
1	Hafid	29	
2	Junaidi Gorik	36	
3	Dewi	24	
4	Agung	51	

"/>

The screenshot shows a list of employees with a "DELETE" icon in the "ACTION" column of each row. The table data is as follows:

ID	NAME	AGE	ACTION
1	Hafid	29	
2	Junaidi Gorik	36	
3	Dewi	24	
4	Agung	51	

Kita bisa juga menambahkan konfirmasi untuk hapus data berupa *popup Javascript*, untuk memastikan bahwa pengguna memang ingin menghapus data tersebut

```

echo Html::a('<i class="glyphicon glyphicon-trash"></i>', ['employee/delete', 'id'=>$employee->id],
            ['onclick'=>'return (confirm("Apakah data mau dihapus?"))?true:false;']);

```

Sebelum menguji cobanya, buat fungsi *actionDelete* yang berisi perintah hapus data.

```

public function actionDelete($id)
{

```

```

    $model = Employee::findOne($id);
    $model->delete();
    return $this->redirect(['index']);
}

```

Kode `$this->redirect(['index']);` berfungsi untuk mengalihkan ke fungsi `actionIndex`.

Mari kita uji coba lagi dengan mengakses URL berikut.

∞ <http://localhost/basic/web/employee/index>

ID	NAME	ACTION
1	Hafid	
2	Junaidi Gorik	

Ketika tombol hapus diklik maka akan muncul konfirmasi hapus. Jika kita memilih `Cancel` maka `popup` akan tertutup dan tidak ada aksi apapun. Namun jika kita memilih `OK` maka `popup` akan tertutup dan Yii akan menjalankan fungsi `actionDelete`.

Selanjutnya, kita perlu membuat tombol Create data `employee`, yang mana bisa kita letakkan di atas tabel daftar `employee` pada file `index.php`

```
echo Html::_('a('Create', ['create'], ['class'=>'btn btn-primary']));
```

ID	NAME	AGE	ACTION
1	Hafid	29	
2	Junaidi Gorik	36	

E. 5. Paging

Jika data `employee` kita cukup banyak, maka sebaiknya tampilan daftar `employee` dibatasi dalam setiap halamannya, misal per halaman lima baris data. Nah, implementasi `paging` di Yii cukup mudah, yaitu dengan memanfaatkan dua `class` yaitu `[[yii\data\Pagination]]` dan `[[yii\widgets\LinkPager]]`.

Buat fungsi `actionPaging` di `EmployeeController`, atau kita juga bisa memodifikasi fungsi `actionIndex`.

```

function actionPaging()
{
    $query = Employee::find();
    $count = $query->count();

    $pagination = new \yii\data\Pagination([
        'totalCount' => $count,
        'defaultPageSize' => 5,
    ]);
}

```

```

$employees = $query->offset($pagination->offset)
->limit($pagination->limit)
->all();

return $this->render('paging', [
    'employees' => $employees,
    'pagination' => $pagination,
]);
}

```

Properti `defaultPageSize` untuk mengatur jumlah *record employee* yang ditampilkan per halamannya (*default*-nya: 20). Langkah selanjutnya mari kita buat *view paging*.

`@app/views/employee/paging.php`

```

<?php
use yii\helpers\Html;
?>
<h1>Daftar Employee</h1>
<?php
echo Html::a('Create', ['create'], ['class'=>'btn btn-primary']);
echo "<hr>";
echo "<table class='table table-bordered table-striped'>";
echo "<tr>";
echo "<th>ID</th>";
echo "<th>NAME</th>";
echo "<th>AGE</th>";
echo "<th>ACTION</th>";
echo "</tr>";
foreach($employees as $employee) {
    echo "<tr>";
    echo "<td>".$employee->id."</td>";
    echo "<td>".$employee->name."</td>";
    echo "<td>".$employee->age."</td>";
    echo "<td>";
    echo Html::a('<i class="glyphicon glyphicon-pencil"></i>', ['employee/update', 'id'=>$employee->id]);
    echo Html::a('<i class="glyphicon glyphicon-trash"></i>', ['employee/delete', 'id'=>$employee->id],
        ['onclick'=>'return (confirm("Apakah data mau dihapus?") ?true:false);']);
    echo "</td>";
    echo "</tr>";
}
echo "</table>";
// display pagination
echo \yii\widgets\LinkPager::widget([
    'pagination' => $pagination,
]);

```

Poin utama adalah di bagian akhir kode di atas, yaitu.

```

// display pagination
echo \yii\widgets\LinkPager::widget([
    'pagination' => $pagination,
]);

```

Hanya dengan modifikasi ini, kita telah selesai menerapkan *paging* di Yii. Mari kita uji coba *paging* dengan mengakses URL berikut.

∞ <http://localhost/basic/web/employee/paging>

The screenshot shows a Yii application interface. At the top, there's a header bar with a back button, a search bar containing 'Search', and links for Home, About, Contact, and Login. Below the header is a navigation bar with 'My Company' on the left and 'Home', 'About', 'Contact', 'Login' on the right. The main content area has a title 'Daftar Employee' and a 'Create' button. Below this is a table with columns: ID, NAME, AGE, and ACTION. The table contains five rows of employee data. At the bottom of the table are navigation buttons for page 1 of 2, and on the right side are a magnifying glass icon for search and a dropdown menu.

ID	NAME	AGE	ACTION
1	Hafid	29	
2	Junaidi Gorik	36	
3	Dewi	24	
4	Agung	51	
5	Susanti	32	

Jangan kaget jika mudah sekali ☺, masihkan anda mau berjibaku dengan coding *paging* secara manual?. Jika *paging* lebih kecil sama dengan lima maka tampilan *paging* otomatis akan disembunyikan.

E. 6. Sorting

Sorting atau mengurutkan data berdasarkan kolom tertentu juga bisa dilakukan dengan mudah di Yii yaitu dengan memanfaatkan class [[yii\data\Sort]]. Untuk mencobanya, mari kita buat fungsi actionSorting sebagai berikut.

```
function actionSorting()
{
    $query = Employee::find();
    $employees = $query->orderBy([
        'name' => SORT_DESC,
        'age' => SORT_DESC,
    ])
    ->all();

    return $this->render('sorting', [
        'employees' => $employees,
    ]);
}
```

Kemudian untuk *view*-nya hampir sama dengan *view* *paging.php*, hanya saja nama *file*-nya *sorting.php*. Silahkan di coba.

E. 7. Filtering

Filtering merupakan proses memilah data berdasarkan parameter tertentu, atau dengan kata lain pencarian. Pada contoh ini kita akan membuat fitur pencarian nama employee. Pertama yang harus dilakukan adalah membuat form pencarian.

```
public function actionFiltering($name="")
{
    $employees = Employee::find();
    if(!empty($name)) $employees = $employees->where(['name'=>$name]);
    $employees = $employees->all();
    return $this->render('filtering', [
        'employees' => $employees,
    ]);
}
```

Duplicate file *index.php* kemudian ubah namanya menjadi *filtering.php*. Lalu tambahkan pada bagian atasnya, formulir pencarian.

Perhatian:

Jika perlu, *review* ulang materi tentang CRUD+ ini, gunakan studi kasus tabel lain.

F. Migrasi Basis Data

F. 1. Definisi

Migrasi basis data atau *database migration* adalah fitur di Yii yang memungkinkan kita memigrasi basis data dengan mudah. Kita perlu menuliskan semua skema dari basis data yang digunakan pada aplikasi, sehingga jika dikemudian hari kita akan melakukan instalasi aplikasi kita, maka cukup dengan perintah:

```
| yii migrate
```

Maka skema tersebut akan terinstalasi pada DBMS, tentu sebelumnya kita harus membuat basis data dan setting konfigurasi untuk koneksi basis data dulu.

Lantas apa kelebihannya jika dibandingkan cara biasa yaitu menggunakan fitur impor basis data?

Kalau bicara tentang kemudahan, jelas lebih mudah menggunakan cara biasa, karena kita tidak perlu repot-repot menuliskan kode migrasinya. Namun di sini kita berbicara tentang pengembangan sebuah perangkat lunak, dimana basis data merupakan bagian dari itu. Sehingga dengan dibuatkan migrasinya maka perubahan skema akan lebih mudah dilakukan serta terekam di *source code control*.

F. 2. Membuat Migrasi Basis Data

Untuk membuat kode migrasi, kita bisa menggunakan CMD dengan perintah sebagai berikut.

```
| yii migrate/create <name>
```

Misalnya kita ingin membuat migrasi untuk tabel *book* berikut.

```
CREATE TABLE IF NOT EXISTS `book` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(150) NOT NULL,
  `author` varchar(50) NOT NULL,
  `publisher` varchar(50) NOT NULL,
  `price` int(11) NOT NULL,
  `stock` int(5) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Maka pada CMD kita jalankan perintah berikut.

```
| yii migrate/create create_book_table
```

```
c:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\basic>yii migrate/create create_book_table
Yii Migration Tool (based on Yii v2.0.11.2)

Create new migration 'C:\xampp\htdocs\basic\migrations\m170214_060009_create_book_table.php'? (yes|no) [no]:yes
New migration created successfully.

c:\xampp\htdocs\basic>
```

Maka akan tercipta file migrasi di @app/migrations/ dengan nama yang unik yaitu menggunakan format ini m<YYMMDD_HHMMSS>_<Name>

```
<?php
use yii\db\Migration;
/**
 * Handles the creation of table `book`.
 */
class m170214_060009_create_book_table extends Migration
{
```

```

    /**
     * @inheritdoc
     */
    public function up()
    {
        $this->createTable('book', [
            'id' => $this->primaryKey(),
        ]);
    }

    /**
     * @inheritdoc
     */
    public function down()
    {
        $this->dropTable('book');
    }
}

```

Yii tidak benar-benar membuatkan kita kode migrasinya, itu hanya *template*-nya saja. Class PHP di atas *extends* dari class [[yii\db\Migration]]. Adapun fungsi `up` digunakan untuk mengimplementasikan kode untuk membuat perubahan pada basis data misalnya `create table`, sedangkan fungsi `down` digunakan untuk mengembalikan kepada keadaan semua atau kebalikan dari fungsi `up`, misalnya `drop table`.

Kode migrasi untuk tabel `book` akan tampak sebagai berikut.

```

<?php

use yii\db\Migration;

/**
 * Handles the creation of table `book`.
 */
class m170214_060009_create_book_table extends Migration
{
    /**
     * @inheritdoc
     */
    public function up()
    {
        $tableOptions = null;
        if ($this->db->driverName === 'mysql') {
            // http://stackoverflow.com/questions/766809/whats-the-difference-
            // between-utf8-general-ci-and-utf8-unicode-ci
            $tableOptions = 'CHARACTER SET utf8 COLLATE utf8_unicode_ci
                ENGINE=InnoDB';
        }

        $this->createTable('book', [
            'id' => $this->primaryKey(),
            'title' => $this->string(150)->notNull(),
            'author' => $this->string(50)->notNull(),
            'publisher' => $this->string(50)->notNull(),
            'price' => $this->decimal(10,2)->notNull(),
            'stock' => $this->integer(5)->notNull(),
        ],$tableOptions);
    }

    /**
     * @inheritdoc
     */
    public function down()
    {
        $this->dropTable('book');
    }
}

```

Pahami dan bandingkan dengan kode SQL.

Pada create tabel, ketika kita gunakan kode

```
| $this->primaryKey()
```

Maka hal itu sama saja dengan kita meng-*create field* ID dengan definisi

```
| int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY
```

Bagaimana jika misalnya *field* id adalah *string* namun *primary key*?, maka kita bisa gunakan kode seperti berikut

```
$this->createTable('book', [
    'id' => $this->string(15)->notNull(),
    ...
    'PRIMARY KEY(id)',
], $tableOptions);
```

Kita juga bisa menambahkan beberapa data untuk mengisi tabel *book*.

```
$this->batchInsert('book',
    ['title', 'author', 'publisher', 'price', 'stock'],
    [
        ['Pemrograman PHP', 'Anton', 'Elexmedia', 65000, 5],
        ['Belajar AngularJs', 'Budi', 'Digipub', 40000, 3],
        ['Membuat Website Menggunakan Yii', 'Dira', 'Andi Offset', 50000, 4],
    ]
);
```

Di samping itu, kita bisa melakukan perintah SQL DDL (data *definition language*, seperti *create table*, *alter table*, dll) maupun DML (data *manipulation language*, seperti *insert*, *update*, *delete*) lainnya.

Ulangi langkah-langkah di atas hingga tercipta semua tabel, tentu saja semua dilakukan dengan manual.

Sebenarnya ada *tools* yang bisa kita gunakan untuk meng-*generate* kode migrasi, namun bukan resmi dari Yii, silakan googling ya ☺

F. 3. Menjalankan Migrasi Basis Data

Setelah semua tabel dibuat migrasinya dalam *folder* @app/migrations, maka langkah selanjutnya adalah menjalankan *migrate* fungsi *up* melalui CMD.

```
| yii migrate/up
```

Bisa juga cukup menggunakan perintah *yii migrate*

```
c:\xampp\htdocs\basic>yii migrate/up
Yii Migration Tool (based on Yii v2.0.11.2)

Creating migration history table "migration"...Done.
Total 1 new migration to be applied:
    m170214_060009_create_book_table

Apply the above migration? (yes|no) [no]:yes
*** applying m170214_060009_create_book_table
    > create table book ... done (time: 0.411s)
    > insert into book ... done (time: 0.077s)
*** applied m170214_060009_create_book_table (time: 0.801s)

1 migration was applied.

Migrated up successfully.

c:\xampp\htdocs\basic>
```

Informasi yang muncul di CMD menyatakan bahwa migrasi berhasil yaitu *create* tabel dan *insert*. Mari kita cek benarkan tabel *book* telah tercipta di basis data melalui phpmyadmin.

The screenshot shows the phpMyAdmin interface for the 'yii2basic' database. In the left sidebar, under the 'information_schema' section, there is a 'New' entry which contains the 'book' table. The main panel displays the 'book' table structure with columns: #, Nama, Jenis, Penyortiran, Atribut, Kosong, Bawaan, Ekstra, and Tindakan. The table has 6 rows of data. Below the table, a list of migrations is shown: 'book', 'employee', and 'migration'. The 'book' migration is currently selected.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Ekstra	Tindakan
1	id	int(11)			Tidak	Tidak ada	AUTO_INCREMENT	Ut
2	title	varchar(150)			Tidak	Tidak ada		Ut
3	author	varchar(50)			Tidak	Tidak ada		Ut
4	publisher	varchar(50)			Tidak	Tidak ada		Ut
5	price	decimal(10,2)			Tidak	Tidak ada		Ut
6	stock	int(5)			Tidak	Tidak ada		Ut

	id	title	author	publisher	price	stock
Ubah	1	Pemrograman PHP	Anton	Elexmedia	65000.00	5
Ubah	2	Belajar AngularJs	Budi	Digipub	40000.00	3
Ubah	3	Membuat Website Menggunakan Yii	Dira	Andi Offset	50000.00	4

Untuk mengembalikan migrasi atau membatalkannya maka gunakan perintah

```
| yii migrate/down
```

```
c:\> C:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\basic>yii migrate/down
Yii Migration Tool (based on Yii v2.0.11.2)

Total 1 migration to be reverted:
    m170214_060009_create_book_table

Revert the above migration? (yes|no) [no]:yes
*** reverting m170214_060009_create_book_table
    > drop table book ... done (time: 0.293s)
*** reverted m170214_060009_create_book_table (time: 0.346s)

1 migration was reverted.

Migrated down successfully.

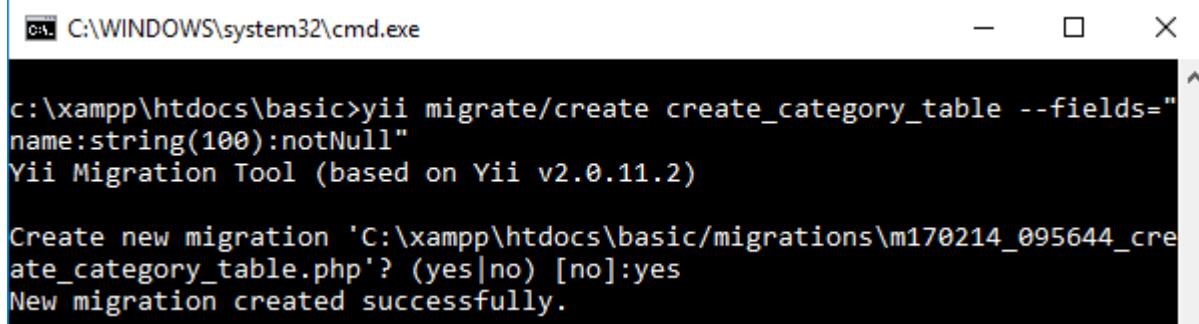
c:\xampp\htdocs\basic>
```

Secara default, ketika kita menjalankan perintah `yii migrate` maka semua file migrasi di dalam direktori `@app/migrations` akan dieksekusi.

F. 4. Membuat Migrasi Tingkat Lanjutan

Kita bisa menggunakan parameter `fields` untuk meng-generate tabel `migration`:

```
| yii migrate/create create_category_table --fields="name:string(100):notNull"
```



```
C:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\basic>yii migrate/create create_category_table --fields="name:string(100):notNull"
Yii Migration Tool (based on Yii v2.0.11.2)

Create new migration 'C:\xampp\htdocs\basic\migrations\m170214_095644_create_category_table.php'? (yes|no) [no]:yes
New migration created successfully.
```

Berikut ini hasilnya

```
<?php
...
class m170214_095644_create_category_table extends Migration
{
    /**
     * @inheritdoc
     */
    public function up()
    {
        $this->createTable('category', [
            'id' => $this->primaryKey(),
            'name' => $this->string(100)->notNull(),
        ]);
    }
    ...
}
```

Catatan:

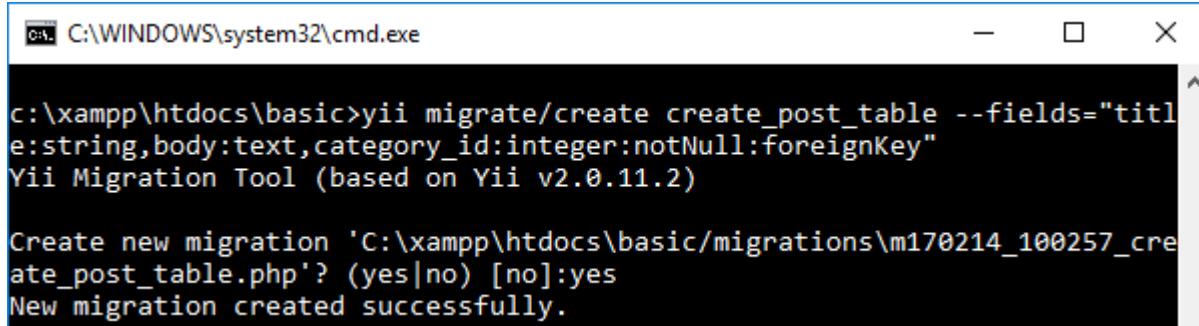
primary key akan otomatis ditambahkan dan secara *default* akan nama *field*-nya adalah id. Namun jika kita ingin menggunakan nama lain maka kita harus tentukan secara spesifik misalnya `--fields="username:primaryKey"`.

Relasi antar tabel juga bisa kita definisikan, misal untuk contoh, kita akan membuat relasi antara tabel *post* dengan tabel *category*, melalui fungsi `foreignkey`

```
| yii migrate/create create_post_table --
fields="title:string,body:text,category_id:integer:notNull:foreignkey(category)"
```

Parameter nama tabel pada fungsi `foreignkey` ini sifatnya opsional, artinya jika tidak didefinisikan maka Yii akan otomatis menggunakan nama *field*-nya tanpa `_id` sebagai nama tabelnya. Pada contoh di atas kodennya `"category_id:integer:notNull:foreignkey(category)"`, bisa diganti atau disingkat dengan `"category_id:integer:notNull:foreignKey"`. Sehingga format *field* `category_id` maka akan menjadi menjadi nama tabel *category*, *field* `post_id` menjadi *post*, dst

```
| yii migrate/create create_post_table --
fields="title:string,body:text,category_id:integer:notNull:foreignkey"
```



```
C:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\basic>yii migrate/create create_post_table --fields="title:string,body:text,category_id:integer:notNull:foreignkey"
Yii Migration Tool (based on Yii v2.0.11.2)

Create new migration 'C:\xampp\htdocs\basic\migrations\m170214_100257_create_post_table.php'? (yes|no) [no]:yes
New migration created successfully.
```

Hasilnya

```
<?php
use yii\db\Migration;
```

```
class m170214_100257_create_post_table extends Migration
{
    public function up()
    {
        $this->createTable('post', [
            'id' => $this->primaryKey(),
            'title' => $this->string(),
            'body' => $this->text(),
            'category_id' => $this->integer()->notNull(),
        ]);

        // creates index for column `category_id`
        $this->createIndex(
            'idx-post-category_id',
            'post',
            'category_id'
        );
    }

    // add foreign key for table `category`
    $this->addForeignKey(
        'fk-post-category_id',
        'post',
        'category_id',
        'category',
        'id',
        'CASCADE'
    );
}

public function down()
{
    // drops foreign key for table `category`
    $this->dropForeignKey(
        'fk-post-category_id',
        'post'
    );

    // drops index for column `category_id`
    $this->dropIndex(
        'idx-post-category_id',
        'post'
    );

    $this->dropTable('post');
}
}
```

Selanjutnya mari kita jalankan *migrate up* ke tiga tabel yang telah kita buat kode *migration*-nya

```
c:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\basic>yii migrate/up
Yii Migration Tool (based on Yii v2.0.11.2)

Total 3 new migrations to be applied:
    m170214_060009_create_book_table
    m170214_095644_create_category_table
    m170214_100257_create_post_table

Apply the above migrations? (yes|no) [no]:yes
*** applying m170214_060009_create_book_table
    > create table book ... done (time: 0.273s)
    > insert into book ... done (time: 0.046s)
*** applied m170214_060009_create_book_table (time: 0.454s)

*** applying m170214_095644_create_category_table
    > create table category ... done (time: 0.421s)
*** applied m170214_095644_create_category_table (time: 0.468s)

*** applying m170214_100257_create_post_table
    > create table post ... done (time: 0.233s)
    > create index idx-post-category_id on post (category_id) ... done (time: 0.231s)
    > add foreign key fk-post-category_id: post (category_id) references category (id) ... done (time: 0.724s)
*** applied m170214_100257_create_post_table (time: 1.255s)

3 migrations were applied.

Migrated up successfully.
```

Mari kita cek di database menggunakan tools Designer pada phpmyadmin

Jika file migration kita tidak terdapat pada direktori @app/migrations, melainkan pada direktori lain, maka kita bisa mengaturnya dengan menambahkan parameter `-migrationPath` yang diisi dengan lokasi folder migrasi kita.

```
| yii migrate --migrationPath=@app/lokasi_folder_migrasi_kamu
```

Selengkapnya silakan pelajari dari panduan resmi Yii

<http://www.yiiframework.com/doc-2.0/guide-db-migrations.html>

Catatan:

Penulisan kode migrasi ini dimaksudkan tidak hanya untuk memudahkan proses migrasi saja, namun juga agar setiap perubahan struktur tabel pada aplikasi juga akan ikut terekam historinya, terutama jika kita menggunakan *source code control*.

G. Kesimpulan

Class ActiveRecord adalah *class* yang memudahkan kita dalam menulis kode terkait interaksi dengan basis data. Di samping itu, *Class* ini juga telah dilengkapi dengan validasi data sehingga coding CRUD menjadi lebih cepat. Yii juga memiliki fitur migrasi basis data yang berfungsi melakukan instalasi tabel pada basis data melalui CMD.

Jika pada bab ini, kita belajar tentang bagaimana membuat aplikasi CRUD secara manual. Maka pada bab selanjutnya kita akan belajar tentang salah satu fitur unggulan *framework* Yii, yaitu kode *generator Gii*. Salah satu fitur dari Gii adalah meng-*generate* aplikasi CRUD tanpa koding.

Halaman ini sengaja dikosongkan!

6

Gii: *Code Generator*

Pada bab ini kita akan belajar tentang:

- Konfigurasi Gii
- Cara menggunakan Gii
- Modifikasi kode hasil generate Gii

Yii mempunyai *tools* yang berfungsi untuk menghasilkan kode program, bernama Gii. *Tools* ini sangat membantu para pemrogram perangkat lunak untuk mempercepat pekerjaannya terutama untuk hal-hal yang umum dan sering dilakukan, salah satunya adalah aplikasi CRUD. CRUD mempunyai pola kode yang sama sehingga bisa dibuat *tools* untuk meng-generate kodennya.

A. Konfigurasi

Gii merupakan *extension* dalam bentuk *module* yang secara *default* telah terinstalasi dan siap untuk digunakan ketika kita melakukan instalasi Yii sehingga tidak perlu melakukan setting apapun. Namun ada baiknya kita tahu dimana letak konfigurasinya.

Mari kita buka file @app/config/web.php

```
if (YII_ENV_DEV) {
    // configuration adjustments for 'dev' environment
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = [
        'class' => 'yii\debug\Module',
        // uncomment the following to add your IP if you are not connecting from
        localhost.
        //'allowedIPs' => ['127.0.0.1', '::1'],
    ];

    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = [
        'class' => 'yii\gii\Module',
        // uncomment the following to add your IP if you are not connecting from
        localhost.
        //'allowedIPs' => ['127.0.0.1', '::1'],
    ];
}
```

Kita dapati bahwa *namespace* Gii adalah [[yii\gii\Module]], yang hanya akan jalan pada mode “YII_ENV_DEV” atau *development*, artinya *tools generator* ini hanya boleh dipakai ketika masih dalam tahap pengembangan aplikasi, tentunya hal ini berkaitan dengan keamanan.

Kita bisa mengubah mode dari *development* menjadi *production*, melalui *file entry script* yaitu

```
@app/web/index.php
```

```
<?php
// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');
```

Perintahnya, cukup dengan meng-*comment* dua baris kode tersebut maka Yii berubah menjadi *mode production* dan tentunya *tools* Gii tidak akan bisa dipakai.

Secara *default*, Gii hanya bisa diakses melalui *localhost*, namun untuk tujuan tertentu kita bisa membuatnya agar bisa diakses menggunakan IP tertentu melalui properti *allowedIPs*. Misalnya:

```
$config['modules']['gii'] = [
    'class' => 'yii\gii\Module',
```

```
|     'allowedIPs' => ['127.0.0.1', '::1', '192.168.0.1']
|];
```

B. Cara Penggunaan

Pada dasarnya *tools Gii* bisa akses melalui *web browser* melalui URL berikut:

∞ <http://localhost/basic/web/gii>

Maka kita akan menjumpai tampilan sebagai berikut:

The screenshot shows the 'Welcome to Gii' page of the Yii Code Generator. At the top, there's a navigation bar with links for Home, Help, and Application. Below the header, the main content area is titled 'Welcome to Gii' with the subtitle 'a magical tool that can write code for you'. It lists six generators:

- Model Generator**: Described as generating an ActiveRecord class for a database table. Includes a 'Start »' button.
- CRUD Generator**: Described as generating a controller and views for CRUD operations. Includes a 'Start »' button.
- Controller Generator**: Described as helping to quickly generate a new controller class with actions and views. Includes a 'Start »' button.
- Form Generator**: Described as generating a view script file for a form. Includes a 'Start »' button.
- Module Generator**: Described as helping to generate skeleton code for a Yii module. Includes a 'Start »' button.
- Extension Generator**: Described as helping to generate files for a Yii extension. Includes a 'Start »' button.

Ada 6 *generator* kode yaitu:

- *Model Generator*

Untuk meng-*generate class Active Record* (model) untuk tabel tertentu pada basis data.

- *CRUD Generator*

Untuk meng-*generate* operasi CRUD berdasarkan *class model* tertentu

- *Controller Generator*

Untuk meng-*generate class Controller*

- *Form Generator*

Untuk meng-*generate* tampilan formulir berdasarkan *class model* tertentu

- *Module Generator*

Untuk meng-*generate template* dari *module* di Yii, jika kita ingin membuat sub aplikasi.

- *Extension Generator*

Untuk meng-*generate template* dari *extension* di Yii, jika kita ingin membuat *extension* sendiri.

Pada bagian ini, kita hanya akan membahas dua *generator* penting yaitu *Model Generator* dan *CRUD Generator*.

B. 1. Meng-*generate Model Active Record*

Dengan menggunakan *generator* ini, kita bisa meng-*generate class* model *Active Record* berdasarkan tabel tertentu pada basis data.

Caranya, klik pada tombol *Start* di bagian *Model Generator*

Welcome to Gii a magical tool that can write code for you

Start the fun with the following code generators:

- Model Generator**
This generator generates an ActiveRecord class for the specified database table.
[Start »](#)
- CRUD Generator**
This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.
[Start »](#)
- Controller Generator**
This generator helps you to quickly generate a new controller class with one or several controller actions and their corresponding views.
[Start »](#)

Isi formulir dengan memasukkan nama tabel pada kolom *table name*. Nama-nama tabel dalam basis data akan secara otomatis muncul dalam dropdown kolom *table name*.

Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name

book

book
Book

Pada kolom Model *Class* isikan nama modelnya, *please care* terhadap bentuk huruf!

Perhatian!

book beda dengan Book beda dengan BOOK, sering sekali membuat frustasi pemula ☹

Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name

book

Model Class

Book

Namespace

app\models

Base Class

yii\db\ActiveRecord

Database Connection ID

db

Untuk sementara biarkan apa adanya.

Penjelasan:

- *Namespace* = app\models,

Lihat kembali penjelasan tentang *namespace* pada bab “Model View Controller”.

- *Base Class* = [[yii\db\ActiveRecord]],

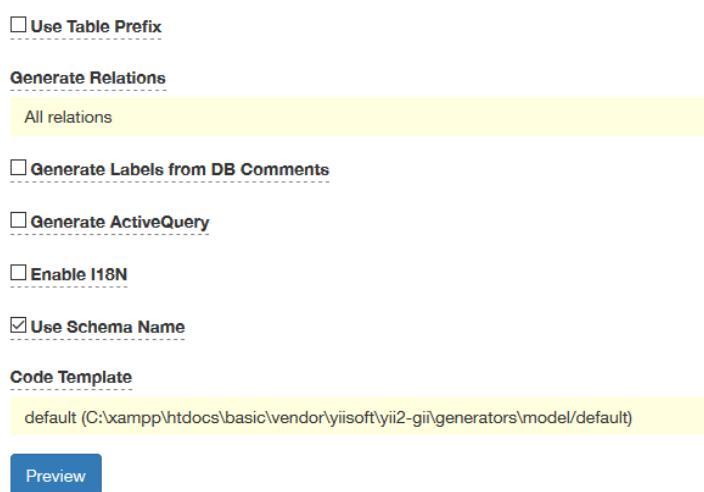
Adalah *class* yang akan di-extends oleh model ini (lihat kembali dasar-dasar Active Record pada bab “Bekerja dengan Basis Data”)

- *Database Connection ID* = db,

Lihat dikonfigurasi @app/config/web.php

```
...
],
'db' => require(__DIR__ . '/db.php'),
'urlManager' => [
...
]
```

Selanjutnya



- *Use Table Prefix*

Jika kita menggunakan tabel *prefix* misal tbl_ , tb_ dsb

- *Generate Relations*

Jika kita *check* maka akan meng-*generate* relasi tabel, namun pastikan tabel kita memang berelasi di basis data.

- *Generate Labels from DB Comments*

Jika kita menuliskan komentar pada setiap label di tabel maka bisa digenerate menjadi label.

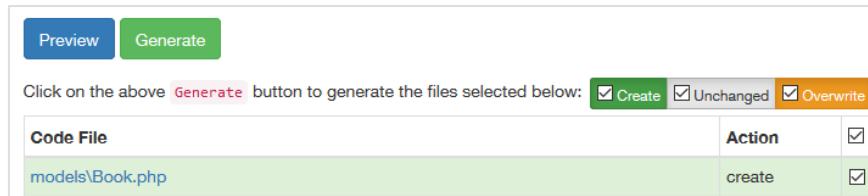
- *Generate ActiveQuery*

Berfungsi meng-*generate* template class ActiveQuery yang merupakan tambahan dari *class* model kita, yaitu bisa berisi fungsi-fungsi *query* tertentu seperti, perhitungan jumlah *record*, rata-rata dsb. Dipisahkan dalam *class* tersendiri supaya lebih mudah dalam pengelolaannya.

- *Enable I18n*

Akan meng-*generate* semua teks dalam format I18N yang mendukung *internationalization*. Yaitu dengan format `Yii::t("teks");`

Jika semua sudah, maka klik tombol *Preview*



Yii akan meng-*generate* file @app/models/Book.php, kita bisa melihat kode yang akan di-*generate* seperti apa, dengan mengklik pada kolom *Code File*.

The screenshot shows the Yii Code Generator interface. In the center, there is a code editor window titled "models\Book.php" containing the following PHP code:

```

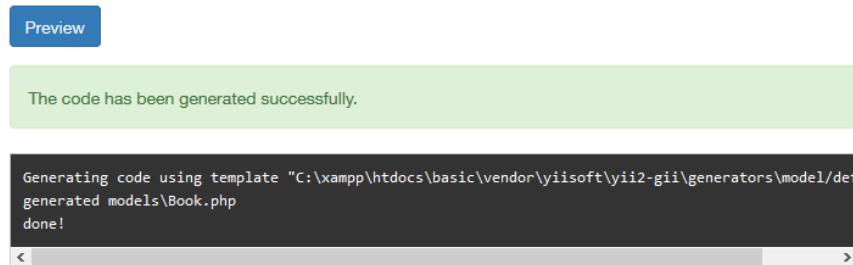
class Book extends \yii\db\ActiveRecord
{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'book';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['title', 'author', 'publisher', 'price', 'stock'], 'required'],
            [['price'], 'number'],
            [['stock'], 'integer'],
            [['title'], 'string', 'max' => 150],
            [['author', 'publisher'], 'string', 'max' => 50],
        ];
    }
}

```

On the right side of the interface, there is a sidebar with several checkboxes. One of them, "Overwrite", is checked.

Jika sudah klik tombol *Generate*



Selesai untuk satu model.

Kalau kita lihat hasil *generate file* model Book, khususnya pada fungsi *rules*, maka kita dapat bahwa tipe data yang kita set pada tabel dibaca sebagai fungsi *rules*.

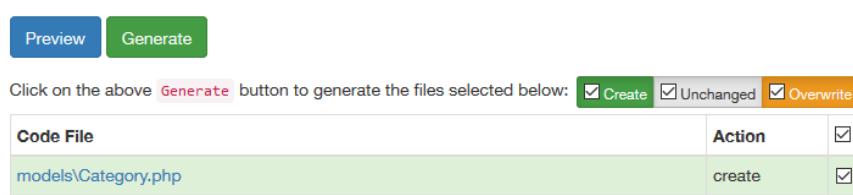
```

public function rules()
{
    return [
        [['title', 'author', 'publisher', 'price', 'stock'], 'required'],
        [['price'], 'number'],
        [['stock'], 'integer'],
        [['title'], 'string', 'max' => 150],
        [['author', 'publisher'], 'string', 'max' => 50],
    ];
}

```

Not null diterjemahkan sebagai *field* yang *required* atau harus diisi.

Lakukan untuk meng-*generate* tabel lainnya.



Jika tabel tersebut memiliki relasi dengan tabel lain, maka otomatis relasi antar tabelnya pun juga dibaca dan diimplementasikan pada model dalam bentuk fungsi

```

/*
 * @return \yii\db\ActiveQuery

```

```
/*
public function getPosts()
{
    return $this->hasMany(Post::className(), ['category_id' => 'id']);
}
```

[Preview](#) [Generate](#)

Click on the above [Generate](#) button to generate the files selected below:

Create Unchanged Overwrite

Code File	Action	<input checked="" type="checkbox"/>
models\Post.php	create	<input checked="" type="checkbox"/>

Berikut ini relasi yang terbentuk pada model Post.

```
/**
 * @return \yii\db\ActiveQuery
 */
public function getCategory()
{
    return $this->hasOne(Category::className(), ['id' => 'category_id']);
}
```

Tidak perlu lagi kita membuat model secara manual, tinggal *generate* dan jadi.. mudah sekali bukan? Kita juga masih dizinkan memodifikasi model hasil *generate* tersebut sesuai dengan keinginan kita.

Di samping itu, kita juga diizinkan untuk membuat *template* Gii sendiri jika kita mau. Panduan tentang hal ini bisa anda jumpai pada tautan berikut.

∞ <https://github.com/yiisoft/yii2-gii/tree/master/docs/guide/README.MD>

Tidak cukup sampai di sini, ada cara yang lebih mudah lagi dalam meng-*generate* model yaitu kita diizinkan untuk meng-*generate* model dari semua tabel sekaligus. Caranya sebagai berikut:

Masih pada formulir model *generator*, isi *field table name* dengan karakter bintang *

Kemudian klik tombol *Preview*, maka akan tampil daftar file model yang akan di-*generate*.

[Preview](#) [Generate](#)

Click on the above [Generate](#) button to generate the files selected below:

Create Unchanged Overwrite

Code File	Action	<input type="checkbox"/>
models\Book.php	create	<input checked="" type="checkbox"/>
models\Category.php	create	<input checked="" type="checkbox"/>
models\Employee.php	create	<input checked="" type="checkbox"/>
models\Migration.php	create	<input type="checkbox"/>
models\Post.php	create	<input checked="" type="checkbox"/>

Kemudian klik tombol *Generate*.

Preview

The code has been generated successfully.

```

Generating code using template "C:\xampp\htdocs\basic\vendor\yiisoft\yii2-gii\generators\model\def
generated models\Book.php
generated models\Category.php
generated models\Employee.php
skipped models\Migration.php
generated models\Post.php
done!

```

Selesai, maka semua model akan ter-*generate*.

Perhatian:

Untuk sementara tabel *user* atau model *user* jangan di-*generate* dulu, karena ada perlakuan khusus yang akan dibahas pada bab berikutnya. *Skip* atau *uncheck* dulu.

B. 2. Menggenerate Operasi CRUD

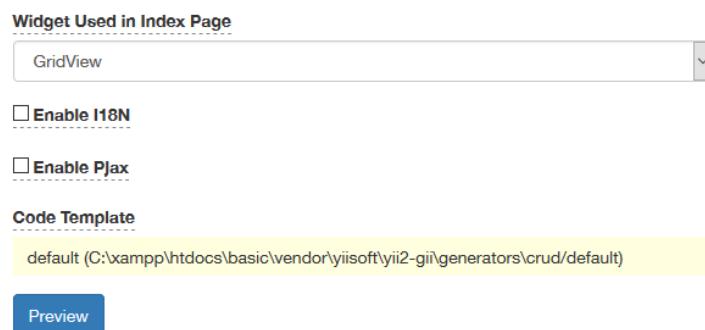
Pada bab sebelumnya, kita telah belajar tentang bagaimana caranya membuat aplikasi CRUD. Nah, ternyata Gii juga punya fitur untuk meng-*generate* kode CRUD. Berikut ini langkah-langkahnya:

1. Pada Gii, pilih menu CRUD Generator, kemudian pada formulir CRUD Generator, isilah *field-fieldnya*

∞ <http://localhost/basic/web/gii/crud>

The screenshot shows the Yii Code Generator interface. On the left, there's a sidebar with several generator options: Model Generator, CRUD Generator (which is currently selected and highlighted in blue), Controller Generator, Form Generator, Module Generator, and Extension Generator. The main content area is titled "CRUD Generator". It contains three input fields: "Model Class" with the value "app\models\Book", "Search Model Class" with the value "app\models\BookSearch", and "Controller Class" with the value "app\controllers\BookController". At the top of the page, the URL "localhost/basic/web/gii/crud" is visible in the browser's address bar, along with standard browser navigation and search tools.

- *Field Model Class* diisi dengan nama *class* yang telah dibuat sebelumnya lengkap dengan *namespace*-nya
- *Field Search Model Class* diisi dengan nama model yang digunakan untuk pencarian, dimana *class* ini akan di-*generate* oleh Gii sebagai *base* model untuk Gridview. Untuk penamaannya bebas namun praktik terbaik adalah sama dengan nama modelnya lalu ditambahkan kata *Search*
- *Field Controller Class*, adalah nama *controller* untuk CRUD ini yang kodenya akan di-*generate* oleh Gii
- *Field View path* adalah lokasi *folder view* akan di-*generate*, jika kosong maka akan di-*generate* ke lokasi @app/views/*ControllerID*, misalnya jika *controller*-nya CategoryController maka akan di-*generate* ke *folder* @app/views/category/. Sebaiknya dikosongkan saja.
- *Widget Used in Index Page*, bentuk tampilan indexnya Gridview atau Listview
- *Enable I18N* untuk internationalisation.
- *Enable Pjax* untuk mengaktifkan Pjax sehingga ketika *sorting* dan *paging* tidak akan *refresh page*, atau *loading page* menggunakan metode ajax. Materi tentang Pjax akan dibahas berikutnya.



2. Klik tombol *Preview*, maka akan muncul daftar *file* yang akan di-*generate*, kita bisa mengklik nama *file* untuk melihat kode seperti apa yang akan di-*generate*

Code File	Action	<input checked="" type="checkbox"/>
controllers\BookController.php	create	<input checked="" type="checkbox"/>
models\BookSearch.php	create	<input checked="" type="checkbox"/>
views\book_form.php	create	<input checked="" type="checkbox"/>
views\book_search.php	create	<input checked="" type="checkbox"/>
views\book\create.php	create	<input checked="" type="checkbox"/>
views\book\index.php	create	<input checked="" type="checkbox"/>
views\book\update.php	create	<input checked="" type="checkbox"/>
views\book\view.php	create	<input checked="" type="checkbox"/>

3. Kemudian klik tombol *Generate*

```
Generating code using template "C:\xampp\htdocs\basic\vendor\yiisoft\yii2-gii\generators\crud\default"
generated controllers\BookController.php
generated models\BookSearch.php
generated views\book\_form.php
generated views\book\_search.php
generated views\book\create.php
generated views\book\index.php
generated views\book\update.php
generated views\book\view.php
done!
```

Selesai untuk satu CRUD, lakukan untuk CRUD lainnya.

Selamat mencoba ☺

B. 3. Uji Coba Hasil Generator

Setelah kita meng-*generate* CRUD menggunakan Gii, maka sekarang saatnya menguji coba hasilnya melalui *web browser*. Pada contoh ini kita akan menguji coba CRUD *category* yang telah kita *generate* sebelumnya.

1. Akses CRUD *category* menggunakan URL berikut.

∞ <http://localhost/basic/web/category/index>

2. Maka otomatis kita akan diarahkan ke *controller category* dengan fungsi *actionIndex*.

No results found.

Terlihat data *Gridview* masih kosong

- Mari kita coba membuat kategori baru melalui tombol *Create Category*, maka akan muncul formulir *create category*

- Tanpa menginput apapun, coba klik tombol *Create*

Akan muncul galat berupa peringatan bahwa *field title* tidak boleh kosong, hal ini disebabkan karena pada model *Category* fungsi *rules* telah didefinisikan bahwa *field title* sebagai *required*. Hal ini juga telah kita bahas pada bab sebelumnya.

- Kemudian isi formulir tersebut, minimal *field title*-nya lalu tekan tombol *Create*, maka kita akan dibawa ke halaman *view*

ID	1
Name	Kesehatan

Pada bagian ini kita bisa mengedit atau menghapus data *category*.

6. Klik tautan *Category* untuk kembali ke *index* atau tabel

#	ID	Name	
1	1	Kesehatan	

7. Pada kolom paling kanan dari tabel *Gridview*, kita jumpai tiga buah *icon* yaitu

- Mata = untuk menampilkan *view*
- Pensil = untuk menampilkan formulir edit data
- Tempat sampah = untuk menghapus

8. Silakan dicoba untuk mengedit dan menghapus data

Selesai.

Lakukan percobaan untuk CRUD yang lain.

C. Modifikasi Kode Hasil *Generate*

Hasil *generate* CRUD masih berupa kode standard, untuk itu kita perlu melakukan beberapa modifikasi sesuai dengan keinginan kita. Supaya sistematis maka penjelasan tentang modifikasi di sini menggunakan pendekatan MVC.

C. 1. *Modifikasi View*

Kita akan mulai dengan memodifikasi kode dari sisi view, atau dalam hal ini (CRUD), berarti *file-file* yang berada pada *folder views*. Penjabaran pada masing-masing view akan dipecah lagi berdasarkan *widget* yang digunakan. Sebagaimana yang telah sedikit disinggung pada bab “*Model View Controller*” bahwa *widget* merupakan blok atau bagian dari tampilan yang dapat digunakan kembali. Pada setiap kode *view* hasil *generate Gii*, Yii menggunakan *widget-widget* standardnya untuk memformat tampilan.

View Index

View index merupakan tampilan utama sebuah CRUD, umumnya ditampilkan dalam bentuk tabular atau daftar, sehingga widget yang umum digunakan adalah Gridview atau Listview.

Misalnya pada *view index category*. Kita bisa menyembunyikan kolom dengan id dengan cara meng-*comment* (// atau /* */) kolom tersebut pada Gridview.

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],
        // 'id',
        'name',
        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
```

Hasilnya

#	Name	
1	Kesehatan	
2	Komputer	
3	Hukum	

View Form

C. 2. *Modifikasi Controller*

C. 3. Modifikasi Model

Misalnya:

- *Field* yang ditampilkan pada *Gridview* hanya *Title*, *Description*, *created_at*, dan *created_by*
- *Field* *created_at*, *updated_at*, *created_by*, *updated_by* tidak ditampilkan pada form. Namun diisi oleh sistem karena nilainya sudah bisa dipastikan.
- *Field* *created_at* ditampilkan dalam format yang mudah dibaca oleh pengguna yaitu “tgl – bulan – tahun jam: menit” serta *field* *created_by* ditampilkan

Untuk melakukannya maka berikut ini langkah-langkahnya:

1. Modifikasi file @app/views/category/index.php

Uncomment field yang ingin ditampilkan

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],
        // 'id',
        'title',
        'description',
        'created_at',
        // 'updated_at',
        'created_by',
        // 'updated_by',
        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
```

Categories

#	Title	Description	Created At	Created By
1	Buku Tulis	Buku buat tulis menulis	(not set)	(not set)

2. Modifikasi file @app/views/category/_form.php

Hapus *field* *created_at*, *updated_at*, *created_by*, *updated_by*

```
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>
<?= $form->field($model, 'description')->textInput(['maxlength' => true]) ?>
<div class="form-group">
    <?= Html::submitButton($model-
>isNewRecord ? 'Create' : 'Update', ['class' => $model->isNewRecord ? 'btn btn-
success' : 'btn btn-primary']) ?>
</div>
<?php ActiveForm::end(); ?>
```

Perhatian:

Kode di atas adalah potongan dari *file _form.php*, tujuannya hanyalah untuk menunjukkan bahwa keempat *field* tersebut telah dihapus.

3. Supaya keempat *field* yang dihapus di atas bisa diisi secara otomatis oleh sistem maka kita perlu memodifikasi model `app\models\Category` yaitu dengan menambahkan fungsi untuk mengisi keempat *field* tersebut sebelum atau setelah proses *create* atau *update* data.

Yii menyebut fungsi yang seperti ini sebagai *Behavior*. Pada kasus ini fungsi *behavior* melekat pada model (`ActiveRecord`). Dan kabar baiknya adalah kita tidak perlu membuat sendiri fungsi untuk melakukan hal itu, hal ini dikarenakan Yii sudah menyediakan bagi kita ☺.

Ada dua *behavior* yang berperan untuk kasus ini yaitu:

- `Class yii\behaviors\TimestampBehavior`, dan
- `Class yii\behaviors\BlameableBehavior`.

A. 1. *Timestamp Behavior*

Behavior ini berfungsi menginput data *timestamp* (waktu saat proses *create/update*) secara otomatis ketika proses *create* atau *update* data.

Untuk mengimplementasikan *Timestamp Behavior*, kita bisa tambahkan *class behavior*-nya pada fungsi `behaviors()` di model. Kodenya seperti berikut ini.

```
use yii\behaviors\TimestampBehavior;

public function behaviors() {
    return [
        TimestampBehavior::className(),
    ];
}
```

Secara *default*, *Timestamp Behavior* akan mengisi secara otomatis *field* atau atribut dengan nama `created_at` dan `updated_at` dengan waktu saat itu (menggunakan fungsi `time()`). Lalu bagaimana jika nama *field* yang kita gunakan berbeda? Dan kita tidak menggunakan tipe data `timestamp` melainkan misalnya menggunakan tipe data `datetime`. Maka kita bisa mendefinisikan properti `$createdAtAttribute`, `$updatedAtAttribute`, dan `$value` sebagai berikut.

```
use yii\db\Expression;

public function behaviors() {
    return [
        [
            'class' => TimestampBehavior::className(),
            'createdAtAttribute' => 'create_time',
            'updatedAtAttribute' => 'update_time',
            'value' => new Expression('NOW()'),
        ],
    ];
}
```

A. 2. *Blameable Behavior*

Behavior ini berfungsi menginput data *current pengguna* secara otomatis ketika proses *create* atau *update* data.

Untuk mengimplementasikan *Blameable Behavior*, kita bisa tambahkan *Class behavior*-nya pada fungsi `behaviors()` di model. Kodenya seperti berikut ini.

```
use yii\behaviors\BlameableBehavior;

public function behaviors() {
    return [
        BlameableBehavior::className(),
    ];
}
```

Secara *default*, *Blameable Behavior* akan mengisi secara otomatis *field* atau atribut dengan nama `created_by` dan `updated_by` dengan *current pengguna* ID (ID pengguna yang sedang *login*). Lalu bagaimana jika nama *field* yang kita gunakan berbeda? Maka kita bisa medefinisikan properti `$ createdByAttribute` dan `$ updatedByAttribute` sebagai berikut:

```
public function behaviors() {
    return [
        [
            'class' => BlameableBehavior::className(),
            'createdByAttribute' => 'author_id',
            'updatedByAttribute' => 'updater_id',
        ],
    ];
}
```

```

        ],
}
}
```

Nah jika kedua *class Behavior* tersebut digunakan maka penulisannya tinggal digabung jadi satu. Modifikasi file `app\models\Category` dengan menambahkan kode sebagai berikut

```

public function behaviors() {
    return [
        TimestampBehavior::className(),
        BlameableBehavior::className()
    ];
}
```

Jangan lupa tambahkan `use` dua Class tersebut di awal *file*.

```

use Yii;
use yii\behaviors\TimestampBehavior;
use yii\behaviors\BlameableBehavior;
```

Mari kita uji coba.

1. Sebelum melakukan uji coba, kita harus *login* terlebih dahulu dengan *username* **admin** dan kata sandi **admin**

Please fill out the following fields to login:

Username	admin
Password	*****
<input checked="" type="checkbox"/> Remember Me	
Login	

You may login with admin/admin or demo/demo.

2. Lakukan *update* data *Category*

Update Category: Buku Tulis

Title	Buku Tulis
Description	Buku buat tulis menuulis
Update	

Setelah tombol *Update* diklik, maka pada *view* kita bisa melihat bahwa *field updated_at* dan *field updated_by* terisi datanya secara otomatis.

Buku Tulis

ID	1
Title	Buku Tulis
Description	Buku buat tulis menuulis
Created At	(not set)
Updated At	1452038332
Created By	(not set)
Updated By	100

3. Untuk mengubah tampilan *field updated_at*, dan *created_at* menjadi tampilan yang bisa dibaca maka kita bisa menggunakan pemformatan pada *class DetailView* sebagai berikut

```
@app/views/category/view.php
```

```
<?= DetailView::widget([
    'model' => $model,
    'attributes' => [
        ...
        'created_at:datetime',
        'updated_at:datetime',
        ...
    ],
]) ?>
```

ID	1
Title	Buku Tulis
Description	Buku buat tulis menulis
Created At	(not set)
Updated At	Jan 5, 2016, 5:31:43 PM
Created By	(not set)
Updated By	100

Atau kita juga bisa menggunakan cara sebagai berikut

```
<?= DetailView::widget([
    'model' => $model,
    'attributes' => [
        ...
        [
            'attribute' => 'created_at',
            'format' => ['date', 'php:Y-m-d H:i:s']
        ],
        [
            'attribute' => 'updated_at',
            'format' => ['date', 'php:Y-m-d H:i:s']
        ],
        ...
    ],
]) ?>
```

ID	1
Title	Buku Tulis
Description	Buku buat tulis menulis
Created At	(not set)
Updated At	05 Jan 2016 17:31:43
Created By	(not set)

Jika kita jeli, maka ada yang aneh dengan tampilan waktu yang muncul, dimana waktu panduan ini dibuat masih pagi sekitar pukul 9 WIB namun mengapa hasil konversinya menjadi sekitar jam 17 atau 5 sore?

Jawabannya adalah ternyata *timestamp* disimpan dalam zona waktu nol atau *Greenwich* (GMT/UTC). Oleh karena itu kita perlu mengaturnya agar zona waktu sesuai dengan zona waktu indonesia.

- WIB = UTC + 7
- WITA = UTC + 8
- WIT = UTC + 9

Pengaturan zona waktu bisa kita jumpai di PHP Manual *Bab List of Supported Timezones*, dari sana bisa kita dapatkan bahwa:

- WIB = Asia/Jakarta
- WITA = Asia/Makassar

- WIT = Asia/Jayapura

Lalu bagaimana implementasinya di Yii? Caranya cukup dengan menambahkan parameter `timeZone` pada `config (@app/config/web.php)`

```
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'timeZone' => 'Asia/Jakarta',
    'components' => [
```

Silakan dicoba lagi

Title	Buku Tulis
Description	Buku buat tulis menulis
Created At	(not set)
Updated At	06 Jan 2016 08:31:43
Created By	(not set)

Akhirnya waktu telah kembali ☺

4. Sedangkan untuk tampilan `field updated_by`, dan `created_by`, maka kita bisa modifikasi `file @app/views/category/view.php` sebagai berikut.

```
<?php
$username_created_by = $model->created_by;
if($user=User::findIdentity($model->created_by)) {
    $username_created_by=$user->username;
}

$username_updated_by = $model->updated_by;
if($user=User::findIdentity($model->updated_by)) {
    $username_updated_by=$user->username;
}
?>

<?= DetailView::widget([
    'model' => $model,
    'attributes' => [
        ...
        [
            'attribute' => 'created_by',
            'value' => $username_created_by,
        ],
        [
            'attribute' => 'updated_by',
            'value' => $username_updated_by,
        ],
    ],
]) ?>
```

`FindIdentity()` berfungsi mengembalikan objek `User` melalui ID yang diberikan. Lebih jelasnya silakan lihat pada `class model` di `@app/models/User.php`

Karena menggunakan model `User`, jangan lupa `use` dulu

```
| use app\models\User;
```

Title	Buku Tulis
Description	Buku buat tulis menulis
Created At	(not set)
Updated At	06 Jan 2016 08:31:43
Created By	(not set)
Updated By	admin

5. Silakan coba *create* beberapa kategori baru. Lalu lihat halaman index, pada kolom created_at dan created_by

Showing 1-4 of 4 items.				
#	Title	Description	Created At	Created By
1	Buku Tulis	Buku buat tulis menulis	(not set)	(not set)
2	Buku Bacaan	Buku komik, cerita dll	1452048105	100
3	Alat Tulis	Pencil, Bulpoin	1452048134	100

Dengan cara yang kurang lebih sama dengan di view.php mari kita modifikasi *Gridview* @app/views/category/index.php

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],
        // 'id',
        'title',
        'description:ntext',
        'created_at:datetime',
        // 'updated_at',
        [
            'attribute' => 'created_by',
            'value' => function($data) {
                $username_created_by = $data->created_by;
                if($user=User::findIdentity($data->created_by)) {
                    $username_created_by=$user->username;
                }
                echo $username_created_by;
            }
        ],
        // 'updated_by',
        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
```

Sedikit berbeda dengan *DetailView*, untuk mengisi *value* pada *Gridview* menggunakan fungsi *closure*. Parameter *\$data* pada fungsi tersebut merupakan model *Category* per *row*.

Silakan dicoba..

Class 'User' not found

1. in C:\Bitnami\wampstack\apache2\htdocs\basic\views\category\index.php

Ops, ternyata *error*, jangan panik karena pesan *error*-nya sudah sangat jelas bahwa *class User* tidak ditemukan. Karena itu definisikan lokasi *class User* melalui *use*

```
| use app\models\User;
```

Coba lagi, dan hasilnya akan sesuai ekspektasi

Showing 1-4 of 4 items.

#	Title	Description	Created At	Created By
1	Buku Tulis	Buku buat tulis menulis	(not set)	(not set)
2	Buku Bacaan	Buku komik, cerita dll	Jan 6, 2016, 9:41:45 AM	admin
3	Alat Tulis	Pencil, Bulpoin	Jan 6, 2016, 9:42:14 AM	admin
4	Makanan Ringan	Snack, Drink	Jan 6, 2016, 9:42:53 AM	admin

Selesai.

Selengkapnya silakan baca di

∞ <http://www.yiiframework.com/doc-2.0/guide-output-data-widgets.html>

B. Kesimpulan

Yii dengan kode *generator*-nya yaitu Gii akan mempercepat kita untuk membuat aplikasi CRUD. Hal ini merupakan langkah awal yang baik, namun CRUD yang dihasilkan masih umum, artinya perlu penyesuaian dan modifikasi agar memenuhi kebutuhan kita.

Pada bab selanjutnya, kita akan belajar tentang apa salah satu topik yang populer dalam dunia pemrograman web yaitu Ajax. Di samping itu juga akan dikenalkan dengan salah satu teknik Ajax yaitu Pjax serta implementasinya di Yii.

7

Ajax dan Pjax

Pada bab ini kita akan belajar tentang:

- Pengelolaan *asset*
- Implementasi Ajax
- Implementasi Pjax

A. Asset

Pemahaman tentang *Asset* ini penting sebelum kita belajar tentang *Ajax* dan *Pjax*

A. 1. Definisi

Tentu kita tidak sedang membicarakan tentang ilmu akuntansi, karena *Asset* yang penulis maksud di sini masih berkaitan dengan web. Pada Yii, *Asset* adalah *file* yang dapat dirujuk dalam halaman web, contoh: *file* CSS, JavaScript, gambar atau video, dll. *Asset* terletak pada direktori web (@app/web) yang bisa diakses oleh pengguna.

Kita bisa menggunakan *asset* yang berbeda untuk suatu *view* yang berbeda. Disampig itu, kita juga bisa mengelompokkan antara *asset* yang utama atau menjadi *base* dari aplikasi kita atau *asset* yang hanya digunakan untuk *layout* atau *widget* tertentu saja.

A. 2. Manajemen Asset

Yii mengelola asset melalui class [[yii\web\AssetBundle]]. Melalui *class* ini kita bisa mendefinisikan *asset-asset* mana yang akan digunakan pada aplikasi, bagaimana ketergantungan dengan *asset* lain, serta dimana lokasi dari *asset* tersebut.

Berikut ini contoh kode dasar dari penerapan *class* tersebut pada *application template Basic*.

```
<?php
namespace app\assets;
use yii\web\AssetBundle;
class AppAsset extends AssetBundle
{
    public $basePath = '@webroot';
    public $baseUrl = '@web';
    public $css = [
        'css/site.css',
    ];
    public $js = [
    ];
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap\BootstrapAsset',
    ];
}
```

Untuk mengimplementasikan AssetBundle tersebut pada aplikasi kita maka di *view* atau *layout* kita bisa gunakan kode sebagai berikut.

```
| \app\assets\AppAsset::register($this);
```

Ketika kita meletakkan kode di atas di *view* maka Yii akan meng-generate elemen HTML *script* untuk meng-include *file* Javascript dan elemen HTML tautan untuk meng-include *file* CSS.

A. 3. Javascript dan CSS tanpa AssetBundle

Di samping menggunakan *class Asset Bundle*, kita juga bisa memasukkan kode Javascript/JS dan CSS tanpa melalui *class* tersebut, melainkan secara langsung pada *view*-nya. Hal ini bisa dilakukan dengan memanfaatkan fungsi-fungsi pada *class* [[yii\web\View]].

Pertanyaan yang mungkin muncul adalah “Bukankah kita cukup menggunakan tag html <style> atau <script> untuk mengimplementasikan CSS dan JS pada aplikasi web kita?

```
<style type='text/css'>
/* kode css */
</style>

<script>
/* kode js */
</script>
```

Jawabannya adalah kita tetap bisa dan diperbolehkan menggunakan cara normal seperti itu sebagaimana kita diperbolehkan menggunakan kode-kode PHP pada Yii. Ini tidak lebih dari memilih antara kode ini

```
| <?= $this->registerJs("alert('test')"); ?>
```

Atau kode berikut ini

```
<script>
alert('test')
</script>
```

Keduanya memiliki fungsi yang sama. Hanya saja ada hal yang tidak bisa dilakukan jika menggunakan cara normal yaitu ketika posisi CSS/JS mempengaruhi fungsinya.

Misalnya:

- Kode JS yang bergantung pada *library* JS lain, contoh: JQuery-Pjax tidak akan jalan jika *core* JQuery belum dijalankan
- Kode CSS seharusnya diletakkan di antara elemen HTML *head*.

Menurut hemat penulis, lakukan yang normal hingga itu tidak mungkin dilakukan. Hal ini dikarenakan penggunaan fungsi di Yii memang bisa mempermudah, namun ada harga yang harus dibayar (performa), meski itu pun juga tidak signifikan.

Berikut ini beberapa fungsi yang bisa digunakan untuk memasukkan kode css dan Js pada projek aplikasi kita.

registerCss()

```
| public void registerCss ( $css, $options = [], $key = null )
```

Keterangan:

- \$css berisi kode CSS yang akan di-*register* pada suatu *view*
- \$options berisi *array* dari HTML atribut untuk tag <style>.
- \$key adalah kunci yang mengidentifikasi blok kode CSS. Jika kosong, maka akan parameter \$css akan menjadi kunci. Jika dua blok kode CSS diregister dengan kunci yang sama maka yang terakhir akan menimpa yang sebelumnya.

Contoh:

```
$this->registerCss (
    '.text-blue {
        color: blue;
    }
", [ 'type'=>'text/css' ]);
```

Meski kode di atas diletakkan di *view*, namun hasil keluarannya diletakkan pada elemen HTML *head* oleh Yii.

```
-- -----
11 <link href="/basic/web/assets/basic.css?r=bbb3" rel="stylesheet" type="text/css">
12 <style type="text/css">
13     .text-blue {
14         color: blue;
15     }
16 </style></head>
17 <body>
```

registerCssFile()

```
| public void registerCssFile ( $url, $options = [], $key = null )
```

Keterangan:

- \$url berisi lokasi dari *file* CSS yang di-*register*.
- \$options merupakan *array* dari atribut HTML untuk elemen *link*. Mengacu pada yii\helpers\Html::cssFile() untuk opsi yang didukung. Di samping itu ada satu parameter spesial yang bukan elemen *link* yaitu *depends* yang merupakan nama dari *asset-asset bundles* berformat *array* yang mana *file* CSS tersebut tergantung padanya.

- \$key (sama dengan penjelasan di registerCss).

Contoh:

```
$this->registerCssFile("@web/css/print.css", [
    'depends' => [
        yii\bootstrap\BootstrapAsset::className()
    ],
    'media' => 'print',
], 'css-print-theme');
```

Kode di atas artinya kita meregister file CSS yang terletak di @web/css/print.css dimana file ini hendaknya dimasukkan setelah *Bootstrap Asset* di-register, kemudian medianya print (CSS yang hanya akan bekerja ketika pengguna mencetak halaman web) dengan kunci 'css-print-theme'.

```
8   <title>My Yii Application</title>
9   <link href="/basic/web/assets/de7fed43/css/bootstrap.css" rel="stylesheet"/>
10 <link href="/basic/web/css/print.css" rel="stylesheet" media="print">
11 <link href="/basic/web/css/site.css" rel="stylesheet">
12 <link href="/basic/web/assets/18a1097f/toolbar.css" rel="stylesheet">
```

registerJs()

```
| public void registerJs ( $js, $position = self::POS_READY, $key = null )
```

Keterangan:

- \$js berisi kode Js yang di-register
- \$position merupakan posisi dimana kode JS ditambahkan dalam halaman. Nilai yang bisa digunakan sebagai berikut:
 - POS_HEAD: pada elemen HTML *head*
 - POS_BEGIN: diawal elemen HTML *body*
 - POS_END: diakhiri elemen HTML *body*
 - POS_LOAD: kode JS akan dibungkus dengan fungsi `jQuery(window).load()`. Kode JS akan di-register setelah file jQuery di-register.
 - POS_READY: kode JS akan di lingkup dengan fungsi `jQuery(document).ready()`. Kode JS akan di-register setelah file jQuery di-register.
 - \$key

registerJsFile()

```
| public void registerJsFile ( $url, $options = [], $key = null )
```

Keterangan:

- \$url berisi lokasi file JS yang di-register.
- \$options merupakan array dari atribut HTML untuk elemen *script*.

Di samping itu berikut ini parameter yang bisa digunakan:

- depends: array, asset-asset bundles berformat array yang mana file CSS tersebut tergantung padanya.
- position: lokasi spesifik dimana kode JS ditambahkan ke halaman. Nilai yang bisa digunakan:
 - o POS_HEAD: di elemen HTML *head*
 - o POS_BEGIN: di awal elemen HTML *body*
 - o POS_END: diakhiri elemen *body*. Ini adalah nilai default.
- \$key

Contoh:

```
$this->registerJsFile('@web/js/main.js', [
    'depends' => [
        yii\web\JqueryAsset::className()
    ],
]);
<script src="/basic/web/assets/3a0cd6f2/jquery.js"></script>
<script src="/basic/web/js/main.js"></script>
<script src="/basic/web/assets/54dd6761/yii.js"></script>
```

B. Ajax

B. 1. Definisi

Ajax adalah singkatan dari *asynchronous Javascript and XML*, merupakan istilah populer dalam pemrograman web. Umumnya, penggunaan Ajax adalah untuk memuat suatu bagian halaman tanpa *me-refresh* keseluruhan halaman.

Sebagai gambaran, pada suatu situs web, biasanya yang sering berubah adalah pada bagian tengah atau konten, sedangkan *header*, *footer* dan menu *sidebar* relatif tetap. Maka dengan Ajax kita bisa memuat hanya pada bagian kontennya saja. Hal ini tentunya akan mempercepat dan mengefisienkan *loading* suatu halaman situs web.

B. 2. Cara Kerja

Ajax bekerja dengan menggunakan *engine* XMLHTTP, merupakan *engine* yang disematkan secara *default* pada *web browser* dan memiliki kemampuan memuat data dari server secara *asynchronous*. *Asynchronous* memungkinkan pemuatan data secara paralel tanpa menunggu proses sebelumnya selesai.

Dahulu, sebelum semua *web browser* menyematkan *engine* XMLHTTP, teknik *asynchronous* dilakukan dengan memanfaatkan elemen HTML Iframe dan Frame. Namun saat ini, semua *web browser* besar telah menggunakan *engine* tersebut sehingga kita bisa menerapkan teknik Ajax dengan mudah dan efisien.

B. 3. Implementasi

Sejak versi 2, Yii telah membuang semua *inline* Ajax pada *widget-widget*-nya sehingga kita perlu koding secara manual. Meskipun demikian, secara *default*, Yii telah menyertakan JQuery yaitu *library* Javascript yang memudahkan kita dalam mengimplementasikan Ajax pada aplikasi kita.

Founder Yii pernah menjelaskan tentang hal ini.

qiangxue commented on Aug 28, 2013

In 2.0 we removed most in-page js code. You should write explicit js code in external js files to achieve similar result as in 1.1. The code is very trivial though.

Sebagai informasi, secara bertahap Yii akan membuang jQuery dari *core*-nya

Pada panduan ini penulis akan menggunakan contoh kasus penggunaan Ajax yang sangat umum digunakan yaitu *dependent dropdown*.

Persiapan

Mari kita buat *controller* baru untuk latihan *dependent dropdown* pada direktori @app/controllers/

```
<?php
namespace app\controllers;
class AjaxController extends \yii\web\Controller
{}
```

Dropdown ke Textbox

Pada bagian ini, kita akan menyimulasikan sebuah *dropdown* yang berhubungan dengan beberapa *textbox*. Skenarionya adalah apabila data *dropdown* dipilih maka nilai *textbox* akan berubah sesuai dengan *dropdown*-nya.

Pada panduan ini, kita belum akan menggunakan basis data atau model ActiveRecord melainkan hanya menggunakan *array*. Cara ini sebenarnya justru lebih sulit dibandingkan dengan menggunakan basis data, namun kita akan menggunakannya untuk menambah wawasan jika menjumpai kasus yang sama di dunia nyata.

Kita akan menggunakan data buku sebagai contohnya. Pada *controller* tambahkan fungsi *getBooks*, yang berfungsi mengembalikan semua daftar buku yang tersedia.

```
public function getBooks()
{
    $books = [
        ['id'=>'1', 'title'=>'Pemrograman PHP', 'author'=>'Hafid', 'year'=>'2015'],
        ['id'=>'2', 'title'=>'Pemrograman JS', 'author'=>'Juned', 'year'=>'2014'],
        ['id'=>'3', 'title'=>'Database MySQL', 'author'=>'Lily', 'year'=>'2013'],
    ];
    // Jika menggunakan basis data maka:
    // $books = Book::find()->asArray()->all();
```

```

    return $books;
}

```

Kemudian kita buat fungsi `actionBook` untuk menampilkan formulir buku. Supaya mudah, kita menggunakan `DynamicModel`.

```

public function actionBook()
{
    $model = new \yii\base\DynamicModel([
        'title', 'author', 'year'
    ]);
    $model->addRule(['title'], 'string');
    $model->addRule(['description'], 'string');
    $model->addRule(['year'], 'integer');

    return $this->render('book', [
        'model' => $model,
        'books' => $this->getBooks(),
    ]);
}

```

Di sini kita bisa menambahkan beberapa *rule* terhadap *field* yang digunakan.

Kemudian kita buat `view book.php` pada direktori ajax.

```
@app/views/ajax/book.php
```

```

<?php
use yii\widgets\ActiveForm;
use yii\helpers\ArrayHelper;
use yii\helpers\Url;

$form = ActiveForm::begin();
$data = ArrayHelper::map($books, 'id', 'title');
echo $form->field($model, 'title')->dropDownList($data, [
    'prompt'=>'-Choose a title-',
]);
echo $form->field($model, 'author')->textInput();
echo $form->field($model, 'year')->textInput();
ActiveForm::end();

```

Field title berupa dropdown sedangkan *field author* dan *year* berupa input teks biasa. Skenarionya jika pengguna memilih judul buku melalui dropdown *title* maka *field author* dan *year* akan otomatis terisi sesuai dengan judul bukunya.

Kembali ke *controller*, mari kita tambahkan fungsi `actionGetBook()` untuk mendapatkan buku dengan id tertentu

```

public function actionGetBook($id)
{
    $books = $this->getBooks();
    $bookSelected = [];
    foreach($books as $book) {
        if($book['id']===$id) {
            $bookSelected = $book;
        }
    }
    // $bookSelected = Book::findOne($id);
    Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
    return [
        'book' => $bookSelected,
    ];
}

```

Nilai balik dari fungsi ini berformat JSON supaya lebih efisien. Nilai balik ini kemudian akan diproses oleh Ajax.

Kembali ke *view*, kita tambahkan fungsi Ajax untuk me-request fungsi di atas.

```

$this->registerJs('
    $("#dynamicmodel-title").change(function() {
        $.get("'.Url::to(['get-book','id'=>'']).'" + $(this).val(), function(data) {
            $("#dynamicmodel-author").val(data.book.author);
            $("#dynamicmodel-year").val(data.book.year);
        });
    });
')

```

```
| ');
```

`$.get` adalah fungsi Ajax pada JQuery untuk memuat URL dengan *method get*. Perhatikan bahwa format JSON bisa dibaca dengan mudah di Js dengan menggunakan titik.

Mari kita uji coba kode kita tersebut.

```
∞ http://localhost/basic/web/ajax/book
```

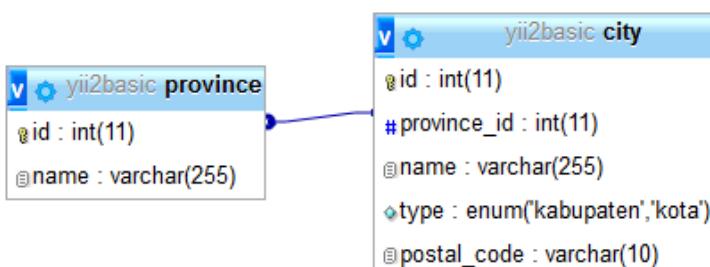
Selesai.

Dropdown ke Dropdown

Dropdown ke dropdown atau yang terkenal dengan sebutan *dependent dropdown*, bisa kita selesaikan dengan cara yang hampir sama dengan cara di atas.

Pada panduan ini kita akan menggunakan data dari tabel basis data, namun tanpa menggunakan model ActiveRecord. Query data dilakukan dengan menggunakan DAO.

Kita akan menggunakan dua tabel yaitu *province* dan *city*. Berikut ini contoh struktur tabelnya.



Baik, mari kita buat fungsi baru pada AjaxController yaitu `getProvinces()`

```
public function getProvinces()
{
    return (new \yii\db\Query())
        ->select('*')
        ->from('province')
        ->orderBy(['name' => SORT_DESC])
        ->all(\yii::$app->db);
}
```

Kemudian kita tambahkan fungsi `actionDepdrop` untuk menampilkan *form*

```
public function actionDepdrop()
{
    $model = new \yii\base\DynamicModel([
        'province_id', 'city_id',
    ]);
    $model->addRule(['province_id'], 'integer');
    $model->addRule(['city_id'], 'integer');

    return $this->render('depdrop', [
        'model' => $model,
        'provinces' => $this->getProvinces(),
    ]);
}
```

Lalu kita buat *view* depdrop.php pada direktori @app/views/ajax/depdrop.php

```
<?php
use yii\widgets\ActiveForm;
use yii\helpers\ArrayHelper;
use yii\helpers\Url;
$form = ActiveForm::begin();
$data = ArrayHelper::map($provinces, 'id', 'name');
echo $form->field($model, 'province_id')->dropDownList($data, [
    'prompt'=>'-Choose a province-',
]);
echo $form->field($model, 'city_id')->dropDownList([], [
    'prompt'=>'-Choose a city-',
]);
ActiveForm::end();
```

Kembali ke *controller*, kita tambahkan fungsi actionGetCities() untuk mendapatkan data kota dengan id provinsi tertentu

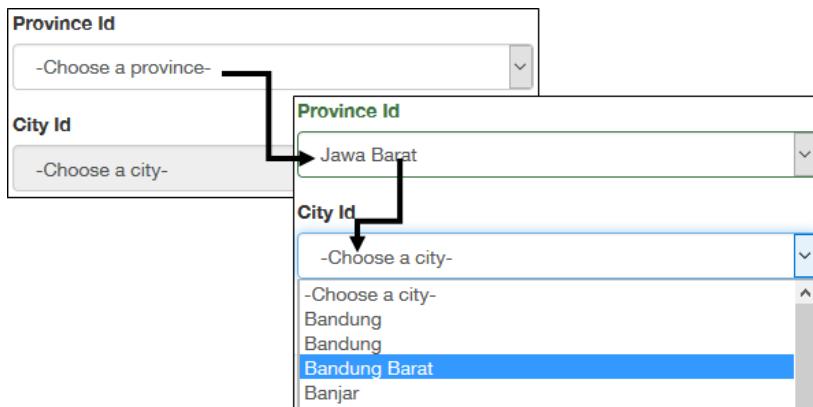
```
public function actionGetCities($province_id)
{
    $cities = (new \yii\db\Query())
        ->select('*')
        ->from('city')
        ->where([
            'province_id'=>$province_id,
        ])
        ->all(\yii::$app->db);
\Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
return [
    'cities' => $cities,
];
}
```

Nah sekarang tinggal membuat fungsi Ajax di *view*

```
$this->registerJs('
    $("#dynamicmodel-city_id").attr("disabled",true);
    $("#dynamicmodel-province_id").change(function() {
        $.get(".Url::to(['get-
cities','province_id=>''']).'" + $(this).val(), function(data) {
            select = $("#dynamicmodel-city_id")
            select.empty();
            var options = "<option value='\\\'>-Choose a city-</option>";
            $.each(data.cities, function(key, value) {
                options += "<option value='\\\"+value.id+\\\'>" + value.name +</option>";
            });
            select.append(options);
            $("#dynamicmodel-city_id").attr("disabled",false);
        });
    });
');
```

Pertama kali elemen *dropdown city* kita set menjadi *disabled*, kemudian setelah data diperoleh maka data *city* baru di*enable* lagi.

Hasil dari *request* Ajax yang berformat JSON tersebut berisi banyak kota, kemudian kita ekstrak datanya menggunakan fungsi `$.each` untuk kemudian digunakan untuk membangun *tag* HTML *option*. Setelah itu, *HTML option* ditambahkan/*append* ke elemen *dropdown*-nya.



Selesai. Mudah sekali bukan?

Ajax Submit Formulir

Penanganan *submit* formulir menggunakan Ajax agak sedikit berbeda dengan cara normal. Pada Yii kita bisa menggunakan *event* on *BeforeSubmit* yaitu suatu *event* yang terjadi ketika semua validasi formulir sukses.

Berikut ini contoh view formulir (formulir normal).

```
<?php $form = ActiveForm::begin([
    'options' => [
        'id' => 'formX',
    ],
    'action' => ['create'],
    'method' => 'post',
]); ?>
<?= $form->field($model, 'title')->textInput() ?>
<?= $form->field($model, 'price')->textInput() ?>
<div class="form-group">
    <?= Html::submitButton('Add to list', ['class' => 'btn btn-primary']) ?>
</div>
<?php ActiveForm::end(); ?>
```

Hanya saja kita gunakan kode JS untuk men-trigger *event* *beforeSubmit* pada formulir dengan id formX

```
<?php
$this->registerJs('
$(document).on("beforeSubmit","#formX",function(event) {
    var form = $(this);
    if (form.find(".has-error").length) {
        return false;
    }
    $.ajax({
        url: form.attr("action"),
        type: form.attr("method"),
        data: form.serialize(),
        success: function (response) {
            // kode jika sukses
        }
    });
    return false;
});
');
```

Selanjutnya merupakan kode ajax biasa, silakan bereksplorasi.

Selesai.

C. Pjax

C. 1. Definisi

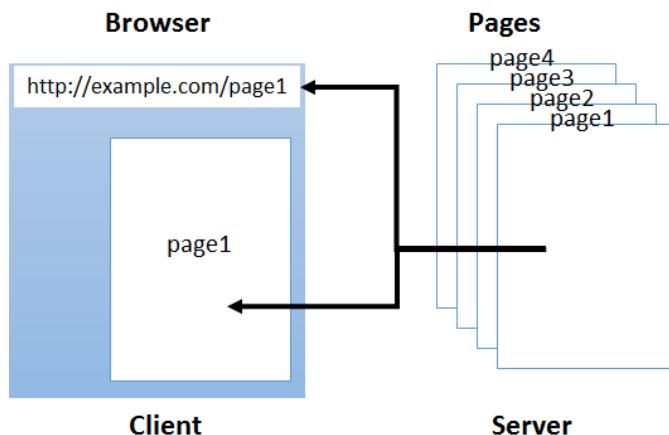
Pjax singkatan dari *push state + Ajax*, merupakan *plugin* jQuery yang menggunakan Ajax dan HTML5 pushState untuk memberikan pengalaman *browsing* yang cepat dengan menggunakan tautan sebenarnya, judul halaman yang sesuai, dan tombol *back* pada *web browser* yang bekerja.

Web *official* Pjax adalah <https://github.com/defunkt/jquery-pjax>, namun Yii mengembangkan sendiri jquery-pjax-nya, yang mana ada sedikit perbedaan dan penyesuaian dengan Yii.

C. 2. Cara Kerja

Cara kerja Pjax adalah mengambil konten halaman web dari server melalui Ajax dan mengganti konten dari suatu halaman yang sedang tampil. Kemudian memperbarui URL *Address* dan *state web browser* menggunakan HTML5 pushState tanpa me-refresh halaman (*layouts*, JS, CSS). Seolah-olah ada *loading* seluruh halaman padahal itu *loading* konten dari halaman tersebut menggunakan Ajax dan HTML5 pushState.

Berikut ini ilustrasinya.



C. 3. Implementasi

Sebagaimana yang dijelaskan diawal bahwa Yii menggunakan Pjax versi sendiri, dimana ada sedikit *enhancement*. Silakan lihat pada URL berikut

∞ <https://github.com/yiisoft/jquery-pjax>

Panduan dan opsi terkait Pjax secara lengkap bisa kita jumpai pada tautan tersebut.

Pada Yii, Pjax ditangani oleh *class* [[*yii\widgets\Pjax*]]. *Class* ini merupakan *widget* yang mempunyai dua fungsi yaitu *begin* dan *end*. Cukup dengan membungkus suatu konten atau *widget* lain dengan *widget* Pjax maka Pjax sudah bisa berjalan pada halaman tersebut.

Pada panduan ini, agar lebih mudah dalam memahaminya, maka kita akan menggunakan CRUD *category* yang telah kita *generate* menggunakan Gii pada bab sebelumnya sebagai studi kasus.

Pjax pada Gridview

Mari kita buka *view index.php* pada direktori @app/views/category/, tambahkan *widget* Pjax

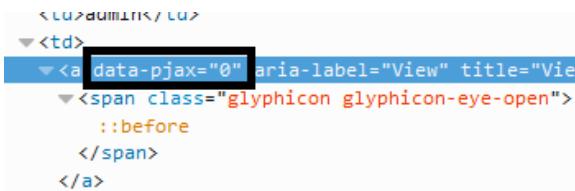
```
<?php \yii\widgets\Pjax::begin(['timeout'=>5000, 'id'=>'pjax-gridview']); ?>
<?= GridView::widget([
...
]); ?>
<?php \yii\widgets\Pjax::end() ?>
```

Timeout adalah maksimal waktu (satuan *milisecond*) yang diberikan untuk Pjax dalam me-request data melalui Ajax dari server, jika lebih maka *request* akan dilakukan dengan cara normal tanpa Ajax. Kita bisa menge-set-nya menjadi *false* untuk waktu yang tak terbatas.

Hanya dengan kode yang sesingkat itu, maka secara umum, Pjax pada Gridview sudah *ready* atau bekerja. Fungsi *paging*, *sorting*, dan *searching* otomatis sudah menggunakan Pjax.

Silakan dicoba.

Adapun pada *actionColumn*, yaitu tautan *view*, *update*, dan *delete*, Pjax belum aktif karena secara *default* telah dinonaktifkan oleh Yii yaitu dengan menambahkan atribut *data-pjax="0"* pada tautan.



Oleh karena itu, untuk mengaktifkannya, kita perlu hapus atribut tersebut dari tautan. Caranya modifikasi Gridview pada `view index.php`

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ...
        [
            'class' => 'yii\grid\ActionColumn',
            'buttons' => [
                'view' => function ($url, $model) {
                    $icon='<span class="glyphicon glyphicon-eye-open"></span>';
                    return Html::a($icon,$url);
                },
                'update' => function ($url, $model) {
                    $icon='<span class="glyphicon glyphicon-pencil"></span>';
                    return Html::a($icon,$url);
                },
                'delete' => function ($url, $model) {
                    $icon='<span class="glyphicon glyphicon-trash"></span>';
                    return Html::a($icon,$url,[
                        'data-confirm'=>"Are you sure you want to delete this item?",
                        'data-method'=>'post',
                    ]);
                },
            ],
        ],
    ],
]); ?>
```

Kita diizinkan untuk meng-override tautan *action* atau bahkan menambahkannya melalui parameter *buttons*. Setelah itu, pada *controller* CategoryController (@app/controllers/CategoryController.php) kita ubah fungsi `render` menjadi `renderAjax`.

Kita akan mencoba untuk `actionView` terlebih dahulu.

```
public function actionView($id)
{
    return $this->renderAjax('view', [
        'model' => $this->findModel($id),
    ]);
}
```

Oke mari kita uji coba di *web browser*.

ID	1
Title	Buku Tulis
Description	Buku buat tulis menulis

Berhasil, *loading view tanpa refresh*, dan kita lihat URL sudah menunjukkan lokasi yang benar, artinya Pjax berhasil mem-push URL ke *web browser*.

Namun ada beberapa catatan.

1. Harusnya bingkai Pjax melingkupi judul dan tombol *Create Category*, tidak hanya Gridview saja. Sehingga ketika pindah halaman dalam hal ini halaman *view* maka judul halaman index dan tombol *Create Category* akan muncul seperti gambar di atas.

Berikut ini gambaran bingkai Pjax

#	Title	Description	Created At	Created By
1	Buku Tulis	Buku buat tulis menulis	(not set)	(not set)
2	Buku Bacaan	Buku komik, cerita dll	Jan 5, 2016, 6:41:45 PM	admin

Oleh karena itu kita bisa pindahkan kode `Pjax::begin` ke posisi di atas judul

```
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="category-index">
<?php
Pjax::begin(['timeout'=>false, 'id'=>'pjax-gridview']);
?>

<h1><?= Html::encode($this->title) ?></h1>
<?php // echo $this->render('_search', ['model' => $searchModel]); ?>
```

2. Catatan kedua adalah apabila halaman *view* di-refresh (dengan menekan F5 atau tombol *refresh* ada *web browser*) maka akan menjadi seperti berikut.

Buku Tulis

[Update](#) [Delete](#)

ID 1
Title Buku Tulis
Description Buku buat tulis menulis
Created At (not set)
Updated At 05 Jan 2016 19:10:14
Created By (not set)
Updated By admin

Yap, seolah tidak ada *layout* karena renderAjax itu hanya akan me-*render* konten halaman saja tanpa *layout* utama. Oleh karena itu perlu kita bedakan, jika *request* melalui Pjax maka kita bisa menggunakan renderAjax, selain itu gunakan *render* normal.

```
public function actionView($id)
{
    if(Yii::$app->request->isAjax){
        return $this->renderAjax('view', [
            'model' => $this->findModel($id),
        ]);
    }
    else{
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }
}
```

Perintah `Yii::$app->request->isAjax` untuk mendeteksi *request* Ajax dari *web browser* dengan mendeteksi *header request*-nya. Di samping itu kita juga bisa menggunakan perintah `Yii::$app->request->isPjax`, untuk lebih spesifik ke Pjax.

Pada `view.php` ini kita juga perlu tambahkan tombol kembali ke halaman index atau Gridview

```
<?= Html::a('Back', ['index'], ['class' => 'btn btn-success']) ?>
```

Kita bisa lakukan hal yang sama untuk `actionCreate` dan `actionUpdate`.

```
public function actionCreate()
{
    $model = new Category();
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        if(Yii::$app->request->isAjax){
            return $this->renderAjax('create', [
                'model' => $model,
            ]);
        }
        else{
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }
}
```

Pjax pada Fungsi Hapus

Pada fungsi hapus atau `actionDelete`, Pjax ternyata tidak jalan. Oleh karena itu kita perlu mengambil alih fungsi pada tombol *delete*. Kita akan menggunakan Javascript untuk me-*request* fungsi `actionDelete`, yang tentunya setelah pengguna mengkonfirmasinya, baru setelah itu Gridview di- *refresh*.

Pada tautan *delete*, kita perlu hapus `data-confirm` karena memicu *popup confirm* yang di-generate oleh Yii. Kita akan gunakan fungsi sendiri untuk menampilkan konfirmasi dalam bentuk *popup* sehingga *action* setelah pengguna melakukan konfirmasi *popup* tersebut bisa kita tangani.

Tambahkan penanda pada tautan *delete* di Gridview misalnya dengan menggunakan atribut *class CSS*, *set* nilainya menjadi `pjaxDelete`

```
'delete' => function ($url, $model) {
    $icon='<span class="glyphicon glyphicon-trash"></span>';
    return Html::a($icon,$url,[
        'class'=>'pjaxDelete'
    ]);
},
```

Kemudian mari kita buat kode Javascript untuk menjalankan perintah tertentu ketika pengguna menekan tombol *delete* tersebut.

```
$this->registerJs('
/* fungsi ini akan dijalankan ketika class pjaxDelete diklik */
$(".pjaxDelete").on("click", function (e) {
    /* cegah tautan menjalankan default action */
    e.preventDefault();
    if(confirm("Are you sure you want to delete this item?")){
        /* request actionDelete dengan method post */
        $.post($(this).attr("href"), function(data) {
            /* reload gridview */
            $.pjax.reload("#pjax-gridview", {"timeout":false});
        });
    }
});');
');
```

Informasi:

Fungsi `actionDelete` hanya menerima *request* menggunakan *method post*, oleh karena itu kita menggunakan `$.post`, lihat pada CategoryController fungsi *behaviors*

Lalu pada *controller* `actionDelete`, kita perlu sedikit modifikasi dengan menghapus fungsi *redirect*, sebagai berikut

```
public function actionDelete($id)
{
    $this->findModel($id)->delete();
    //return $this->redirect(['index']);
}
```

Karena fungsi *redirect* telah ditangani oleh Javascript melalui Pjax *reload*.

Selesai. Silakan dicoba.

Pjax pada Submit Form

Pada formulir *create* maupun *update*, fungsi *submit* masih belum menggunakan Pjax, sehingga halaman ikut ter-refresh. Bagaimana dengan ajax *submit* yang telah kita bahas sebelumnya?. Sedikit berbeda, *submit* formulir pada Pjax lebih sederhana.

Ada dua hal yang perlu kita lakukan untuk mengimplementasikannya, yaitu dengan membungkus formulir dengan Pjax dan menambahkan opsi `data-pjax` pada form. Maka formulir akan secara otomatis Pjax *ready*.

Mari kita modifikasi `view @app/views/category/_form.php`, sebagaimana panduan di atas serta tambahkan tombol *back* untuk kembali ke *Gridview*.

```
<?php
use yii\helpers\Html;
use yii\widgets\ActiveForm;
use yii\widgets\Pjax;
Pjax::begin([
    'id'=>'pjax-form', 'timeout'=>false,
]);
?>
<?php $form = ActiveForm::begin([
    'options' => ['data-pjax' => true ]
]); ?>
<?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>
<?= $form->field($model, 'description')->textInput(['maxlength' => true]) ?>
```

```

<div class="form-group">
    <?= Html::submitButton($model-
>isNewRecord ? 'Create' : 'Update', ['class' => $model->isNewRecord ? 'btn btn-
success' : 'btn btn-primary']) ?>
    <?= Html::a('Back', ['index'], ['class' => 'btn btn-success']) ?>
</div>

<?php ActiveForm::end(); ?>
<?php Pjax::end(); ?>

```

Kemudian pada *controller* fungsi actionCreate dan actionUpdate perlu kita modifikasi untuk *render*-nya serta jika perlu kita bisa tambahkan *flash message*. Berikut ini contohnya.

```

public function actionCreate()
{
    $model = new Category();
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        Yii::$app->session->setFlash('success', 'Data berhasil disimpan');
        if(Yii::$app->request->isAjax) {
            $model = new Category();
            return $this->renderAjax('create', [
                'model' => $model,
            ]);
        }
        else{
            return $this->redirect(['view', 'id' => $model->id]);
        }
    } else {
        ...
    }
}

public function actionUpdate($id)
{
    $model = $this->findModel($id);
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        Yii::$app->session->setFlash('success', 'Data berhasil disimpan');
        if(Yii::$app->request->isAjax) {
            return $this->renderAjax('update', [
                'model' => $model,
            ]);
        }
        else{
            return $this->redirect(['view', 'id' => $model->id]);
        }
    } else {
        ...
    }
}

```

Karena menggunakan *flash message*, kita bisa gunakan *widget Alert* \app\widgets\Alert::widget(); yang telah kita buat pada bab sebelumnya. Sebenarnya *widget* ini telah kita pasang di *layout*, hanya saja pada *request* by Pjax, *layout* tidak ikut tersentuh sehingga *widget* tersebut pun tidak dijalankan. Oleh karena itu kita bisa tambahkan pada view _form.php di atas

```

Pjax::begin([
    'id'=>'pjax-form', 'timeout'=>false,
]);
?>
<?php
if(Yii::$app->request->isAjax)
    echo \app\widgets\Alert::widget();
?>
<?php $form = ActiveForm::begin([
    'options' => ['data-pjax' => true ]
]); ?>

```

Berikut ini hasilnya

Home / Categories

Update Category: Buku Tulis

Data berhasil disimpan

Title

Buku Tulis

Description

Buku buat tulis menulis

Update

Back

Kita bisa menambahkan fungsi untuk mereload gridview setelah proses submit. Caranya tambahkan kode Js berikut.

```
<?php
$this->registerJs('
    $("#pjax-form").on("pjax:end", function() {
        $.pjax.reload("#pjax-gridview",{
            "timeout": false,
            "url": ".\yiy\helpers\Url::to(['index'])",
            "replace": false,
        });
    });
');
```

Selesai.

Pjax pada Modal

Kita telah belajar bagaimana menggunakan Pjax untuk memuat suatu konten tanpa *refresh* halaman web. Nah agar lebih interaktif, kita bisa menampilkan *view*, formulir *create* dan *update* pada sebuah *modal* (Twitter Bootstrap).

Mari kita mulai dengan mempersiapkan *modal*-nya. Yii telah membuatkan kita *widget modal* [[yiy\bootstrap\Modal]] sehingga lebih mempercepat kerja kita.

Pada *view index.php* (@app/views/category/index.php) kita tambahkan *widget modal* pada bagian paling bawah *view index.php*, sebagai berikut.

```
<?php
Modal::begin([
    'id' => 'categoryModal',
]);
Pjax::begin([
    'id'=>'pjax-modal', 'timeout'=>false,
'enablePushState'=>false,
'enableReplaceState'=>false,
]);
Pjax::end();
Modal::end();
?>
```

Tentu saja kita harus menambahkan kode berikut diawal *script* dari *file index.php*

```
| use yiy\bootstrap\Modal;
```

Yang menjadi catatan bahwa pada kode di atas kita telah membuat *modal* dengan id *categoryModal*, serta kita juga menambahkan blok Pjax dengan id *pjax-modal*. Parameter *enablePushState* dan *enableReplaceState* diset *false* dimaksudkan untuk mencegah perubahan URL oleh HTML5 *pushState*, alasannya adalah karena halaman akan dimuat pada *modal* sehingga URL seharusnya tetap, namun hal ini tidak harus.

Modal ini tidak akan nampak pada tampilan web kecuali jika dipanggil.

Kemudian, pada Gridview *file index.php* tersebut atau tepatnya pada *actionColumn*, kita modifikasi tautan tombol *view* dan *update*. Yaitu dengan menambahkan parameter yang berfungsi men-trigger *modal*.

```
[[
    'class' => 'yiy\grid\ActionColumn',
```

```
'buttons' => [
    'view' => function ($url, $model) {
        $icon='<span class="glyphicon glyphicon-eye-open"></span>';
        return Html::a($icon,$url,[
            'data-toggle'=>"modal",
            'data-target'=>"#categoryModal",
        ]);
    },
    'update' => function ($url, $model) {
        $icon='<span class="glyphicon glyphicon-pencil"></span>';
        return Html::a($icon,$url,[
            'data-toggle'=>"modal",
            'data-target'=>"#categoryModal",
        ]);
    },
],
```

Kita juga perlu menambahkan juga parameter tersebut untuk tombol *Create Category*.

Mari sekarang kita *test*, dan bukan sulap bukan sihir, ketika kita klik tautan *view* atau *update* pada Gridview maka akan keluar *modal* dalam bentuk *popup*.



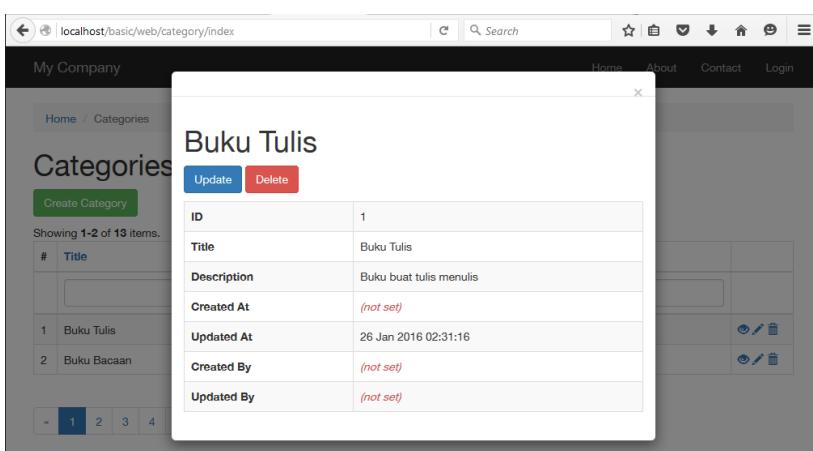
Sebuah *modal* akan muncul, namun *blank* alias kosong. Yap, kita perlu buatkan kode untuk mengisi konten yang ada di dalam modal tersebut. Berbekal informasi dari dokumentasi Twitter Bootstrap di getbootstrap.com, disana ada contoh kode yang akan dijalankan ketika *modal* tampil yaitu

```
$("#ID_MODAL").on("shown.bs.modal", function (event) {
    // ...
})
```

Kode di atas bisa kita sematkan di *view index.php* untuk me-*load* konten sesuai URL dari tautan (*view*, *update*, dan *create*) ke dalam *modal*.

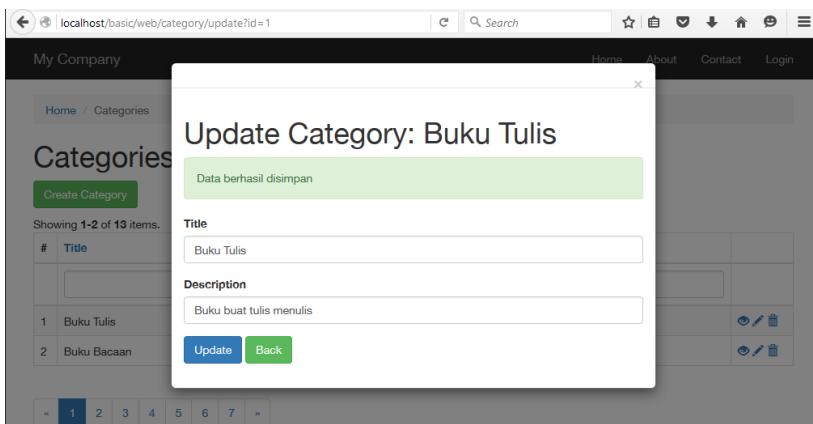
```
<?php
$this->registerJs('
...
$("#categoryModal").on("shown.bs.modal", function (event) {
    var button = $(event.relatedTarget)
    var href = button.attr("href")
    $.pjax.reload("#pjax-modal",{
        "timeout":false,
        "url": href,
        "replace": false,
    });
})
');
```

Kode *pjaxDelete* adalah kode yang sebelumnya, sehingga tidak perlu diubah. Yang perlu diubah cukup kode yang di bawah untuk menangani *modal*.



Wow, sebuah modal dengan gagah berani muncul beserta konten yang sesuai dengan tautan.

Demikian juga untuk tautan edit



Tombol *update*, juga berhasil di-submit pada *modal*. Hanya saja tombol *Back* pada *modal* menjadi tidak berfungsi. Oleh karena itu perlu kita ubah, menjadi tombol untuk menutup *modal*.

```
<?= Html::a('Close', ['index'], [
    'class' => 'btn btn-success',
    'onclick'=>'',
        $('#categoryModal').modal("hide");
        return false;
    '
]) ?>
```

Selesai dan silakan diuji coba.

Realtime Data View

Realtime data view adalah tampilan data yang berubah secara *realtime*. Ada banyak teknik yang dapat kita lakukan untuk membuat hal ini. Berikut ini beberapa diantaranya

Ajax Long Polling

Ajax *long polling* adalah salah satu metode yang bisa digunakan untuk membuat realtime data *view*. Metode ini menggunakan fungsi *setTimeout* yang menggunakan Ajax untuk *me-request* data dari server, dan jika sukses maka akan mengulangi hal ini lagi.

Berikut ini gambaran kodennya.

```
(function poll(){
    setTimeout(function(){
        $.ajax({ url: "server.php", success: function(data) {
            // implementasi kode di sini
            poll();
        }, dataType: "json"});
    }, 3000);
})();
```

Contoh implementasinya, misalnya kita ingin menampilkan data tabel *Gridview* yang harus selalu *updated*. Maka cara yang paling mudah adalah dengan mengecek secara berkala apakah ada data yang berubah. Jika ada data yang berubah maka *Gridview* akan di-refresh menggunakan Pjax, sehingga lebih efisien.

Berikut ini contoh *view* pada @app/category/index.php

```
<?php
use yii\grid\GridView;
use yii\widgets\Pjax;
use yii\helpers\Url;
?>
<?php Pjax::begin(['timeout'=>false, 'id'=>'gridview']); ?>
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ...
    ]
]); ?>
<?php Pjax::end() ?>
```

```

<?php
$this->registerJs('
    var currentData = "";
    var check = function(){
        setTimeout(function(){
            $.ajax({ url: "'.Url::to(['category/check']).'", success: function(data) {
                if(currentData!=data.lastId){
                    currentData = data.lastId;
                    $.pjax({
                        url:"'.Url::to(['category/index']).'",
                        container:"#gridview",
                        timeout:false,
                        replace: false,
                    }).done(function(data) {
                        check();
                    });
                }
                else{
                    check();
                }
            }, dataType: "json"});
        }, 5000);
    }
    check();
');

```

Adapun fungsi actionIndex pada CategoryController menggunakan fungsi renderAjax jika *request* dilakukan menggunakan Ajax

```

public function actionIndex()
{
    $searchModel = new CategorySearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
    if(Yii::$app->request->isAjax){
        return $this->renderAjax('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    else{
        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
}

```

Sedangkan berikut ini contoh fungsi actionCheck sederhana, untuk mendeteksi adanya perubahan data.

```

public function actionCheck()
{
    $category = Category::find()->select('id')->orderBy('id DESC')->one();
    Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
    return ['lastId'=>$category->id];
}

```

Idealnya kita deteksi perubahan data dari *field* yang menunjukkan waktu terakhir diperbarui.

HTML5 Server Sent Event (SSE)

HTML SSE adalah suatu teknologi dimana sebuah *web browser* menerima update otomatis dari server melalui koneksi HTTP. Tidak semua *web browser* mendukung SSE. <http://caniuse.com/#feat=eventsource>

Jika Ajax *Long Polling* menggunakan *engine* XMLHTTP / Ajax untuk me-*request* data ke server secara berkala, maka SSE menggunakan *web browser* untuk me-*request* data secara berkala.

Berikut ini contoh kode Javascript untuk mengimplementasikan HTML5 SSE

```

var source = new EventSource("server.php");
source.onmessage = function(event) {
    // implementasi Pjax
}

```

server.php adalah *file* PHP di server yang di-request secara berkala. Fungsi *onmessage* adalah fungsi yang dijalankan ketika SSE mendapat data dari server.

Sebagai contoh, kita akan mengkonversi kode pada Ajax *Long Polling* menjadi versi HTML SSE.

Fungsi actionCheck agak sedikit berbeda dengan versi sebelumnya.

```
public function actionCheck()
{
    header('Content-Type: text/event-stream');
    header('Cache-Control: no-cache');
    $category = Category::find()->select('id')->orderBy('id DESC')->one();
    echo 'data: {"lastId": "'.$category->id.'"}';
    echo "\n\n";
    flush();
}
```

Dan kode Javascript pada *view index.php* menjadi sebagai berikut.

```
<?php
$this->registerJs('
var currentData = "";
var source = new EventSource("'.Url::to(['category/check']).'");
source.onmessage = function(event) {
    var data = JSON.parse(event.data);
    if(currentData!=data.lastId){
        currentData = data.lastId;
        $.pjax({
            url:"'.Url::to(['category/index']).'",
            container:"#gridview",
            timeout:false,
            push: false,
        }).done(function(data) {
        });
    }
}, false);
');
```

D. Kesimpulan

Ajax bukanlah teknologi baru, namun masih menjadi pilihan hingga saat ini. Implementasinya pada Yii juga cukup flexible, dimana kita bisa menggunakan hasil keluaran dalam format Json sehingga lebih efisien. Sedangkan untuk urusan *reload* suatu bagian dari halaman, kita bisa mengandalkan Pjax. Kombinasi yang tepat pada dua metode ini akan membuat aplikasi kita lebih efisien dan responsif sebagai mana yang dicontohkan pada bagian “*Realtime Data View*”.

Pada bab selanjutkan kita akan belajar tentang *layout* aplikasi yaitu salah satunya belajar tentang bagaimana membuat tampilan *layout* yang berbeda untuk setiap *view* yang berbeda. Di samping itu juga belajar tentang bagaimana memecah aplikasi menjadi beberapa sub aplikasi menggunakan *module*.

Halaman ini sengaja dikosongkan!

8

Layout dan Module Aplikasi

Pada bab ini kita akan belajar tentang:

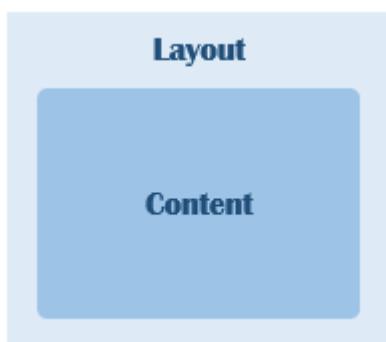
- Konsep dan implementasi *layout*
- Konsep dan implementasi *module*
- Pembuatan CRUD pada *module*

A. Bekerja dengan *Layout*

A. 1. *Definisi*

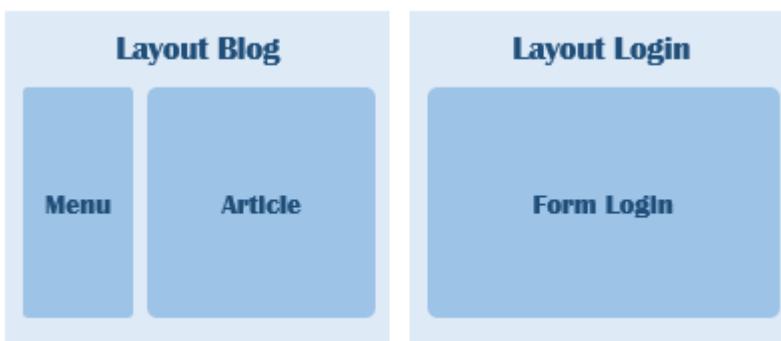
Layout merupakan bagian yang bertanggung jawab atas tampilan aplikasi Yii kita. Karena berkaitan dengan tampilan sehingga dalam MVC, *layout* termasuk *view*, yang mana kita bisa jumpai kode untuk *layout* di @views/layouts.

Sebenarnya pada bab-bab awal kita telah sedikit membahas tentang *layouts* dimana digambarkan sebagai sebuah pigura.



Pada sebuah halaman situs web yang akan sering berubah adalah pada bagian *content*, namun demikian *layout*-pun juga boleh berubah mengikuti halaman yang dibuka misalnya halaman *login*, *panel* administrasi, dsb.

Berikut ilustrasinya.



Pada contoh di atas, *layout blog* memiliki dua kolom yaitu menu dan *article*. Sedangkan *layout login* hanya memiliki satu kolom yaitu formulir *login*. Baik menu, *article* dan formulir *login* merupakan *content*.

Di Yii kita bisa menentukan jenis *layouts* melalui *controller* [[yii\web\Controller]], yang mana ada parameter *layout* yang bisa kita *set*, dan jika tidak kita *set* maka secara *default* akan merujuk ke *layout* main (@app/views/layouts/main.php).

Berikut ini contoh pengaturan *layout* pada *controller* yang akan berlaku umum pada *controller* tersebut.

```
| public $layout = 'login';
```

Sedangkan pada fungsi *action*, kita bisa menentukan *layout* yang hanya akan berlaku pada *action* tersebut saja.

```
| $this->layout = 'login';
```

A. 2. *Implementasi*

Mari kita coba praktikkan di Yii.

Layout Satu Kolom

Kita akan coba membuat *layout* satu kolom sederhana, yang akan kita gunakan untuk *layout* login.

- Buat file dengan nama login.php pada direktori @app/views/layouts/

```
<?php
use yii\helpers\Html;
use app\assets\AppAsset;
AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body>
<?php $this->beginBody() ?>

<div class="wrap">
    <div class="container">
        <?= $content ?>
    </div>
</div>

<?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

- Buat sebuah controller bernama TestController untuk menguji coba *layout* kita, @app/controllers/TestController.php

```
<?php
namespace app\controllers;
use Yii;
use yii\web\Controller;
class TestController extends Controller
{
    public $layout = 'login';

    public function actionLogin()
    {
        // render view login
        return $this->render('form-login');
    }
}
```

- Buat file view dummy saja, @app/views/test/form-login.php

```
<h1> Form Login </h1>
<p>
Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. L
orem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet.
</p>
```

- Mari kita uji coba, melalui URL /test/login



Form Login

Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet.
ipsum sit dolor amet.

Selesai

Layout Dua Kolom

Kita akan coba membuat *layout* dua kolom sederhana, yang akan kita gunakan untuk *layout blog*.

1. Buat file dengan nama blog.php pada direktori @app/views/layouts/

```
<?php
use yii\helpers\Html;
use app\assets\AppAsset;
AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body>
<?php $this->beginBody() ?>

<div class="wrap">
    <div class="container">
        <div class="row">
            <div class="col-md-2">
                <?= $this->blocks['sidebar']; ?>
            </div>
            <div class="col-md-10">
                <?= $content; ?>
            </div>
        </div>
    </div>
</div>

<?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

2. Tambahkan fungsi actionBlog pada TestController

```
public function actionBlog()
{
    // select layouts
    $this->layout = 'blog';
    // render view blog
    return $this->render('blog');
}
```

3. Buat file view dummy saja, @app/views/test/blog.php

```
<?php
use yii\widgets\Block;
?>
```

```
<?php Block::begin([
    'id'=>'sidebar'
]); ?>
<h1> Menu </h1>
<ul>
    <li>Home</li>
    <li>About</li>
    <li>Contact</li>
</ul>
<?php Block::end(); ?>

<h1> Article </h1>
<p> Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor ame
t. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet
. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet.
Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet.
Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. Lorem ipsum sit dolor amet. <
/p>
```

4. Mari kita uji coba, melalui URL /test/blog



Menu Article

- Home
- About
- Contact

Lore ipsum sit dolor amet. Lore ipsum sit dolor amet. Lc
ipsum sit dolor amet. Lore ipsum sit dolor amet. Lore ip:
dolor amet. Lore ipsum sit dolor amet. Lore ipsum sit dc
amet. Lore ipsum sit dolor amet. Lore ipsum sit dolor ar

Selesai, Seru kan? ☺

B. Bekerja dengan Module

B. 1. Definisi

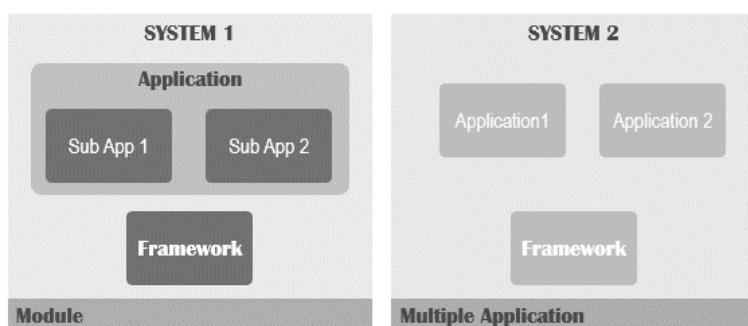
Module adalah sub aplikasi yang lengkap MVC-nya namun tidak bisa berdiri sendiri melainkan bergantung pada aplikasi utama. Satu aplikasi bisa mempunyai banyak sub aplikasi. Dengan adanya sub aplikasi maka pengembangan aplikasi bisa lebih *modular* yaitu bisa dipecah berdasarkan *module*-nya.

Misalnya pada sebuah sistem yang besar dengan tim yang besar, pembuatan aplikasi ERP. Maka kita bisa membagi aplikasi ERP menjadi beberapa *module* seperti: HRIS, Accounting, Payroll, dll. Dimana masing-masing *module* tersebut bisa dikerjakan secara paralel oleh tim yang berbeda.

Di samping modules, sebenarnya ada pendekatan lain yang juga ditawarkan oleh Yii untuk menangani banyak aplikasi yang saling terhubung. Yaitu sebagaimana yang dicontohkan pada *application template Advanced*, dimana ada aplikasi *frontend* dan aplikasi *backend* yang saling terhubung.

Perbedaan utama dari *module* adalah pada *template Advanced*, “sub aplikasinya” bisa berdiri sendiri.

Berikut ini gambaran sederhananya.



Secara *default* kode *module* akan diletakkan pada direktori @app/modules, dengan nama *class*-nya adalah *Module*

B. 2. Implementasi Module

Membuat Module

Pada panduan ini kita akan membuat *module* admin yang nanti akan digunakan untuk meletakkan CRUD-CRUD manajemen aplikasi kita. Untuk membuat *module*, kita bisa menggunakan *tools* Gii yang akan membantu kita meng-generate kode dasar dari *module*.

Berikut ini langkahnya:

1. Akses gii
 - ∞ <http://localhost/basic/web/gii>
2. Pilih “Module Generator”

Module Generator

This generator helps you to generate the skeleton c

Module Class

Module ID

3. Masukkan *namespace class Module* yang ingin kita generate yaitu app\modules\admin\Module, dengan *module* Id-nya **admin**.

Module Class

Module ID

4. Kemudian klik tombol *Preview*

Code Template

```
default (C:\Bitnam\wampstack\apache2\htdocs)
```

Preview

5. Ada 3 file yang akan di-generate yaitu

```
@app/modules/admin/Module.php  
@app/modules/admin/controllers/DefaultController.php  
@app/modules/admin/default/index.php
```

Klik tombol *Generate*

Preview Generate

Click on the above **Generate** button Create Unchanged Overwrite
to generate the files selected below:

Code File	Action	<input checked="" type="checkbox"/>
modules\admin\Module.php	create	<input checked="" type="checkbox"/>
modules\admin\controllers\DefaultController.php	create	<input checked="" type="checkbox"/>
modules\admin\views\default\index.php	create	<input checked="" type="checkbox"/>

Selesai

```
Generating code using template "C:\Bitnami\wampstack\apache2\htdocs\yii\generators\crud\templates\module"
generated modules\admin\Module.php
generated modules\admin\controllers\DefaultController.php
generated modules\admin\views\default\index.php
done!
```

The module has been generated successfully.

To access the module, you need to add this to your application configuration:

```
<?php
.....
'modules' => [
    'admin' => [
        'class' => 'app\modules\admin\Module',
    ],
],
.....
```

Sebagai sub aplikasi, maka kita perlu menambahkan *module* yang telah kita bikin ke konfigurasi aplikasi utama, sebagaimana yang dicontohkan di atas. @app/config/web.php

```
$config = [
'id' => 'basic',
'basePath' => dirname(__DIR__),
'bootstrap' => ['log'],
'timeZone' => 'Asia/Jakarta',
'modules' => [
    'admin' => [
        'class' => 'app\modules\admin\Module',
    ],
],
```

Setelah ditambahkan dikonfigurasi, maka sekarang kita bisa mengakses modules admin menggunakan URL sebagai berikut

∞ <http://localhost/basic/web/admin/default/index>

admin/default/index

This is the view content for action "index". The action belongs to the controller "app\modules\admin\controllers\DefaultController" in the "admin" module.

You may customize this page by editing the following file:

C:\Bitnami\wampstack\apache2\htdocs\basic\modules\admin\views\default\index.php

Pada direktori *module* admin atau @app/modules/admin, kita menjumpai dua *folder* yaitu *controllers* dan *views*, silakan tambahkan satu *folder* lagi yaitu model yang di dalamnya akan kita isi dengan model-model yang spesifik dengan *modules* admin.

Tambahkan *folder* model pada direktori @app/modules/admin

Sekarang kita bisa lihat bahwa *module* adalah sub aplikasi yang lengkap MVC-nya. Namun meskipun demikian modules tidak bisa berdiri sendiri sebagaimana aplikasi. Hal ini dikarenakan *entry point* dari *module* masih menginduk ke *entry point* aplikasi utama.

B. 3. Menambahkan CRUD pada Module

Setelah kita membuat *module* admin, maka kini saatnya kita menambahkan CRUD ke *module* admin tersebut. Sebagai contoh, kita akan menambahkan CRUD *Category* melalui Gii.

CRUD Generator

This generator generates a controller and views that implement a model.

Model Class

app\models\Category

Search Model Class

app\modules\admin\models\CategorySearch

Controller Class

app\modules\admin\controllers\CategoryController

View Path

@app/modules/admin/views/category

Arahkah semua kecuali model *class* ke direktori *modules* admin kita yaitu @app/modules/admin/

Enable I18N

Click on the above **Generate** button to generate the Create Unchanged Overwrite

Code File	Action	<input type="checkbox"/>
modules\admin\controllers\CategoryController.php	create	<input checked="" type="checkbox"/>
modules\admin\models\CategorySearch.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category_form.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category_search.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category\create.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category\index.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category\update.php	create	<input checked="" type="checkbox"/>
modules\admin\views\category\view.php	create	<input checked="" type="checkbox"/>

Jika sudah, klik tombol *Generate*

The code has been generated successfully.

```
Generating code using template "C:\Bitnami\wampstack\generated\modules\admin\controllers\CategoryController"
generated modules\admin\models\CategorySearch.php
generated modules\admin\views\category\_form.php
generated modules\admin\views\category\_search.php
generated modules\admin\views\category\create.php
generated modules\admin\views\category\index.php
generated modules\admin\views\category\update.php
generated modules\admin\views\category\view.php
done!
```

Saatnya uji coba, mari kita akses CRUD *Category* menggunakan URL berikut

∞ <http://localhost/basic/web/admin/category>

Hasilnya

localhost/basic/web/admin/category

My Company

Home / Categories

Categories

Create Category

Showing 1-13 of 13 items.

#	ID	Title	Description
1	1	Buku Tulis	Buku buat tulis menulis

Selesai. Lanjutkan untuk CRUD-CRUD yang lain.

C. Kesimpulan

Pada bab selanjutnya kita akan belajar tentang *extension*.

9

Bekerja dengan *Extension*

Pada bab ini kita akan belajar tentang:

- Definisi *extension* di Yii
- Cara menggunakan *extension*
- Cara membuat *extension* hingga *publish*

A. Memahami *Extension*

Jika diterjemahkan secara harfiah, maka *extension* adalah tambahan atau perpanjangan. Namun yang dimaksudkan di sini adalah paket kode terpisah yang didesain untuk digunakan pada aplikasi Yii dan menyediakan fitur-fitur yang siap digunakan.

Kata kuncinya adalah terpisah, artinya kode ini diluar kode utama atau *core* dari *framework* Yii. Pada saat kita melakukan instalasi paket *framework* Yii, maka isi paket selain *core* Yii juga ada beberapa *extension* yang disertakan yaitu:

- yiisoft/yii2-bootstrap (*widget* Twitter Boostrap)
- yiisoft/yii2-swiftmailer (*helper* untuk kirim email melalui SMTP)
- yiisoft/yii2-codeception (*helper* untuk uji coba)
- yiisoft/yii2-debug (*module* untuk *debug*)
- yiisoft/yii2-gii (*module* untuk *generate* kode)
- yiisoft/yii2-faker (*helper* untuk membuat data *dummy*)

Kita bisa lihat bahwa *extension* itu hanya istilah umum dimana bentuknya bisa berupa *widget*, *components*, *module* (sub aplikasi), *helpers* (fungsi tertentu), dsb.

Kode *extension* bersifat *reusable* atau dapat digunakan kembali jika dibutuhkan. Dengan menggunakan *extension* tentu bisa mempercepat pemrogram aplikasi dalam mengembangkan aplikasinya. Ada banyak *extension* yang mendukung Yii dengan berbagai fungsinya. Kita bisa jumpai di packagist.org atau di *official site* Yii.

Catatan

Istilah "*extension*" yang digunakan di sini adalah paket *software* yang mengacu spesifik pada Yii. Sedangkan paket *software* lain yang bisa digunakan tanpa Yii maka akan digunakan istilah "*package*" atau "*library*".

B. Menggunakan Extension

Sebelum menggunakan *extension*, kita perlu menginstalnya terlebih dahulu. Umumnya, *extension* bisa kita instalasi menggunakan Composer sehingga lebih mudah.

Agar lebih mudah memahaminya, mari kita instalasi beberapa *extension* yang dibutuhkan untuk pengembangan aplikasi studi kasus kita. Semua *extension* ini kita instalasi menggunakan Composer:

B. 1. kartik/yii2-widgets

Definisi

kartik/yii2-widgets adalah kumpulan *widget-widget* penting untuk mempercantik tampilan antarmuka, yaitu: datepicker, timepicker, select2, depdrop, dll

∞ <https://github.com/kartik-v/yii2-widgets>

Instalasi

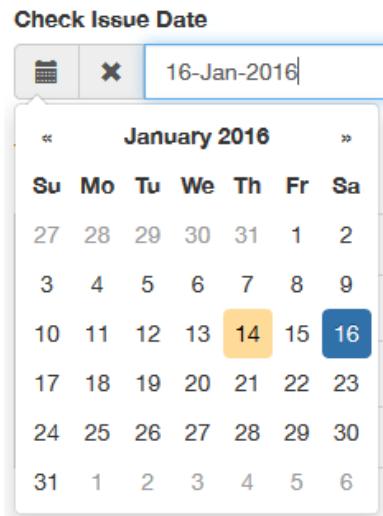
Pada CMD jalankan perintah berikut

```
| composer require kartik-v/yii2-widgets "*"
```

Penggunaan

Karena merupakan kumpulan *widget* sehingga tergantung *widget*-nya. Berikut ini contoh untuk widget DatePicker.

```
use kartik\widgets\DatePicker;
// usage without model
echo '<label>Check Issue Date</label>';
echo DatePicker::widget([
    'name' => 'check_issue_date',
    'value' => date('d-M-Y', strtotime('+2 days')),
    'options' => ['placeholder' => 'Select issue date ...'],
    'pluginOptions' => [
        'format' => 'dd-M-yyyy',
        'todayHighlight' => true
    ]
]);
```



B. 2. yiisoft/yii2-imagine

Definisi

yiisoft/yii2-imagine adalah *extension* yang merupakan *wrapper* dari *library Imagine*, yaitu *library* untuk manipulasi *image*. *Extension* ini merupakan *extension* yang dikembangkan dan didukung oleh Yii.

Instalasi

Pada CMD jalankan perintah berikut

```
| composer require --prefer-dist yiisoft/yii2-imagine "*"
```

Implementasi

Contoh berikut adalah kode untuk merotasi *image*.

```
use yii\imagine\Image;
// frame, rotate and save an image
Image::frame('path/to/image.jpg', 5, '666', 0)
    ->rotate(-8)
    ->save('path/to/destination/image.jpg', ['quality' => 50]);
```

Pada studi kasus, kita menggunakan *extension* ini untuk meng-*crop* foto profil, membuat *watermark* produk, membuat *thumbnail* produk.

B. 3. vova07/yii2-imperavi-widget

Definisi

Extension ini adalah sebuah *wrapper* untuk libari Imperavi Redactor (<https://imperavi.com/redactor/>), *high quality WYSIWYG editor*.

Sebagai informasi, *library* Imperavi Redactor sendiri adalah *library* berbayar namun sejak komunitas Yii membeli lisensi OEM maka kita bisa menggunakannya secara gratis dengan syarat aplikasi kita *base on* Yii.

Instalasi

Pada CMD jalankan perintah berikut

```
| Composer require --prefer-dist vova07/yii2-imperavi-widget "*"
```

Perhatian:

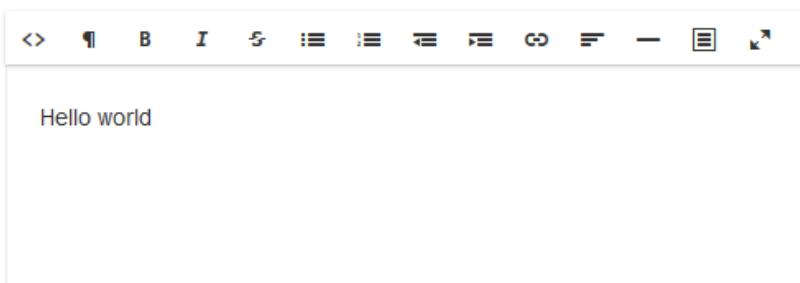
Versi Redactor yang *free* (OEM) adalah versi yang ada pada *extension* di atas. Adapun versi yang terdapat pada situs web resmi Redactor tetap berbayar.

Implementasi

Pada studi kasus kita, *extension* ini akan kita gunakan untuk tampilan *editor* deskripsi produk.

Berikut ini contoh implementasinya

```
use vova07\imperavi\Widget;
echo Widget::widget([
    'name' => 'redactor',
    'settings' => [
        'lang' => 'id',
        'minHeight' => 200,
        'plugins' => [
            'clips',
            'fullscreen'
        ]
    ],
]);
```



B. 4. hscstudio/yii2-chart

Definisi

Extension ini merupakan *wrapper* dari ChartNewJs. Yaitu *library* untuk meng-generate tampilan *chart* atau *diagram*.

```
∞ https://github.com/hscstudio/yii2-chart
```

Instalasi

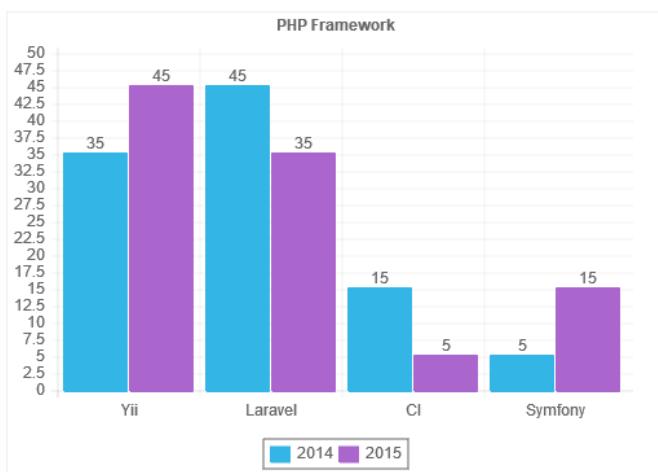
Pada CMD jalankan perintah berikut

```
| composer require --prefer-dist yiisoft/yii2-chart "~1.0"
```

Implementasi

Berikut ini contoh implementasinya.

```
use hscstudio\chart\ChartNew;
echo ChartNew::widget([
    'type'=>'bar', # pie, doughnut, line, bar, horizontalBar, radar, polar, stackedBar, polarArea
    'title'=>'PHP Framework',
    'labels'=>['Yii','Laravel','CI','Symfony'],
    'datasets' => [
        ['title'=>'2014','data'=>[35,45,15,5]],
        ['title'=>'2015','data'=>[45,35,5,15]],
    ],
]);
```



C. Membuat Extension

Setelah mengetahui apa itu *extension*, serta bagaimana menggunakannya, maka kini sudah waktunya penulis menurunkan ilmu kepada anda yaitu membuat sendiri *extension*.

Pada bab sebelumnya, sebenarnya kita telah membuat *extension* yaitu *widget Alert*, namun pada bagian ini kita fokus agar *extension* itu bisa kita instalasi menggunakan Composer sehingga dapat digunakan oleh orang lain di seluruh dunia. Wow hebat bukan? ☺

C. 1. Persiapan Kode Dasar

Sekali lagi kita akan gunakan *tools Gii* untuk membuat kode dasar *extension* kita.

1. Akses Gii

∞ <http://127.0.0.1/basic/web/gii>

2. Pilih *Extension Generator*

Extension Generator

This generator helps you to generate the files needed by
a Yii extension.

[Start »](#)

3. Isi formulir tersebut

- *Vendor name* berisi *username* Github kita,
- *Package name* adalah nama dari *extension* kita, direkomendasikan diawali dengan *yii2-*
- *Namespace* atau alias, sebenarnya bebas namun standardnya adalah *username* Github*nama extension*\

Info:

hscstudio adalah *username* Github milik penulis

Extension Generator

This generator helps you to generate the files needed by a Yii extension.

Please read the Extension Guidelines before creating an extension.

Vendor Name

hscstudio

Package Name

yii2-alert

Namespace

hscstudio\alert\

Type

yii2-extension

Keywords

yii2,extension>alert

License

MIT

Title

yii2-alert

Description

Widget twitter bootstrap for Yii framework 2.0

Author Name

Hafid Mukhlasin

Author Email

hafidmukhlasin@gmail.com

Output Path

@app/runtime/tmp-extensions

Code Template

default (C:\Bitnami\wampstack\apache2\htdocs\basic\vendor\yii

Preview

4. Klik tombol *Preview*

Preview
Generate

Click on the above **Generate** button to generate the files selected below:

Code File	Action	<input checked="" type="checkbox"/>
runtime\tmp-extensions\yii2-alert\composer.json	create	<input checked="" type="checkbox"/>
runtime\tmp-extensions\yii2-alert\AutoloadExample.php	create	<input checked="" type="checkbox"/>
runtime\tmp-extensions\yii2-alert\README.md	create	<input checked="" type="checkbox"/>

5. Klik tombol *Generate*, maka *extension* akan ter-*generate* di

```
@app/runtime/tmp-extensions
```

Pada layar kita juga jumpai panduan untuk mengunggah *extension* kita, namun kita tahan dulu.

Selesai

C. 2. Koding Extension

Setelah kode dasar jadi, kini saatnya kita coding implementasi dari *extension* yang akan kita buat, yang dalam hal ini adalah *widget Alert*. Hasil *generate* Gii ada tiga *file* yaitu composer.json, AutoloadExample.php serta README.md.

Nah, yang perlu kita contoh adalah *file*

```
@app/runtime/tmp-extensions /yii2-alert/AutoloadExample.php
```

```
<?php
namespace hscstudio\alert;
/**
 * This is just an example.
 */
class AutoloadExample extends \yii\base\Widget
{
    public function run()
    {
        return "Hello!";
    }
}
```

Kita akan mengkonversi *widget Alert* yang telah kita buat ke dalam format di atas. Caranya, silakan salin *file widget Alert @app/widgets/Alert.php* ke dalam direktori *@app/runtime/tmp-extensions/yii2-alert/Alert.php*, lalu sesuaikan sebagai berikut

```
<?php
namespace hscstudio\alert;
use Yii;
class Alert extends \yii\bootstrap\Widget
{
    public function init()
    {
        parent::init();
        if (Yii::$app->session->hasFlash('success')) {
            echo '<div class="alert alert-success">';
            echo Yii::$app->session->setFlash('success');
            echo '</div>';
        }
        else if (Yii::$app->session->hasFlash('error')) {
            echo '<div class="alert alert-danger">';
            echo Yii::$app->session->setFlash('error');
            echo '</div>';
        }
    }
}
```

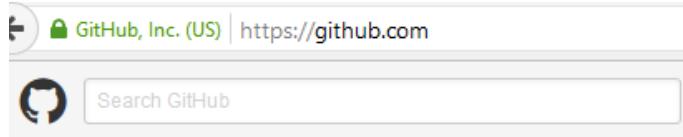
Ya, cukup dengan mengubah *namespace*-nya. Jika sudah maka *file AutoloadExample* bisa kita hapus.

C. 3. Mengunggah Extension

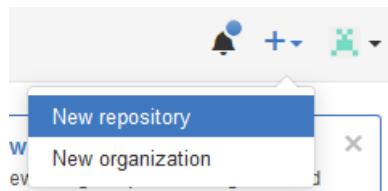
Sebelum kita unggah *extension* kita ke Github, maka kita perlu *create repository* (repo) baru di Github

Buat Repo di Github

- Caranya, *login* ke <https://github.com>



- Klik tanda + di sebelah kanan, dan pilih *New repository*



- Isi formulir *create repo*, sebagai berikut

Create a new repository

A repository contains all the files for your project, including the r

Owner	Repository name
hscstudio	/ <input type="text" value="yii2-alert"/>
Great repository names are short and memorable. Need inspiration?	
Description (optional)	
widget Twitter Bootstrap for Yii Framework 2.0	
<input checked="" type="radio"/> Public Anyone can see this repository. You choose who can commit.	
<input type="radio"/> Private You choose who can see and commit to this repository.	
<input type="checkbox"/> Initialize this repository with a README This will let you immediately clone the repository to your computer. Skip	
Add .gitignore: <input type="button" value="None"/>	Add a license: <input type="button" value="None"/>
Create repository	

Catatan:

hscstudio itu *username* dari *repository* Github penulis, silakan disesuaikan dengan repo anda.

- Lalu klik tombol *Create repository*

Jika sudah kita akan dibawa ke halaman

∞ <https://github.com/hscstudio/yii2-alert>

Inilah alamat URL dari repo kita, sekarang dunia sudah bisa melihatnya ☺. Di halaman itu juga ada panduan tentang bagaimana caranya unggah *file* dari repo lokal ke Github. Yang perlu kita catat adalah alamat URL gitnya sebagai berikut.

∞ <https://github.com/hscstudio/yii2-alert.git>

Unggah Repo Local ke Github

Sekarang saatnya kita unggah *extension* kita ke Github, pada CMD

1. Jalankan perintah berikut:

```
| cd C:\xampp\htdocs\basic\runtime\tmp-extensions\yii2-alert
| git init
```

Perintah di atas adalah perintah untuk menginisialisasi atau membuat repo git di lokal komputer kita pada *folder* direktori @app/runtime/tmp-extensions/yii2-alert atau tergantung dimana *file* di-generate saat menggunakan Gii.

2. Kemudian daftarkan (*commit*) ekstensi *file* kita ke repo Git lokal kita. Jalankan perintah berikut

```
| git add -A
| git commit -m "init"
```

3. Hubungkan repo lokal dengan Github

```
| git remote add origin https://github.com/hscstudio/yii2-alert.git
```

4. Unggah repo lokal ke Github

```
| git push -u origin master
```

maka kita akan diminta *username* dan kata sandi Github

```
C:\Windows\system32\cmd.exe
C:\Bitnami\wampstack\apache2\htdocs\basic\runtime\tmp-extensions\y
push -u origin master
Username for 'https://github.com': hscstudio
Password for 'https://hscstudio@github.com':
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.05 KiB | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/hscstudio/yii2-alert
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Selesai.

Bukan sulap bukan sihir, sekarang ekstensi *file* kita telah muncul di Github.

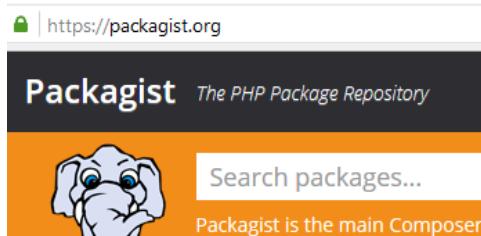


Hubungkan Github ke Packagist

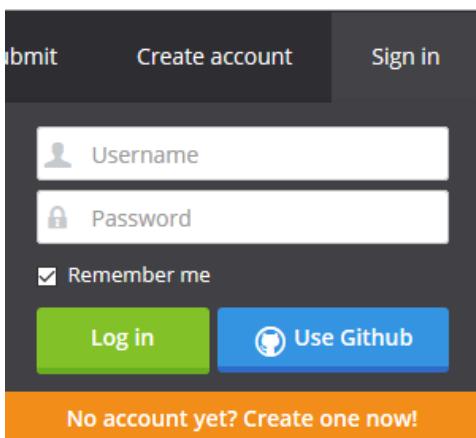
Agar *extension* kita bisa diinstalasi dengan mudah menggunakan Composer maka kita perlu hubungkan antara akun repo dari *extension* kita di Github dengan Packagist.

Caranya:

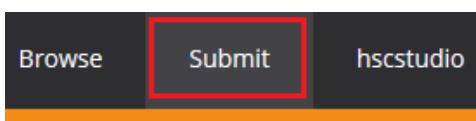
1. Akses packagist.org



2. Silakan *login*, gunakan “Use Github”



3. Setelah *login*, klik menu *Submit* di panel atas



4. Pada formulir *submit package*, masukkan alamat URL *extension* kita sebagai berikut

Submit package

Repository URL (Git/Svn/Hg)

Check

Klik tombol *Check*, wah ternyata Packagist mendeteksi bahwa ada paket yang namanya mirip dengan nama *extension* kita. Klik tombol *Submit*.

Submit package

Repository URL (Git/Svn/Hg)

Notice: One or more similarly named packages have already been submitted to Packagist. If this is a fork read the notice above regarding VCS Repositories.

Similarly named packages:
[voskobovich/yii2-alert](#)

The package name found for your repository is: hscstudio/yii2-alert.
 press Submit to confirm.

Submit

Hore.. berhasil

★ hscstudio/yii2-alert

composer require hscstudio/yii2-alert

This package is not auto-updated. Please set up the [GitHub Service Hook](#) for Packagist so that it gets updated whenever you push!

Widget twitter bootstrap for Yii framework 2.0

Abandon **Delete** **Update** **Edit**

Namun kalau kita perhatikan bahwa paket yang telah kita buat belum *auto update*, artinya ketika *extension* di Github diperbaharui maka paket kita tidak serta merta terbarukan.

Oleh karena itu kita perlu membuatnya *auto update*. Berikut caranya.

1. Akses URL ini

∞ <https://packagist.org/profile/>

2. Klik *Show API Token*

Your API Token

Show API Token

You can use your API token to interact with the Packagist API.

3. Salin *token* yang muncul
4. Buka halaman *extension* kita di Github, melalui URL

∞ <https://github.com/hscstudio/yii2-alert>

sesuaikan dengan URL *extension* Github anda.

5. Klik tab *Settings*

hscstudio / yii2-alert

Code Issues Pull requests Wiki Pulse Graphs Settings

6. Pilih menu *Webhooks & services*, lalu pilih *Add service*

hscstudio / yii2-alert

Code Issues P

Options Collaborators Branches Webhooks & services Deploy keys

7. Pada tombol menu *Add service*, cari Packagist, lalu klik Packagist

Add service ▾

Available Services

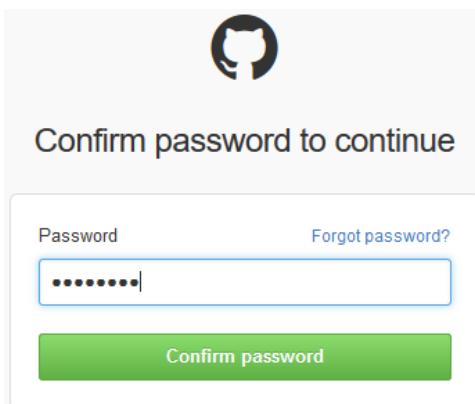
pac

Packagist

PythonPackages

DjangoPackages

8. Github meminta konfirmasi kata sandi akun Github kita



9. Masukkan *token* yang kita peroleh dari Packagist, kemudian klik *Add service*

User

Token

Domain

Active
We will run this service when an event is

Add service

10. Jika sudah, klik Packagist

Services

 **GitHub integrations directory**
Everything you need to build software is now highest quality integrations made to help you

Services are pre-built integrations that perform cert services check out our [service hooks guide](#).

Packagist

11. Klik uji coba *service* dan selesai.

Services / **Manage Packagist** Test service

Okay, the test payload is on its way. X

C. 4. Uji coba Extension

Mari kita coba instalasi *extension* kita menggunakan perintah berikut

```
| composer require --prefer-dist hscstudio/yii2-alert " "*
```

Namun apa yang kita dapat? Ternyata terjadi galat yang intinya paket *extension* kita tidak ditemukan. Sebagai berikut:

```
C:\Bitnami\wampstack\apache2\htdocs\basic>composer require --prefer-dist hscstudio/yii2-alert "*"
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Your requirements could not be resolved to an installable set of packages.

  Problem 1
    - The requested package hscstudio/yii2-alert could not be found in any version, there may be a typo in the package name.

  Potential causes:
    - A typo in the package name
    - The package is not available in a stable-enough version according to your minimum-stability setting
      see <https://groups.google.com/d/topic/composer-dev/_g3ASeIFrc/discussion> for more details.

  Read <https://getcomposer.org/doc/articles/troubleshooting.md> for further common problems.

Installation failed, reverting ./composer.json to its original content.

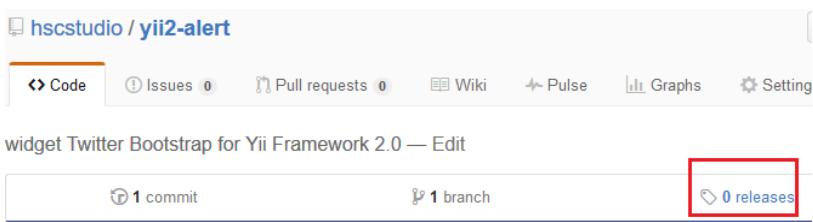
C:\Bitnami\wampstack\apache2\htdocs\basic>
```

Tenang, jangan panik. Hal ini terjadi karena kita belum *me-release extension* kita di Github, oleh karena itu untuk menginstalnya maka kita perlu ubah tanda * menjadi dev-master

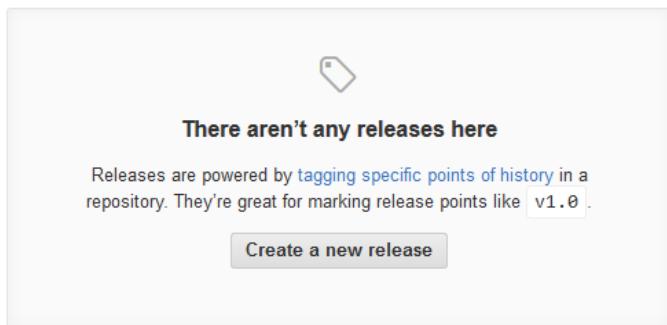
| composer require --prefer-dist hscstudio/yii2-alert "dev-master"

Sebagai penanda bahwa kita ingin melakukan instalasi versi *development*. Namun tentu ini bukan pilihan yang baik terutama bagi calon pengguna *extension* kita nanti ☺. Karena itu kita *release* saja. Caranya,

1. Klik menu *release*



2. Kemudian klik tombol “Create a new release”



2. Pada formulir *release* masukkan sebagai berikut

3. Klik *Publish release*

This is a pre-release
We'll point out that this release is ident

Publish release

Save draft

Selesai

Latest release

 v1.0.0
 c47ea06

Initial
hscstudio released this 2 hours ago
Release inisial

Downloads

 [Source code \(zip\)](#)
 [Source code \(tar.gz\)](#)

Hore.. akhirnya *extension* pertama kita telah berhasil kita buat.

D. Kesimpulan

Di Packagist, kita bisa jumpai banyak *extension* Yii yang bisa kita gunakan untuk aplikasi kita. Menggunakan *extension-extension* tersebut tentu bisa mempercepat waktu kita dalam mengembangkan aplikasi, namun meskipun demikian kita harus jeli untuk menggunakannya.

Gunakan *extension* yang memang dibutuhkan dan terpercaya. Kita bisa melihat statistik di Github dan membaca *issue-issue* yang muncul, agar penggunaan *extension* tersebut tidak menyebabkan masalah pada aplikasi kita dikemudian hari.

Pada bab selanjutnya kita akan belajar tentang konsep dan implementasi *code testing* yaitu uji coba otomatis untuk memastikan aplikasi kita berjalan sesuai dengan ekspektasi dan fokus pengembangan aplikasi jelas.

Halaman ini sengaja dikosongkan!

10

Code Testing

Pada bab ini kita akan belajar tentang:

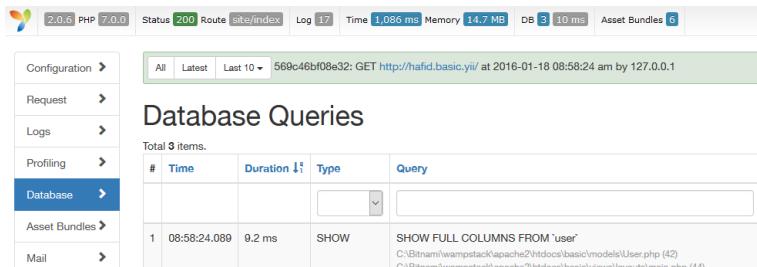
- Apa itu *code testing*
- Cara melakukan uji coba
- Cara membuat *code testing*
- Cara membuat data untuk uji coba

A. Apa itu *Code Testing*?

A. 1. *Definisi*

Uji coba aplikasi merupakan bagian yang sangat penting dalam pengembangan sebuah perangkat lunak. Suka atau tidak suka, sebenarnya setiap saat ketika sedang mengembangkan sebuah perangkat lunak maka kita melakukan uji coba tersebut.

Sebagai contoh, setelah kita menulis kode program (PHP) maka kita melakukan uji coba melalui *web browser*, kadangkala kita gunakan fungsi `die` atau `var_dump` untuk mengecek hasil keluaran dari suatu variabel. Atau kadang juga kita menggunakan *debug toolbar* (*panel* bawah aplikasi Yii) untuk sekedar melihat apakah *query* yang di-*generate* oleh Yii seperti yang kita inginkan. Atau terkadang juga, kita uji coba dengan meng-*input* data ke formulir dan melihat hasilnya di basis data, dsb



The screenshot shows the Yii Framework's debug toolbar interface. At the top, it displays system information: 2.0.6 PHP 7.0.0, Status 200, Route site/index, Log 17, Time 1,086 ms, Memory 14.7 MB, DB 3 10 ms, Asset Bundles 6. Below this is a navigation menu with links for Configuration, Request, Logs, Profiling, Database (which is currently selected), Asset Bundles, and Mail. A green header bar indicates the current route is site/index and the log count is 17. The main area is titled "Database Queries" and shows a table with one item. The table has columns: #, Time, Duration, Type, and Query. The single entry is: # 1, Time 08:58:24.089, Duration 9.2 ms, Type SHOW, and Query: SHOW FULL COLUMNS FROM `user` C:\Bitnami\wampstack\apache2\htdocs\basic\models\User.php (42) C:\Bitnami\wampstack\apache2\htdocs\basic\views\layouts\main.php (44)

Apapun itu, yang pasti kita senantiasa melakukan uji coba, yang tanpa kita sadari bahwa hal itu dilakukan secara berulang-ulang.

Lantas, mengapa ada topik ini?

Pada bagian ini kita akan belajar untuk membuat *code testing* otomatis, sehingga untuk melakukan uji coba aplikasi maka cukup dengan menjalankan sedikit perintah dan semua uji coba akan berjalan dengan sendirinya. Adapun setelah semua selesai diuji maka hasilnya akan ditampilkan pada kita.

A. 2. Kapan Melakukan *Code Testing*?

Dalam pengembangan perangkat lunak ada istilah *Test-Driven Development* (TDD) dan *Behavior-Driven Development* (BDD) yaitu pendekatan pengembangan perangkat lunak dengan menggambarkan perilaku dari bagian program atau seluruh fitur sebagai seperangkat skenario atau tes sebelum menulis kode program, baru kemudian mempersiapkan segala sesuatu (termasuk menulis kode program) untuk memastikan tes ini berhasil sesuai dengan yang diinginkan.

Sebagai contoh untuk membangun sebuah fitur, maka mengikuti langkah-langkah berikut:

1. Membuat *code testing* baru yang menggambarkan fitur tersebut.
2. Jalankan *code testing* tersebut
3. Tulis kode program sederhana yang penting lulus uji coba
4. Jalankan semua uji coba dan pastikan semua lulus.
5. Perbaiki kode program dan pastikan lulus uji coba juga.

Setelah langkah itu dilakukan makan diulang lagi untuk fitur berikutnya. Jika fitur berubah maka *code testing* juga harus diubah.

Bagaimana? Kelihatan tidak masuk akal? Buang-buang waktu?

Alasan untuk membuat *code testing* sebelum implementasi adalah bahwa hal itu akan membuat kita fokus pada apa yang ingin kita capai.

Pendekatan yang telah dijelaskan di atas sangat cocok untuk mengembangkan aplikasi jangan panjang dengan proyek aplikasi yang kompleks, namun tentu saja berlebihan jika kita hanya ingin membuat aplikasi yang sederhana.

Jadi kapan kita membuat *code testing*? Jangan dijawab dulu, namun mari kita belajar bagaimana sebenarnya *code testing* itu.

B. Tools untuk *Code Testing*

B. 1. *Selayang Pandang*

Untuk membuat *code testing* yang otomatis (yang otomatis itu uji cobanya, bukan membuat *code testing*-nya ☺) maka kita perlu menggunakan *tools*. Sebenarnya ada banyak *tools* yang bisa digunakan untuk melakukan uji coba, namun yang akan digunakan di sini adalah Codeception. Hal ini karena Yii telah terintegrasi dengan Codeception dan didukung secara resmi oleh Yii.

Ada tiga jenis uji coba yang bisa dilakukan dengan menggunakan *tools* ini, yaitu *unit testing*, *functional testing*, dan *acceptance testing*.

Unit Testing

Untuk memverifikasi bahwa suatu kode aplikasi berjalan sesuai dengan yang diharapkan. Codeception menggunakan PHPUnit sebagai *backend* untuk menjalankan *tests*.

- Kelebihan
 1. Relatif lebih cepat dibandingkan dengan jenis uji coba lain
 2. Melingkupi fitur yang jarang digunakan
 3. Dapat menguji stabilitas dari *core* aplikasi
 4. Anda cukup dianggap pemrogram aplikasi yang baik jika menulis *unit testing*
- Kekurangan
 1. Tidak menguji hubungan antar kode
 2. Kurang stabil karena sangat sensitif terhadap perubahan kode

Functional Testing

Untuk memverifikasi skenario dari perspektif bisnis proses menggunakan *browser emulation*.

- Kelebihan
 1. Kecepatan sedang, diantara *unit* dan *acceptance test*
 2. Dapat memberikan informasi uji coba yang lebih rinci
 3. Dapat menunjukkan kode baik kepada *manager* maupun klien
 4. Cukup stabil, karena hanya perubahan kode utama atau perubahan *framework* yang bisa membuatnya bermasalah.
- Kekurangan

Tidak dapat menguji Javascript dan Ajax

Acceptance Testing

Untuk memverifikasi skenario dari perspektif pengguna menggunakan *web browser*.

- Kelebihan
 1. Dapat dijalankan pada setiap situs web
 2. Dapat menguji coba ajax dan Javascript
 3. Dapat ditampilkan baik kepada *manager* maupun klien
 4. Sangat stabil karena perubahan kode dan teknologi tidak akan berpengaruh secara signifikan
- Kekurangan
 1. Lebih lambat karena menjalankan *web browser*
 2. Jumlah uji coba yang lebih sedikit
 3. Tidak stabil dalam menjalankan uji coba karena berkaitan dengan Javascript yang bisa jadi mengeluarkan hasil yang tak terduga.

Yii sebenarnya telah memberikan kita panduan yang bisa anda jumpai di `@app/tests/README.md`, itu artinya mengikuti panduan disana atau mengikuti panduan di buku ini sebenarnya sama saja ☺.

B. 2. Instalasi Codeception

Untuk melakukan uji coba maka kita perlu instalasi *tools* Codeception. Instalasi bisa dilakukan dengan menggunakan Composer, sebagai berikut.

```
| composer global require "codeception/codeception=2.0.*"
composer global require "codeception/specify=*>
composer global require "codeception/verify=*>
```

Setelah itu jalankan perintah berikut

```
| composer global status
```

```
C:\Bitnami\wampstack\apache2\htdocs\basic>composer global status
Changed current directory to C:/Users/hafid/AppData/Roaming/Composer
No local changes
```

Tentu akan sedikit berbeda dengan yang muncul di komputer anda.

Setelah itu, daftarkan `<directory>/vendor/bin` atau dalam contoh di komputer penulis adalah

```
C:/Users/hafid/AppData/Roaming/Composer/vendor/bin
```

ke variabel *environment* Path di OS. Lihat Bab sebelumnya tentang “Memahami Composer” untuk mengetahui cara mendaftarkan suatu direktori ke Path.

Tujuannya adalah agar perintah `codecept` bisa dijalankan dari lokasi manapun di CMD kita.

```
C:\WINDOWS\system32\cmd.exe
c:\Bitnami\wampstack\apache2\htdocs\basic\tests>
codecept
Codeception version 2.0.14
```

Kemudian *tools* yang kita perlukan adalah *Faker*, digunakan untuk meng-generate data *dummy*. Instalasinya bisa menggunakan perintah berikut

```
| composer require --dev yiisoft/yii2-faker:*
```

Hanya saja, secara *default* *Faker* adalah paket yang disertakan dalam *template basic*, sehingga kita tidak perlu menginstalnya lagi.

B. 3. Menjalankan Uji Coba

Oleh karena Yii telah membuatkan kepada kita contoh dari *code testing*, maka kita bisa mencobanya sebelum membuat kodennya sendiri.

Lalu apa yang di uji coba oleh Yii menggunakan contoh dari *code testing*?

Yang di uji coba adalah *application template* yaitu fitur-fitur bawaan Yii seperti halaman *about*, *contact* dan *login*. Dengan contoh tersebut, diharapkan bisa memberikan gambaran tentang bagaimana seharusnya *code testing* tersebut dibuat.

Membuat Basis Data untuk Uji Coba

Sebelumnya kita perlu membuat basis data untuk uji coba, misal `yii2_basic_tests`, kemudian kita konfigurasi koneksi basis data untuk uji coba ini pada *file* berikut.

```
@app/tests/codeception/config/config.php
```

```
'db' => [
    'dsn' => 'mysql:host=localhost;dbname=yii2_basic_tests',
    'username' => 'root',
    'password' => ' kata sandi basis data mu',
    'charset' => 'utf8',
],
```

Jika posisi *current* di *root* direktori (`C:\xampp\htdocs\basic`) maka pindah dulu ke direktori *tests* yaitu dengan menjalankan perintah berikut.

```
| cd tests
```

Adapun perintah untuk migrasi data sebagai berikut.

```
| codeception\bin\yii migrate
```

Untuk OS Linux jalankan `codeception/bin/yii migrate`

Perintah di atas akan menjalankan *file* migrasi di direktori `@app/migrations`, hanya jika ada.

```
c:\Bitnami\wampstack\apache2\htdocs\basic\tests>codeception\bin\yii migrate
Yii Migration Tool (base on Yii v2.0.9)

Total 1 new migration to be applied:
    m160118_041129_create_employee_table
```

Tambahkan konfigurasi untuk mengaktifkan fitur pretty URL, dengan mengedit *file* `@app/tests/codeception/config/config.php`

```
'urlManager' => [
    'enablePrettyUrl' => true,
    'showScriptName' => false,
],
```

Kemudian pastikan bahwa pada direktori `@app/web` terdapat *file* `.htaccess` (jika menggunakan *web server* apache), sesuaikan kode pada *file* `.htaccess` sebagai berikut:

```
# use mod_rewrite for pretty URL support
RewriteEngine on
# If a directory or a file exists, use the request directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# Otherwise forward the request to index.php
RewriteRule . index-test.php
```

Silahkan menyesuaikan jika anda menggunakan *web server* nginx, yang pada intinya merubah *entry point* dari `index.php` menjadi `index-test.php`

Build Test Suite

Fungsinya adalah untuk meng-generate ulang `class -class` uji coba sesuai dengan konfigurasi pada *file* `yml`.

Jalankan perintah berikut

```
| codecept build
```

```
c:\Bitnami\wampstack\apache2\htdocs\basic\tests>codecept build
Building Actor classes for suites: acceptance, functional, unit
\AcceptanceTester includes modules: PhpBrowser, WebDriver
AcceptanceTester.php generated successfully. 51 methods added
\FunctionalTester includes modules: Filesystem, Yii2
FunctionalTester.php generated successfully. 63 methods added
```

Jalankan Web Server

Untuk memulai uji coba jenis *functional* dan *acceptance test* maka kita membutuhkan *web server*.

Jalankan *web server* apache atau nginx Anda.

Edit Konfigurasi Menyesuaikan Web Server

Caranya, edit *file* `codeception.yml` di direktori `@app/tests/` pada bagian akhir yaitu URL `http://localhost:8080/index-test.php` menjadi sebagai berikut

```
config:
    # the entry script URL (with host info) for functional and acceptance tests
```

```
# PLEASE ADJUST IT TO THE ACTUAL ENTRY SCRIPT URL
# test_entry_url: http://localhost:8080/index-test.php
test_entry_url: http://localhost/basic/web/index-test.php
```

Entry point pada saat uji coba menggunakan index-test.php dan bukan index.php karena pada index-test.php ada pengaturan untuk *environment testing* yang bisa jadi berbeda dengan *production*.

Termasuk juga, kita harus menyesuaikan konfigurasi di @app/tests/codeception/acceptance.suite.yml dengan menghapus port 8080.

```
config:
  PhpBrowser:
    # PLEASE ADJUST IT TO THE ACTUAL ENTRY POINT WITHOUT PATH INFO
    url: http://localhost
  WebDriver:
    # url: http://localhost:8080
    # browser: firefox
    # restart: true
```

Jangan salah ya http://localhost, bukan http://localhost/basic/web.

Jalankan Code Testing

Jalankan *code testing* dengan mengeksekusi perintah berikut

```
| # run all available tests
| codecept run
```

Hasilnya sebagai berikut

```
C:\WINDOWS\system32\cmd.exe
29. I fill field "textarea[name="ContactForm[body]"]",
28. I fill field "input[name="ContactForm[subject]"]",
27. I fill field "input[name="ContactForm[email]"]", "t
26. I fill field "input[name="ContactForm[name]"]", "te

FAILURES!
Tests: 12, Assertions: 59, Failures: 1.
```

Gagal. Aplikasi kita tidak lolos 😞, dimana ada 12 *test*, 59 *assertions (step testing)*, dan 1 *failures*. Jangan bersedih yah hehe.

Hasil keluaran dari uji coba ini jika ada bisa jumpai pada direktori @app/basic/tests/_output

B. 4. Pembahasan Hasil Uji Coba

Mari kita bahas dengan menjalankan *code testing* per jenis uji coba

Acceptance Tests

Perintah berikut untuk menjalankan *acceptance test*

```
| codecept run acceptance
```

```
C:\WINDOWS\system32\cmd.exe
26. I fill field "input[name="ContactForm[name]"]",

FAILURES!
Tests: 4, Assertions: 22, Failures: 1.

C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Ternyata kegagalan ada di *unit testing*.

Pada CMD terlihat bahwa galat hanya terjadi pada formulir *contact*,

```
Scenario Steps:
32. I don't see element "#contact-form"
31. I click "contact-button"
30. I fill field "input[name="ContactForm[\\"
```

Nah setelah penulis telusuri pada file

```
@app\tests\codeception\acceptance>ContactCept.php
```

```
46 $contactPage->submit([
47     'name' => 'tester',
48     'email' => 'tester@example.com',
49     'subject' => 'test subject',
50     'body' => 'test content',
51     'verifyCode' => 'testme',
52 ]);
53 if (method_exists($I, 'wait')) {
54     $I->wait(3); // only for selenium
55 }
56 $I->dontSeeElement('#contact-form');
57 $I->see('Thank you for contacting us.
```

Dan setelah penulis uji coba melalui *web browser* dengan menggunakan URL ini

```
∞ http://127.0.0.1/basic/web/site/contact
```

ternyata ada galat pada captcha

Verification Code

The verification code is incorrect.

Setelah dilihat di SiteController

```
public function actions()
{
    return [
        'error' => [
            'class' => 'yii\web\ErrorAction',
        ],
        'captcha' => [
            'class' => 'yii\captcha\CaptchaAction',
            'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
        ],
    ];
}
```

Tertulis jika YII_ENV_TEST maka *captcha* cukup gunakan testme.

Namun setelah dicek pada file entry script @app/web/index-test.php, ternyata beda.

```
| defined('YII_ENV') or define('YII_ENV', 'test');
```

Karena itu mari kita ubah konstanta di SiteController dari YII_ENV_TEST menjadi YII_ENV, setelah itu kemudian kita uji coba lagi

```
C:\WINDOWS\system32\cmd.exe
Ensure that login works (LoginCept)
Ok
Ok
-----
Time: 3.6 seconds, Memory: 20.00Mb
OK (4 tests, 23 assertions)
```

Wow.. berhasil

Functional Tests

Perintah berikut untuk menjalankan *functional test*

```
| codecept run functional
```

```
C:\WINDOWS\system32\cmd.exe
Ensure that login works (LoginCept)
=====
=====
=====
Time: 1.62 seconds, Memory: 26.00Mb
OK (4 tests, 23 assertions)
```

Sukses, jadi tidak perlu dibahas

Unit Tests

Perintah berikut untuk menjalankan *unit test*

```
| codecept run unit
```

```
C:\WINDOWS\system32\cmd.exe
nWrongPassword)
Test login correct (tests\codeception\unit\models\LoginFo
t)
=====
=====
Time: 1.45 seconds, Memory: 20.00Mb
OK (4 tests, 14 assertions)
```

Sukses, jadi tidak perlu dibahas

Akhirnya sukses semua, dan kita menemukan satu *bug* ☺ Laporkan.

B. 5. Membuat Code Testing

Pada bab sebelumnya kita telah membuat CRUD untuk tabel *employee*, nah kita bisa jadikan itu sebagai studi kasus untuk uji coba.

Unit Test

Uji coba ini digunakan untuk memverifikasi apakah suatu kode aplikasi berjalan sesuai dengan yang diharapkan. Pada kasus CRUD *employee*, maka yang cocok dan umum untuk diuji coba apakah kode tersebut berjalan sesuai dengan yang diharapkan adalah model *Employee*.

Pada model *Employee* kita jumpai fungsi *rules* yang berisi aturan-aturan validasi. Nah itulah yang akan jadi target uji coba kita, “apakah validasinya benar-benar telah sesuai dengan yang diharapkan atau belum”.

Mari kita analisis.

Tabel *employee* memiliki skema (*id*, *name varchar(50)*, *age int(3)*), kita bisa lihat pada model *Employee* @app/models/Employee.php pada bagian fungsi *rules*

```
public function rules()
{
    return [
        [['name', 'age'], 'required'],
        [['name'], 'string'],
        [['age'], 'integer'],
    ];
}
```

Aturan-aturan tersebut bisa kita uji coba. Atribut *name* dan *age* tidak boleh kosong, atribut *name* harus *string*, dan atribut *age* harus *integer*. Nah dari *rule* ini kita bisa buatkan uji coba untuk validasinya apakah sudah OK sesuai dengan *rules* tersebut atau belum.

Mari kita buat uji cobanya

Codeception menyediakan *generator* untuk membuat kode *template unit testing*

```
| codecept generate:test unit Employee
```

Maka akan tercipta file uji coba bertipe TEST

```
 @app/tests/codeception/unit/EmployeeTest.php
```

sebagai berikut.

```
<?php
class EmployeeTest extends \Codeception\TestCase\Test
{
    protected $tester;

    protected function _before()
    {
    }

    protected function _after()
    {
    }

    public function testMe()
    {

}
```

Fungsi `_before` akan dijalankan sebelum uji coba dimulai, sedangkan fungsi `_after` sebaliknya yaitu akan dijalankan setelah uji coba berakhir. Adapun fungsi `testMe` hanyalah contoh fungsi dimana *code testing* kita diimplementasikan.

Agar fitur-fitur Yii bisa berjalan dengan baik, maka kita perlu merombak *class* `EmployeeTest` menjadi sebagai berikut

```
<?php
class EmployeeTest extends yii\codeception\TestCase
{
    use Codeception\Specify;

    protected function setUp()
    {
    }

    protected function tearDown()
    {
    }

    // tests
    public function testMe()
    {
    }
}
```

Sederhana saja, fungsi `setUp` akan dijalankan sebelum *tests* sebagai persiapan misalnya data awal, sedangkan fungsi `tearDown` akan dijalankan setelah uji coba misal hapus data uji coba. Adapun fungsi `testMe` adalah *code testing* kita misalnya uji coba *input* data.

Lihat bahwa *class* `test` yang kita buat meng-*extends* *class* `[[yii\codeception\TestCase]]` yaitu *base class* untuk semua *codeception unit test*.

Mari kita buat fungsi baru pada *class* `EmployeeTest` sebagai berikut

```
protected $model;

protected function setUp()
{
    parent::setUp();
    // uncomment the following to load fixtures for user table
    //$this->loadFixtures(['user']);
    $this->model = new \app\models\Employee();
}

public function testNameValidation()
{
    $model = $this->model;
```

```

    $this->specify("name is required", function() use ($model) {
        $model->name = null;
        expect('model should not validate', $model->validate(['name']))->false();
    });
}

```

Tidak susah memahami kode di atas, dimana uji coba itu untuk memastikan apakah benar atribut *name* itu *required*? Maka dibuatkan kasus dimana atribut *name* diset sebagai *null*, kemudian diuji coba dengan menjalankan fungsi *validate*, dan jika *false* atau gagal maka uji coba ini terpenuhi.

Kita juga bisa mengetest, untuk kasus *required* dengan memasukkan nilai yang normal, berhasil apa tidak?

```

$this->specify("name is required", function() use ($model) {
    $model->name = 'hafid';
    expect('model should validate', $model->validate(['name']))->true();
});

```

Atau kita uji coba atribut *age* dengan memasukkan nilai yang bukan *integer*

```

public function testAgeValidation()
{
    $model = $this->model;

    $this->specify("age is required", function() use ($model) {
        $model->age = null;
        expect('model should not validate', $model->validate(['age']))->false();
    });

    $this->specify("age is required", function() use ($model) {
        $model->age = 31;
        expect('model should validate', $model->validate(['age']))->true();
    });

    $this->specify("age is int", function() use ($model) {
        $model->age = 'test';
        expect('model should not validate', $model->validate(['age']))->false();
    });
}

```

Pada contoh di atas kita telah membuat dua fungsi uji coba yaitu *testNameValidation()* yang berisi dua *assertion* dan *testAgeValidation()* yang berisi tiga *assertion*.

Sekarang waktunya menjalankan uji coba, pada CMD gunakan perintah berikut

```
| codecept run unit
```

```
C:\WINDOWS\system32\cmd.exe
Time: 3.53 seconds, Memory: 22.00Mb
OK (6 tests, 19 assertions)
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Berhasil, ☺

Kok ada enam *test*? Padahal kita cuman membuat dua uji coba dengan lima *assertions*? Ya karena bawaan Yii juga ikut diproses.

Nah kita bisa jalankan uji coba spesifik pada *file* uji coba tertentu

```
| codecept run unit EmployeeTest.php
```

```
C:\WINDOWS\system32\cmd.exe
Time: 3.36 seconds, Memory: 20.00Mb
OK (2 tests, 5 assertions)
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Atau kita juga bisa spesifik ke *assertion* tertentu.

```
| codecept run unit EmployeeTest.php:testNameValidation
```

```
C:\WINDOWS\system32\cmd.exe
Time: 2.34 seconds, Memory: 20.00Mb
OK (1 test, 2 assertions)
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Done.

Functional Test

Uji coba ini digunakan untuk memverifikasi skenario dari perspektif bisnis proses menggunakan *browser emulation*.

Pada uji coba ini yang cocok untuk di uji coba adalah *controller* atau dalam hal ini EmployeeController. Apakah *create*, *read*, *update* *delete* itu sesuai dengan yang diinginkan.

Mari kita buat file uji coba bertipe CEPT(diakhiri dengan cept) dengan menggunakan *generator*

```
| codecept generate:cept functional Employee
```

Maka akan tercipta file EmployeeCept.php pada direktori

```
@app/tests/codeception/functional/
```

```
<?php
$I = new FunctionalTester($scenario);
$I->wantTo('perform actions and see result');
```

\$I adalah *instance* dari class FunctionalTester yang bisa kita jumpai pada direktori yang sama yaitu @app/tests/codeception/functional. Pada class FunctionalTester ini, kita bisa menjumpai beberapa fungsi penting yang bisa kita gunakan untuk uji coba antara lain:

- wantTo

Digunakan untuk menjelaskan secara garis besar dari uji coba yang akan kita lakukan.

```
| $I->wantTo('Memastikan "Create Employee" bekerja');
```

- amOnPage

Digunakan untuk mengalihkan lokasi web ke URL tertentu contoh

```
| $I->amOnPage(['site/view', 'page'=>'about']);
$I->amOnPage('index-test.php?site/index');
$I->amOnPage('http://localhost/index-test.php?site/index');
```

- amGoingTo

Digunakan untuk menjelaskan tiap langkah dari uji coba yang kita lakukan.

```
| $I->amGoingTo('submit formulir dengan data input kosong');
```

- expectTo

Digunakan untuk menjelaskan ekspektasi atau harapan pada satu langkah yang dilalui.

```
| $I->expectTo('Validasi galat muncul!');
```

- see

Digunakan untuk melihat adakah suatu teks yang terlihat (di *acceptance* banyak dieksplor)

```
| $I->see('Name cannot be blank.');
```

- dontSee
Kebalikan dari fungsi see
- fillField
Digunakan untuk menginputkan data ke *field* dari *form*

```
| $I->fillField('input[name="Employee[name]"]', 'Hafid');
```

- click
Digunakan untuk mengklik suatu tautan

```
| $I->click('button[type="submit"]);
```

- submitForm
Digunakan untuk mensubmit sebuah formulir beserta datanya

```
| $I->submitForm('#login', [
    'login' => 'admin',
    'password' => '123456'
]);
```

- seeRecord
Digunakan untuk melihat adakah *record* tertentu di basis data

```
| $I->seeRecord('\app\models\Employee', ['name'=>'Hafid']);
```

- dontSeeRecord
Kebalikan dengan fungsi seeRecord. Biasanya digunakan untuk mengecek apakah proses hapus data berhasil.

```
| $I->dontSeeRecord('\app\models\Employee', ['name'=>'Hafid']);
```

Masih banyak lagi fungsi-fungsi lain, selengkapnya silakan lihat pada *class FunctionalTester*.

Berbekal pengetahuan tentang fungsi-fungsi tersebut, kita akan mencoba membuat *code testing* untuk fungsi *create employee* apakah berfungsi dengan baik. Kode lengkapnya kurang lebih sebagai berikut

```
<?php
/* @var $scenario Codeception\Scenario */
$I = new FunctionalTester($scenario);
/* TESTING CREATE EMPLOYEE */
$I->wantTo('Memastikan "Create Employee" bekerja');
$I->amOnPage(['employee/create']);

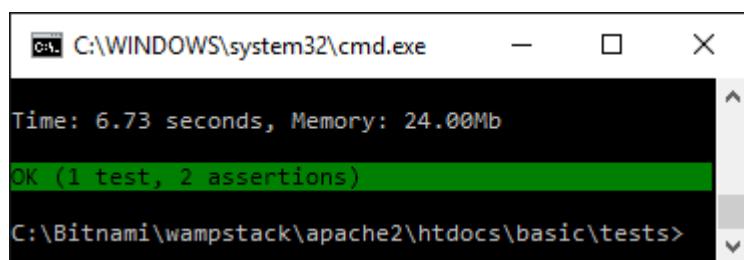
$I->amGoingTo('submit form dengan data input kosong');
$I->click('button[type="submit"]');
$I->expectTo('Validasi galat muncul!');
$I->see('Name cannot be blank.');
$I->see('Age cannot be blank.');

$I->amGoingTo('submit form dengan data');
$I->fillField('input[name="Employee[name]"]', 'Hafid');
$I->fillField('input[name="Employee[age]"]', '30');
$I->click('button[type="submit"]');
$I->expectTo('ada data Hafid di tabel employee');
$I->seeRecord('\app\models\Employee', ['name'=>'Hafid']);
```

Untuk uji coba fungsi lain silakan dijelajahi sendiri ya ☺

Mari kita jalankan *code testing* ini melalui CMD

```
| codecept run functional EmployeeCept.php
```



```
C:\WINDOWS\system32\cmd.exe
Time: 6.73 seconds, Memory: 24.00Mb
OK (1 test, 2 assertions)
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Done.

Acceptance Test

Uji coba ini digunakan untuk memverifikasi skenario dari perspektif pengguna menggunakan *web browser*. Secara *default*, codeception telah menggunakan *module* PhpBrowser untuk melakukan uji coba yaitu *web browser* dengan fungsi yang terbatas.

Sedikit berbeda dengan setting diawal, pada uji coba ini kita akan melakukan pengaturan URL pada file @app/tests/codeception/acceptance.suite.yml dengan menggunakan URL lengkap yaitu http://localhost/basic/web

```

config:
    PhpBrowser:
        # PLEASE ADJUST IT TO THE ACTUAL ENTRY POINT WITHOUT PATH INFO
        url: http://localhost/basic/web
        WebDriver:
            url: http://localhost:8080
            browser: firefox
            restart: true

```

Efek dari perubahan ini akan membuat contoh *acceptance testing* bawaan Yii akan galat, namun untuk sementara lupakan itu (ada ketidak konsistenan di sini).

Mari kita generate file code testing bertipe CEPT

```
| codecept generate:cept acceptance Employee
```

Maka akan tercipta file EmployeeCept.php pada direktori

@app/tests/codeception/acceptance/

```

<?php
$I = new AcceptanceTester($scenario);
$I->wantTo('perform actions and see result');

```

\$I adalah *instance* dari class AcceptanceTester yang bisa kita jumpai pada direktori yang sama yaitu @app/tests/codeception/acceptance. Fungsi-fungsi pada class AcceptanceTester hampir sama dengan class FunctionalTester ini.

Kode lengkapnya kurang lebih sebagai berikut

```

<?php
/* @var $scenario Codeception\Scenario */
$I = new AcceptanceTester($scenario);
$I->wantTo('Memastikan "Employee" bekerja');

/* TESTING INDEX EMPLOYEE */
$I->amGoingTo('cek halaman daftar employee');
$I->amOnPage('employee/index');
$I->expectTo('judul halaman "Daftar Employee", dan tombol Create');
$I->see('Daftar Employee','h1');
$I->see('CREATE','a');

/* TESTING CREATE EMPLOYEE */
$I->wantTo('Memastikan "Create Employee" bekerja');
$I->amGoingTo('cek halaman create employee');
$I->amOnPage('employee/create');
$I->see('Employee','h1');
$I->see('SIMPAN','button');
$I->seeElement('input', ['name' => 'Employee[name]']);
$I->seeElement('input', ['name' => 'Employee[age]']);

$I->amGoingTo('submit formulir dengan data input kosong');
$I->click('button[type="submit"]');
$I->expectTo('Validasi galat muncul!');
$I->see('Name cannot be blank.');
$I->see('Age cannot be blank.');

$I->amGoingTo('submit formulir dengan data data');
$I->fillField('input[name="Employee[name]"]', 'Hafid');
$I->fillField('input[name="Employee[age]"]', '30');
$I->click('button[type="submit"]');
$I->expectTo('submit formulir berhasil');
$I->see('Data berhasil disimpan');

```

Sekilas nampak sama dengan kode pada *functional testing*, namun ingat bahwa uji coba ini hanya fokus dari perspektif atau sudut pandang pengguna. Pengguna hanya melihat apa yang tampil di *web browser*, tidak lebih dari itu.

Mari kita jalankan *code testing* ini melalui CMD

```
| codecept run acceptance EmployeeCept.php
```

```
C:\WINDOWS\system32\cmd.exe
Time: 8.82 seconds, Memory: 20.00Mb
OK (1 test, 9 assertions)
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>
```

Done.

Sebagaimana yang disebutkan di atas bahwa *module web browser* yang digunakan adalah PhpBrowser dengan segala keterbatasannya. Nah sekarang kita akan mencoba menggunakan module lain yaitu WebDriver.

Module ini benar-benar akan menjalankan *web browser* yang telah terpasang pada komputer kita misalnya Firefox sehingga kita melihat simulasi yang nyata.

Silakan baca panduan resminya pada tautan berikut: <http://codeception.com/docs/modules/WebDriver>,

1. Unduh *Selenium*

```
∞ http://docs.seleniumhq.org/download/
```

2. Buka CMD baru, jalankan *Selenium* dengan perintah

```
| java -jar selenium-server-standalone-2.xx.xxxx.jar
```

Pada OS Windows, penulis harus menambahkan parameter berikut, yaitu menyertakan lokasi dari *file web browser executable*-nya.

```
| java -jar selenium-server-standalone-2.xx.xxxx.jar -Dwebdriver.firefox.bin="D:\Program Files (x86)\Mozilla Firefox\Firefox.exe"
```

Server *Selenium* harus *running* ketika uji coba dilakukan. Pada kode di atas *current directory* pada CMD sama dengan direktori *file selenium-server*, jika beda maka kita harus menuliskan direktorinya secara lengkap misal

```
| java -jar c:\downloads\selenium-server-standalone-2.xx.xxxx.jar
```

Contoh pada komputer penulis

```
C:\WINDOWS\system32\cmd.exe - java -jar C:\U...
C:\Users\hafid>java -jar C:\Users\hafid\Downloads\selenium-server-standalone-2.52.0.jar webdriver.firefox.
bin="D:\Program Files (x86)\Mozilla Firefox\Firefox.e
xe"
18:31:19.073 INFO - Launching a standalone Selenium S
erver
```

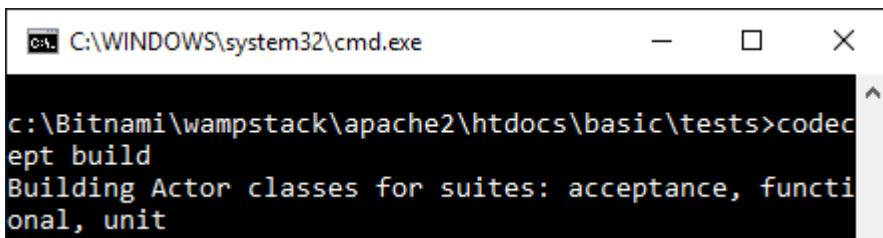
3. Modifikasi *file @app/tests/codeception/acceptance.suite.yml*, aktifkan *module* WebDriver.

```
class_name: AcceptanceTester
modules:
    enabled:
        - PhpBrowser
    # you can use WebDriver instead of PhpBrowser to test javascript and ajax.
    # This will require you to install selenium. See http://codeception.com/docs/04-
AcceptanceTests#Selenium
    # "restart" option is used by the WebDriver to start each time per test-
    file new session and cookies,
    # it is useful if you want to login in your app in each test.
        - WebDriver
    config:
        PhpBrowser:
    # PLEASE ADJUST IT TO THE ACTUAL ENTRY POINT WITHOUT PATH INFO
        url: http:localhost/basic/web
```

```
WebDriver:
  url: http://localhost/basic/web
  browser: firefox
  restart: true
```

4. Setelah itu kita harus *build* lagi, karena terjadi perubahan *module*

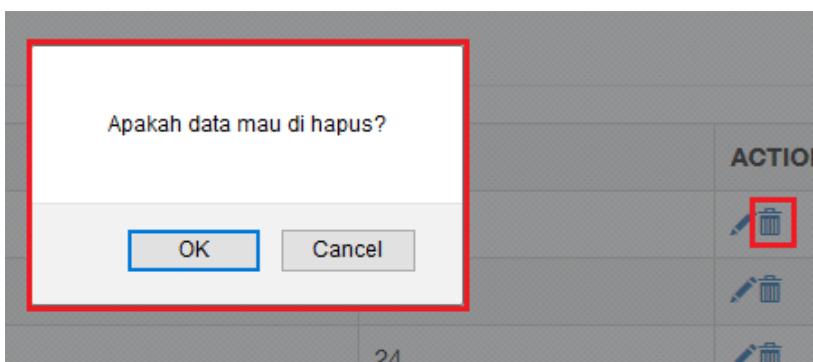
```
| codecept build
```



```
c:\Bitnami\wampstack\apache2\htdocs\basic\tests>codecept build
Building Actor classes for suites: acceptance, functional, unit
```

Perintah ini juga akan meng-*generate* ulang file @app/tests/codeception/acceptance/AcceptanceTester.php, sehingga fungsi-fungsi pada class ini akan menyesuaikan dengan kemampuan *module*-nya.

Sebagai contoh kita akan menguji fitur *delete record* yang mana menggunakan Javascript *popup* untuk konfirmasi hapus

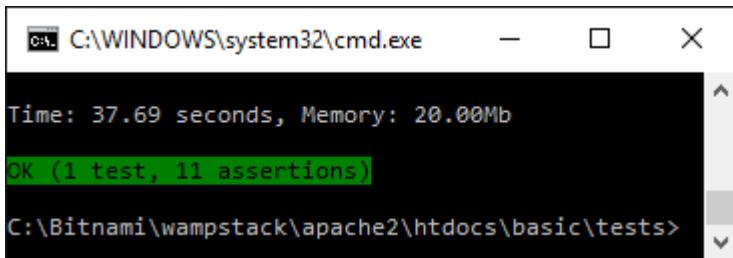


Module WebDriver mampu menangani hal ini, yaitu menekan tombol *OK* atau *Cancel*. Kodennya kira-kira sebagai berikut:

```
$I = new AcceptanceTester($scenario);
$I->wantTo('Memastikan "Employee" bekerja');
$I->amOnPage('employee/index');
$I->see('Daftar Employee', 'h1');
$I->click('a[href*="delete?id=1"]');
if (method_exists($I, 'wait')) {
    $I->wait(3); // only for selenium
}
$I->acceptPopup();
if (method_exists($I, 'wait')) {
    $I->wait(3); // only for selenium
}
$I->see('Daftar Employee', 'h1');
```

Jika uji coba ini dijalankan maka kita akan lihat bahwa *web browser* benar-benar dijalankan.

```
| codecept run acceptance EmployeeCept.php
```



```
c:\Bitnami\wampstack\apache2\htdocs\basic\tests>codecept run acceptance EmployeeCept.php
Time: 37.69 seconds, Memory: 20.00Mb
OK (1 test, 11 assertions)
```

Jika uji coba ini gagal, mungkin dikarenakan server Selenium belum dijalankan atau tidak ada data *employee* dengan id = 1.

Pada uji coba menggunakan *web driver* ini kita perlu menambahkan waktu tunggu setelah proses *submit* atau klik, hal ini dikarenakan proses uji cobanya memang benar-benar langsung pada *web browser*.

```

if (method_exists($I, 'wait')) {
    $I->wait(3); // only for selenium
}

```

Keterangan: kode di atas akan menunggu selama 3 detik.

Format Cest

Format Cest adalah format *code testing* untuk *functional* dan *acceptance* (sebenarnya *unit* juga) *testing*, yang mempunyai karakteristik *file* berakhiran Cest dan kodenya berupa *class* dan fungsi-fungsi. Ingat, pada contoh-contoh yang telah kita bahas sebelumnya yaitu *functional* dan *acceptance* test, kita menggunakan format cept, yang mana tidak ada berbentuk *class*.

Format cept biasanya lebih mudah untuk difahami oleh non pemrogram aplikasi sedangkan format cest lebih memudahkan pemrogram aplikasi dalam mengelola kodenya. Jadi silakan dipilih yang paling sesuai dengan kebutuhan.

Untuk membuat format cest kita juga bisa menggunakan generator

```
| codecept generate:cest functional Employee
```

Kode di atas akan meng-generate *functional testing* yaitu @app/testing/codeception/functional/EmployeeCest.php

```

<?php
use \AcceptanceTester;

class EmployeeCest
{
    public function _before(AcceptanceTester $I)
    {
    }

    public function _after(AcceptanceTester $I)
    {
    }

    // tests
    public function tryToTest(AcceptanceTester $I)
    {
    }
}

```

Pada kode di atas, *block-block* uji coba yang pada format cest hanya dipisahkan dengan komentar, maka pada format ini bisa kita kelompokkan masing-masing ke dalam satu fungsi tersendiri.

Adapun untuk *acceptance testing* kita bisa menggunakan perintah yang hampir sama

```
| codecept generate:cest acceptance Employee
```

Perintah tersebut akan meng-generate *acceptance testing* pada direktori @app/testing/codeception/acceptance/EmployeeCest.php

Berikut ini contoh penerapan format Cest untuk *acceptance*

```

<?php
use \AcceptanceTester;

class EmployeeCest
{
    public function _before(AcceptanceTester $I)
    {
    }

    public function _after(AcceptanceTester $I)
    {
    }

    // tests
    public function testIndex(AcceptanceTester $I)
    {
        $I->amGoingTo('cek halaman daftar employee');
        $I->amOnPage('employee/index');
        if (method_exists($I, 'wait')) {
            $I->wait(3); // only for selenium
        }
    }
}

```

```

        }
        $I->expectTo('judul halaman "Daftar Employee", dan tombol Create');
        $I->see('Daftar Employee','h1');
        $I->see('CREATE','a');
    }
}

```

Lanjutkan untuk uji coba yang lain.

Untuk menjalankan uji coba untuk file ini kita bisa gunakan perintah

```
| codecept run acceptance EmployeeCest.php
```

Atau bisa juga menjalankan uji coba untuk perfungsi, misalnya fungsi testIndex saja

```
| codecept run acceptance EmployeeCest.php:testIndex
```

Format Page

Format page ini merupakan *class* yang bisa kita gunakan dalam *functional* dan *acceptance test*. Format ini menggambarkan halaman web dan fungsi-fungsinya. Keuntungan menggunakan format *page* adalah kode uji coba bisa digunakan kembali.

Mari kita generate kode page

```
| codecept generate:pageobject Employee
```

Maka akan tercipta file EmployeePage.php yang terletak pada direktori@app/tests/codeception/_pages/EmployeePage.php. Supaya kita bisa memaksimalkan fitur-fitur Yii untuk page ini maka kita extends dengan *class* [[yii\codeception\BasePage]] dengan catatan bahwa kita tidak akan menggunakan fitur openBy yang dimiliki oleh *class* BasePage ini dikarenakan permasalah URL yang penulis bahas diawal.

Berikut ini contoh kode dari EmployeePage.

```

<?php
namespace tests\codeception\_pages;
use yii\codeception\BasePage;

class EmployeePage extends BasePage
{
    public static $URL = 'employee/index';

    public static function route($param='')
    {
        return static::$URL.$param;
    }

    public function start($param='')
    {
        $this->actor->amOnPage(self::route($param));
        $this->wait(3);
    }

    public function create($name=null, $age=null)
    {
        $this->actor->click('Create');
        $this->wait(3);
        $this->actor->fillField('input[name="Employee[name]"]', $name);
        $this->actor->fillField('input[name="Employee[age]"]', $age);
        $this->actor->click('button[type="submit"]');
        $this->wait(3);
    }

    public function delete($id=null)
    {
        $this->actor->click('a[href*="delete?id='.$id.'"]');
        $this->wait(3);
        if (method_exists($this->actor, 'acceptPopup')) {
            $this->actor->acceptPopup(); // only for selenium
        }
        $this->wait(3);
    }

    public function wait($time)

```

```

    {
        if (method_exists($this->actor, 'wait')) {
            $this->actor->wait($time); // only for selenium
        }
    }
}

```

Kode `$this->actor` merupakan `$I` atau objek *testing guy* yang dalam hal ini bisa FunctionalTester atau AcceptanceTester, untuk lebih jelasnya kita bisa melihat pada *parent class*-nya yaitu BasePage

```

public function __construct($I)
{
    $this->actor = $I;
}

```

Class EmployeePage di atas, bisa kita implementasikan pada *code testing* kita, sebagai contoh, pada *acceptance test format cest* yang telah kita buat yaitu `@app/tests/codeception/acceptance/EmployeeCest.php`

```

<?php
use \AcceptanceTester;
use tests\codeception\_pages\EmployeePage;

class EmployeeCest
{
    protected $page;

    public function _before(AcceptanceTester $I)
    {
        $this->page = new EmployeePage($I);
    }

    public function _after(AcceptanceTester $I)
    {
    }

    // tests
    public function testIndex(AcceptanceTester $I)
    {
        $I->amGoingTo('cek halaman daftar employee');
        $this->page->start();
        $I->expectTo('judul halaman "Daftar Employee", dan tombol Create');
        $I->see('Daftar Employee','h1');
        $I->see('CREATE','a');
        if (method_exists($I, 'wait')) {
            $I->wait(3); // only for selenium
        }
    }

    public function testCreate(AcceptanceTester $I)
    {
        $I->amGoingTo('submit formulir dengan data data');
        $this->page->start();
        $this->page->create('Hafid',31);
        $I->expectTo('submit formulir berhasil');
        $I->see('Data berhasil disimpan');
    }

    public function testDelete(AcceptanceTester $I)
    {
        $I->amGoingTo('hapus data data');
        $this->page->start();
        $this->page->delete(1);
        $I->expectTo('data berhasil dihapus');
        $I->dontSeeLink('a[href*="delete?id=1"]');
    }
}

```

Terlihat bahwa *code testing* kita lebih rapi.

C. Fixture Data

Fixture juga merupakan bagian yang cukup penting dalam sebuah uji coba. Hal ini karena *fixture* ini digunakan untuk mempersiapkan lingkungan uji coba yang sesuai dengan kebutuhan dari suatu uji coba.

Sebagai contoh sederhana, misalnya kita ingin membuat uji coba update data, yaitu *update* data *employee* dengan nama hafid menjadi Hafid Mukhasin. Bayangkan jika nama hafid tidak ada dalam basis data, maka uji coba akan mengalami kegagalan, padahal kegagalan tersebut hanya disebabkan oleh data yang tidak ditemukan, bukan pada kode aplikasi kita.

C. 1. Manual Data

Mari kita buat *class EmployeeFixture* (nama *fixture* hendaknya diakhiri dengan kata *Fixture*) pada direktori `@app/tests/codeception/fixtures/`

```
<?php
namespace app\tests\codeception\fixtures;

use yii\test\ActiveFixture;

class EmployeeFixture extends ActiveFixture
{
    public $modelClass = 'app\models\Employee';
}
```

Kemudian kita perlu siapkan data untuk uji coba. Buat file `@app/tests/codeception/fixtures/data/employee.php` yang berisi

```
<?php
return [
    'employee1' => [
        'name' => 'Hafid',
        'age' => 31,
    ],
    'employee2' => [
        'name' => 'Junaidi',
        'age' => 37,
    ],
    'employee3' => [
        'name' => 'Fuad',
        'age' => 28,
    ],
];
```

Kita juga boleh menuliskannya sebagai berikut

```
<?php
return [
    ['name' => 'Hafid', 'age' => 31],
    ['name' => 'Junaidi', 'age' => 37],
    ['name' => 'Fuad', 'age' => 28],
];
```

Setelah itu baru kita gunakan pada uji coba kita.

Sebagai contoh ketika pada *unit testing* cukup dengan menambahkan fungsi *fixtures*

```
<?php
use yii\codeception\TestCase;
use app\tests\codeception\fixtures\EmployeeFixture;

class EmployeeTest extends TestCase
{
    use Codeception\Specify;

    public function fixtures()
    {
        return [
            'employees' => EmployeeFixture::className(),
        ];
    }
}
```

Pada *functional* ataupun *acceptance testing* khususnya format *cept*, kita bisa menggunakan *fixture* sebagai berikut

```
<?php

use app\tests\codeception\fixtures\EmployeeFixture;
use tests\codeception\_pages\EmployeePage;

$I = new AcceptanceTester($scenario);
$I->wantTo('Memastikan "Employee" bekerja');

$employeeFixture = new EmployeeFixture();
$employeeFixture->load();
```

Sedangkan pada format *cest*, kita bisa implementasikan pada fungsi *_before*, lihat contoh berikut

```
<?php
use \AcceptanceTester;
use tests\codeception\_pages\EmployeePage;
use app\tests\codeception\fixtures\EmployeeFixture;

class EmployeeCest
{
    protected $page;

    public function _before(AcceptanceTester $I)
    {
        $this->page = new EmployeePage($I);
        $employeeFixture = new EmployeeFixture();
        $employeeFixture->load();
    }
}
```

Dengan menambahkan *fixture* pada saat uji coba ini maka basis data untuk uji coba akan senantiasa diperbarui sebelum uji coba dijalankan, sehingga tidak ada lagi kemungkinan terjadi galat, dan uji coba berjalan sesuai dengan yang diharapkan.

C. 2. Faker Data

Faker data adalah *generator* data *fake* atau palsu. Pada bagian sebelumnya, kita perlu membuat atau lebih tepatnya mengarang data untuk uji coba secara manual. Kalau kita hanya perlu satu atau dua data tentu tidak ada masalah jika dilakukan secara manual. Namun jika misalnya kita ingin melakukan uji coba untuk *paging* pada Gridview dimana data minimal berjumlah 21 atau mungkin uji coba aplikasi dengan banyak data maka menjadi tidak efisien jika lakukan secara manual.

Oleh karena itu, kita perlu menggunakan *tools* untuk menggenerate data tersebut. *Tools* yang bisa kita gunakan adalah *faker*, *yiisoft/yii2-faker*, merupakan *extension official* Yii yang secara *default* sudah termasuk dalam instalasi Yii kita. Kabar baiknya adalah *tools* ini juga secara *default* telah diintegrasikan dengan uji coba, melalui *config*. Silakan lihat di `@app/tests/codeception/config/config.php`

```
<?php
/**
 * Application configuration shared by all test types
 */
return [
    'language' => 'en-US',
    'controllerMap' => [
        'fixture' => [
            'class' => 'yii\faker\FixtureController',
            'fixtureDataPath' => '@tests/codeception/fixtures',
            'templatePath' => '@tests/codeception/templates',
            'namespace' => 'tests\codeception\fixtures',
        ],
    ],
    'components' => [

```

Keterangan:

- templatePath adalah lokasi *template* dari *fixture*
- fixtureDataPath adalah lokasi hasil *generate* data *faker* yang kemudian dijadikan sebagai data *fixture*

Oleh karena lokasi data *fixture* kita adalah di `@tests/codeception/fixtures/data` maka kita perlu sesuaikan konfigurasi di atas menjadi sebagai berikut

```
<?php
return [
```

```
'language' => 'en-US',
'controllerMap' => [
    'fixture' => [
        'class' => 'yii\faker\FixtureController',
        'fixtureDataPath' => '@tests/codeception/fixtures/data',
        'templatePath' => '@tests/codeception/templates',
        'namespace' => 'tests\codeception\fixtures',
    ],
],
```

Lalu bagaimana cara menggunakan uji coba pada *fixture* kita? Inilah yang akan kita bahas sekarang ☺

Caranya cukup mudah yaitu mula-mula kita membuat sebuah *template faker* terlebih dahulu dengan nama employee.php pada direktori @tests/codeception/templates, sebagai berikut:

```
<?php
return [
    [
        'name' => $faker->name,
        'age' => $faker->numberBetween(15, 75),
    ]
];
```

\$faker adalah objek dari *Faker generator*

Tentang fungsi-fungsi yang didukung oleh *faker* selengkapnya bisa kita baca pada tautan ini <https://github.com/fzaninotto/Faker>

Cara meng-*generate* menjadi *fixture* data adalah menggunakan CMD

```
| codeception\bin\yii fixture/generate employee
```

Tentu saja untuk Linux jalankan perintah berikut.

```
| codeception/bin/yii fixture/generate employee
```

Perintah di atas adalah perintah untuk meng-*generate* *fixture* data berdasarkan *template employee* di @tests/codeception/templates untuk kemudian di-*generate* menjadi data *fixture* di @tests/codeception/fixtures/data.

```
C:\WINDOWS\system32\cmd.exe
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>codeception\bin\yii fixture/generate employee
Fixtures will be generated under the path:
    C:\Bitnami\wampstack\apache2\htdocs\basic\tests\codeception\fixtures\data

Templates will be taken from path:
    C:\Bitnami\wampstack\apache2\htdocs\basic\tests\codeception\templates

    * employee
Generate above fixtures? (yes|no) [no]:yes
The following fixtures template files were generated:

    * employee
```

Sekarang mari kita lihat hasil *generate*-nya di:

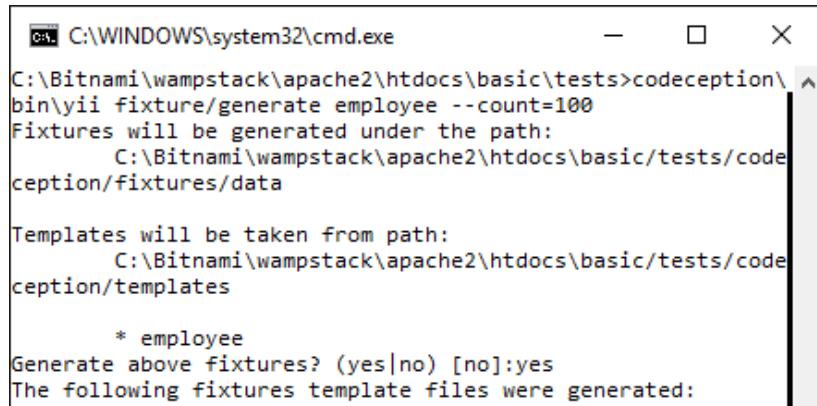
```
@tests/codeception/fixtures/data/employee.php
```

```
<?php
return [
    [
        [
            [
                'name' => 'Quinton Reichert MD',
                'age' => 50,
            ],
        ],
        [
            [
                'name' => 'Kelton Kiehn',
                'age' => 48,
            ],
        ],
    ];
];
```

Data ini mungkin berbeda dengan di tempat anda, karena datanya *random*. Lalu bagaimana jika kita perlu misalnya 100 data atau 1000 data *fixture*?

Caranya cukup dengan menambahkan parameter `--count=(jumlah data yang diinginkan)`

```
∞ codeception\bin\yii fixture/generate employee --count=100
```



```
C:\Bitnami\wampstack\apache2\htdocs\basic\tests>codeception\bin\yii fixture/generate employee --count=100
Fixtures will be generated under the path:
    C:\Bitnami\wampstack\apache2\htdocs\basic\tests\codeception\fixtures\data

Templates will be taken from path:
    C:\Bitnami\wampstack\apache2\htdocs\basic\tests\codeception\templates

    * employee
Generate above fixtures? (yes|no) [no]:yes
The following fixtures template files were generated:
```

Dan lihat apa yang dihasilkannya?

Mudah sekali bukan? ☺

D. Kesimpulan

Uji coba merupakan sesuatu kegiatan yang dilakukan secara berulang-ulang oleh pemrogram aplikasi, dalam rangka memastikan sistem aplikasi yang dibuat bekerja sesuai dengan yang diharapkan. Oleh karena itu, lahirlah *tools-tools* yang memudahkan para pemrogram aplikasi tersebut untuk melakukan uji coba, salah satunya adalah Codeception. Di Yii, Codeception telah terintegrasi dengan disertai *sample code*-nya sehingga akan mempermudah bagi kita untuk mengimplementasikannya pada aplikasi.

Pada bab selanjutnya, kita akan belajar tentang metode otentifikasi pengguna, baik otentifikasi melalui basis data biasa maupun melalui *social media*.

Halaman ini sengaja dikosongkan!

11 Otentikasi Pengguna

Pada bab ini kita akan membahas tentang:

- Pemrosesan kata sandi
- *Login & signup* menggunakan basis data
- *Login & signup* menggunakan *socia media*

Otentikasi digunakan untuk memastikan hanya pengguna yang berhak saja yang bisa dan boleh mengakses sebuah sistem aplikasi. otentikasi bisa juga dimaknai sebagai sebuah pintu utama atau *filter* utama keamanan dari aplikasi. Oleh karenanya, penting untuk kemudian kita bahas sehingga diharapkan sistem yang akan kita bangun benar-benar mengikuti kaidah-kaidah pokok ini.

Pada sebuah aplikasi, otentikasi merujuk pada fitur *login*. Namun sebelum kita membahas tentang otentikasi, alangkah baiknya kita mengetahui konsep dan teknologi dibalik otentikasi.

A. Bekerja dengan Kata Sandi

A. 1. Apa itu Kata Sandi?

Kata sandi atau *password* biasa diartikan sebagai kata yang digunakan untuk masuk ke dalam sistem. Setiap kita hendak mengakses aplikasi, misalnya *webmail*, *social media*, atau aplikasi-aplikasi yang menyangkut privasi data maka tampilan pertama yang akan disuguhkan kepada kita adalah formulir *login*. Formulir ini biasanya terdiri dari dua inputan yaitu *username / email* dan kata sandi.

Pada poin ini, kata sandi merupakan hal vital yang harusnya hanya pengguna tersebut saja yang boleh tahu, bahkan pemilik aplikasi seharusnya tidak boleh mengetahui apa kata sandi dari pengguna aplikasinya. Hal ini berbeda dengan *username / email* yang bisa sembarang orang tahu. Oleh karena itu, untuk menjaga kerahasiaan tersebut makanya kita sering jumpai bahwa formulir *input-an* kata sandi akan di-*masking* dengan tanda asterik (*).

Pada HTML kita bisa menerapkannya dengan menggunakan *type password* pada *input*

```
| <input type="password" />
```

Berikut ini tampilan *field*-nya.



Implementasi pada Yii, cukup dengan menambahkan fungsi *passwordInput()* pada *field* formulir, maka *field* kata sandi akan dimasking.

```
| <?= $form->field($model, 'username') ?>
| <?= $form->field($model, 'password')->passwordInput() ?>
```

Kemanan di sisi *client* tidak berhenti disitu saja, umumnya aplikasi berbasis web akan memaksa pengguna menggunakan protokol SSL atau https saat melakukan transaksi penting, salah satunya adalah transaksi login. Hal ini dimaksudkan untuk mencegah terbacanya kata sandi atau data sensitif yang dikirimkan melalui jaringan internet (*man in the middle attack*).

Demikian juga dari sisi server, etikanya, sebuah kata sandi tidak boleh disimpan begitu saja ke dalam basis data dalam bentuk *plain text*, melainkan perlu dienkripsi terlebih dahulu dengan menggunakan metode enkripsi searah (*one way encryption*) sehingga tidak bisa didekripsi lagi. Oleh karena itu, jika kita lupa kata sandi pada sebuah sistem yang *proven* keamanannya, maka sistem itu tidak akan mengingatkan kita dengan mengirim kata sandi ke *email*, melainkan melalui mekanisme ubah kata sandi.

A. 2. Bagaimana Kata Sandi yang Ideal itu?

Sebelum menjawab pertanyaan tersebut, marilah kita merenung sejenak dengan menjawab pertanyaan berikut:

- Apa yang biasanya kita gunakan sebagai kata sandi?
- Apakah kita biasa menggunakan kata sandi yang sama untuk banyak aplikasi?
- Apakah kita secara berkala mengganti kata sandi untuk keamanan?
- Seberapa sering kita lupa dengan kata sandi kita sendiri?
- Bagaimana kita mengingat & menjaga kata sandi kita?

Jawaban dari pertanyaan-pertanyaan di atas akan bisa menggambarkan tentang seberapa peduli kita terhadap kata sandi. Jadi intinya,

- Hendaknya kata sandi itu bukan merupakan bagian dari identitas kita, misal: nama, no telp, nama ortu, nama pacar, tanggal lahir, alamat dsb. Jika merupakan bagian dari identitas kita maka akan mudah ditebak. Solusinya misalnya dengan mengkombinasikan dengan kata lain atau menggunakan karakter alay.
- Hendaknya tidak semua aplikasi menggunakan kata sandi yang sama, kita bisa membaginya misalnya berdasarkan jenisnya.
- Hendaknya kata sandi secara berkala kita ganti.
- Agar mudah diingat maka hendaknya kata sandi dapat kita baca. Misal : “4ku \$4y4n6 k4mu!”, jadi bukan dengan mencatatnya.

Sekarang kebayangan bagaimana kata sandi yang ideal itu?. So, jangan asal unik, aneh dan panjang, yang justru akan merepotkan kita karena tiap kali login harus lihat kontekstan dulu.

Oke topik ini hanya menambah wawasan saja, jangan diambil hati ☺

A. 3. Penanganan Kata Sandi di Yii

Sebenarnya tidak ada penanganan khusus untuk kata sandi di Yii, sebagai contoh aturan agar kata sandi minimal 8 karakter, atau kata sandi harus merupakan kombinasi angka dan abjad, dsb. Namun aturan itu bisa kita terapkan melalui pendekatan umum.

Di Yii, kita bisa menerapkan aturan tersebut pada model, tepatnya pada fungsi *rules*. Berikut ini contohnya:

```
<?php
public function rules()
{
    return [
        ...
        ['password', 'required'],
        ['password', 'string', 'min' => 8],
    ];
}
```

Aturan di atas artinya:

- atribut *password* harus diisi (*required*),
- atribut *password* harus bertipe *string*
- panjang minimal atribut *password* harus 8 karakter

Untuk penanganan yang lebih spesifik, Yii menyediakan validasi tipe *match* dengan menggunakan *regular expression* atau *regex*.

Contoh 1:

```
[['password'], 'match',
    'pattern' => '[0-9]{6,}',
    'message' => 'Password harus angka min 6 karakter',
],
```

Penjelasan *pattern*:

- [0-9] artinya *password* hanya akan cocok jika diisi dengan angka
- {6, } artinya panjang karakter minimal enam dan maksimal tidak terbatas.

Jika tidak terpenuhi maka akan menampilkan pesan sesuai parameter *message* di atas.

Contoh: 123456

Contoh 2:

```
[ ['password'], 'match',
  'pattern' => '/[A-Z]{3,6}/i',
  'message' => 'Password harus abjad min 3
                  karakter dan max 6 karakter'
],
```

Penjelasan *pattern*:

- [A-Z] artinya *password* hanya akan cocok jika diisi dengan abjad
- {3,6} artinya panjang karakter minimal 3 dan maksimal 6.
- /i artinya abjadnya boleh huruf kecil atau kapital

Contoh: abc, ABC, Abc

Contoh 3:

```
[ ['password'], 'match',
  'pattern' => '/[A-Z0-9]{6,}/i',
  'message' => 'Password boleh kombinasi
                  angka dan abjad min 6 karakter'
],
```

Penjelasan *pattern*:

- [A-Z0-9] artinya *password* boleh kombinasi abjad dan atau angka
- {6,} artinya panjang karakter minimal 6
- /i artinya abjadnya boleh huruf kecil atau kapital

Contoh: abcdefg, 123456, abc123, 123ABC

Contoh 4:

```
[ ['password'], 'match',
  'pattern' => '(([A-Z][0-9] | [0-9][A-Z]) {6,})/i',
  'message' => 'Password harus kombinasi
                  angka dan abjad min 6 karakter'
],
```

Penjelasan *pattern*:

- [A-Z][0-9] artinya *password* harus diawali dengan abjad dan diikuti angka
- | atau
- [0-9] [A-Z] artinya *password* harus diawali dengan angka dan diikuti abjad
- {6,} artinya panjang karakter minimal 6
- /i artinya abjadnya boleh huruf kecil atau kapital

Contoh: abc123, 123abc

Contoh 5:

```
[ ['password'], 'match',
  'pattern' => '(?=.*[A-Z])(?=.*[a-z])(?=.*\d)\w{6,}',
  'message' => 'Password harus kombinasi
                  angka dan abjad, min 1 huruf kapital,
                  1 huruf kecil dan 1 angka, serta
                  min 6 karakter'
],
```

Penjelasan *pattern*:

- (?=.*[A-Z]) artinya minimal ada 1 huruf kapital
- (?=.*[a-z]) artinya minimal ada 1 huruf kecil
- (?=.*[0-9]) artinya minimal ada 1 angka
- \w artinya abjad atau angka
- {6,} artinya panjang karakter minimal 6, misal: Abc123, 123Abc, 1A2b3C

Bagaimana? mudah sekali bukan?

Selengkapnya kita bisa baca-baca di sini

∞ http://www.tutorialspoint.com/php/php_regular_expression.htm

dan untuk uji coba *regular expression* secara daring bisa di sini

∞ <https://regex101.com>

A. 4. Enkripsi & Verifikasi Kata Sandi

Sebagaimana yang telah dijelaskan di atas tentang keamanan kata sandi, maka peran dari Yii adalah melakukan enkripsi di sisi server sebelum disimpan ke dalam basis data. Enkripsi yang dimaksud di sini adalah metode enkripsi searah atau *hashing*.

Pada umumnya, banyak *programmer* PHP melakukan hashing kata sandi dengan menggunakan algoritma MD5 or SHA1 yang merupakan metode enkripsi searah yang cukup populer. Namun pada perkembangannya, dengan kecanggihan perangkat komputer, saat ini kita bisa mereverse “mendekripsi” kata sandi yang dienkripsi menggunakan MD5 yaitu dengan metode *brute force attacks*.

Oleh karena itu, untuk meningkatkan keamanan kata sandi, berkembanglah metode-metode enkripsi lain untuk menanggulangi serangan brute force, dan pilih terbaik saat ini adalah *bcrypt* atau di PHP kita bisa menggunakan fungsi *crypt*.

Yii sendiri menyediakan dua fungsi *helpers* untuk melakukan *hashing* menggunakan *crypt* dan untuk melakukan verifikasi terhadap hasil *hashing*.

Enkripsi Password

Untuk melakukan enkripsi menggunakan Yii, caranya cukup mudah yaitu menggunakan fungsi `generatePasswordHash` dengan parameter *plain text* dari kata sandi, sebagai berikut:

```
| $hash = Yii::$app->getSecurity()->generatePasswordHash ($password) ;
```

Variabel `$hash` akan berisi nilai dari *password hash*.

```
| A9NNIGDc36nAQUiQVbMe3CR9xZ_8x489
```

Dan uniknya, hasil dari *hash* akan berubah ubah sesuai waktu. Ini lah yang menyebabkan *reverse* dari metode enkripsi ini jauh lebih sulit dan butuh waktu lama.

Nilai *hash* kata sandi ini lah yang nantikan akan disimpan ke dalam basis data. Sehingga siapapun termasuk juga admin atau pemilik aplikasi tidak tahu apa kata sandi dari pengguna aplikasinya. Demikianlah memang idealnya.

Verifikasi Kata Sandi

Verifikasi kata sandi digunakan untuk mengecek apakah kata sandi yang dimasukkan oleh pengguna sesuai dengan yang telah tersimpan di basis data. Proses ini biasanya dilakukan ketika *login*.

Karena kata sandi tersimpan dalam bentuk *hash* dan tidak bisa didekripsi maka untuk mengecek apakah pengguna menginputkan kata sandi yang sesuai atau tidak pada saat login, diperlukan suatu metode verifikasi tertentu. Yii menggunakan fungsi `validatePassword()` dengan dua parameter yaitu kata sandi input dari pengguna dan *password hash* dari basis data.

```
| if (Yii::$app->getSecurity()->validatePassword($password, $hash)) {
    // all good, logging user in
} else {
    // wrong password
}
```

A. 5. Generate Random Teks

Di samping fungsi untuk enkripsi dan verifikasi kata sandi, Yii juga menyediakan fungsi untuk meng-*generate* teks secara acak. Kapan kita menggunakan ini? Sebagai contoh ketika kita mengeset ulang kata sandi melalui email maka kita perlu meng-*generate token*, dan menyimpan *token* tersebut ke basis data serta mengirimkannya ke pengguna melalui *email* untuk mengecek apakah pengguna yang meminta *reset* kata sandi itu memang benar-benar pengguna pemilik akun. Intinya kode *token* yang di-*generate* harus unik sehingga jangan sampai mudah ditebak oleh *cracker*.

Untuk meng-*generate* teks secara acak di Yii kita bisa menggunakan fungsi berikut:

```
| $key =Yii::$app->getSecurity()->generateRandomString();
```

Perhatian:

Untuk menggunakan fitur ini maka *extension OpenSSL* harus terinstalasi dan *enable* di PHP

Silakan berimajinasi, dan jangan ragu-ragu untuk memanfaatkan fitur keren ini.

A. 6. Enkripsi & Dekripsi Data

Selain menyediakan metode enkripsi searah yang tidak bisa didekripsi, Yii juga menyediakan metode enkripsi data yang bisa didekripsi. Misalnya: ada data yang sangat rahasia sehingga hanya orang-orang tertentu saja yang boleh membacanya maka kita bisa menggunakan metode ini. Contoh lain adalah untuk menyimpan data *cookie* di *web browser*, supaya tidak mudah dibaca pengguna maka kita perlu mengekripsinya(Yii telah melakukannya pada kasus *cookie*).

Fungsi yang digunakan adalah `encryptByPassword()` dengan dua parameter yaitu :

- \$data = data teks yang akan dienkripsi
 - \$secretKey = kata sandi untuk enkripsi / dekripsi data, bebas
- ```
| $encryptedData = Yii::$app->getSecurity()->encryptByPassword($data, $secretKey);
```

Sedangkan untuk dekripsi kita bisa gunakan fungsi `decryptByPassword`

```
| $data = Yii::$app->getSecurity()->decryptByPassword($encryptedData, $secretKey);
```

Gunakan sesuai dengan kebutuhan.

## B. Login & Register Menggunakan Basis Data

Sebenarnya yii telah membuatkan *template advance* yang telah lengkap otentikasinya yaitu *login* dan *register* menggunakan basis data. Adapun di *template basic*, otentikasinya belum menggunakan basis data alias masih statis.

Tujuan dari panduan ini adalah agar kita bisa lebih memahami tentang bagaimana otentikasi di Yii bekerja, serta bisa menerapkannya pada *template basic*. Oleh karena itu, studi kasus yang dipakai pada bagian ini adalah menggunakan *template basic*.

### A. 1. Persiapan Basis Data

Oleh karena *login* menggunakan basis data, maka kita memerlukan membuat sebuah tabel untuk menampung data *username* dan kata sandi *login* dari pengguna atau kemudian kita sebut sebagai tabel *user*.

| user                 |                |
|----------------------|----------------|
| id                   | : int(11)      |
| username             | : varchar(50)  |
| auth_key             | : varchar(32)  |
| password_hash        | : varchar(255) |
| password_reset_token | : varchar(255) |
| email                | : varchar(255) |
| status               | : smallint(6)  |
| created_at           | : int(11)      |
| updated_at           | : int(11)      |

Struktur dari tabel *user* di atas merujuk pada praktik terbaik dari Yii. Kita bisa melihatnya pada tautan berikut.

∞ [https://github.com/yiisoft/yii2-app-advanced/blob/master/console/migrations/m130524\\_201442\\_init.php](https://github.com/yiisoft/yii2-app-advanced/blob/master/console/migrations/m130524_201442_init.php)

```
$this->createTable('{{%user}}', [
 'id' => $this->primaryKey(),
 'username' => $this->string()->notNull()->unique(),
 'auth_key' => $this->string(32)->notNull(),
 'password_hash' => $this->string()->notNull(),
 'password_reset_token' => $this->string()->unique(),
 'email' => $this->string()->notNull()->unique(),
 'status' => $this->smallInteger()->notNull()->defaultValue(10),
 'created_at' => $this->integer()->notNull(),
 'updated_at' => $this->integer()->notNull(),
], $tableOptions);
```

Kode di atas adalah kode untuk migrasi basis data pada *template advanced*. Salin kode di atas dan letakkan ke dalam direktori *@app/migrations/*

### Apa itu Migrasi Basis Data?

Migrasi basis data adalah fitur untuk migrasi tabel pada basis data melalui CMD.

Mengapa kita perlu membuat migrasi padahal kita bisa menggunakan fitur impor SQL?

Tujuannya adalah agar pengembangan basis data juga masuk menjadi bagian dari pengembangan aplikasi. Pada saat pengembangan, perubahan dari struktur basis data masih dimungkinkan terjadi, misalnya ketika terjadi penambahan atau perubahan fitur aplikasi. Dengan adanya migrasi basis data maka selain memudahkan saat *deployment* juga memudahkan untuk melacak histori perubahan kode kita.

Secara *default, file* migrasi basis data diletakkan pada direktori @app/migrations/, sebagaimana yang dijelaskan di atas. Kemudian kita bisa menjalankan perintah untuk migrasi basis data dengan menggunakan kode berikut pada CMD.

```
| yii migrate
```

c:\Bitnami\wampstack\apache2\htdocs\basic> yii  
migrate  
Yii Migration Tool (based on Yii v2.0.6)  
Total 1 new migration to be applied:  
m130524\_201442\_init  
Apply the above migration? (yes|no) [no]:

Lalu ketikkan:

```
| yes
```

Kemudian tekan *enter* dan selesai.

Apply the above migration? (yes|no) [no]:yes  
\*\*\* applying m130524\_201442\_init  
 > create table {{%user}} ... done (time: 0.266s)  
\*\*\* applying m130524\_201442\_init (time: 0.439s)  
  
Migrated up successfully.

Silakan cek di basis data, maka akan tercipta tabel *user*..

Lalu bagaimana dengan tabel yang lain?

Yii hanya menyediakan untuk tabel *user* saja, jika kita ingin implementasikan untuk tabel lain maka kita perlu membuat migrasinya sendiri.

### Membuat Migrasi Basis Data

Setelah kita mengetahui tentang penggunaan migrasi basis data, maka sekarang saatnya kita belajar untuk membuatnya sendiri. Kita akan banyak bermain dengan CMD.

Mari kita awali dengan membuat kode dasar dari migrasi basis data. Berikut ini formatnya.

```
| yii migrate/create <name>
```

Sebagai contoh, kita akan membuat migrasi untuk tabel *student* dengan struktur tabel sebagai berikut.

| Kolom          | Keterangan                           |
|----------------|--------------------------------------|
| <i>id</i>      | <i>Primary Key, Auto Increment</i>   |
| <i>name</i>    | <i>Varchar(50), Not Null</i>         |
| <i>code</i>    | <i>Varchar(15), Not Null, Unique</i> |
| <i>gender</i>  | <i>Boolean, Not Null, Default 1</i>  |
| <i>age</i>     | <i>Integer(3), Null</i>              |
| <i>address</i> | <i>Text, Null</i>                    |

Maka jalankan kode berikut

```
| yii migrate/create create_student_table
```

```
c:\Bitnami\wampstack\apache2\htdocs\basic> yii
migrate/create create_student_table
Yii Migration Tool (base on Yii v2.0.6)

Create new migration
'c:\Bitnami\wampstack\apache2\htdocs\basic\migrations
\m160112_231852_create_student_table.php'? (yes|no)
[no]:yes
New migration created successfully.
```

Maka pada direktori @app/migrations akan tercipta file dengan nama m160112\_231852\_create\_student\_table.php. Nama file tersebut sesuai dengan nama class-nya, yang mana nama tersebut di-generate oleh Yii dengan format m<YYMMDD\_HHMMSS>\_<Name>. Sehingga akan berbeda jika anda mempraktekkannya sekarang.

Lalu apa isi dari file migrasi basis data yang di-generate?

```
<?php

use yii\db\Schema;
use yii\db\Migration;

class m160112_231852_create_student_table extends Migration
{
 public function up()
 {

 }

 public function down()
 {
 echo "m160112_231852_create_student_table cannot be reverted.\n";
 return false;
 }

 /*
 // Use safeUp/safeDown to run migration code within a transaction
 public function safeUp()
 {

 }

 public function safeDown()
 {
 }
 */
}
```

Yii hanya membuatkan kode dasarnya, sedangkan implementasinya diserahkan kepada kita. Ada dua fungsi yaitu *up* dan *down*. Fungsi *up* itu isinya nanti adalah kode untuk membuat atau memodifikasi tabel basis data, sedangkan *down* itu isinya adalah kode untuk menghapus tabel basis data.

```
public function up()
{
 $this->createTable('student', [
 'id' => $this->primaryKey(),
 'name' => $this->string(50)->notNull(),
 'code' => $this->string(15)->notNull()->unique(),
 'gender' => $this->boolean()->notNull()->defaultValue(1),
 'age' => $this->integer(3),
 'address' => $this->text(),
]);
}

public function down()
```

```

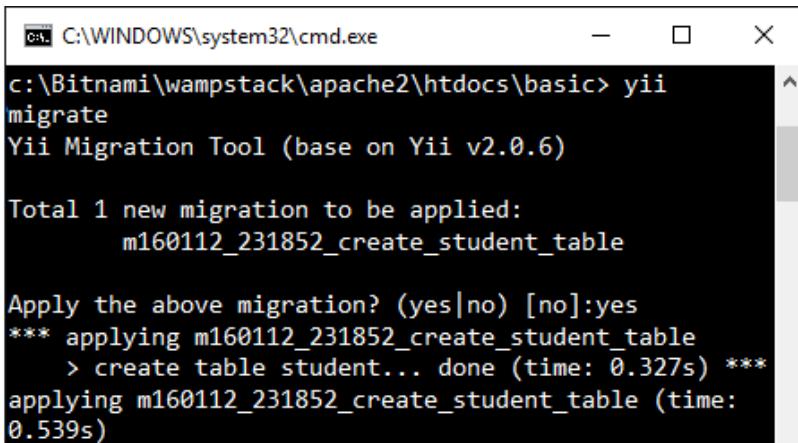
 echo "m160112_231852_create_student_table can't be reverted.\n";
 $this->dropTable('student');
 return false;
}

```

Setelah itu, kita bisa instalasi tabel *student* melalui CMD

```
| yii migrate
```

Maka Yii akan otomatis mencari *file* migrasi basis data di direktori @app/migrations, jika ditemukan maka akan muncul konfirmasi apakah kita akan memigrasinya? Maka ketik yes.



```

C:\WINDOWS\system32\cmd.exe
c:\Bitnami\wampstack\apache2\htdocs\basic> yii
migrate
Yii Migration Tool (base on Yii v2.0.6)

Total 1 new migration to be applied:
 m160112_231852_create_student_table

Apply the above migration? (yes|no) [no]:yes
*** applying m160112_231852_create_student_table
 > create table student... done (time: 0.327s)
applying m160112_231852_create_student_table (time:
0.539s)

```

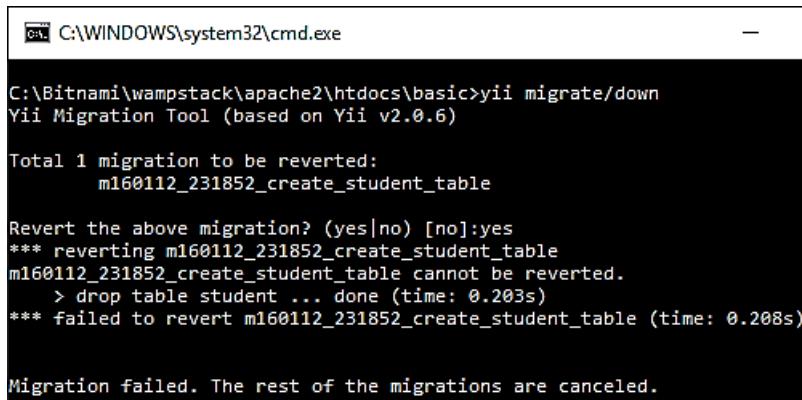
Mari kita lihat hasilnya di basis data.

| Name           | Type        | Collation       | Attributes | Null | Default | Extra          |
|----------------|-------------|-----------------|------------|------|---------|----------------|
| <b>id</b> 🌈    | int(11)     |                 |            | No   | None    | AUTO_INCREMENT |
| <b>name</b>    | varchar(50) | utf8_general_ci |            | No   | None    |                |
| <b>code</b> 💬  | varchar(15) | utf8_general_ci |            | No   | None    |                |
| <b>gender</b>  | tinyint(1)  |                 |            | No   | 1       |                |
| <b>age</b>     | int(3)      |                 |            | Yes  | NULL    |                |
| <b>address</b> | text        | utf8_general_ci |            | Yes  | NULL    |                |

Jika kita melihat tampilan seperti diatas berarti kita berhasil membuat tabel ini.

Nah karena ini hanya coba-coba, mari kita hapus tabel ini. Caranya juga melalui CMD

```
| yii migrate/down
```



```

C:\WINDOWS\system32\cmd.exe
C:\Bitnami\wampstack\apache2\htdocs\basic>yii migrate/down
Yii Migration Tool (based on Yii v2.0.6)

Total 1 migration to be reverted:
 m160112_231852_create_student_table

Revert the above migration? (yes|no) [no]:yes
*** reverting m160112_231852_create_student_table
m160112_231852_create_student_table cannot be reverted.
 > drop table student ... done (time: 0.203s)
*** failed to revert m160112_231852_create_student_table (time: 0.208s)

Migration failed. The rest of the migrations are canceled.

```

Sekarang jika kita cek di basis data, maka tidak akan kita temui lagi yang namanya tabel *student*.

Mudah sekali bukan?

Selengkapnya silakan baca di *guide* Yii pada topik “Database Migration”.

## A. 2. Membuat Fitur Login

Setelah tabel *user* siap, maka langkah selanjutnya adalah implementasi otentikasi di Yii.

### Konfigurasi Komponen User

Ada sebuah komponen yang bertugas mengelola status otentikasi pengguna yaitu komponen *user*. Komponen ini memiliki parameter *identityClass* yang merujuk pada suatu model. Model inilah yang nantinya mengimplementasikan *class* [[yii\web\IdentityInterface]] yaitu sebuah *class* untuk otentikasi.

Untuk mengkonfigurasi komponen *user*, kita bisa buka file @app/config/web.php, pada bagian *components*,

```
...
$config = [
 ...
 'components' => [
 ...
 'user' => [
 'identityClass' => 'app\models\User',
 'enableAutoLogin' => true,
],
 ...
]
]
```

Pastikan konfigurasinya seperti di atas yang mana *identityClass* merujuk ke model *User* yang terletak di app\models\User. Konfigurasi ini sesuai dengan bawaan dari *template basic* sehingga tidak perlu diotak-atik lagi.

### Implementasi [[yii\web\IdentityInterface]] pada model User

Sebagaimana yang disebutkan sebelumnya bahwa proses otentikasi di Yii bertumpu pada *class* IdentityInterface, oleh karena itu model yang digunakan untuk otentikasi yang dalam hal ini adalah model *User* harus mengimplementasikan *class* IdentityInterface ini.

Sekarang mari kita bedah, *class* IdentityInterface memiliki beberapa fungsi yaitu:

- *findIdentity()*: digunakan untuk mendapatkan *instance* dari *class identity* menggunakan ID pengguna spesifik. Hal ini berlaku ketika kita menangani status *login* menggunakan *session*.
- *findIdentityByAccessToken()*: hampir sama dengan fungsi sebelumnya yaitu untuk mendapatkan *instance* dari *identity class* menggunakan *access token*. Hal ini berlaku biasanya digunakan untuk mekanisme otentikasi pada aplikasi RESTful yang *stateless* (tidak mendukung *session*).
- *getId()*: fungsi ini mengembalikan ID dari pengguna yang telah di-*instance* menggunakan fungsi di atas.
- *getAuthKey()*: fungsi ini mengembalikan nilai *key* yang digunakan untuk memverifikasi *login* berbasis *cookie*.
- *validateAuthKey()*: fungsi ini berfungsi memverifikasi kevalidan dari *key* yang tersimpan di *cookie web browser* pada login berbasis *cookie*.

Tidak semua fungsi tersebut digunakan, namun wajib diimplementasikan meskipun berupa fungsi kosong. Sebagai contoh, pada aplikasi RESTful maka kita cukup menggunakan *findIdentityByAccessToken()* dan *getId()*.

Okey, mari kita buka model *User*

```
<?php
namespace app\models;
class User extends \yii\base\Object implements \yii\web\IdentityInterface
{
 public $id;
 public $username;
 public $password;
 public $authKey;
 public $accessToken;
 private static $users = [
 '100' => [
 'id' => '100',
 'username' => 'admin',
 'password' => 'admin',
 'authKey' => 'test100key',
 'accessToken' => '100-token',
],
];
}
```

Model *User* bawaan *template basic* telah mengimplementasikan *class* IdentityInterface, hanya saja data penggunanya masih statis dan belum terkoneksi ke tabel *user* melalui *class* ActiveRecord.

Tugas kita adalah mengubahnya agar terkoneksi ke tabel *user*. Secara umum, berikut ini langkah-langkahnya.

1. Ubah *extendsnya* ke *class* [[yii\db\ActiveRecord]];
2. Tambahkan konstanta status *user*
3. Tambahkan fungsi *tableName* yang mengembalikan nama tabel *user*.
4. Tambahkan *behavior* (opsional) untuk otomatisasi *input* atribut *created\_at* dan *updated\_at*
5. Tambahkan fungsi *rules*

@app/models/User.php

```
<?php
namespace app\models;
use Yii;
use yii\web\IdentityInterface;
use yii\db\ActiveRecord;
use yii\behaviors\TimestampBehavior;

class User extends ActiveRecord implements IdentityInterface
{
 const STATUS_DELETED = 0;
 const STATUS_ACTIVE = 10;

 public static function tableName()
 {
 return '{{%user}}';
 }

 public function behaviors()
 {
 return [
 TimestampBehavior::className(),
];
 }

 public function rules()
 {
 return [
 ['status', 'default', 'value' => self::STATUS_ACTIVE],
 ['status', 'in', 'range' => [self::STATUS_ACTIVE, self::STATUS_DELETED]],
];
 }
}
```

Kemudian modifikasi fungsi yang berasal dari implementasi *class Identity Interface* sebagaimana yang dijelaskan sebelumnya.

```
public static function findIdentity($id)
{
 return static::findOne(['id' => $id, 'status' => self::STATUS_ACTIVE]);
}

public static function findIdentityByAccessToken($token, $type = null)
{
}

public function getId()
{
 return $this->primaryKey();
}

public function getAuthKey()
{
 return $this->auth_key;
}

public function validateAuthKey($authKey)
{
 return $this->getAuthKey() === $authKey;
}
```

Kemudian tambahkan dua fungsi lagi yaitu *findByUsername* dan *validatePassword*. Untuk *findByUsername* silakan disesuaikan

```
public static function findByUsername($username)
{
 return static::findOne(['username' => $username, 'status' => self::STATUS_ACTIVE]);
}
```

sedangkan untuk validatePassword, maka gunakan fungsi validatePassword

```
public function validatePassword($password)
{
 return Yii::$app->security->validatePassword($password, $this->password_hash);
}
```

Tambahkan dua fungsi lagi yang berguna jika kita ingin membuat fungsi untuk mengubah kata sandi.

```
public function generatePasswordResetToken()
{
 $this->password_reset_token = Yii::$app->security-
>generateRandomString() . '_' . time();
}

public function removePasswordResetToken()
{
 $this->password_reset_token = null;
}
```

### ***Uji coba login***

Sebelum kita melakukan uji coba *login*, maka kita perlu meregistrasikan data pengguna ke tabel secara manual. Hal ini dikarenakan kita belum membuat fitur *signup*. Silakan buka basis data, tabel *user*, lalu tambahkan data pengguna.

Field yang diisi cukup tiga *field* saja yaitu: *username*, *password\_hash*, dan *email*. Untuk *username* dan *email* bisa diisi bebas misalnya: admin, admin@gmail.com, sedangkan *password\_hash* bisa dihasilkan dengan menggunakan fungsi berikut.

```
$password_hash = Yii::$app->getSecurity()->generatePasswordHash(' kata sandi
pilihanmu');
```

Tampilkan variabel *\$password\_hash* di atas pada *view*, pada contoh ini, kita bisa meletakkan kode berikut di @app/views/site/index.php

```
| echo Yii::$app->getSecurity()->generatePasswordHash('123456');
```

---

Setelah mendapatkan hasil *hash password*, hapus kode tersebut dari *view index.php* □

---

Lalu kita akses *view site/index*

localhost/basic/web/index.php



\$2y\$13\$qsMMc1phl6LureFFqfx5PuaCWjYV5GcETtqfLudpn8GTSIILrKecu

sehingga bisa kita salin dan *insert* ke basis data.

```
INSERT INTO `user` (`id`, `username`, `auth_key`, `password_hash`, `password_reset_to
ken`, `email`, `status`, `created_at`, `updated_at`)
VALUES (NULL, 'admin', 'rahasia-
gan', '$2y$13$qsMMc1phl6LureFFqfx5PuaCWjYV5GcETtqfLudpn8GTSIILrKecu', NULL, 'admin@gm
ail.com', '10', '', ''');
```

Jika sudah di-*input* ke dalam basis data, maka langkah selanjutnya adalah *login* ke aplikasi menggunakan data pengguna yang baru kita masukkan tadi.

Home / Login

## Login

Please fill out the following fields to login:

|                                                 |       |
|-------------------------------------------------|-------|
| Username                                        | admin |
| Password                                        | ..... |
| <input checked="" type="checkbox"/> Remember Me |       |
| <input type="button" value="Login"/>            |       |

Silakan tekan tombol *login* dan kita akan berhasil masuk ke *sistem* ☺. Wah bisa jadi *hacker* dong kita? Hehe.

### Menggunakan [[yii\web\User]]

Beberapa hal yang perlu kita ketahui berkaitan dengan *class* [[yii\web\User]] terutama terkait dengan *login* pengguna.

- Mengecek apakah sudah login atau belum

```
// whether the current user is a guest (not authenticated)
$isGuest = Yii::$app->user->isGuest;
```

Jika \$isGuest *true* maka pengguna belum *login*, dan hal ini biasanya digunakan untuk metode untuk menampilkan atau menyembunyikan menu.

- Mengakses atribut pengguna yang sedang *login*

```
// the current user identity. Null if the user is not authenticated.
$identity = Yii::$app->user->identity;
$identity->username;
$identity->email;
$identity->status;

// the ID of the current user. Null if the user not authenticated.
$id = Yii::$app->user->id;
```

- Proses *Login*

```
// find a user identity with the specified username.
// note that you may want to check the password if needed
$identity = User::findOne(['username' => $username]);

// logs in the user
Yii::$app->user->login($identity);
```

- Proses *Logout*

```
| Yii::$app->user->logout();
```

## A. 3. Membuat Fitur Registrasi

Bagian ini merupakan tambahan saja, jika sebelumnya kita registrasi data pengguna secara manual maka sekarang kita akan buatkan *form*-nya.

### Menambahkan Beberapa Fungsi Pada Model User

Sebelum kita membuat formulir *signup*, kita perlu menambahkan beberapa *method* pada model *User* yang berguna untuk menyederhanakan proses *signup*. Yaitu *setPassword* untuk menghash kata sandi dan *generateAuthKey* untuk meng-generate string *auth\_key* (digunakan untuk verifikasi fitur *remember* pada *login*)

```
public function setPassword($password)
{
 $this->password_hash = Yii::$app->security->generatePasswordHash($password);
}

public function generateAuthKey()
{
 $this->auth_key = Yii::$app->security->generateRandomString();
}
```

## Membuat Model SignupForm

Buat model @app/models/SignupForm.php

```
<?php
namespace app\models;
use app\models\User;
use yii\base\Model;
use Yii;
class SignupForm extends Model
{
 public $username;
 public $email;
 public $password;
 public function rules()
 {
 return [
 ['username', 'filter', 'filter' => 'trim'],
 ['username', 'required'],
 ['username', 'unique', 'targetClass' => '\app\models\User', 'message' =>
'This username has already been taken.'],
 ['username', 'string', 'min' => 2, 'max' => 255],
 ['email', 'filter', 'filter' => 'trim'],
 ['email', 'required'],
 ['email', 'email'],
 ['email', 'string', 'max' => 255],
 ['email', 'unique', 'targetClass' => '\app\models\User', 'message' => 'Th
is email address has already been taken.'],
 ['password', 'required'],
 ['password', 'string', 'min' => 6],
];
 }
 public function signup()
 {
 if ($this->validate()) {
 $user = new User();
 $user->username = $this->username;
 $user->email = $this->email;
 $user->setPassword($this->password);
 $user->generateAuthKey();
 if ($user->save()) {
 return $user;
 }
 }
 return null;
 }
}
```

Kita bisa menerapkan aturan kata sandi di sini, sebagaimana yang telah dibahas pada bagian sebelumnya sehingga keamanannya lebih terjaga.

Fungsi *signup* pada model ini akan diakses ketika formulir *signup* di-submit yaitu di SiteController fungsi actionSignup. Fungsinya adalah men-create pengguna baru sesuai dengan *username*, kata sandi dan *email* dari formulir *signup* yang di-input oleh pengguna.

## Membuat View Formulir Signup

Untuk lebih mudahnya, silakan buat file @app/views/site/signup.php yang isinya sama dengan formulir *login*. Kemudian silakan disesuaikan.

```
<?php $form = ActiveForm::begin(['id' => 'form-signup']); ?>
<?= $form->field($model, 'username') ?>
<?= $form->field($model, 'email') ?>
<?= $form->field($model, 'password')->passwordInput() ?>
<div class="form-group">
 <?= Html::submitButton('Signup', ['class' => 'btn btn-
primary', 'name' => 'signup-button']) ?>
</div>
<?php ActiveForm::end(); ?>
```

### Membuat Fungsi actionSignup pada Class SiteController

Tambahkan fungsi actionSignup pada class app\controllers\SiteController

```
public function actionSignup()
{
 $model = new \app\models\SignupForm();
 if ($model->load(Yii::$app->request->post()) && $model->signup()) {
 Yii::$app->getUser()->login($user);
 return $this->goHome();
 }
}

return $this->render('signup', [
 'model' => $model,
]);
}
```

fungsinya adalah untuk menampilkan formulir *signup* dan memprosesnya.

Sebagaimana yang dijelaskan pada model SignupForm di atas bahwa ada pemanggilan method *signup* untuk men-create pengguna baru, jika berhasil maka proses *login* dijalankan.

### Menambahkan Tautan Signup Pada Menu

Karena berada di SiteController dan actionSignup maka kita bisa mengakses formulir *signup* melalui tautan site/signup, untuk memudahkan pengguna agar tidak perlu mengetikkan URL tersebut setiap kali melakukan registrasi maka kita bisa tambahkan tautan tersebut pada menu utama.

Caranya, silakan buka file @app/views/layouts/main.php

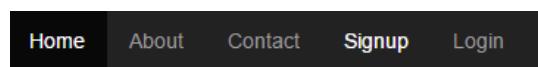
Tambahkan menu tersebut di bawah menu *Contact*

```
| ['label' => 'Signup', 'url' => ['/site/signup']],
```

Kira-kira sebagai berikut:

```
echo Nav::widget([
 'options' => ['class' => 'navbar-nav navbar-right'],
 'items' => [
 ['label' => 'Home', 'url' => ['/site/index']],
 ['label' => 'About', 'url' => ['/site/about']],
 ['label' => 'Contact', 'url' => ['/site/contact']],
 ['label' => 'Signup', 'url' => ['/site/signup']],
],
 Yii::$app->user->isGuest ? (
```

Maka jika kita akses melalui aplikasi, akan muncul menu baru yaitu *Signup*



Jika kita klik menu *Signup* akan masuk ke view formulir *signup*

## Signup

Please fill out the following fields to signup:

**Username**

**Email**

**Password**

**Signup**

Lakukan registrasi. Selesai.

## B. Login Melalui Akun Social Media

Login melalui *social media* merupakan fitur yang umumnya dimiliki oleh aplikasi web modern untuk tujuan kemudahan. Hal ini tentunya memberikan pilihan menarik bagi pengguna untuk masuk ke dalam aplikasi kita, cukup dengan menggunakan akun *social media* yang telah mereka miliki.

### A. 1. Definisi

Mekanisme otentifikasi aplikasi menggunakan akun *social media* dimungkinkan karena adanya standardisasi teknologi seperti: Oauth/OAuth2 dan OpenID.

#### **Apa itu OAuth?**

Menurut wikipedia (<http://en.wikipedia.org/wiki/OAuth>) adalah singkatan dari *open standard to authorization*. OAuth sebuah protokol yang membuat aplikasi kita bisa mengakses dengan aman data pengguna pada aplikasi server yang mendukung OAuth tentunya dengan persetujuan dari si pengguna itu sendiri.. OAuth sampai buku ini disusun telah mencapai versi 2.0. Hampir semua situs web besar mendukung OAuth sebut saja Google, Facebook, Twitter.

∞ <http://oauth.net/2/>

#### **Apa itu OpenID?**

OpenID menggunakan protokol OAuth. Misi open ID adalah membuat satu akun aja untuk semua situs web. Kalo selama ini kita membuat akun satu per satu di setiap situs web misal Google akun sendiri, facebook, twitter, dll. Mengapa kita tidak membuat satu akun saja yang bisa digunakan di semua situs web?.

∞ <http://openid.net/>

### A. 2. Skenario

Ketika aplikasi kita menerapkan fitur *login* menggunakan *social media*, maka aplikasi akan mendapatkan data informasi tentang pengguna yang *login* menggunakan *social media* tersebut. *Social media* yang akan kita gunakan pada panduan ini adalah Facebook, Google, Twitter, Github. Namun percayalah, jika kita bisa mengimplementasikan ke satu *social media* maka akan mudah melakukan untuk yang lain.

Jenis data yang diperoleh dari proses otentifikasi menggunakan *social media* bisa berbeda-beda tergantung dari *social media* yang digunakan.

- Facebook : *name, email* dan ID unik.
- Google: *name[], displayName, email, ID unik, gender, dsb*
- Twitter: *name, screen name, location, description, ID unik, dll*
- Github: *login, name, ID unik, email, dll*

Untuk apa data tersebut? dan bagaimana proses *login*-nya?

Ada banyak mekanisme otentifikasi menggunakan *social media*, karena semua terserah masing-masing pemrogram aplikasi. Namun untuk menyamakan persepsi dan memudahkan kita dalam memahami panduan ini, maka berikut ini skenarionya:

1. Pengguna memiliki pilihan untuk *login* menggunakan akun *social media* mereka. Adapun tautan untuk *login* akan kita letakkan di bagian bawah formulir *login* normal

2. Ketika pengguna menekan tautan *login* melalui *social media* tertentu, maka akan muncul *popup login* ke *social media* tersebut dan formulir persetujuan untuk mengakses data pribadi pengguna dari akun tersebut.
3. Jika diijinkan oleh pengguna, maka aplikasi akan mendapatkan data pengguna yang *login* sebagaimana yang kita singgung di atas, kita fokus ketiga data yaitu ID unik, *username*, nama dan *email*.
4. Data tersebut akan kita gunakan untuk mengecek adakah pengguna di basis data aplikasi kita yang memiliki id/*username* yang sama dengan id/*username* dari social media tersebut (id dan *username social media* disimpan pada tabel *user\_social\_media*).
5. Jika id/*username* tersebut ditemukan maka pengguna di-*login*-kan secara otomatis. Selesai
6. Jika id/*username* tidak ditemukan maka dicek apakah *email* dari *social media* tersebut telah terdaftar pada tabel *user*
7. Jika terdaftar maka akun *social media* tersebut dimasukkan ke dalam tabel *user* dimaksud, dan di-*login*-kan secara otomatis. Selesai
8. Jika belum terdaftar, maka akan didaftarkan dan di-*login*-kan secara otomatis menggunakan data tersebut kecuali untuk *social media* Twitter. Selesai.
9. Karena secara *default* data dari twitter tidak ada data *email* maka kita akan dialihkan ke formulir *signup* agar pengguna melakukan registrasi.

Kira-kira demikian skenario yang akan kita gunakan pada panduan ini. Jika sudah memahaminya maka anda bisa menggunakan skenario lain yang mungkin lebih sesuai dengan kebutuhan anda.

Jangan lupa karena pada skenario ini kita menggunakan tambahan yaitu tabel *user\_social\_media*, maka kita harus membuat tabel tersebut dengan struktur sebagai berikut.

<b>user_social_media</b>	
◆	<i>social_media</i> : enum('facebook','google','twitter','github')
◎	<i>id</i> : varchar(255)
◎	<i>username</i> : varchar(255)
◎	<i>user_id</i> : int(11)
#	<i>created_at</i> : int(11)
#	<i>updated_at</i> : int(11)

### A. 3. Instalasi Extension

Yii telah menyediakan *extension* untuk menerapkan fitur ini pada aplikasi kita, yang bernama *yii2-authclient*. *Extension AuthClient* ini adalah *extension official* Yii 2 yang mendukung otentifikasi melalui *third-party service*: Oauth, OAuth2 dan OpenID. *Extension* ini mendukung banyak *social media*, antara lain: Facebook, Google, Github, Twitter, dll. Dan jika itu masih kurang maka kita bisa menambahkan sendiri.

Meski merupakan *extension* yang dibuat oleh *Yii dev*, namun *extension* ini tidak otomatis terinstalasi sewaktu kita instalasi Yii. Oleh karenanya kita perlu melakukan instalasi, sebagai berikut:

```
| composer require --prefer-dist yiisoft/yii2-authclient "*"
```

atau silakan ikuti dokumentasinya di

∞ <http://www.yiiframework.com/doc-2.0/ext-authclient-index.html>

### A. 4. Token Social Media

Untuk alasan keamanan, komunikasi antar dua aplikasi yaitu aplikasi kita dan aplikasi *social media* harus menggunakan *token* atau kunci. *Token* berbeda dengan dengan kata sandi untuk *login* ke *social media* tersebut. Oleh karena itu untuk menerapkan fitur ini maka kita perlu mendapatkan *token* tersebut. *Token* tersebut bisa kita dapatkan melalui *social media* penyedia *service* ini.

Berikut ini beberapa contoh cara untuk mendapatkan kunci dari beberapa *social media*.

#### Perhatian:

Perubahan *user interface* dari *social media* begitu cepat, maka bisa jadi ketika anda membaca panduan ini maka tampilan atau langkah-langkahnya mungkin sudah berbeda. Cukup, ambil “benang merah” dari panduan ini

#### Facebook

Berikut ini langkah-langkah untuk mendapatkan *token* Facebook.

1. Akses alamat ini <https://developers.facebook.com/apps>

2. Jika kita belum *login* ke Facebook, maka kita akan diarahkan ke halaman *login*, silakan *login* dahulu dengan akun kita.

**Masuk Facebook**

Anda harus masuk untuk melanjutkan.

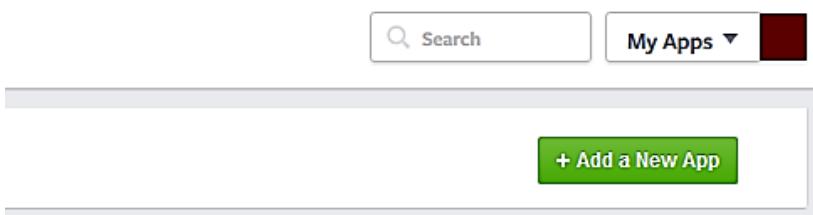
Email atau  
Telepon: hafid\_jmbr@yahoo.com

Kata sandi:   Biarkan saya tetap masuk

**Masuk** atau Mendaftar Facebook

Lupa kata sandi?

3. Klik tombol *Add a New App*



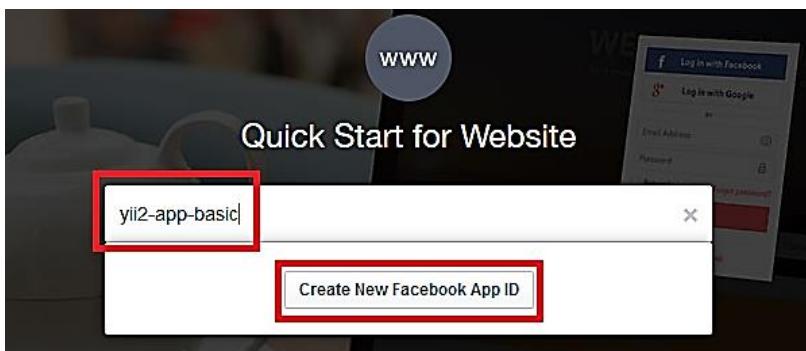
4. Pilih *platform* aplikasi kita yaitu “situs web”

### Add a New App

Select a platform to get started



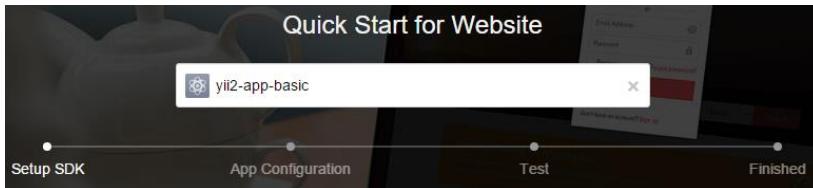
5. Tuliskan nama aplikasi kita, misal “yii2-app-basic”, lalu klik tombol *Create New Facebook App ID*.



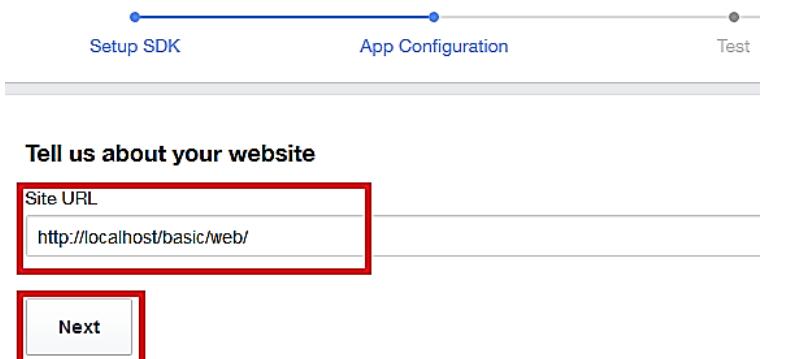
6. Pilih kategori aplikasi, lalu klik tombol *Create New Facebook App ID*



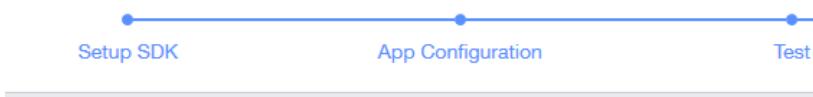
7. Pada tampilan ini, silakan scroll ke bawah



Maka masukkan URL aplikasi anda, dan klik Next



8. Scroll ke bawah dan klik tautan “Skip to Developer Dashboard”

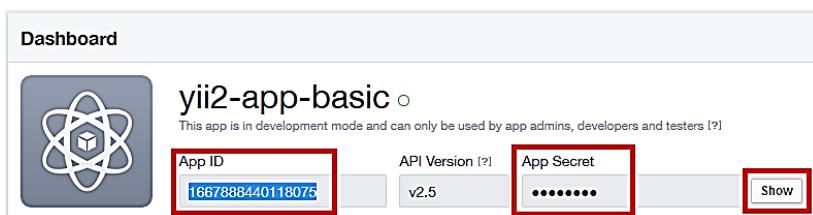


### Next Steps

Congratulations! You have added the Facebook SDK to your project. You are now in the ne Facebook. What do you want to do next: [Skip to Developer Dashboard](#) or [Documentation](#)



9. Maka kita akan dialihkan ke halaman developer dashboard.



10. Catat app ID dan app Secret (klik tombol Show).

Selesai.

## Google

Berikut ini langkah-langkah untuk mendapatkan *token* Google.

1. Akses alamat ini <https://console.developers.google.com/project>

Jika belum *login* ke Google maka akan muncul formulir *login*, silakan *login* dahulu



One account. All of Google.

Sign in to continue to Google Cloud Platform

2. Klik tombol *Create Project*

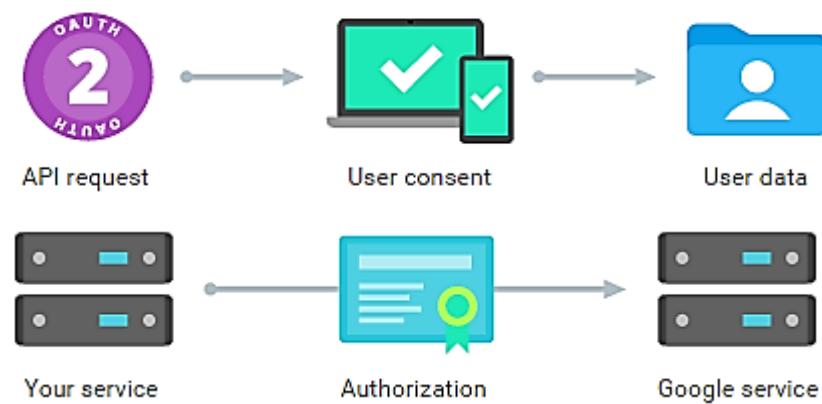
3. Masukkan nama *project* kita misal yii2-app-basic, dan klik tombol *Create*

4. Kita akan dialihkan ke halaman *dashboard project* yang baru kita buat, klik tautan *Enable and Manage API*

5. Pilih Google+ API

Kenapa kita memilih Google+ APIs bukan yang lain?

Jawabannya adalah kita bisa mengakses data pengguna menggunakan OAuth 2.0 pada API ini dan bisa digunakan untuk interaksi server ke server antara aplikasi kita dengan Google. Mekanisme bisa dijelaskan pada bagan berikut:



6. Klik *Enable*

The screenshot shows the Google Developers Console interface. On the left, there's a sidebar with 'API Manager' selected. Under 'Google+ API', the 'Overview' tab is active. A prominent blue button labeled 'Enable API' is centered on the page, with a red box drawn around it to indicate it as a key step.

7. Jika sudah, masuk ke menu *Credential*

Overview

This screenshot shows the 'Credentials' page for the Google+ API. It displays a message: 'This API is enabled, but you can't use it in your project until you create credentials. Click 'Go to Credentials' to do this now (strongly recommended.)'. A blue button labeled 'Go to Credentials' is highlighted with a red box.

Atau bisa juga dengan menekan tombol menu kiri

This screenshot shows the 'API Manager' overview page again. The 'Credentials' menu item in the sidebar is highlighted with a red box.

8. Tentukan dimana kita akan memanggil atau menggunakan API ini, lalu kemudian klik tombol “*What credential do I need*”

## Add credentials to your project

1 Find out what kind of credentials you need

We'll help you set up the correct credentials

If you wish you can skip this step and create an API key, client

Which API are you using?

Determines what kind of credentials you need.

Google+ API

Where will you be calling the API from?

Determines which settings you'll need to configure.

Web server (e.g. node.js, Tomcat)

What credentials do I need?

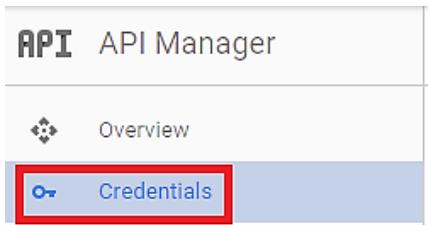
9. Jika sudah, maka akan muncul seperti berikut

## Add credentials to your project

- ✓ Find out what kind of credentials you need  
Calling Google+ API from a web server

[Cancel](#)

10. Klik menu credential lagi

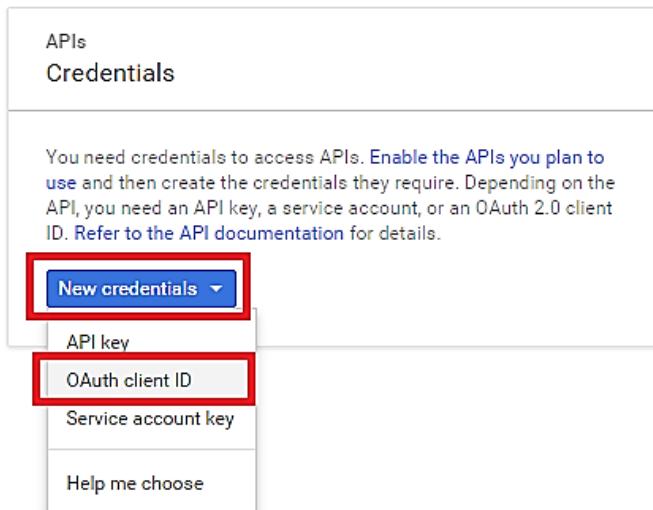


11. Maka kita akan dibawa ke halaman *Credential*,

### Credentials

[Credentials](#)   [OAuth consent screen](#)   [Domain verification](#)

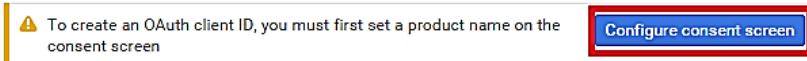
Pada tombol *New credentials*, plih submenu OAuth *client ID*



12. Klik tombol *Configure consent screen*



Create client ID



*Consent Screen* adalah tampilan yang akan muncul pertama kali ketika pengguna memilih *login* menggunakan Google.

13. Kita perlu konfigurasi melalui formulir berikut:

Credentials    OAuth consent screen    Domain verification

---

Email address 

Product name shown to users

Homepage URL (Optional)

Product logo URL (Optional) 



This is how your logo will look to end users  
Max size: 120x120 px

**Save** **Cancel**

14. Lalu kita akan dibawa kembali ke formulir *Create Client ID*, pilih *Web Application*



Create client ID

Application type

- Web application
- Android [Learn more](#)
- Chrome App [Learn more](#)
- iOS [Learn more](#)
- PlayStation 4
- Other

15. Masukkan data sebagai berikut, sesuaikan dengan aplikasi anda,

Name

Authorized JavaScript origins  
Enter JavaScript origins here or redirect URIs below (or both)   
Cannot contain a wildcard ([http://\\*.example.com](http://*.example.com)) or a path (<http://example.com/subdir>).

Authorized redirect URIs  
Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public

**Perhatian:**

*Authorized URL* adalah URL yang akan dituju pertama kali ketika tombol *login social media* diklik yaitu <http://localhost/basic/web/site/auth?authclient=google>

kemudian klik *Create*

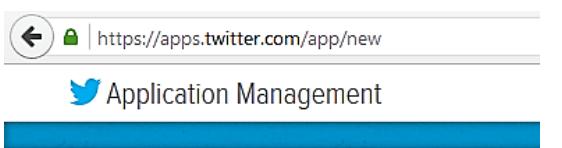


16. Dan akhirnya yang kita cari muncul juga. Silakan dicatat ya..

**Twitter**

Berikut ini langkah-langkah untuk mendapatkan *token* Twitter

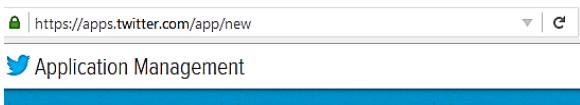
1. Akses alamat ini <https://dev.twitter.com/apps/new>



## Access denied

You are not authorized to access this page.

Silalah *login* dulu, maka setelah itu akan muncul formulir *create an application*



## Create an application

2. Isi formulir *create an application* tersebut

### Application Details

**Name \***  
yii2-app-basic

*Your application name. This is used to attribute the source of a tweet and in user-facing authorization requests.*

**Description \***  
yii2 application basic template

*Your application description, which will be shown in user-facing authorization requests.*

**Website \***

Your application's publicly accessible home page, where users can go to do what they want with your application. This URL is used in the source attribution for tweets created by your application and in the links shared by users.

**Callback URL**

Where should we return after successfully authenticating? OAuth 1.0a applications must provide a callback URL. This URL is used to return the user to your application after the user has been authenticated. It is not required to use the same URL as your website.

**Perhatian:**

Callback URL adalah URL yang akan dituju ketika proses otentikasi dengan *social media* berhasil yaitu <http://localhost/basic/web/site/success-callback>. Kita akan mengaturnya nanti pada bagian implementasi.

- Setujui *Developer Agreement* dengan meng-checklist *Yes, I agree*. Kemudian tekan tombol “Create your Twitter Application”

otherwise fails to comply or is inconsistent with any part of this Agreement. If you exceed the Rate Limit, Twitter may attempt to circumvent Rate Limits, controls to limit use of the Twitter APIs or the terms and conditions of this Agreement. Your ability to use the Licensed Materials may be temporarily suspended or permanently blocked.

Yes, I agree

**Having trouble creating your application?**

If you're having trouble fulfilling application creation requirements, please contact our Platform team via the "I have a question not covered by these points" option of the contact form at <https://support.twitter.com>.

[Create your Twitter application](#)

Horee.. berhasil ☺

 Application Management

Your application has been created. Please take a moment to review and update your application's settings.

## yii2-app-basic

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

- Klik tab *Keys and Access Tokens*



- Catat *consumer key* dan *consumer secret*

## yii2-app-basic

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

### Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	TZ7pRyHLIA...TR5JZTEuU0
Consumer Secret (API Secret)	gCR8XqG...fbBvax2qlffjX3YYB2c3nE8uiWSzjqgkKh
Access Level	Read and write (modify app permissions)
Owner	hafidmukhlasin
Owner ID	1678966496

6. Pada tab *permissions*, pilih type access menjadi *Read only*.

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

### Access

What type of access does your application need?

[Read more about our Application Permission Model.](#)

Read only

Read and Write

Read, Write and Access direct messages

*Note:*

*Changes to the application permission model will only reflect in access tokens issued after the change. Existing access tokens will need to re-negotiate existing access tokens to alter the permission level.*

[Update Settings](#)

Selesai

### Github

Berikut ini langkah-langkah untuk mendapatkan *token* Github

1. Akses <https://github.com/settings/applications/new>



Sign in to GitHub

Username or email address

Password [Forgot password?](#)

[Sign in](#)

2. Isi formulir dan klik *Register Application*

**Register a new OAuth application**

**Application name**  
yii2-basic-app  
Something users will recognize and trust

**Homepage URL**  
http://localhost/basic/web  
The full URL to your application homepage

**Application description**  
Yii2 Basic Application Template  
This is displayed to all potential users of your application

**Authorization callback URL**  
http://localhost/basic/web/site/auth?authclient=github  
Your application's callback URL. Read our [OAuth documentation](#)

---

**Perhatian:**  
*Authorized callback URL* adalah URL yang akan dituju pertama kali ketika tombol *login social media* diklik yaitu http://localhost/basic/web/site/auth?authclient=github

Klik tombol *Register Application*

**Register application**

### 3. Catat Client ID & Client Secret



## A. 5. Konfigurasi

Setelah mendapatkan *token* dari masing-masing *social media*, maka langkah selanjutnya adalah melakukan konfigurasinya pada aplikasi Yii kita. Caranya pun sangat sederhana, yaitu dengan menambahkan *class* [[*yii\authclient\Collection*]] pada konfigurasi *components* di file

@app/main/web.php

```
'components' => [
 'authClientCollection' => [
 'class' => 'yii\authclient\Collection',
 'clients' => [
 'facebook' => [
 'class' => 'yii\authclient\clients\Facebook',
 'clientId' => 'facebook_client_id',
 'clientSecret' => 'facebook_client_secret',
],
 'google' => [
 'class' => 'yii\authclient\clients\GoogleOAuth',
],
],
],
],
```

```

 'clientId' => 'google_client_id',
 'clientSecret' => 'google_client_secret',
],
 'twitter' => [
 'class' => 'yii\authclient\clients\Twitter',
 'consumerKey' => 'twitter_consumer_key',
 'consumerSecret' => 'twitter_consumer_secret',
],
 'github' => [
 'class' => 'yii\authclient\clients\GitHub',
 'clientId' => 'github_client_id',
 'clientSecret' => 'github_client_secret',
],
],
],
...
]

```

Silakan ganti `client_id` dan `client_secret` dengan *token* yang telah kita peroleh dari akun *social media*. Silakan disesuaikan tergantung *social media*-nya.

Setelah konfigurasi, sekarang kita bisa menampilkan tautan untuk *login* menggunakan *social media* dan kabar baiknya, Yii telah menyediakan *widget* untuk itu yaitu [[`yii\authclient\widgets\AuthChoice`]].

```

<?= yii\authclient\widgets\AuthChoice::widget([
 'baseAuthUrl' => ['site/auth']
]); ?>

```

Kode di atas akan menampilkan tautan *icon social media* sebagaimana yang didefinisikan pada konfigurasi. Letakkan kode tersebut di bawah formulir *login*

```
@app/views/site/login.php
```

`baseAuthUrl` berisi URL yang dalam hal ini diarahkan ke `SiteController` fungsi `auth`, fungsinya untuk pemrosesan otentifikasi menggunakan *social media*. Mekanismenya akan kita bahas pada bagian selanjutnya

```

41 <?php ActiveForm::end(); ?>
42
43 <div class="col-lg-offset-1" style="color:#999;">
44 <?= yii\authclient\widgets\AuthChoice::widget([
45 'baseAuthUrl' => ['site/auth']
46]); ?>
47 </div>
48 </div>

```

Waktunya uji coba, silakan akses formulir *login*

```
∞ site/login
```

Home / Login

## Login

Please fill out the following fields to login:

Username

Password

Remember Me

**Login**



Facebook



Google



Twitter



GitHub

Ada 4 *icon social media* sesuai dengan setting yang kita lakukan di konfigurasi.

Untuk *social media* lain yang tidak didukung oleh Yii2, maka kita bisa membuat sendiri, ikuti panduan di sini

∞ <https://github.com/yiisoft/yii2-authclient/blob/master/docs/guide/creating-your-own-auth-clients.md>

## A. 6. Implementasi

Untuk membuat otentikasi menggunakan *social media* bekerja pada aplikasi Yii kita maka pada *controller*, kita perlu membuat sebuah fungsi bertipe *action* yang mengimplementasikan *class* [[*yii\authclient\AuthAction*]]. Fungsi inilah yang akan menangani komunikasi antara aplikasi kita dengan *social media*. Pada panduan ini, fungsi tersebut diberi nama *auth*.

Penamaan fungsi ini bebas, dan disesuaikan atau berkaitan dengan parameter *baseAuthUrl* pada *widget AuthChoice*.

```
<?= yii\authclient\widgets\AuthChoice::widget([
 'baseAuthUrl' => ['site/auth']
]); ?>
```

Pada implementasi *class AuthAction* tersebut kita juga perlu mengatur parameter *successCallback*. Parameter ini mereferensikan sebuah fungsi yang bertugas menangani atau memproses hasil dari otentikasi yang dilakukan oleh fungsi *auth*.

Untuk lebih jelasnya marilah kita lihat *skeleton* kode berikut:

```
class SiteController extends Controller
{
 public function actions()
 {
 return [
 'auth' => [
 'class' => 'yii\authclient\AuthAction',
 'successCallback' => [$this, 'successCallback'],
],
]
 }

 public function successCallback($client)
 {
 $attributes = $client->getUserAttributes();
 // user login or signup comes here
 }
}
```

---

### Perhatian

Jangan asal mengubah SiteController menjadi seperti kode di atas, *please deh* !. Karena pada SiteController sudah ada fungsi *actions()* maka letakkan *block auth* pada *items array* berikutnya. Lihat contoh di bawah ini

```
public function actions()
{
 return [
 'error' => [
 'class' => 'yii\web\ErrorAction',
],
 'captcha' => [
 'class' => 'yii\captcha\CaptchaAction',
 'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
],
 'auth' => [
 'class' => 'yii\authclient\AuthAction',
 'successCallback' => [$this, 'successCallback'],
],
];
}
```

### Analisis Data Social Media

Nah, untuk mengetahui isi dari variabel *\$attributes* maka kita bisa menampilkannya menggunakan fungsi *print\_r* atau *var\_dump*,

```
public function successCallback($client) {
 // get user data from client
 $attributes = $client->getUserAttributes();
```

```

 print_r($attributes);
 exit;
}

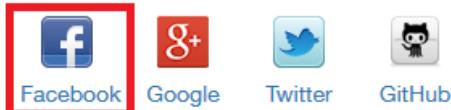
```

Kemudian kita coba *login* menggunakan *social media*, maka kita akan melihat bahwa hasil keluarannya berbeda-beda tergantung *social media* yang digunakan, yang kita butuhkan pada panduan ini hanyalah id, *username social media* tersebut, serta nama dan *email* untuk registrasi.

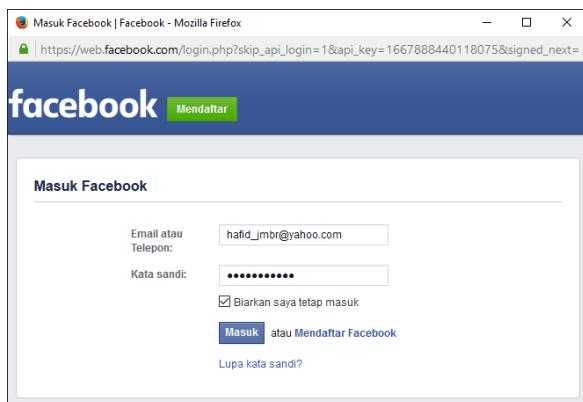
Hasil dari percobaan penulis, sebagai berikut

#### ❖ Facebook

1. Klik tombol Facebook pada formulir *login* aplikasi kita



2. Maka akan muncul *popup, login* ke Facebook



3. Pada jendela konfirmasi, klik tombol *Oke*



4. Maka akan muncul data sebagai berikut..

```

Array (
 [name] => Hafid Mukhlasin
 [email] => hafid_jmbr@yahoo.com
 [id] => 10207297161724551
)

```

Data tersebut sudah memenuhi kebutuhan kita. Adapun cara mengakses data tersebut adalah sebagai berikut:

```

$attributes['id']; // id unik
$attributes['email']; // username dan email
$attributes['name']; // nama

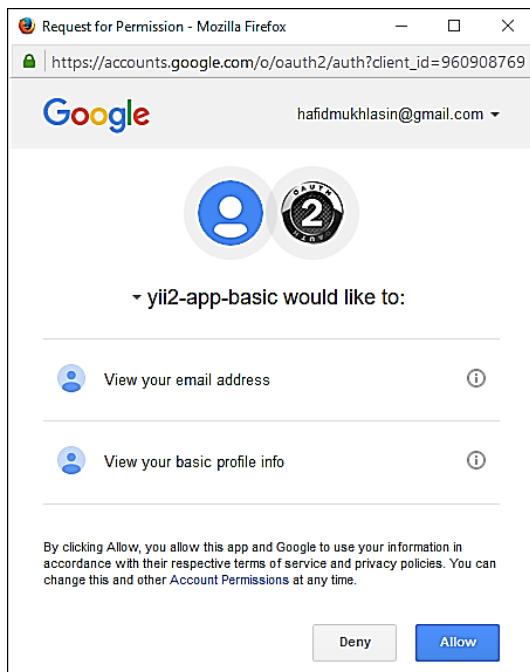
```

#### ❖ Google

Klik tombol Google pada formulir *login* aplikasi kita



1. Maka akan muncul *popup*, login ke Google jika kita belum *login*
2. Pada jendela konfirmasi, klik tombol *allow*



3. Maka akan muncul data sebagai berikut..

```
Array (
 [emails] => Array (
 [0] => Array (
 [value] => hafidmukhlasin@gmail.com
 [type] => account
)
)
 [id] => 103806943133437885384
 [displayName] => Hafid Mukhlasin
)
```

**Note:**

Sebenarnya ada banyak data namun penulis hanya menampilkan bagian yang relevan.

Data tersebut sudah memenuhi kebutuhan kita. Adapun cara mengakses data tersebut adalah sebagai berikut:

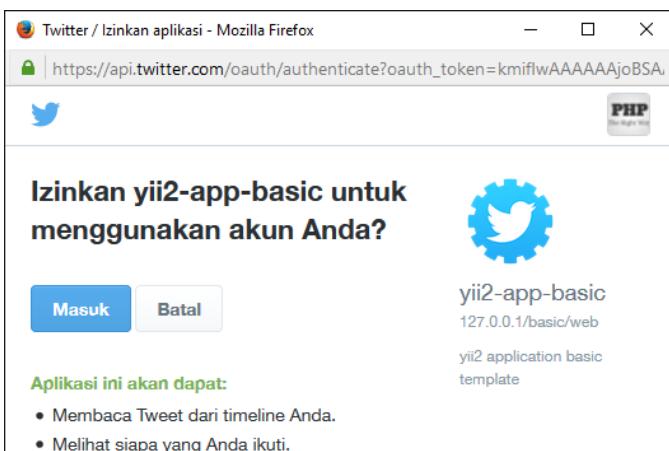
```
$attributes['emails'][0]['value']; // username dan email
$attributes['id']; // id unik
$attributes['displayName']; // nama
```

❖ **Twitter**

Klik tombol Twitter pada formulir *login* aplikasi kita



1. Maka akan muncul *popup*, login ke Twitter jika kita belum *login*
2. Pada jendela konfirmasi, klik tombol Masuk



3. Maka akan muncul data sebagai berikut..

```
Array (
 [id] => 1678966496
 [name] => Hafid Mukhlasin
 [screen_name] => hafidmukhlasin
)
```

#### Note:

Sebenarnya ada banyak data namun penulis hanya menampilkan bagian yang relevan.

Data tersebut sudah memenuhi kebutuhan kita. Adapun cara mengakses data tersebut adalah sebagai berikut:

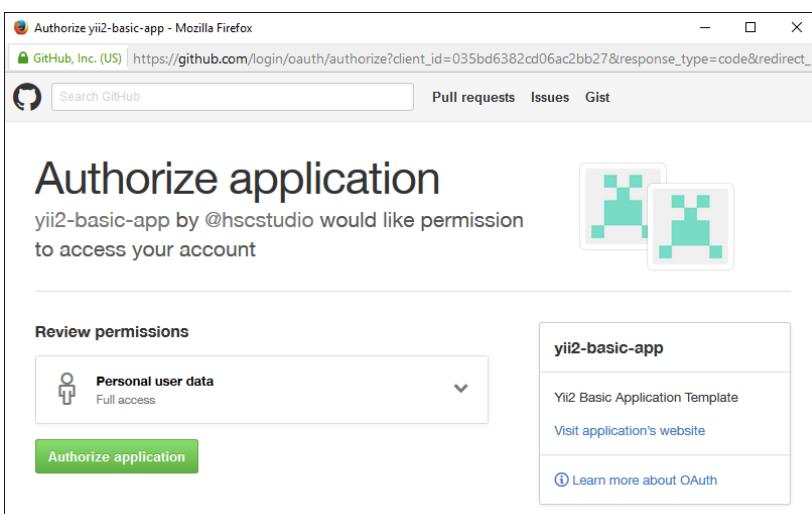
```
$attributes['id']; // id unik
$attributes['screen_name']; // username
$attributes['name']; // nama
// email tidak ada
```

#### ❖ Github

Klik tombol Github pada formulir *login* aplikasi kita



1. Maka akan muncul *popup, login* ke Github, silakan *login* dahulu
2. Pada jendela konfirmasi, klik tombol *Authorize application*



3. Maka akan muncul data sebagai berikut..

```
Array (
 [login] => hscstudio
 [id] => 2976897
 [name] => Hafid Mukhlasin
```

```
|) [email] => milisstudio@gmail.com
```

---

**Note:**

Sebenarnya ada banyak data namun penulis hanya menampilkan bagian yang relevan.

Data tersebut sudah memenuhi kebutuhan kita. Adapun cara mengakses data tersebut adalah sebagai berikut:

```
| $attributes['id']; // id unik
| $attributes['login']; // username
| $attributes['name']; // nama
| $attributes['email']; // email
```

**Penerapan Data Social Media**

Setelah kita mendapatkan data yang dibutuhkan, maka sekarang kita bisa terapkan dengan memodifikasi fungsi successCallback pada SiteController

- Baca data dari *social media*

Mari kita modifikasi fungsi successCallback pada SiteController, dimana kita tambahkan kode untuk mendapatkan nilai dari atribut *social media*.

```
public function successCallback($client) {
 // get user data from client
 $attributes = $client->getUserAttributes();

 // set default value
 $safe_attributes = [
 'social_media'=> '',
 'id'=> '',
 'username'=> '',
 'name'=> '',
 'email'=> '',
];

 // get value from user attributes base on social media
 if ($client instanceof \yii\authclient\clients\Facebook) {
 $safe_attributes = [
 'social_media'=> 'facebook',
 'id'=> $attributes['id'],
 'username'=> $attributes['email'],
 'name'=> $attributes['name'],
 'email'=> $attributes['email'],
];
 }
 else if ($client instanceof \yii\authclient\clients\GoogleOAuth) {
 $safe_attributes = [
 'social_media'=> 'google',
 'id'=> $attributes['id'],
 'username'=> $attributes['emails'][0]['value'],
 'name'=> $attributes['displayName'],
 'email'=> $attributes['emails'][0]['value'],
];
 }
 else if ($client instanceof \yii\authclient\clients\Twitter) {
 $safe_attributes = [
 'social_media'=> 'twitter',
 'id'=> $attributes['id'],
 'username'=> $attributes['screen_name'],
 'name'=> $attributes['name'],
 'email'=> '--',
];
 }
 else if ($client instanceof \yii\authclient\clients\GitHub) {
 $safe_attributes = [
 'social_media'=> 'github',
 'id'=> $attributes['id'],
 'username'=> $attributes['login'],
 'name'=> $attributes['name'],
 'email'=> $attributes['email'],
];
 }
}
```

```

 }
}
```

Pada kode di atas dilakukan pengecekan berdasarkan jenis *social media* yang digunakan. Hal ini dikarenakan data dari *social media* tidak seragam sebagaimana yang telah kita coba sebelumnya.

Agar lebih rapi, kita bisa memindahkan kode di atas menjadi sebuah fungsi baru pada SiteController. Hasilnya sebagai berikut:

```

public function successCallback($client) {
 // call safeAttributes method for properly format data
 $attributes = $this->safeAttributes($client);
}

public function safeAttributes($client){
 // get user data from client
 $attributes = $client->getUserAttributes();

 // set default value
 $safe_attributes = [
 'social_media'=> '',
 'id'=> '',
 'username'=> '',
 'name'=> '',
 'email'=> '',
];

 // get value from user attributes base on social media
 if ($client instanceof \yii\authclient\clients\Facebook) {
 $safe_attributes = [
 'social_media'=> 'facebook',
 'id'=> $attributes['id'],
 'username'=> $attributes['email'],
 'name'=> $attributes['name'],
 'email'=> $attributes['email'],
];
 } else if ($client instanceof \yii\authclient\clients\GoogleOAuth) {
 $safe_attributes = [
 'social_media'=> 'google',
 'id'=> $attributes['id'],
 'username'=> $attributes['emails'][0]['value'],
 'name'=> $attributes['displayName'],
 'email'=> $attributes['emails'][0]['value'],
];
 } else if ($client instanceof \yii\authclient\clients\Twitter) {
 $safe_attributes = [
 'social_media'=> 'twitter',
 'id'=> $attributes['id'],
 'username'=> $attributes['screen_name'],
 'name'=> $attributes['name'],
 'email'=> '-',
];
 } else if ($client instanceof \yii\authclient\clients\GitHub) {
 $safe_attributes = [
 'social_media'=> 'github',
 'id'=> $attributes['id'],
 'username'=> $attributes['login'],
 'name'=> $attributes['name'],
 'email'=> $attributes['email'],
];
 }

 return $safe_attributes;
}
```

- Mengecek apakah data *social media* itu ada di basis data

Pada tahap ini kita melakukan pencarian apakah data yang kita dapatkan dari *social media* pengguna tadi terdapat di dalam basis data tabel *user\_social\_media* (model *app\models\UserSocialMedia*)

```

public function successCallback($client) {
 // call safeAttributes method for properly format data
 $attributes = $this->safeAttributes($client);

 // find data social media in basis data
 $user_social_media = UserSocialMedia::find()
 ->where([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
])
 ->one();

 // if data found
 if($user_social_media) {

 }
 else{

 }
}

```

**Perhatian:**

Karena kita menggunakan *class UserSocialMedia* maka seharusnya kita menambahkan kode *use app\models\UserSocialMedia* pada bagian atas SiteController.

- Aksi yang dilakukan jika data *social media* ditemukan dalam tabel *user\_social\_media*

Jika data *social media* ditemukan dalam tabel *user\_soial\_media* maka dicek dulu apakah pengguna yang dimaksud statusnya aktif. Jika pengguna yang dimaksud aktif maka sistem akan me-*login*-kan otomatis, namun jika pengguna tidak aktif maka pesan galat di *set*.

```

// if data found
if($user_social_media){
 // get user from relation
 $user = $user_social_media->user;
 // check user is active
 if($user->status==User::STATUS_ACTIVE){
 // do automatic login
 Yii::$app->user->login($user);
 }
 else{
 Yii::$app->session->setFlash('error','Login gagal, status user tidak aktif');
 }
 // finish
}

```

**Perhatian:**

Karena kita menggunakan *class User* maka seharusnya kita menambahkan kode *use app\models\User* pada bagian atas SiteController.

- Aksi yang dilakukan jika data *social media* tidak ditemukan dalam tabel *user\_social\_media*

Jika data *social media* tidak ditemukan dalam tabel *user\_soial\_media* maka akan dilakukan pengecekan apakah data tersebut khususnya *email* ada di tabel *user*.

```

// if data not found
// check if email social media exists in tabel user
$user = User::find()
 ->where([
 'email' => $attributes['email']
])
 ->one();

```

- Aksi yang dilakukan jika data *social media* ditemukan dalam tabel *user*

Jika data *social media* ditemukan dalam *tabel user* maka dicek dulu apakah pengguna yang dimaksud statusnya aktif. Jika pengguna yang dimaksud aktif maka sistem akan menambah data *social media* ke tabel *user\_social\_media* serta meloginkan otomatis, namun jika pengguna tidak aktif maka pesan galat di *set*.

```

// if user found
if($user){

```

```

// check user is active
if($user->status==User::STATUS_ACTIVE) {
 // add to table user social media
 $user_social_media = new UserSocialMedia([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
 'user_id'=>$user->id,
]);
 $user_social_media->save();

 // do automatic login
 Yii::$app->user->login($user);
}
else{
 Yii::$app->session->setFlash('error','Login gagal, status user tidak aktif');
}
}

```

- Aksi yang dilakukan jika data *social media* tidak ditemukan dalam tabel *user*

Jika data *social media* tidak ditemukan dalam tabel *user* maka dicek dulu apakah *social media*nya Twitter? Mengapa Twitter? Karena Twitter secara *default* tidak menyediakan atribut *email* untuk diakses.

Jika bukan Twitter maka sistem mendaftarkan data dari *social media* ke tabel *user*, jika sukses maka ditambahkan ke tabel *user\_social\_media*, baru kemudian di-*login*-kan secara otomatis.

Namun jika Twitter maka attribute *social media* disimpan kedalam *session* agar bisa digunakan pada fungsi lain dalam hal ini adalah fungsi *signup*. Kemudian sistem akan mengalihkan ke formulir *login* melalui pengaturan variabel *successUrl* pada *current action*.

```

else{
 // check if social media not twiiter
 if($attributes['social_media']!='twitter'){
 // do automatic signup
 $password = Yii::$app->security->generateRandomString(6);
 $user = new User([
 'username' => $attributes['username'],
 'email' => $attributes['email'],
 'password' => $password,
]);
 $user->generateAuthKey();
 $user->generatePasswordResetToken();
 if($user->save()){
 $user_social_media = new UserSocialMedia([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
 'user_id'=>$user->id,
]);
 $user_social_media->save();
 // do automatic login
 Yii::$app->user->login($user);
 }
 else{
 Yii::$app->session->setFlash('error','Login gagal, galat saat registrasi');
 }
 }
 else{
 // save data attributes to session
 $session = Yii::$app->session;
 $session['attributes']=$attributes;

 // redirect to signup, via property successUrl
 $this->action->successUrl = Url::to(['signup']);
 }
}

```

#### **Perhatian:**

Karena kita menggunakan *class* *Url* maka seharusnya kita menambahkan kode `use yii\helpers\Url;` pada bagian atas *SiteController*.

Sehingga kode lengkap untuk fungsi *successCallback* adalah sebagai berikut

```

public function successCallback($client) {
 // call safeAttributes method for properly format data
 $attributes = $this->safeAttributes($client);

 // find data social media in basis data
 $user_social_media = UserSocialMedia::find()
 ->where([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
])
 ->one();

 // if data found
 if($user_social_media){
 // get user from relation
 $user = $user_social_media->user;
 // check user is active
 if($user->status==User::STATUS_ACTIVE){
 // do automatic login
 Yii::$app->user->login($user);
 }
 else{
 Yii::$app->session-
 >setFlash('error','Login gagal, status user tidak aktif');
 }
 // finish
 }
 else{
 // if data not found
 // check if email social media exists in tabel user
 $user = User::find()
 ->where([
 'email' => $attributes['email']
])
 ->one();
 // if user found
 if($user){
 // check user is active
 if($user->status==User::STATUS_ACTIVE){
 // add to table user social media
 $user_social_media = new UserSocialMedia([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
 'user_id'=>$user->id,
]);
 $user_social_media->save();

 // do automatic login
 Yii::$app->user->login($user);
 }
 else{
 Yii::$app->session-
 >setFlash('error','Login gagal, status user tidak aktif');
 }
 }
 else{
 // check if social media not twiiter
 if($attributes['social_media']!='twitter'){
 // do automatic signup
 $password = Yii::$app->security->generateRandomString(6);
 $user = new User([
 'username' => $attributes['username'],
 'email' => $attributes['email'],
 'password' => $password,
]);
 $user->generateAuthKey();
 $user->generatePasswordResetToken();
 if($user->save()){
 $user_social_media = new UserSocialMedia([

```

Karena pemrosesan data tersebut berkaitan dengan fungsi registrasi maka kita perlu modifikasi juga fungsi `actionSignup`. Intinya adalah agar data dari *social media* juga ikut terekam.

```

public function actionSignup()
{
 $model = new \app\models\SignupForm();

 // use session
 $session = Yii::$app->session;
 $attributes = $session['attributes'];

 if ($model->load(Yii::$app->request->post())) {
 if ($user = $model->signup()) {
 if ($session->has('attributes')){
 // add data user_social_media
 $user_social_media = new UserSocialMedia([
 'social_media' => $attributes['social_media'],
 'id'=>(string)$attributes['id'],
 'username'=>$attributes['username'],
 'user_id'=>$user->id,
]);
 $user_social_media->save();
 }
 }

 if (Yii::$app->getUser()->login($user)) {
 return $this->goHome();
 }
 }

 if ($session->has('attributes')){
 // set form field with data from social media
 $model->username = $attributes['username'];
 $model->email = $attributes['email'];
 }

 return $this->render('signup', [
 'model' => $model,
]);
}

```

#### A. 7. *Uji Coba Sistem*

Untuk membuktikan apakah sistem berjalan sesuai skenario kita atau tidak maka kita perlu menguji cobanya menggunakan semua kemungkinan atau skenario uji coba. Berikut ini skenario dan hasil uji cobanya.

## Skenario Uji Coba 1

### ❖ Skenario:

User login menggunakan *social media* Facebook, dimana akun *social media* tersebut belum ada di basis data.

Berikut ini data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password_hash</b>	<b>password_r</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$qsMM	NULL	admin@gmail.com	10
2	test	o34ZhOd	\$2y\$13\$bAbAF	NULL	test@gmail.com	10

Sedangkan data tabel *user\_social\_media* kosong

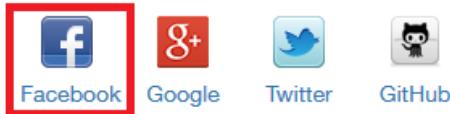
### ❖ Ekspektasi:

Sistem akan meregistrasi secara otomatis yaitu dengan menambahkan data pengguna dari *social media* ke dalam tabel *user*, serta menambahkan data tersebut ke tabel *user\_social\_media*, kemudian sistem melakukan *login* atas pengguna baru tersebut.

### ❖ Uji Coba

Uji coba ini menggunakan akun Facebook dengan *username* hafid\_jmbr@yahoo.com

Setelah menekan tombol Facebook



Kemudian muncul *popup login* Facebook, setelah *login* dan menyetujui permintaan izin pengaksesan data *social media*, maka tampilan dialihkan ke halaman *home*, dengan tampilan menu sebagai berikut

Home	About	Contact	Signup	Logout (hafid_jmbr@yahoo.com)
------	-------	---------	--------	-------------------------------

Tampilan di atas menunjukkan bahwa sistem telah *login* dengan *username* hafid\_jmbr@yahoo.com. Untuk memastikan apakah benar-benar berhasil registrasi ke tabel *user* dan *user\_social\_media*, maka kita cek kedua tabel tersebut.

Data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password_hash</b>	<b>password_r</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$qsMM	NULL	admin@gmail.com	10
2	test	o34ZhOd	\$2y\$13\$bAbAF	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPWaSE	\$2y\$13\$Elr9Ae	nbmdC8ovBll	hafid_jmbr@yahoo.com	10

Data tabel *user\_social\_media*

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3

### ❖ Kesimpulan

Karena berhasil login otomatis, data di tabel *user* dan *user\_social\_media* bertambah maka skenario uji coba pertama ini dinyatakan berhasil

## Skenario Uji Coba 2

### ❖ Skenario:

Pengguna *login* menggunakan *social media* Facebook, dimana akun tersebut sudah ada di basis data.

Berikut ini data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password_ha</b>	<b>password_r</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$qsMM	NULL	admin@gmail.com	10
2	test	o34ZhOdl	\$2y\$13\$bAbAf	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPwaSE-	\$2y\$13\$Elr9Ae	nbmdC8ovBll ADbsovmyDl	hafid_jmbr@yahoo.com	10

Sedangkan data tabel user\_social\_media sebagai berikut

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3

#### ❖ Ekspektasi:

Sistem akan melakukan *login* otomatis tanpa menambahkan data apapun pada tabel *user* dan *user\_social\_media*

#### ❖ Uji Coba

Uji coba ini menggunakan akun Facebook dengan *username* hafid\_jmbr@yahoo.com

Setelah menekan tombol Facebook



Kemudian muncul *popup login* Facebook, kemudian terjadi proses, dan setelah itu tampilan dialihkan ke halaman *home*, dengan tampilan menu sebagai berikut

Home	About	Contact	Signup	Logout (hafid_jmbr@yahoo.com)
------	-------	---------	--------	-------------------------------

Tampilan di atas menunjukkan bahwa sistem telah *login* dengan *username* hafid\_jmbr@yahoo.com. Untuk memastikan apakah sistem tidak menambahkan data baru ke tabel *user* dan *user\_social\_media*, maka kita cek kedua tabel tersebut.

Data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password_ha</b>	<b>password_r</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$qsMM	NULL	admin@gmail.com	10
2	test	o34ZhOdl	\$2y\$13\$bAbAf	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPwaSE-	\$2y\$13\$Elr9Ae	nbmdC8ovBll ADbsovmyDl	hafid_jmbr@yahoo.com	10

Data tabel *user\_social\_media*

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3

#### ❖ Kesimpulan

Karena berhasil *login* otomatis, dan data di tabel *user* dan *user\_social\_media* tidak bertambah maka skenario uji coba kedua ini dinyatakan berhasil.

### ***Skenario Uji coba 3***

#### ❖ Skenario:

Pengguna *login* menggunakan *social media* Google, dimana akun tersebut belum ada di tabel *user\_social\_media*, hanya saja *email* Google tersebut terdaftar pada tabel *user*.

---

#### **Catatan:**

Untuk mencoba skenario ini kita bisa menambah data baru menggunakan fitur *signup* yang telah kita buat sebelumnya

---

Berikut ini data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password</b>	<b>password</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$	NULL	admin@gmail.com	10
2	test	o34ZhOdl	\$2y\$13\$	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPwaSE+	\$2y\$13\$	nbmdC8 ADbsvr	hafid_jmbr@yahoo.com	10
4	hafid	1x1Wuic.	\$2y\$13\$	NULL	hafidmukhlasin@gmail.com	10

Sedangkan data tabel user\_social\_media kosong

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3

#### ❖ Ekspektasi:

Sistem akan melakukan *login* otomatis tanpa menambahkan data apapun pada tabel *user* dan hanya menambahkan ke tabel user\_social\_media

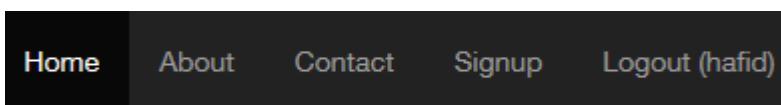
#### ❖ Uji coba

Pada uji coba ini, penulis menggunakan akun Google dengan *username* hafidmukhlasin@gmail.com.

Setelah menekan tombol Google



Kemudian muncul *popup login* Google, setelah *login* dan menyetujui permintaan izin pengaksesan data *social media*, maka tampilan dialihkan ke halaman *home*, dengan tampilan menu sebagai berikut



Tampilan di atas menunjukkan bahwa sistem telah login dengan *username* hafid. Untuk memastikan apakah sistem tidak menambahkan data baru ke tabel *user* melainkan hanya menambahkan ke tabel user\_social\_media saja, maka kita cek kedua tabel tersebut.

Data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password</b>	<b>password</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$	NULL	admin@gmail.com	10
2	test	o34ZhOdl	\$2y\$13\$	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPwaSE+	\$2y\$13\$	nbmdC8 ADbsvr	hafid_jmbr@yahoo.com	10
4	hafid	1x1Wuic.	\$2y\$13\$	NULL	hafidmukhlasin@gmail.com	10

Data tabel user\_social\_media

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3
google	103806943133437885384	hafidmukhlasin@gmail.com	4

#### ❖ Kesimpulan

Karena berhasil *login* otomatis, dan data di tabel *user* tidak bertambah serta data user\_social\_media bertambah sesuai data *social media* maka skenario uji coba ketiga ini dinyatakan berhasil ☺

### **Skenario Uji Coba 4**

#### ❖ Skenario:

Pengguna *login* menggunakan *social media* Twitter, dimana akun tersebut belum ada di tabel *user* dan user\_social\_media.

Berikut ini data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password</b>	<b>password</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-g	\$2y\$13\$	NULL	admin@gmail.com	10
2	test	o34ZhOdl	\$2y\$13\$	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPwaSEt	\$2y\$13\$	nbmdC8 ADbsovr	hafid_jmbr@yahoo.com	10
4	hafid	1x1Wuic.	\$2y\$13\$	NULL	hafidmukhlasin@gmail.com	10

Sedangkan data tabel *user\_social\_media* kosong

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3
google	103806943133437885384	hafidmukhlasin@gmail.com	4

#### ❖ Ekspektasi:

Sistem menyimpan data dari *social media* ke *session*, kemudian sistem akan mengalihkan ke formulir registrasi/*signup*. Kemudian sistem menge-set nilai *field* formulir registrasi dengan data *social media* yang telah disimpan pada *session*.

Sistem menampilkan formulir registrasi yang sebagian telah terisi data *social media*. Pengguna kemudian melengkapi data formulir *registrasi* dan men-submit-nya, dan jika sukses maka sistem akan melakukan *login* untuk pengguna dimaksud. Data tabel *user* dan *user\_social\_media* bertambah dari data *social media* tersebut.

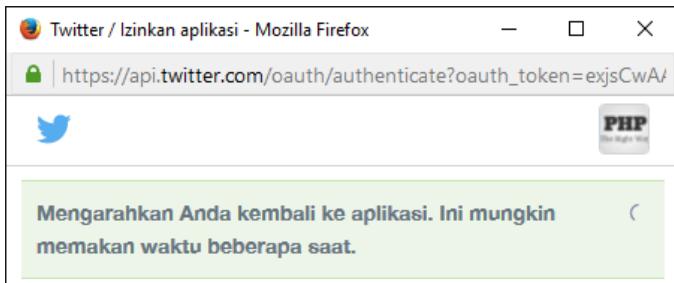
#### ❖ Uji Coba

Pada uji coba ini, penulis menggunakan akun Twitter dengan *username* @hafidmukhlasin.

Setelah menekan tombol Twitter



Kemudian muncul *popup login* Twitter, setelah *login* dan menyetujui permintaan izin pengaksesan data *social media*, maka tampilan dialihkan



ke halaman registrasi pengguna, dengan tampilan sebagai berikut

Home / Signup

## Signup

Please fill out the following fields to signup:

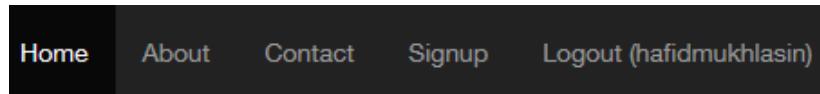
**Username**  
Hafid Mukhlasin

**Email**  
-

**Password**  
[Redacted]

**Signup**

Setelah formulir diisi dengan lengkap dan tombol *Signup* diklik maka sistem mengalihkan ke halaman *home* dengan tampilan sebagai berikut.



Tampilan di atas menunjukkan bahwa pengguna dengan nama hafidmukhlasin berhasil *login* pada sistem.

Mari kita cek tabel *user* dan tabel *user\_social\_media*

Data tabel *user*

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>passwo</b>	<b>passwo</b>	<b>email</b>	<b>status</b>
1	admin	rahasia-	\$2y\$13\$	NULL	admin@gmail.com	10
2	test	o34ZhO	\$2y\$13\$	NULL	test@gmail.com	10
3	hafid_jmbr@yahoo.com	RPWaSl	\$2y\$13\$	nbmdC&ADbsovi	hafid_jmbr@yahoo.com	10
4	hafid	1x1Wui	\$2y\$13\$	NULL	hafidmukhlasin@gmail.com	10
5	hafidmukhlasin	peNCzLZGmQ3	\$2y\$13\$	NULL	milisstudio@gmail.com	10

Data tabel *user\_social\_media*

<b>social_media</b>	<b>id</b>	<b>username</b>	<b>user_id</b>
facebook	10207669019340759	hafid_jmbr@yahoo.com	3
google	103806943133437885384	hafidmukhlasin@gmail.com	4
twitter	1678966496	hafidmukhlasin	5

### ❖ Kesimpulan

Karena berhasil mengalihkan ke formulir registrasi sekaligus menampilkan data *social media* yang sesuai pada *field*, sistem berhasil *login* dengan *username* hafidmukhlasin atau sama dengan *username* Twitter yang dipakai, kemudian ada penambahan data di tabel *user* dan *user\_social\_media* maka skenario uji coba keempat ini dinyatakan berhasil ☺

## B. Kesimpulan

Yii telah menyederhanakan mekanisme otentikasi pengguna pada aplikasi, baik itu melalui basis data maupun *social media*. Panduan pada buku ini hanyalah contoh, yang mana anda diperkenankan menggunakan versi anda sendiri yang sesuai dengan kebutuhan.

Sebenarnya ada banyak sekali topik yang bisa kita gali atau kasus berkaitan dengan otentikasi ini. Misalnya: detail pengguna atau profil, fitur ubah kata sandi aktifasi pengguna menggunakan *email* konfirmasi, dsb. Namun untuk buku ini, penulis rasa sudah cukup sampai di sini dulu. Silakan dikembangkan sendiri.

Pada bab selanjutnya, kita akan masuk ke ranah yang lebih dalam lagi setelah otentikasi yaitu otorisasi. Kita akan belajar tentang apa yang bisa dilakukan oleh pengguna setelah melakukan otentikasi. Kita juga belajar tentang bagaimana membedakan hak akses dari masing-masing pengguna berdasarkan levelnya.

Halaman ini sengaja dikosongkan

# 12 Otorisasi Pengguna

Pada bab ini kita akan belajar tentang:

- Definisi otorisasi, *ACF vs RBAC*
- Implementasi otorisasi
- Penggunaan *extension* otorisasi

## A. Gambaran Umum

Otorisasi pengguna adalah pengaturan hak akses pengguna terhadap aplikasi. Hal ini berbeda dengan otentikasi yang hanya mengecek apakah pengguna sudah *login* atau belum saja, otorisasi mengatur apa yang boleh dan tidak boleh dilakukan oleh pengguna yang telah *login* atau belum pada aplikasi

Otorisasi cocok diterapkan untuk aplikasi yang memiliki beberapa level pengguna, misalnya level *guest* yang hanya bisa *view*, level *member* bisa mengunduh, dan level *admin* yang bisa melakukan apapun yang bisa dilakukan oleh dua level pengguna sebelumnya.

Yii menyediakan dua metode otorisasi, yaitu *Access Control Filter* (ACF) dan *Role-Based Access Control* (RBAC). Pada dasarnya keduanya sama-sama *access control*,

## B. Access Control Filter

### B. 1. Definisi

ACF adalah metode otorisasi yang sederhana yang cocok digunakan untuk aplikasi yang membutuhkan pengaturan hak akses yang *simple*. Sesuai dengan namanya, ACF bekerja dengan menyaring setiap *action* pada suatu *controller* ketika pengguna melakukan *request* terhadap suatu *action*.

### B. 2. ACF Dasar

Mari kita lihat contoh penggunaan ACF pada suatu *controller* (misalnya: SiteController)

```
<?php
...
use yii\filters\AccessControl;
...
class SiteController extends Controller
{
 public function behaviors()
 {
 return [
 'access' => [
 'class' => AccessControl::className(),
 'only' => ['logout', 'signup'],
 'rules' => [
 [
 'actions' => ['signup'],
 'allow' => true,
 'roles' => ['?'],
],
 [
 'actions' => ['logout'],
 'allow' => true,
 'roles' => ['@'],
],
],
],
];
 }
}
```

Pada kode di atas ACF diimplementasikan menggunakan fungsi *behaviors* dengan *access* dengan *class* `[[yii\filters\AccessControl]]`. Berikut ini penjelasan dari parameter yang digunakan.

- *Only* berisi daftar fungsi *action* yang dikenai *filter access* ini, yaitu *logout* (*actionLogout*), dan *signup* (*actionSignup*). Jika parameter ini tidak didefinisikan artinya *filter access* akan diimplementasikan pada semua fungsi *action*.
- *Rules* berisi daftar aturan *filter action* tentang boleh atau tidaknya pengguna tertentu mengakses suatu *action*.
  - o *Actions* berisi daftar fungsi *action* yang dikenai suatu aturan
  - o *Allow* bernilai *boolean* yang menunjukkan boleh tidaknya suatu *access*
  - o *Roles* berkaitan dengan siapa yang dikenai aturan tersebut. Tanda `?` menunjukkan pengguna yang belum *login* atau *guest*, tanda `@` menunjukkan pengguna yang sudah *login / authenticated user*. Di samping itu kita bisa menggunakan teks lain (nama *role*: *admin*, *staff*) yang akan men-trigger pemanggilan fungsi `yii\web\User::can()`, dimana masuk ke ranah RBAC.

*Rules* pertama bermakna fungsi *signup* diperbolehkan untuk diakses oleh pengguna yang belum *login*. Sedangkan *rules* kedua artinya fungsi *logout* hanya diperbolehkan untuk diakses oleh pengguna yang sudah *login*.

Dari contoh di atas kita bisa menarik kesimpulan sementara bahwa pengguna yang telah *login* atau *authenticated user* tidak lantas bisa melakukan semua *action*. Kita bisa menentukan, mana saja *action* yang boleh dilakukan oleh pengguna dan mana yang tidak.

### B. 3. Customize ACF

Lalu bagaimana jika kita ingin membuat *action* yang boleh diakses oleh pengguna yang sudah *login* maupun yang belum *login*? Maka kita perlu gunakan dua *roles* sekaligus.

```
'rules' => [
 [
 'actions' => ['index'],
 'allow' => true,
 'roles' => ['?','@'],
],
],
```

Ketika kita meng-*attach class* `[[yii\filters\AccessControl]]` pada *controller* maka secara *default* semua *action* tidak akan bisa diakses oleh pengguna.

Ada dua hal yang akan terjadi ketika pengguna mengakses suatu *action* yang mana dia tidak memiliki hak akses akan *action* tersebut.

1. Jika pengguna belum *login* atau *guest*, maka ACF akan memanggil fungsi `yii\web\User::loginRequired()` untuk mengalihkan alamat *web browser* ke halaman *login*.
2. Jika pengguna sudah *login* atau *authenticated user* maka ACF akan menjalankan `[[yii\web\ForbiddenHttpException]]`.

## Forbidden (#403)

You are not allowed to perform this action.

Kita juga bisa memodifikasi pesan galat di atas dengan menggunakan parameter *denyCallback*.

```
<?php
public function behaviors()
{
 return [
 'access' => [
 'class' => AccessControl::className(),
 ...
 'denyCallback' => function ($rule, $action) {
 throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk
mengakses halaman '.$action->id.' ini!');
 }
],
],
}
```

Contoh di atas bisa kita implementasikan pada *CategoryController*, dimana *controller* tersebut belum diimplementasikan otorisasi.

Seharusnya ACF hanyalah sebatas *on / off* pengaksesan suatu *action* tidak lebih dari itu. Jika otorisasi sudah menggunakan pembagian atau pengelompokan pengguna misalnya ada *admin*, *staff*, *member*, dsb, maka sudah waktunya

menggunakan RBAC. Kita bisa saja mengoprek ACF dan memaksanya bekerja layaknya RBAC hanya hal itu akan lebih ribet dan tidak sesuai dengan praktik terbaik Yii.

Hal ini terkait dengan dimana kita harus menyimpan data *role* dari pengguna. Untuk kasus yang sangat sederhana misalnya kita mempunya tiga jenis pengguna yaitu *guest*, *authenticated user* dan seorang *admin*, maka kita bisa memaksimalkan ACF yaitu dengan menggunakan parameter *matchCallback* pada *rules*.

Asumsinya, ID admin adalah 1, maka mari kita bisa membuat *controller* ini agar hanya bisa diakses oleh pengguna dengan ID sama dengan 1 atau admin. Lihat kode berikut.

```
'access' => [
 'class' => AccessControl::className(),
 'rules' => [
 [
 'allow' => true,
 'roles' => ['@',1],
 'matchCallback' => function ($rule, $action) {
 $user = \Yii::$app->user;
 if(in_array($user->id, $rule->roles))
 return true;
 }
],
 ...
],
]
```

## B. 4. Dynamic ACF

Apakah mungkin menggunakan *dynamic ACF*?

Jawabannya adalah mungkin sekali, namun kita perlu media penyimpanan semisal basis data untuk menyimpan data *route* (*module*, *controller*, *action*) dan ID pengguna yang diperbolehkan mengakses *action* tersebut. Sebagai contoh, kita buat tabel auth sebagai berikut.

yii2basic auth	
module :	varchar(60)
controller :	varchar(60)
action :	varchar(60)
user_id :	int(11)

Kemudian dengan menggunakan Gii, kita *generate* model tabel auth.

Berikut ini contoh implementasinya pada *matchCallback*.

```
'rules' => [
 [
 'allow' => true,
 'matchCallback' => function ($rule, $action) {
 $moduleId = $action->controller->module->id;
 $controllerId = $action->controller->id;
 $actionId = $action->id;
 $userId = \Yii::$app->user->id;
 $auth = \app\models\Auth::find()
 ->where([
 'module' => $moduleId,
 'controller' => $controllerId,
 'action' => $actionId,
 'user_id' => $userId,
])
 ->count();
 if($auth>0) return true;
 }
],
]
```

Kita bisa menggunakan parameter *action* pada *matchCallback* untuk mendapatkan tiga bagian penting dalam sebuah *route* yaitu ID *module*, ID *controller*, ID *action*. Untuk ID *module*, apabila tidak berada pada *module*, maka akan berisi ID dari aplikasi \Yii::\$app->id (lihat di config/web.php).

Kemudian dengan menggunakan ActiveRecord, dilakukan pengecekan apakah *action* dan pengguna tersebut terdaftar dalam basis data. Jika data ditemukan maka pengguna diizinkan mengakses, dan jika tidak maka akses ditolak.

Sebelum mengujinya, kita perlu masukkan secara manual data *action* yang diizinkan untuk ID pengguna tertentu ke dalam basis data tabel auth.

module	controller	action	user_id
basic	category	create	1
basic	category	index	1

Setelah itu kita bisa uji coba dengan mengakses fungsi *action* pada CategoryController. Maka kita hanya akan bisa mengakses suatu *action* yang telah terdaftar di basis data saja.

## B. 5. Implementasi ACF

Lalu, bagaimana cara mengimplementasikan ACF pada semua *controller*? Apakah dilakukan dengan manual yaitu mengesetnya satu persatu?

Jawabannya adalah tergantung apakah rule ACF tersebut spesifik atau tidak. Jika spesifik, maka kita perlu set ACF satu persatu pada setiap *controller*. Jika tidak spesifik maka kita bisa gunakan banyak cara di Yii, misalnya dengan membuat *base controller* yang telah diimplementasikan ACF, bisa juga melalui *event beforeAction*, melalui *behaviour access control*, serta kita bisa juga meletakkannya pada *Bootstrap*.

### Implementasi ACF pada Application Event di Config

Berikut ini contoh implementasi ACF pada *application event beforeAction*, pada *config* kita bisa gunakan *trigger* on *beforeAction*, yaitu *trigger* yang akan dijalankan sebelum *action* eksekusi.

```
<?php
$config = [
 'id' => 'basic',
 'basePath' => dirname(__DIR__),
 'bootstrap' => ['log'],
 'on beforeAction' => function($event) {
 $action = $event->action;
 $moduleID = $action->controller->module->id;
 $controllerID = $action->controller->id;
 $actionID = $action->id;
 $user = \Yii::$app->user;
 $userID = $user->id;
 if(!in_array($controllerID, ['default', 'site'])){
 $auth = \app\models\Auth::find()
 ->where([
 'module' => $moduleID,
 'controller' => $controllerID,
 'action' => $actionID,
 'user_id' => $userID,
])
 ->count();
 if($auth==0) {
 if (!$action instanceof \yii\web\ErrorAction) {
 if ($user->getIsGuest()) {
 $user->loginRequired();
 } else {
 throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
 }
 }
 }
 },
 },
],
```

Kode “if(!in\_array(\$controllerID, ['default', 'site'])) {“ dimaksudkan agar implementasi ACF dikecualikan untuk *controller default* dan *site*, ini hanya contoh. Sedangkan kode “if (!\$action instanceof \yii\web\ErrorAction) {” dimaksudkan untuk mencegah pemanggilan rekursif terhadap ErrorAction, hal ini dikarenakan Yii menggunakan *action* untuk menangani *action galat*.

### Implementasi ACF pada Application Event di Bootstrap

Cara lain untuk mengimplementasikan ACF adalah melalui *Application Event* pada *Bootstrap*.

Pertama kita buat dulu *class Auth* (tidak harus bernama *Auth*) yang implements *class [[yii\base\BootstrapInterface]]*.

```
@app/components/Auth.php
```

```
<?php
namespace app\components;
use Yii;
class Auth implements \yii\base\BootstrapInterface
{
 public function bootstrap($app)
 {
 $app->on(\yii\base\Application::EVENT_BEFORE_ACTION, function ($event) {
 $action = $event->action;
 $moduleId = $action->controller->module->id;
 $controllerId = $action->controller->id;
 $actionId = $action->id;
 $user = \Yii::$app->user;
 $userId = $user->id;
 if(!in_array($controllerId,['default','site'])) {
 $auth = \app\models\Auth::find()
 ->where([
 'module' => $moduleId,
 'controller' => $controllerId,
 'action' => $actionId,
 'user_id' => $userId,
])
 ->count();
 if($auth==0) {
 if (!($action instanceof \yii\web\ErrorAction)) {
 if ($user->getIsGuest()) {
 $user->loginRequired();
 } else {
 throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
 }
 }
 }
 });
 });
 }
}
```

Kemudian pada *config* kita cukup menambahkan *class* ini pada *application bootstrap*

```
<?php
...
$config = [
 'id' => 'basic',
 'basePath' => dirname(__DIR__),
 'bootstrap' => [
 'log',
 'app\components\Auth',
],
 'components' => [

```

Selesai.

### **Implementasi ACF pada Application Behaviour di Config**

Kita juga mengimplementasikan ACF pada *Application Behaviour* melalui *class* [[*yii\base\ActionFilter*]]. Jika pada *event* kita menggunakan kode *on*, maka pada *behavior* kita menggunakan *as*.

Pertama kita harus buat dulu sebuah *class behavior* kita yang *extends class ActionFilter*

```
@app/component/AccessControl.php
```

```
<?php
namespace app\components;
use Yii;
class AccessControl extends \yii\base\ActionFilter
{
 public function beforeAction($action)
 {
 $moduleId = $action->controller->module->id;
 $controllerId = $action->controller->id;
```

```

$actionID = $action->id;
$user = \Yii::$app->user;
$userId = $user->id;
if(!in_array($controllerID,['default','site'])){
 $auth = \app\models\Auth::find()
 ->where([
 'module' => $moduleID,
 'controller' => $controllerID,
 'action' => $actionID,
 'user_id' => $userId,
])
 ->count();
 if($auth==0) {
 if (!$action instanceof \yii\web\ErrorAction) {
 if ($user->getIsGuest()) {
 $user->loginRequired();
 } else {
 throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
 }
 }
 }
}
return true;
}
}

```

Kemudian pada *config* (@app/config/web.php) kita bisa definisikan *behavior* yang telah kita buat tersebut.

```

<?php
...
$config = [
 'id' => 'basic',
 'basePath' => dirname(__DIR__),
 'bootstrap' => ['log'],
 'as access' => [
 'class' => 'app\components\AccessControl',
],
]

```

Selesai.

Cara ketiga ini yang lebih disukai penulis, karena sesuai dengan peruntukannya. Kita tentu bisa saja memotong daging dengan pisau biasa asal tajam, tapi tentu sebaiknya menggunakan pisau khusus daging, kira-kira demikian analoginya.

## B. 6. User Interface ACF

Pada bagian sebelumnya, kita harus menginputkan data hak akses pada tabel auth sebelum kita menguji coba apakah ACF bekerja dengan baik. Bagi pemrogram aplikasi, tentunya melakukan hal ini adalah hal yang *simple*, hanya saja untuk *administrator* yang hanya pengguna akhir tentu sangat merepotkan jika harus menginput satu persatu secara manual.

Berdasarkan hal tersebut maka menyediakan *user interface* (UI) dari ACF merupakan sebuah keharusan pada aplikasi yang menerapkan ACF.

Lalu bagaimana? Tidak ada referensi tentang hal ini.

Oke, mari kita berfikir sederhana, bahwa sebenarnya yang paling utama dibutuhkan dalam pembuatan UI ACF ini adalah daftar semua *route* (*module*, *controller*, dan *action*) dari aplikasi kita.

Alur dari UI ACF dimulai dari tampilan daftar pengguna baru kemudian pada masing-masing pengguna ada *action* untuk menentukan hak aksesnya yang berisi daftar semua *route* dan *checklist*-nya.

Mari kita buat dulu tampilan daftar pengguna dengan menggunakan *tools Gii*, *CRUD Generator*.

∞ <http://localhost/basic/web/gii/crud>

Isi kolom *Model Class* dengan app\models\User dan kolom *Controller Class* dengan app\controller\AuthController

## CRUD Generator

This generator generates a controller and view model.

### Model Class

app\models\User

### Search Model Class

### Controller Class

app\controllers\AuthController

Klik *preview*, dan cukup *checklist* tiga file saja yaitu AuthController, index, dan view, karena kita hanya membutuhkan tiga file itu.

**Preview** **Generate**

Click on the above **Generate** button  Create  Unchanged  Overwrite to generate the files selected below:

Code File	Action	<input type="checkbox"/>
controllers\AuthController.php	create	<input checked="" type="checkbox"/>
views\auth\_form.php	create	<input type="checkbox"/>
views\auth\create.php	create	<input type="checkbox"/>
views\auth\index.php	create	<input checked="" type="checkbox"/>
views\auth\update.php	create	<input type="checkbox"/>
views\auth\view.php	create	<input checked="" type="checkbox"/>

Klik tombol *Generate*.

Modifikasi file @app/views/auth/index.php

```
<?php
use yii\helpers\Html;
use yii\grid\GridView;
$this->title = 'Authorization';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="auth-index">
 <h1><?= Html::encode($this->title) ?></h1>
 <?= GridView::widget([
 'dataProvider' => $dataProvider,
 'columns' => [
 ['class' => 'yii\grid\SerialColumn'],
 'username',
 [
 'header' => 'Auth',
 'format' => 'raw',
 'value' => function ($model) {
 return Html::a("<i class='glyphicon glyphicon-lock'></i>",
 ['view', 'id'=>$model->id],
 ['class'=>'btn btn-danger btn-xs'])
 };
],
],
]);
 </div>
?>
```

Intinya dengan menambahkan tautan ke actionView dimana *action* ini akan kita tampilkan semua *route* yang bisa di-checklist.

# Authorization

Showing 1-5 of 5 items.

#	Username	Auth
1	admin	
2	test	

Selanjutnya kita tambahkan fungsi `getRoutes` pada `controller` untuk mendapatkan semua `route (module, controller, action)` dari aplikasi kita. Berikut ini ada `code snippet` yang penulis ambil dan rangkum dari `extension hscstudio/yii2-mimin`

```
public function getRoutes($app, $moduleId, $namespace, $user_id)
{
 $routes = [];
 $path = @Yii::getAlias('@' . str_replace('\\', '/', $namespace));
 foreach (scandir($path) as $file) {
 if (strcmp(substr($file, -14), 'Controller.php') === 0) {
 $controllerID = \yii\helpers\Inflector::camel2id(substr(basename($file), 0, -14));
 $className = $namespace . '\yii\helpers\Inflector::id2camel($controllerID)' .
 'Controller';
 $controller = Yii::createObject($className, [$controllerID, $app]);
 $controllerID = $controller->uniqueId;
 foreach ($controller->actions() as $actionID => $value) {
 $auth = \app\models\Auth::find()->where([
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'user_id'=>$user_id,
])->count();
 $routes[] = [
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'auth'=>$auth,
];
 }
 }

 $class = new \ReflectionClass($controller);
 foreach ($class->getMethods() as $method) {
 $name = $method->getName();
 if ($method->isPublic() && !$method->isStatic() && strpos($name, 'action') === 0 && $name !== 'actions') {
 $actionID = \yii\helpers\Inflector::camel2id(substr($name, 6));
 $auth = \app\modules\Auth::find()->where([
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'user_id'=>$user_id,
])->count();
 $routes[] = [
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'auth'=>$auth,
];
 }
 }
 }
 return $routes;
}
```

Inti dari kode di atas adalah mendapatkan semua `controller` dengan cara men-scan semua `file` dalam `folder controllers` yang terindikasi merupakan `controller`.

```
| foreach (scandir($path) as $file) {
```

Kemudian dari file controller tersebut dideteksi semua fungsi *actionnya*.

```
$controller = Yii::createObject($className, [$controllerID, $app]);
$controllerID = $controller->uniqueId;
foreach ($controller->actions() as $actionID => $value) {
```

Lalu pada setiap fungsi *action*, dicek apakah *action* tersebut telah terdaftar pada basis data tabel auth atau belum.

```
$auth = \app\modules\Auth::find()->where([
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'user_id'=>$user_id,
])->count();
```

Hasil keluaran dari fungsi ini adalah *routes* dan auth yang bernilai 1 atau 0.

```
$routes[] = [
 'module'=>$moduleId,
 'controller'=>$controllerID,
 'action'=>$actionID,
 'auth'=>$auth,
];
```

Kemudian kita harus memodifikasi fungsi *actionView* untuk memanggil fungsi *getRoutes* di atas.

```
public function actionView($id)
{
 $app = \Yii::$app;
 $moduleId = $app->id;
 $namespace = trim($app->controllerNamespace, '\\') . '\\';
 $routes = $this->getRoutes($app, $moduleId, $namespace, $id);
 foreach ($app->getModules() as $moduleID => $child) {
 if (($module = $app->getModule($moduleId)) !== null) {
 $namespace = trim($module->controllerNamespace, '\\') . '\\';
 $routes = array_merge($routes, $this->getRoutes($module, $moduleId, $namespace, $id));
 }
 }

 return $this->render('view', [
 'model' => $this->findModel($id),
 'routes' => $routes,
]);
}
```

Kemudian kita juga perlu memodifikasi *view* @app/views/auth/view.php, yaitu menampilkan *routes* dan *checklist*.

```
<?= DetailView::widget([
 'model' => $model,
 'attributes' => [
 'id',
 'username',
],
]) ?>

| Modules | Controllers | Actions | Auth |
|---------|-------------|---------|------|
|---------|-------------|---------|------|


```

```
?>
</table>
```

Mari kita lihat hasilnya

∞ <http://localhost/basic/web/auth/view?id=1>

## Auth user#1

<b>Id</b>	1		
<b>Username</b>	admin		
<hr/>			
Modules	Controllers	Actions	Auth
basic	auth	index	<input type="checkbox"/>
basic	auth	view	<input type="checkbox"/>
basic	category	index	<input checked="" type="checkbox"/>
basic	category	view	<input type="checkbox"/>
basic	category	create	<input checked="" type="checkbox"/>

Kita lihat bahwa pada kolom Auth tampilan *checklist* sudah sesuai dengan data pada tabel auth, dimana jika terdaftar maka akan ter-checklist.

Langkah selanjutnya adalah kita akan membuat fitur untuk menentukan hak akses melalui *checklist* tersebut. Ketika pengguna mencentang *checkbox* maka akan melakukan proses *insert* atau *delete data routes* pada tabel auth.

Oleh karena itu, kita ubah sedikit kode untuk *checkbox*

```
<td><?= Html::checkbox('auth[]', $row['auth'], [
 'class' => 'processAuth',
 'data-module' => $row['module'],
 'data-controller' => $row['controller'],
 'data-action' => $row['action'],
]); ?></td>
```

Kemudian kita tambahkan Javascript untuk melakukan *request* ke fungsi pemrosesan auth secara Ajax.

```
<?php
$this->registerJs('
$(".processAuth").on("click", function (e) {
 module = $(this).attr("data-module");
 controller = $(this).attr("data-controller");
 action = $(this).attr("data-action");
 user_id = '.$model->id.';
 checked = $(this).prop("checked");
 var link = "' . Url::to(['process-auth']) . '?module=' + module +
 '&controller=' + controller + '&action=' + action + '&user_id=' + user_id +
 '&checked=' + checked;
 $.get(link, function(data) {
 alert(data)
 });
});
');
```

Selanjutnya, pada kita buat fungsi *action* untuk memproses auth yaitu *actionProcessAuth()*

∞ @app/controllers/AuthController.php

```
public function actionProcessAuth($module, $controller, $action, $user_id, $checked)
{
 $params = [
 'module' => $module,
 'controller' => $controller,
 'action' => $action,
 'user_id' => $user_id,
];
 $auth = Auth::find()->where($params)->count();
 if ($checked) {
```

```

if($auth==0) {
 $model = new Auth($params);
 $model->save();
}
return "success inserted";
}
else{
 if($auth>0) {
 Auth::find()->where($params)->delete();
 }
 return "success deleted";
}
}

```

Kode di atas hanyalah *insert* atau *delete* data *route* dari tabel *auth*.

## C. Role Base Access Control

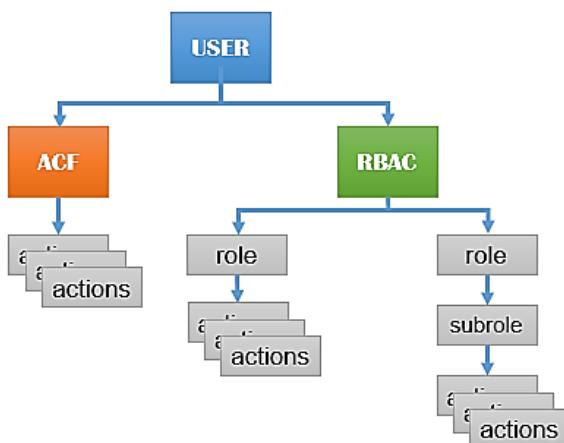
### C. 1. Definisi

RBAC adalah mekanisme pengaturan hak akses berdasarkan *role* pada aplikasi. Mudahnya *role* bisa diartikan sebagai *group* dari pengguna meskipun kenyataanya *role* lebih tepat diartikan sebagai *group* dari hak akses. Contoh role: *admin*, *member*, *author*, dst.

#### Apa perbedaannya dengan ACF?

Perbedaan mendasar antara RBAC dan ACF meskipun keduanya sama-sama *access control*, yaitu RBAC menggunakan konsep *role* dalam pengaturan hak aksesnya sedangkan ACF hanya menggunakan konsep *on off* saja.

Berikut ini gambaran sederhana dari ACF vs RBAC.



#### Apa kelebihannya dibandingkan dengan ACF?

RBAC memungkinkan dan memudahkan kita membuat hak akses pengguna secara bertingkat sebagaimana yang digambarkan pada bagan di atas.

#### Konsep Dasar

Sebuah *role* merupakan kumpulan dari *permission* (contoh *permission*: membuat artikel, mengedit artikel). Sebuah *role* bisa dipasangkan ke satu atau lebih pengguna. Untuk mengecek apakah seorang pengguna mempunyai *permission* tertentu maka kita bisa mengeceknya melalui *rolenya*.

Hubungan antara *role* dan *permission* bisa menjadi sebuah *rule*. Sebuah *rule* merupakan kode yang akan dieksekusi ketika terjadi pengecekan *role* dan *permission*. Contoh sederhana, sebuah situs web berita dengan beberapa *author*, kita bisa mengatur agar yang berhak untuk mengedit artikel adalah *author*-nya.

*Role* dan *permission* bisa diatur menjadi sebuah hierarki hak akses. Sebuah *role* bisa memiliki beberapa *role* lain dan *permission*. Sebuah *permission* juga bisa memiliki beberapa *permission* lain. Sub *role* pada Yii juga bisa memiliki sub dari sub *role* hingga tidak terbatas. Demikian juga dengan *permission*.

Apa contoh *role*? *Administrator*, bagian keuangan, *marketing*, *author*, dll.

Apa contoh *permission*? Misalnya: mengelola artikel, mengelola data pengguna, dll.

Sebenarnya, ada satu istilah lagi dalam RBAC yaitu *operation*, namun istilah ini tidak ada pada RBAC Yii. *Operation* sebenarnya merupakan level di bawah *permission*, sehingga satu *permission* bisa memiliki banyak *operation*. Contoh *operation* adalah membuat artikel, mengedit artikel, menghapus artikel, dll.

Dengan definisi tersebut, maka kita bisa tarik kesimpulan bahwa *operation* bisa kita padankan dengan fungsi *action* (*actionCreate*, *actionUpdate*, dll). Pada *guide official* Yii tentang RBAC, sebenarnya ada sedikit kerancuan, dimana pada *guide* dijelaskan bahwa contoh *permission* itu adalah *Create Post*, *Update Post*, dll. Bukankah ini masuk dalam *operation*? atau fungsi *action*?

Jangan bingung karena konsep RBAC bukanlah buatan Yii, melainkan Yii hanya mengadopsinya dengan beberapa penyesuaian.

∞ [https://en.wikipedia.org/wiki/Role-based\\_access\\_control](https://en.wikipedia.org/wiki/Role-based_access_control)

## C. 2. Konfigurasi

Sebelum kita menggunakan RBAC, maka kita harus terlebih dahulu melakukan konfigurasi component *authManager*. Yii mempunyai dua jenis *class* untuk mengontrol otorisasi, yaitu [[*yii\rbac\PhpManager*]] dan [[*yii\rbac\DbManager*]].

Jika kita menggunakan jenis *class* *PhpManager*, maka Yii akan menggunakan *file PHP* biasa untuk menyimpan data otorisasi, ini cocok untuk RBAC yang sederhana dan tidak banyak perubahan data. Sedangkan *class* *DbManager*, menyimpan data otorisasinya menggunakan basis data.

### **PhpManager**

Jika kita menggunakan *PhpManager* maka kita bisa melakukan konfigurasi dengan mendefinisikannya pada *component authManager* dengan *class* [[*yii\rbac\PhpManager*]]:

@app/config/web.php

```
$config = [
 ...
 'components' => [
 ...
 'authManager' => [
 'class' => 'yii\rbac\PhpManager',
],
 ...
],
]
```

Dengan konfigurasi di atas maka sekarang kita bisa mengakses *authManager* dengan kode \Yii::\$app->*authManager*. Secara *default*, *PhpManager* akan menyimpan data RBAC pada direktori @app/rbac. Oleh karena itu pastikan bahwa direktori ini bisa ditulisi.

### **DbManager**

Dengan cara yang sama, kita juga mengkonfigurasi RBAC menggunakan *class* *DbManager*:

```
$config = [
 ...
 'components' => [
 ...
 'authManager' => [
 'class' => 'yii\rbac\DbManager',
],
 ...
],
]
```

Berbeda dengan *PhpManager*, *DbManager* menyimpan data RBAC pada basis data dengan menggunakan empat tabel, yaitu:

- itemTable: tabel untuk menyimpan otorisasi items. *Default*-nya adalah "auth\_item".
- itemChildTable: tabel untuk menyimpan hierarki otorisasi item. *Default*-nya adalah "auth\_item\_child".
- assignmentTable: tabel untuk menyimpan data *assignment* item otorisasi. *Default*-nya adalah "auth\_assignment".
- ruleTable: tabel untuk menyimpan *rules*. *Default*-nya adalah "auth\_rule".

Oleh karena kita harus membuat keempat tabel di atas pada basis data kita. Yii telah menyiapkan migrasi basis data untuk keempat tabel ini, dimana kode migrasinya tersimpan pada direktori @yii/rbac/migrations. Pada CMD kita bisa menjalankan perintah berikut.

| yii migrate --migrationPath=@yii/rbac/migrations

### C. 3. Memasukkan Data RBAC

Pada ACF, kita tidak perlu memasukkan data, karena hanya berupa *on off* saja sehingga cukup berdasarkan data otentifikasi pengguna saja, apakah pengguna sudah atau belum *login*.

Kita perlu membuat *controller* baru (sementara) untuk memasukkan data RBAC secara manual.

### C. 4. Implementasi RBAC

Pada ACF kita bisa mengimplementasikan ACF, salah satunya dengan menggunakan **behavior access** pada **controller** melalui *class* [[*yii\filters\AccessControl*]]. Demikian juga pada RBAC kita bisa melakukan hal yang sama.

```
public function behaviors()
{
 return [
 'access' => [
 'class' => AccessControl::className(),
 'rules' => [
 [
 'allow' => true,
 'actions' => ['action1', 'action2'],
 'roles' => ['author'],
],
],
],
];
}
```

Jika sebelumnya pada ACF kita hanya bisa menggunakan karakter ? dan @ saja maka pada RBAC kita bisa menggunakan *role* atau *permission*. Dengan kita memasukkan karakter selain ? dan @ maka akan men-trigger Yii untuk menjalankan perintah *yii\web\User::can()*

Belajar dari ACF, kita bisa mengimplementasikan RBAC juga pada *Application Behaviour* melalui *class* [[*yii\base\ActionFilter*]].

`@app/component/Rbac.php`

```
<?php
namespace app\components;
use Yii;
class Rbac extends \yii\base\ActionFilter
{
 public function beforeAction($action)
 {
 $actionID = $action->id;
 $user = \Yii::$app->user;
 if ($user->can('/' . $actionID)) {
 return true;
 }

 $controllerID = $action->controller->id;
 if(in_array($controllerID, ['default', 'site'])){
 return true;
 }

 if (!$action instanceof \yii\web\ErrorAction) {
 if ($user->getIsGuest()) {
 $user->loginRequired();
 } else {
 throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk
mengakses halaman ' . $action->id . ' ini!');
 }
 }
 }
}
```

Dengan langkah yang hampir sama dengan ACF, maka kita juga bisa membuat *user interface* untuk RBAC.

Ini tantangan untuk anda! ☺

### C. 5. Menggunakan Extension RBAC

Untuk memudahkan kita mengimplementasi RBAC maka kita bisa menggunakan *extension RBAC manager*.

### **Extension Yii2-admin**

Salah satu *extension* untuk menangani RBAC yang cukup populer adalah buatan guru penulis yaitu yii2-admin yang bisa kita lihat melalui tautan berikut

```
∞ https://github.com/mdmsoft/yii2-admin
```

*Extension* ini mengeksplorasi semua fitur RBAC yang dimiliki Yii. Untuk kasus pengaturan hak akses yang kompleks maka *extension* ini bisa menanganinya dengan baik. Jika kita mempunyai dasar pengetahuan yang baik tentang RBAC maka penulis sangat menyarankan untuk menggunakan *extension* ini.

*Extension* ini tidak akan penulis bahas pada buku ini.

### **Extension Yii2-mimin**

*Extension* lain yang bisa dijadikan alternatif adalah *extension* yii2-mimin

```
∞ https://github.com/hscstudio/yii2-mimin
```

*Extension* ini merupakan versi mini dari yii2-admin. Tidak semua fitur RBAC bisa diterapkan pada *extension* ini, namun tampilan antaramukanya cukup mudah difahami serta memiliki beberapa fitur yang memudahkan kita dalam menerapkan RBAC.

Mari kita menginstalnya menggunakan Composer.

```
| composer require --prefer-dist hscstudio/yii2-mimin "*"
```

Jika sudah, maka lakukan konfigurasi pada `@app/config/web.php`

```
'as access' => [
 'class' => '\hscstudio\mimin\components\AccessControl',
 'allowActions' => [
 // add wildcard allowed action here!
 'site/*',
 'debug/*',
 'mimin/*', // only in dev mode
],
],
...
'modules' => [
 'mimin' => [
 'class' => '\hscstudio\mimin\Module',
],
 ...
],
'components' => [
 'authManager' => [
 'class' => 'yii\rbac\DbManager', // only support DbManager
],
 ...
],
]
```

Untuk keperluan migrasi basis data, maka kita juga harus menambahkan konfigurasi authManager pada `@app/config/`

```
'components' => [
 'authManager' => [
 'class' => 'yii\rbac\DbManager',
],
 ...
],
]
```

Karena *extension* ini menggunakan `[[yii\rbac\DbManager]]` sebagai authManager, maka kita perlu menjalankan migrasi basis data untuk meng-generate tabel-tabel yang diperlukan oleh RBAC

```
| yii migrate --migrationPath=@yii/rbac/migrations
```

```
C:\Bitnami\wampstack\apache2\htdocs\basic>yii migrate --migrationPath=@yii/rbac/migrations
Yii Migration Tool (based on Yii v2.0.6)

Total 1 new migration to be applied:
 m140506_102106_rbac_init

Apply the above migration? (yes|no) [no]:yes
*** applying m140506_102106_rbac_init
 > create table {{%auth_rule}} ... done (time: 2.438s)
 > create table {{%auth_item}} ... done (time: 1.677s)
 > create index idx-auth_item-type on {{%auth_item}} (type)
... done (time: 0.967s)
 > create table {{%auth_item_child}} ... done (time: 1.363s)
```

Di samping itu, untuk keperluan menyimpan data *routing*, maka kita juga harus menjalankan migrasi basis data berikut.

```
| yii migrate --migrationPath=@hscstudio/mimin/migrations
```

```
C:\Bitnami\wampstack\apache2\htdocs\basic>yii migrate --migrationPath=@hscstudio/mimin/migrations
Yii Migration Tool (based on Yii v2.0.6)

Total 1 new migration to be applied:
 m151024_072453_create_route_table

Apply the above migration? (yes|no) [no]:yes
*** applying m151024_072453_create_route_table
 > create table {{%route}} ... done (time: 0.522s)
*** applied m151024_072453_create_route_table (time: 0.821s)
```

Setelah semua prosedur di atas kita lakukan, maka sekarang kita bisa mengelola otorisasi melalui *user interface* dari *extension* ini.

### Mengelola Routing

Untuk mengelola *routing* melalui tautan berikut.

```
∞ http://localhost/path/to/index.php?r=mimin/route
```

## Routes

#	Type	Alias	Name	Status
				on

No results found.

Fitur ini digunakan untuk mendaftarkan *route* dari aplikasi. Kita bisa mendaftarkan secara manual melalui tombol *Create Route*, atau meng-generate secara otomatis melalui tombol *Generate Route*.

Jika kita menekan tombol *Generate route* maka *extension* ini akan akan men-scan seluruh *route* pada aplikasi sebagai mana yang telah kita bahas pada bagian sebelumnya, kemudian menyimpannya ke basis data tabel *route*.

## Routes

Showing 1-20 of 61 items.					
#	Type	Alias	Name	Status	
1	*		/	<input checked="" type="checkbox"/> On	
2	admin	*	/admin/*	<input checked="" type="checkbox"/> On	
3	admin/default	*	/admin/default/*	<input checked="" type="checkbox"/> On	
4	admin/default	index	/admin/default/index	<input checked="" type="checkbox"/> On	
5	auth	*	/auth/*	<input checked="" type="checkbox"/> On	

Pada *route* ini, kita juga bisa melakukan proses CRUD, termasuk meng-on/off-kan suatu *route*. Ada empat kolom yaitu *Type*, *Alias*, *Name* dan *Status*.

- Kolom *Type* menunjukkan controllerID atau *route module* atau bisa juga bertindak sebagai *group* dari *route*. Oleh karena itu kita bebas mengeditnya agar lebih mudah dibaca. Misal *Type admin/default* menjadi “Module Admin”.
- Kolom *Alias* menunjukkan alias dari suatu *route*, dan kita bebas mengeditnya agar lebih nyaman dibaca. Misal *Alias \** menjadi “all”
- Kolom *Name* menunjukkan *route* aplikasi, direkomendasikan untuk tidak mengedit kolom ini, biarkan apa adanya hasil dari *generate* tersebut. Tanda \* artinya semua *action*
- Sedangkan kolom *Status* menunjukkan apakah *route* ini diaktifkan atau tidak. Jika di *set off*, maka route ini tidak akan muncul di *role* sehingga tidak bisa *diassign* ke *role*.

Untuk mengedit atau menghapus suatu route kita bisa gunakan tombol yang tersedia di kolom paling kanan. Berikut ini contoh formulir *update route*.

### Update Route: /category/view

**Name**

**Alias**

**Type**

**Status**  
 On

**Update**

### Mengelola Role

Untuk mengelola *role* melalui tautan berikut.

∞ <http://localhost/path/to/index.php?r=mimin/role>

## Roles

Create Role	
#	Name
No results found.	

Fitur ini berfungsi untuk mendefinisikan *role* atau *level* akses dari pengguna. Misalnya *role admin, author, staf*, dsb. Melalui fitur ini kita kemudian bisa meng-*assign route* pada suatu *role* tertentu.

Pertama, kita harus *create role* dulu dengan menekan tombol *Create Role*.

## Create Role

Name	<input type="text" value="Administrator"/>
<input type="button" value="Create"/>	

Kemudian kita akan dibawa ke tampilan *view* yang juga terdapat fitur pemilihan hak akses *route* untuk *role Administrator*

## Administrator

		<input type="button" value="Update"/>	<input type="button" value="Delete"/>
Name		Administrator	
Created At		06 Feb 2016 17:02:03	
Updated At		06 Feb 2016 17:02:03	
<b>Type</b> <b>Permision</b>			
admin/default	<input type="checkbox"/> -	<input type="checkbox"/> index	
auth	<input type="checkbox"/> -	<input type="checkbox"/> index	<input type="checkbox"/> process-auth
category	<input type="checkbox"/> -	<input type="checkbox"/> check	<input type="checkbox"/> check-sse
debug/default	<input type="checkbox"/> -	<input type="checkbox"/> db-explain	<input type="checkbox"/> download-mail
gii/default	<input type="checkbox"/> -	<input type="checkbox"/> action	<input type="checkbox"/> diff
mimin/role	<input type="checkbox"/> -	<input type="checkbox"/> create	<input type="checkbox"/> delete
			<input type="checkbox"/> index
			<input type="checkbox"/> permission

Untuk meng-assign route pada role Administrator, maka cukup dengan mencentang *checkbox route / permission*.

Lakukan hal yang sama untuk semua role.

## Mengelola Role dari Pengguna

Untuk mengelola *role* dari pengguna melalui tautan berikut.

∞ <http://localhost/path/to/index.php?r=mimin/user>

Users						
<a href="#">Create User</a> Showing 1-5 of 5 items.						
#	Username	Email	Roles	Status	Created At	Updated At
1	admin	admin@gmail.com		<span>Active</span>	31 Dec 1969 16:00:00	31 Dec 1969 16:00:00
2	test	test@gmail.com		<span>Banned</span>	05 Jan 2016 21:01:39	06 Feb 2016 15:47:13

Fitur ini digunakan untuk mengelola data pengguna, ditambah fitur untuk meng-assign akun pengguna pada *role* tertentu.

Cara meng-assign-nya melalui tombol *view* (gambar mata)

Created At	0
Updated At	0
<b>Role</b>	
<input type="text" value="Pilih role ..."/>	
<input type="button" value="Update"/>	

Kemudian kita bisa memilih *role* yang telah kita masukkan pada bagian sebelumnya. Kemudian diakhiri dengan menekan tombol *Update*.



Selesai

### Implementasi pada Button

Kita bisa mengimplementasikan RBAC ini pada *button*/tombol atau tautan, sehingga tombol tersebut hanya akan tampil ketika pengguna memang memiliki hak akses akan *route* atau *action* pada tombol tersebut.

Fungsi yang digunakan adalah `Mimin::checkRoute`. Berikut ini contoh implementasinya pada tombol *Create Category* di

```
@app/views/category/index.php
```

```
use hscstudio\mimin\components\Mimin;
if ((Mimin::checkRoute($this->context->id.'/create'))){
 echo Html::a('Create Category', ['create'], ['class' => 'btn btn-success']);
}
```

### Implementasi pada Menu

*Extension* ini juga memiliki fitur untuk menyaring menu yang akan tampil sesuai dengan hak akses pengguna. Yaitu dengan menggunakan fungsi `Mimin::filterMenu`. Sebagai contoh, hal ini bisa kita implementasikan pada main menu di `@app/views/layouts/main.php`

```
use hscstudio\mimin\components\Mimin;
$menuItems = [
 ['label' => 'Home', 'url' => ['/site/index']],
 ['label' => 'About', 'url' => ['/site/about']],
 ['label' => 'Contact', 'url' => ['/site/contact']],
];

if (\Yii::$app->user->isGuest){
 $menuItems[] = ['label' => 'Login', 'url' => ['/site/login']];
}
else{
 $menuItems[] = ['label' => 'App', 'items' => [
 ['label' =>'Category','url'=>['/category/index']],
 ['label' =>'Product','url'=>['/product/index']],
 ['label' =>'Cart','url'=>['/cart/index']],
]];
 $menuItems[] = [
 'label' => 'Logout (' . \Yii::$app->user->identity->username . ')',
 'url' => ['/site/logout'],
 'linkOptions' => ['data-method' => 'post']
];
}

$menuItems = Mimin::filterMenu($menuItems);

echo Nav::widget([
 'options' => ['class' => 'navbar-nav navbar-right'],
 'items' => $menuItems,
]);
```

### Implementasi pada Gridview

*Extension* ini juga memiliki fitur untuk menyaring *template button* pada *Gridview action column*. Berikut ini contoh implementasinya.

```
use hscstudio\mimin\components\Mimin;
echo GridView::widget([
 'dataProvider' => $dataProvider,
 'searchModel' => $searchModel,
 'columns' => [
 ...
 [
 'class' => 'yii\grid\ActionColumn',

```

```
'template' => Mimin::filterActionColumn([
 'view', 'update', 'delete'
], $this->context->route),
...
],
]);
});
```

Dengan kode di atas maka *button actionColumn* hanya akan menampilkan *button* yang sesuai dengan hak akses saja.

## D. Kesimpulan

Otorisasi adalah mekanisme pengaturan hak akses pengguna terhadap fungsi-fungsi *action* pada aplikasi. Yii menyediakan dua metode otorisasi, yaitu ACF dan RBAC. ACF umumnya digunakan untuk pengaturan hak akses yang sederhana yaitu boleh tidaknya pengguna mengakses suatu *action*, sedangkan RBAC lebih dari itu karena merupakan pengaturan hak akses berdasarkan level / *rolenya*.

Pada bab selanjutnya, kita akan belajar tentang teknik untuk mengunggah *file* ke server, impor data, pelaporan serta pembuatan grafik.

Halaman ini sengaja dikosongkan!

# 13 Unggah, Impor & Pelaporan

Pada bab ini kita akan belajar tentang:

- Mengunggah *file*
- Mengimpor data
- Pelaporan
- Visualisasi Data

## A. Mengunggah *File*

### A. 1. Definisi

Adakalanya, kita membuat aplikasi yang memerlukan fitur unggah *file* ke server. Sebagai contoh sederhana adalah pada formulir update profil foto pengguna, atau formulir unggah pada aplikasi *file sharing*. Pada dasarnya, implementasi proses unggah *file* di Yii adalah sama dengan cara pada umumnya.

Alur kerja proses unggah *file* adalah mula-mula pengguna mengunggah *file* melalui formulir unggah di *web browser*, lalu *web browser* melakukan pengecekan *file* (ukuran, tipe, dll), dan jika file tersebut valid maka kemudian *file* tersebut akan diunggah dan disimpan sementara di server pada direktori *temporary* (temp), dan pada akhirnya aplikasi akan memindahkan ke direktori tertentu.

Ada beberapa hal yang perlu diperhatikan ketika kita akan membuat fitur unggah *file*, diantaranya

- Besar *file* maksimal
- Format *file* (ekstensi *file*) yang diizinkan
- Lokasi *file* di server

### A. 2. Konfigurasi

Umumnya, tidak ada konfigurasi khusus untuk membuat fitur unggah *file*. Namun tidak ada salahnya kita memastikan bahwa server kita siap menerima *file* yang akan diunggah oleh pengguna. Untuk itu kita perlu memeriksa konfigurasinya pada *file* *php.ini*. Pada komputer penulis terletak di C:\Bitnami\wampstack\php\php.ini

```

785 ;;;;;;;;;;;
786 ; File Uploads ;
787 ;;;;;;;;;;;
788
789 ; Whether to allow HTTP file uploads.
790 ; http://php.net/file-uploads
791 file_uploads = On
792
793 ; Temporary directory for HTTP uploaded
794 ; files (will use system default if not
795 ; specified).
796 ; http://php.net/upload-tmp-dir
797 upload_tmp_dir =
798 "C:/Bitnami/WAMPST~1/php/tmp"
799
800 ; Maximum allowed size for uploaded files.
801 ; http://php.net/upload-max-filesize
802 upload_max_filesize = 40M
803
804 ; Maximum number of files that can be
805 ; uploaded via a single request
806 max_file_uploads = 20

```

Berikut ini penjelasannya.

- file\_upload

Pastikan di *set On*, untuk mengizinkan pengguna mengunggah *file* ke server menggunakan PHP

- upload\_tmp\_dir

Merupakan lokasi dimana *file* akan diunggah untuk sementara di server. Kita bisa mengubahnya dan memastikan bahwa direktori tersebut *permission-nya writable* (server Linux/Macintosh)

- upload\_max\_filesize

Menunjukkan maksimal ukuran *file* yang boleh diunggah oleh pengguna, 40M artinya 40 megabytes. Secara *default*, PHP mengeset maksimal 2M, sehingga banyak *newbie* yang bingung ketika gagal mengunggah *file* karena *file* yang diunggahnya ternyata lebih dari dua megabytes. Pertanyaan karena kegagalan ini telah berulang-ulang ditanyakan di forum, apakah anda juga akan melakukannya?

- max\_file\_uploads

Menunjukkan jumlah maksimal *file* yang diunggah melalui satu *request (form)*, untuk kasus unggah banyak *file* dalam satu *form*.

Di samping itu, pastikan juga bahwa lokasi direktori target unggah di *server permission-nya set writable*.

Kita juga perlu memastikan *library* fileinfo diaktifkan, karena Yii menggunakanannya untuk mendekripsi informasi tentang *file* yang diunggah.

```
868 | ;extension=php_bz2.dll
869 | extension=php_curl.dll
870 | extension=php_fileinfo.dll
871 | extension=php_gd2.dll
872 | ;extension=php_gettext.dll
```

Di sisi kode, yaitu formulir unggah, kita harus mendefinisikan *parameter enctype = multipart/form-data*. Parameter ini mengizinkan formulir untuk mengirimkan *file* ke server.

```
<form "enctype"="multipart/form-data">
 ...
 <input type="file" />
 ...
</form>
```

### A. 3. Implementasi

Sebagai studi kasus, kita akan membuat *fitur update* profil pengguna, yang mengizinkan pengguna untuk mengunggah foto profilnya. Lokasi *file* foto di server akan kita simpan pada tabel user\_photo

v yii2basic user_photo	
✉	user_id : int(11)
📷	photo : varchar(255)

Silakan dibuat tabel user\_photo dengan struktur seperti gambar di atas.

#### Model & Validasi

Pada fitur unggah *file*, model memiliki fungsi yang cukup penting yaitu sebagai *validation control*, yang mana definisi dari validasi bisa kita deklarasikan pada model ini. Perlu dicatat bahwa, unggah *file* tidak harus menggunakan model dengan *extends* ke ActiveRecord bahkan tidak menggunakan modelpun bisa. Kita juga bisa menggunakan DynamicModel.

Setelah membuat tabel user\_photo, kemudian kita generate modelnya menggunakan Gii.

# Model Generator

This generator generates an ActiveRecord class

## Table Name

user\_photo

## Model Class

UserPhoto

```
<?php
namespace app\models;
use Yii;
class UserPhoto extends \yii\db\ActiveRecord
{
 public static function tableName()
 {
 return 'user_photo';
 }

 public function rules()
 {
 return [
 ['user_id', 'photo'], 'required'],
 ['user_id'], 'integer'],
 ['photo'], 'string', 'max' => 255]
];
 }

 public function attributeLabels()
 {
 ...
 }

 public function getUser()
 {
 ...
 }
}
```

Yang perlu kita perhatikan adalah pada fungsi *rules*, dimana kita bisa mendefinisikan validasi *file* unggah menggunakan *class* `[[\yiivalidators\FileValidator]]`. *File validator* akan mengecek ekstensi *file*, *size*, MIMEtype, dll. Materi tentang validasi *file* sudah disinggung pada bab tentang “Bekerja dengan Basis Data”.

Pada contoh ini, kita akan membuat skenario validasi foto sebagai berikut:

- Hanya boleh *file* bertipe *image* dengan ekstensi *file* png dan jpg.
- Ukuran *file* maksimal satu *mega bytes*.

Maka pada fungsi *rules* akan nampak sebagai berikut.

```
public function rules()
{
 return [
 ['user_id', 'photo'], 'required'],
 ['user_id'], 'integer'],
 ['photo'], 'file', 'extensions' => ['png', 'jpg','gif'], 'maxSize' => 1024*1
024];
}
```

## View

Pada *view*, akan nampak sebagai berikut.

`@app/site/upload.php`

```
<?php
use yii\widgets\ActiveForm;
use yii\helpers\Html;
?>
```

```

<h1>Upload Foto</h1>
<?php $form = ActiveForm::begin([
 'options' => ['enctype' => 'multipart/form-data']
]) ?>
<?= $form->field($model, 'photo')->fileInput() ?>
<div class="form-group">
 <?= Html::submitButton('Submit', ['class' => 'btn btn-primary']) ?>
</div>
<?php
ActiveForm::end();

```

### **Controller**

Pada SiteController, kita buat fungsi actionUpload

```

public function actionUpload()
{
 $model = new \app\models\UserPhoto();

 if (\Yii::$app->request->post()) {
 $model->photo = \yii\web\UploadedFile::getInstance($model, 'photo');
 $model->user_id = \Yii::$app->user->id;
 if ($model->validate()) {
 $saveTo = 'uploads/' . $model->photo->baseName . '.' . $model->photo-
>extension;
 if ($model->photo->saveAs($saveTo)) {
 $model->save(false);
 Yii::$app->session->setFlash('success', 'Foto berhasil diupload');
 }
 }
 }

 return $this->render('upload', [
 'model' => $model
]);
}

```

Bagian yang cukup penting pada kode di atas adalah penggunaan *class [[\yii\web\UploadedFile]]* yang berfungsi menangani *file* yang diunggah pengguna, dan otomatis kita bisa mendapatkan nama *file* dan ekstensi *file*-nya melalui properti *baseName* dan *extension*. Kemudian fungsi *saveAs* itu digunakan untuk memindahkan *file* unggah dari *temporary folder* ke direktori yang kita inginkan menggunakan fungsi PHP *move\_uploaded\_file*. Pada contoh tersebut *file* diunggah ke *folder uploads (@app/web/uploads)*.

Kode berikutnya adalah *\$model->save(false)*, mengapa *false*? karena fungsi *save* akan men-triger validasi, dimana salah satu validasinya adalah mengecek apakah *file exists* atau tidak, jika tidak *exists* maka validasi gagal. Pada kasus ini, Yii akan mengecek *file temporary* yang telah di pindah menggunakan fungsi *saveAs*, sehingga akan menghasilkan galat jika tidak kita gunakan *parameter false* pada fungsi *save*.

Mari kita coba

∞ <http://localhost/basic/web/site/upload>

## Upload Foto

**Photo**

No file selected.

Browse file dan tekan tombol submit.

Foto berhasil diupload

## Upload Foto

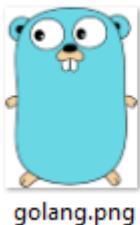
**Photo**

golang.png

Setelah itu kita bisa cek apakah *file* benar-benar telah terunggah pada direktori

@app/web/uploads/

C:\Bitnami\wampstack\apache2\htdocs\basic\web\uploads\



Sukses.

Kita juga bisa menampilkan *file* yang telah berhasil diunggah pada *view upload*. Pada *view upload.php* kita tambahkan kode untuk menampilkan *image*,

```
<?= Html::img(Yii::getAlias('@web') . '/uploads/' . $model->photo, [
 'class'=>'img-thumbnail', 'style'=>'float:right;'
]); ?>
```

Kemudian pada *controller actionUpload* kita perlu sesuaikan lagi.

```
public function actionUpload()
{
 $user_id = \Yii::$app->user->id;
 $model = \app\models\UserPhoto::find()->where([
 'user_id' => $user_id
])->one();

 if (!$model) {
 $model = new \app\models\UserPhoto([
 'user_id' => $user_id
]);
 }

 if (\Yii::$app->request->post()) {
 $model->photo = \yii\web\UploadedFile::getInstance($model, 'photo');
 if ($model->validate()) {
 $saveTo = 'uploads/' . $model->photo->baseName . '.' . $model->photo-
>extension;
 if ($model->photo->saveAs($saveTo)) {
 $model->save(false);
 Yii::$app->session->setFlash('success', 'Foto berhasil diupload');
 }
 }
 }

 return $this->render('upload', [
 'model' => $model
]);
}
```

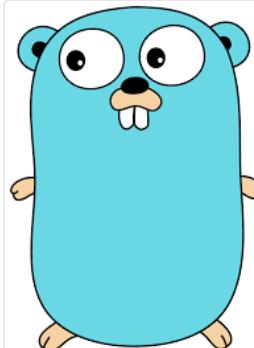
Hasilnya, jika kita akses URL berikut.

∞ <http://localhost/basic/web/site/upload>

## Upload Foto

**Photo**

No file selected.



### A. 4. Mengunggah Banyak File

Ada sedikit perbedaan untuk menangani pengunggahan banyak *file* atau *multiple upload* pada satu *request form*. Sebagai contoh pada kasus *image galery*, kita bisa buat tabel *galery*

v	yii2basic gallery
id	: int(11)
image	: varchar(255)

Lalu kita *generate* modelnya menggunakan Gii. Kemudian kita modifikasi model *Gallery* pada fungsi *rules*

```
public function rules()
{
 return [
 ['image'], 'file', 'extensions'=>['png', 'jpg'],
];
}
```

Pada *view*, kita jadikan bentuk *array* untuk atribut *image*.

@app/views/site/gallery.php

```
<?php
use yii\widgets\ActiveForm;
use yii\helpers\Html;
?>
<h1>Gallery</h1>
<?php $form = ActiveForm::begin([
 'options' => ['enctype' => 'multipart/form-data']
]) ?>
<?= $form->field($model, 'image[]')->fileInput() ?>
<?= $form->field($model, 'image[]')->fileInput() ?>
<?= $form->field($model, 'image[]')->fileInput() ?>
<div class="form-group">
 <?= Html::submitButton('Submit', ['class' => 'btn btn-primary']) ?>
</div>
<?php
ActiveForm::end();
```

Pada kasus di dunia nyata kita bisa gunakan *dynamic file input*, sehingga pengguna bebas menentukan berapa *file* yang akan diunggah pada sekali *submit*.

# Gallery

**Image**

**Browse...** No file selected.

**Image**

**Browse...** No file selected.

**Image**

**Browse...** No file selected.

**Submit**

Di samping menggunakan cara tersebut di atas kita juga bisa menggunakan pendekatan lain. Yaitu menambahkan *parameter multiple* bernilai *true*, sehingga ketika *browse file*, pengguna bisa memilih banyak *file* sekaligus.

Berikut ini contoh kodennya.

```
| <?= $form->field($model, 'image[]')->fileInput(['multiple' => true]) ?>
```

# Gallery

**Image**

**Browse...** 3 files selected.

**Submit**

Kemudian pada SiteController kita buat actionGallery.

```
public function actionGallery()
{
 $model = new \app\models\Gallery();
 if (\Yii::$app->request->post()) {
 $model->image = \yii\web\UploadedFile::getInstances($model, 'image');
 if ($model->validate()) {
 foreach ($model->image as $file) {
 $saveTo = 'uploads/' . $file->baseName . '.' . $file->extension;
 if ($file->saveAs($saveTo)) {
 $model2 = new \app\models\Gallery([
 'image' => $file->baseName . '.' . $file->extension,
]);
 $model2->save(false);
 }
 }
 \Yii::$app->session->setFlash('success', 'Image berhasil diupload');
 }
 }

 return $this->render('gallery', [
 'model' => $model
]);
}
```

Karena atribut *image* dalam bentuk *array*, maka kita harus menggunakan fungsi *getInstances* (sebelumnya *getInstance*) untuk menangkap *multiple file upload* tadi. Kemudian kita gunakan *foreach* untuk memecah *array* dari *file* tersebut, sebelum kemudian dipindahkan dan disimpan *path*-nya.

Jadi ada dua objek yang kita gunakan, objek pertama (*\$model*) untuk menampilkan formulir unggah, dan objek kedua (*\$model2*) untuk proses penyimpanan ke basis data pada setiap iterasi. Bahkan kita perlu tambahkan objek yang ketiga untuk menampilkan *gallery image* di *view*.

```
$model3 = \app\models\Gallery::find()->all();

return $this->render('gallery', [
 'model' => $model,
```

```

 'model3' => $model3,
]);

```

Lalu pada *view gallery.php* kita bisa tambahkan sebagai berikut.

```

foreach($model3 as $file){
 echo Html::img(Yii::getAlias('@web').'/uploads/'.$file->image,[
 'class'=>'img-thumbnail','style'=>'float:left;width:150px;'
]);
}

```

Maka hasilnya sebagai berikut.



## A. 5. Mengamankan File

### Validasi File

Validasi *file* digunakan untuk memastikan *file* yang diunggah oleh pengguna adalah *file* yang valid atau sesuai dengan aturan yang kita terapkan. Aturan yang terkait dengan validasi *file* misalnya pembatasan ekstensi dari *file* dan ukuran maksimal dari *file* yang diunggah. Berikut contoh penerapan aturan tersebut di model.

```

[['photo'], 'file', 'extensions' => 'png, jpg', 'maxSize' => 1024*1024],

```

Apa yang dilakukan Yii dengan definisi aturan ini? Di sisi server, Yii tidak hanya mengecek apakah nama *filenya* berakhiran .png atau .jpg saja namun juga memvalidasi dengan *mimetype*-nya atau *header* dari *file*-nya. Di samping itu Yii juga akan memastikan bahwa ukuran file yang diunggah oleh pengguna tidak lebih dari satu *megabytes*.

Kita bisa juga menambahkan *parameter mimetype*, maka Yii akan men-trigger validasi di sisi *client* untuk *mimetype*-nya.

```

[['photo'], 'file', 'extensions' => 'png, jpg',
'maxSize' => 1024*1024,
'mimeTypes' => 'image/jpeg, image/png',
],

```

---

#### Perhatian:

Validasi *mimetype* untuk mencegah pengguna mengunggah *file* yang *header*-nya berbeda dengan ekstensi *file*-nya.

---

Jika memang yang diunggah adalah *file* bertipe *image*, maka kita bisa gunakan *class* [[yii\validators\ImageValidator]], kelebihannya adalah kita bisa menentukan ukuran dimensi (panjang / lebar) dari *image* yang diunggah oleh pengguna.

```

['photo', 'image', 'extensions' => ['png', 'jpg'],
'mimeTypes' => 'image/jpeg, image/png',
'minWidth' => 100, 'maxWidth' => 1000,
'minHeight' => 100, 'maxHeight' => 1000,
],

```

Pada *view* fungsi *fileInput* kita bisa tambahkan *parameter accept* bernilai *image/\** sehingga ketika pengguna melakukan *browse file*, *file* yang akan muncul hanya yang bertipe *image* saja.

```

<?= $form->field($model, 'image')->fileInput(['accept' => 'image/*']) ?>

```

Pada pengunggahan banyak *file*, kita juga bisa membatasi jumlah maksimal *file*

```

[['image'], 'file', 'extensions' => ['png', 'jpg'],'maxFiles' => 3],

```

*maxFiles* digunakan untuk membatasi jumlah maksimal *file* yang diunggah pada sekali *submit*.

## Lokasi Penyimpanan

Adakalanya kita ingin agar *file* yang diunggah hanya bisa diakses oleh pengguna tertentu saja misalnya pengguna yang telah login. Untuk itu, maka kita tidak boleh mengunggahnya ke direktori yang bisa diakses oleh siapapun (@app/web/). Oleh karena itu kita perlu unggah ke direktori lain, sebagai contoh @app/uploads, maka pada *controller* kita bisa ubah menjadi sebagai berikut.

```
$saveTo = \Yii::getAlias('@app') . '/uploads/' . $model->photo->baseName . '.' . $model->photo->extension;
if ($model->photo->saveAs($saveTo)) {
```

Jika kita unggah pada lokasi diluar web *accessible* atau @app/web maka untuk menampilkannya tidak bisa langsung melalui URL ke *file* tersebut, melainkan harus menggunakan bantuan kode PHP dalam hal ini bisa kita letakkan pada *controllers*.

Kita buat actionDownload pada SiteController

```
public function actionDownload($inline=false) {
 $user_id = \Yii::$app->user->id;
 $model = \app\models\UserPhoto::find()->where([
 'user_id' => $user_id
])->one();
 $path = Yii::getAlias("@app") . '/uploads/';
 $file_download = $path . $model->photo;
 $fake_filename = $model->photo;
 $response = Yii::$app->getResponse();
 $response->sendFile($file_download, $fake_filename, ['inline'=>$inline]);
}
```

Variabel \$fake\_filename bisa kita ganti sesuka kita, tidak harus sama dengan nama *file* aslinya. Fungsi sendFile akan menjalankan fungsi PHP fopen untuk membaca *file* dari direktori yang tidak terjangkau oleh pengguna. Adapun parameter *inline*, jika diset *false* maka akan *force download*, cocok untuk kasus mengunduh *file*.

Pada actionDownload tersebut kita bisa menyaring siapa saja pengguna yang boleh mengunduh *file*.

```
if(\Yii::$app->user->isGuest) {
 throw new \yii\web\NotFoundHttpException('File tidak ditemukan.');
} else{
 $response->sendFile($file_download, $fake_filename, ['inline'=>$inline]);
}
```

Kemudian, pada *view* kita arahkan pemanggilan *image* kepada fungsi actionDownload.

```
<?= Html::img(['download','inline'=>true], [
 'class'=>'img-thumbnail','style'=>'float:right;'
]); ?>
```

## B. Mengimpor Data

Mengimpor data yang dimaksudkan di sini adalah memasukkan data teks yang berasal dari *file* berformat *spreadsheet/tabel* (xls/xlsx/ods) ke tabel di basis data. Adakalanya pengguna telah memiliki data berformat *spreadsheets*, sehingga jika pada aplikasi tersedia fitur untuk mengimpor data tentu akan memudahkan pengguna untuk memasukkan data.

Oleh karena itu, kita membutuhkan *library* yang *powerfull* untuk membaca *file* berformat *spreadsheet*. Pilihan penulis jatuh pada php-excel (<https://phpexcel.codeplex.com>), sebuah *library* yang cukup populer untuk menangani *spreadsheet* menggunakan PHP.

Ada sebuah *extension* untuk Yii yang telah membungkus *library* tersebut yaitu hscstudio/yii2-export (<https://github.com/hscstudio/yii2-export>).

```
| require --prefer-dist hscstudio/yii2-export "1.0.0"
```

Sebagai contoh kasus, kita akan membuat fitur impor data pada CRUD *category* atau lebih tepatnya tabel *category* yang telah kita buat pada panduan sebelumnya.

Maka yang perlu kita siapkan adalah :

1. *File spreadsheet* berisi data *category*
2. Sebuah fungsi *action* dan *view* untuk menampilkan formulir unggah *file excel*.
3. Sebuah fungsi *action* untuk melakukan proses impor data.

Pertama, mari kita mulai dengan mempersiapkan *file spreadsheet* yang berisi data *category*.

A	B
<b>DATA CATEGORY</b>	
title	description
Pemrograman Web	tentang pemrograman web
Desain Grafis	tentang desain grafis
Aplikasi Perkantoran	tentang aplikasi perkantoran
<b>6 Pemrograman Mobile</b>	<b>tentang pemrograman mobile</b>
Jaringan Komputer	tentang jaringan komputer
Sistem Informasi	tentang sistem informasi

Pada contoh *template* di atas, artinya pembacaan *row* data dimulai dari baris ketiga.

Langkah kedua, kita buat fungsi *action* untuk menampilkan formulir unggah. Pada contoh ini kita buat fungsi *actionImport* pada *CategoryController*.

```
public function actionImport()
{
 $modelImport = new \yii\base\DynamicModel([
 'fileImport' => 'File Import',
]);
 $modelImport->addRule(['fileImport'], 'required');
 $modelImport-
 >addRule(['fileImport'], 'file', ['extensions'=>'ods,xls,xlsx'], ['maxSize'=>1024*1024]);
}

return $this->render('import', [
 'modelImport' => $modelImport,
]);
}
```

di sini kita gunakan *DynamicModel* yang hanya berisi satu atribut yang diberikan validasi *file* dengan ekstensi nama *file* yang diperbolehkan hanya ods, xls, dan xlsx serta ukuran maksimal *filenya* 2MB.

Karena di-*render* ke *import*, maka kita harus buat *view* @app/views/category/import.php yang berisi formulir impor data.

```
<?php
use yii\widgets\ActiveForm;
use yii\helpers\Html;
?>
<h1>Import Data</h1>
<?php
$form = ActiveForm::begin([
 'options' => ['enctype'=> 'multipart/form-data'],
]) ?>
<?= $form->field($modelImport, 'fileImport')->fileInput() ?>
<?= Html::submitButton('Import', ['class'=>'btn btn-primary']) ?>
<?php ActiveForm::end() ?>
```

Langkah ketiga adalah memproses *file spreadsheet* yang diunggah dan mengimpornya ke basis data. Mari kita modifikasi *actionImport*

```
public function actionImport()
{
 $modelImport = new \yii\base\DynamicModel([
 'fileImport' => 'File Import',
]);
 $modelImport->addRule(['fileImport'], 'required');
 $modelImport-
 >addRule(['fileImport'], 'file', ['extensions'=>'xls,xlsx'], ['maxSize'=>1024*1024]);

 if (Yii::$app->request->post()) {
 $modelImport-
 >fileImport = \yii\web\UploadedFile::getInstance($modelImport, 'fileImport');
 if ($modelImport->fileImport && $modelImport->validate()) {
 $inputFileType = \PHPEExcel_IOFactory::identify($modelImport->fileImport-
 >tempName);
 $objReader = \PHPEExcel_IOFactory::createReader($inputFileType);
 $objPHPExcel = $objReader->load($modelImport->fileImport->tempName);
 $sheetData = $objPHPExcel->getActiveSheet()-
```

```

>toArray(null,true,true,true);
 $baseRow = 3;
 while (!empty($sheetData[$baseRow] ['A'])) {
 $model = new Category();
 $model->title = (string)$sheetData[$baseRow] ['A'];
 $model->description = (string)$sheetData[$baseRow] ['B'];
 $model->save();
 $baseRow++;
 }
 Yii::$app->getSession()->setFlash('success', 'Success');
}
else{
 Yii::$app->getSession()->setFlash('error', 'Error');
}
}

return $this->render('import', [
 'modelImport' => $modelImport,
]);
}

```

Dengan kita melakukan instalasi *extension* yii2-export, maka *library* PHPExcel secara otomatis langsung bisa kita gunakan. Intinya adalah, objek PHPExcel membaca *file spreadsheet* yang diunggah oleh pengguna, kemudian mengekstraknya menjadi *array*, sehingga kita bisa gunakan perintah while untuk mengiterasi baris *record* sesuai dengan keinginan kita.

Pada contoh ini dimulai dari baris ketiga yang ditandai dengan variabel \$baseRow, serta kolom yang dibaca adalah kolom A untuk *title* dan kolom B untuk *description*.

Mari kita uji coba dengan mengunggah *file spreadsheet* yang telah kita isi dengan data *category* seperti contoh di atas.

∞ <http://localhost/basic/web/category/import>

Success

## Import Data

### File Import

[Browse...](#) No file selected.

[Import](#)

Cek basis data tabel *category*

27	Pemrograman Web	tentang pemrograman web	NULL
28	Desain Grafis	tentang desain grafis	NULL
29	Aplikasi Perkantoran	tentang aplikasi perkantoran	NULL
30	Pemrograman Mobile	tentang pemrograman mobile	NULL
31	Jaringan Komputer	tentang jaringan komputer	NULL

Atau cek actionIndex <http://localhost/basic/web/category>

Showing 1-20 of 26 items.			
#	ID 	Title	Description
1	32	Sistem Informasi	tentang sistem informasi
2	31	Jaringan Komputer	tentang jaringan komputer
3	30	Pemrograman Mobile	tentang pemrograman mobile
4	29	Aplikasi Perkantoran	tentang aplikasi perkantoran

## C. Pelaporan

Fitur pelaporan yang dimaksud di sini adalah fitur untuk menggenerate data dari basis data menjadi format dokumen misalnya *spreadsheet*, PDF, Word, dsb. Penerapan fitur ini dalam dunia nyata memiliki beragam jenis, antara lain: cetak data pendukung, pelaporan, cetak bukti pembayaran, dsb.

Pada panduan ini kita masih akan menggunakan *extension* yii2-export dimana *extension* ini mendukung banyak format dokumen yaitu pdf, docx, doc, odt, odf, xls, xlsx, ppt, pptx. Ada tiga teknologi dibalik *extension* ini sehingga mendukung banyak format *file* yaitu TinyButStrong, PHPExcel, dan MPDF.

Sebagai contoh kasus, kita akan menggenerate data *category* ke dalam format Excel, Word, dan PDF.

### C. 1. Eksport ke Excel

Untuk menggenerate dokumen ke format *Excel*, kita bisa menggunakan PHPExcel dan OpenTBS. Berikut ini penjabarannya.

#### PHPExcel

Pertama kita buat dulu tombol di *view* (@app/views/category/index.php) yang berfungsi sebagai *trigger* untuk menjalankan fungsi *generate Excel*. Letakkan di bawah kode *Gridview*.

```
<!-- EXAMPLE BUTTON EXPORT PHPEXCEL -->
<?= Html::a('Export Excel', ['export-excel'], ['class'=>'btn btn-info']); ?>
```

Sehingga tampilannya akan sebagai berikut.

Langkah kedua, kita buat *file Excel* sebagai *template*. Kita letakkan *file* tersebut di @app/views/category/\_excel.xlsx

A	B	C
DATA CATEGORY		
id	title	description
1		
2		
3		
4		
5		

Kemudian pada CategoryController, kita buat actionExportExcel yang berfungsi menggenerate data ke format *Excel* dengan menggunakan *library* PHPExcel

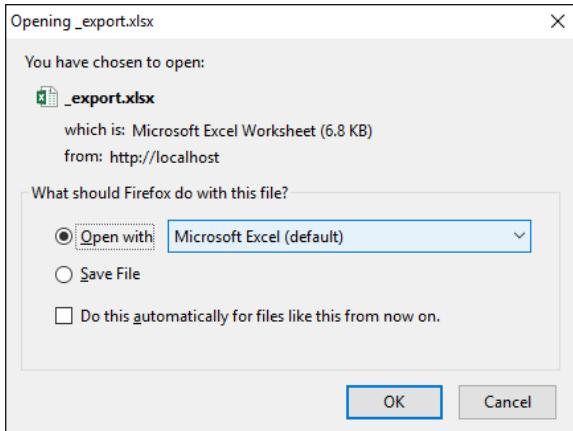
```
public function actionExportExcel()
{
 $searchModel = new CategorySearch();
 $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
 $objReader = \PHPExcel_IOFactory::createReader('Excel2007');
 // set template
 $template = Yii::getAlias('@app/views/category') . '/_export.xlsx';
 $objPHPExcel = $objReader->load($template);
 $activeSheet = $objPHPExcel->getActiveSheet();
 // set orientasi & ukuran kertas
 $activeSheet->getPageSetup()->setOrientation(\PHPExcel_Worksheet_PageSetup::ORIENTATION_LANDSCAPE)
 ->setPaperSize(\PHPExcel_Worksheet_PageSetup::PAPERSIZE_FOLIO);
 $baseRow=3;
 foreach($dataProvider->getModels() as $category){
 $activeSheet->setCellValue('A' . $baseRow, $baseRow-2)
 ->setCellValue('B' . $baseRow, $category->title)
 ->setCellValue('C' . $baseRow, $category->description);
 $baseRow++;
 }
 header('Content-Type: application/vnd.openxmlformats-
```

```

 officedocument.spreadsheetml.sheet');
 header('Content-Disposition: attachment;filename=_export.xlsx');
 header('Cache-Control: max-age=0');
 $objWriter = \PHPExcel\IOFactory::createWriter($objPHPExcel, "Excel2007");
 $objWriter->save('php://output');
 exit;
 }
}

```

Mari kita lihat hasilnya, dengan menekan tombol *Export Excel*



Berikut ini data di *Excel*.

A	B	C	
DATA CATEGORY			
1	id	title	description
3	1	Buku Tulis2	Buku buat tulis menulis
4	2	Buku Bacaan	Buku komik, cerita dll
5	3	Alat Tulis	Pencil, Bulpoin

### OpenTBS

Hampir sama dengan langkah-langkah meng-*generate* data menggunakan PHPExcel, pada OpenTBS kita bisa buat tombol lain untuk membedakan dengan sebelumnya.

```

<!-- EXAMPLE BUTTON EXPORT OPENTBS -->
<?= Html::a('Export Excel2', ['export-excel2'], ['class'=>'btn btn-info']); ?>

```

Kemudian kita buat template yang hampir sama dengan sebelumnya, namun ditambahkan “*formula*”, sebagai berikut.

```
@app/views/category/_export2.xlsx
```

A	B	C	
DATA CATEGORY			
1	id	title	description
3	[data.no;noerr;block=tbs:row]	[data.title;noerr]	[data.description;noerr]

Keterangan.

- data merupakan *variabel* yang nanti akan dikirimkan dari PHP.
- data.no, data.title dan data.description adalah properti yang dimiliki *variabel* data (yang sebagianya merupakan atribut dari tabel *category*)
- noerr untuk mencegah tampilnya galat yang mungkin terjadi
- tbs:row menunjukkan baris baru

Kemudian pada CategoryController, kita buat actionExportExcel2 yang berfungsi meng-*generate* data ke format *Excel* dengan menggunakan *library* OpenTBS

```

public function actionExportExcel2()
{
 $searchModel = new CategorySearch();
 $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
 // Initialize the TBS instance
 $OpenTBS = new \hscstudio\export\OpenTBS; // new instance of TBS
 // set template
}

```

```

$template = Yii::getAlias('@app/views/category').'_export2.xlsx';
// Also merge some [onload] automatic fields (depends of the type of document).
$OpenTBS->LoadTemplate($template);
// $OpenTBS->VarRef['modelName'] = "Category";
$data = [];
$no=1;
foreach($dataProvider->getModels() as $category) {
 $data[] = [
 'no'=>$no++,
 'title'=>$category->title,
 'description'=>$category->description,
];
}

$OpenTBS->MergeBlock('data', $data);
// Output the result as a file on the server. You can change output file
$OpenTBS-
>Show(OPENTBS_DOWNLOAD, '_export.xlsx'); // Also merges all [onshow] automatic fields
.
exit;
}

```

Mari kita lihat hasilnya, dengan menekan tombol *Export Excel* yang kedua

A	B	C
DATA CATEGORY		
id	title	description
1	Buku Tulis2	Buku buat tulis menulis
2	Buku Bacaan	Buku komik, cerita dll
3	Alat Tulis	Pencil, Bulpoin

Berhasil.

## C. 2. Eksport ke Word

Untuk ekspor data ke dalam format *word* atau odt/doc/docx kita bisa menggunakan *library* OpenTBS. Caranya kurang lebih sama dengan *generate* data ke format *Excel*.

Bedanya hanya di *template*-nya saja, yaitu *template* dibuat dengan Ms Word, anda juga bisa membuatnya dengan menggunakan LibreOffice.



## Data Category

id	title	description
[data.no;noerr;block=tbs:row]	[data.title;noerr]	[data.description;noerr]

Simpan ke dalam file @app/views/category/\_export.docx

Jangan lupa pada *view* kita buat tombol *Export Word*

```
<?= Html::a('Export Word', ['export-word'], ['class'=>'btn btn-warning']); ?>
```

Kemudian kita pada *controller* kita buat actionExportWord

```

public function actionExportWord()
{
 $searchModel = new CategorySearch();
 $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
 // Initialize the TBS instance
 $OpenTBS = new \hscstudio\export\OpenTBS; // new instance of TBS
 // set template
 $template = Yii::getAlias('@app/views/category').'_export.docx';
 // Also merge some [onload] automatic fields (depends of the type of document).
 $OpenTBS->LoadTemplate($template);
 // $OpenTBS->VarRef['modelName'] = "Category";
 $data = [];
 $no=1;
 foreach($dataProvider->getModels() as $category) {

```

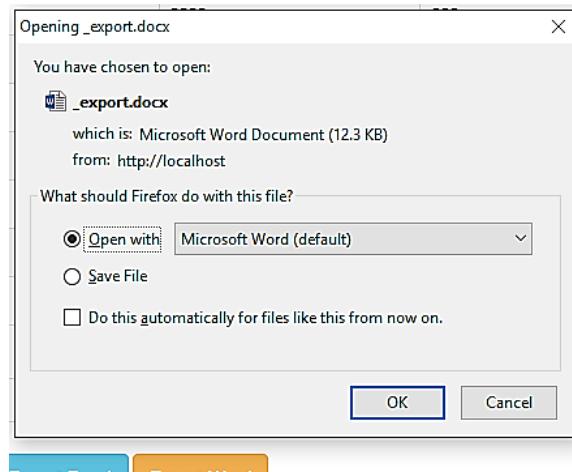
```

 $data[] = [
 'no'=>$no++,
 'title'=>$category->title,
 'description'=>$category->description,
];
 }

 $OpenTBS->MergeBlock('data', $data);
 // Output the result as a file on the server. You can change output file
 $OpenTBS-
>Show(OPENTBS_DOWNLOAD, '_export.docx'); // Also merges all [onshow] automatic fields
.
 exit;
}

```

Mari kita *test*



**Export Excel** **Export Word**

Hasilnya

## Data Category

<b>id</b>	<b>title</b>	<b>description</b>
1	Buku Tulis2	Buku buat tulis menulis
2	Buku Bacaan	Buku komik, cerita dll
3	Alat Tulis	Pencil, Bulpoin

Selesai.

### C. 3. Eksport ke PDF

Untuk meng-*generate* data ke format PDF, kita akan menggunakan *library* mPDF. *Library* ini juga telah terinstalasi ketika kita melakukan instalasi yii2-export.

Pertama, pada *controller*, kita buat fungsi *actionExportPdf*

```

public function actionExportPdf()
{
 $searchModel = new CategorySearch();
 $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
 $html = $this->renderPartial('_export',['dataProvider'=>$dataProvider]);
 $mpdf=new \mPDF('c','A4','','',0,0,0,0,0,0);
 $mpdf->SetDisplayMode('fullpage');
 $mpdf-
>list_indent_first_level = 0; // 1 or 0 - whether to indent the first level of a lis
t
 $mpdf->WriteHTML($html);
 $mpdf->Output();
 exit;
}

```

Kemudian sama dengan cara sebelumnya yaitu membuat *template* dokumen. Namun pada PDF *template*-nya berupa kode PHP dan HTML.

@app/views/category/\_export.php

```
<!DOCTYPE html>
<html>
<head>
 <title>Data Category</title>
 <style type="text/css">
 .page{padding:2cm;}
 table{border-spacing:0; border-collapse: collapse; width:100%;}
 table td, table th{border: 1px solid #ccc;}
 </style>
</head>
<body>
 <div class="page">
 <h1>Data Category</h1>
 <table border="0">
 <tr>
 <th>No</th>
 <th>Title</th>
 <th>Description</th>
 </tr>
 <?php
 $no = 1;
 foreach($dataProvider->getModels() as $category) {
 ?>
 <tr>
 <td><?= $no++ ?></td>
 <td><?= $category->title ?></td>
 <td><?= $category->description ?></td>
 </tr>
 <?php
 }
 ?>
 </table>
 </div>
</body>
</html>
```

Kodenya mirip dengan view, hanya saja *full* HTML, kita juga bisa menerapkan CSS, namun perlu dicoba-coba untuk menampilkan hasil terbaik. Kita juga bisa menyisipkan *image* dsb.

Jangan lupa tambahkan tombol *Export PDF*

```
<!-- EXAMPLE BUTTON EXPORT MPDF -->
<?= Html::a('Export PDF', ['export-pdf'], ['class'=>'btn btn-success']); ?>
```

Berikut ini contoh hasil uji coba *generate* data ke format PDF



## Data Category

No	Title	Description
1	Buku Tulis2	Buku buat tulis menulis
2	Buku Bacaan	Buku komik, cerita dll
3	Alat Tulis	Pencil, Bulpoin

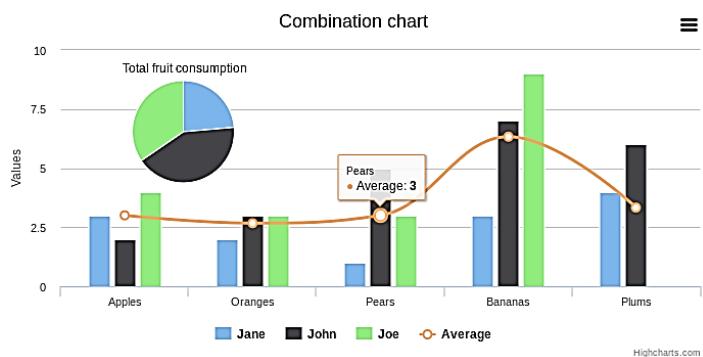
Silakan mencoba.

## D. Visualisasi Data

Visualisasi data yang dimaksud di sini adalah menampilkan data dalam bentuk grafik atau *chart*. Ada banyak *library* yang bisa digunakan untuk membuat tampilan grafik, salah satunya yang cukup populer adalah Highchart (<http://www.highcharts.com/>). *Library* ini berbayar untuk tujuan proyek aplikasi *commercial*.

Highchart banyak digunakan oleh perusahaan besar, semisal Facebook, Twitter, Yahoo, dsb.

Salah satu *extension* Yii yang mem-wrap *library* ini dan cukup populer adalah yii2-highcharts (<https://github.com/miloschuman/yii2-highcharts>). Tidak hanya Highchart, kita juga bisa mengintegrasikan *library* lain yang berafiliasi dengan Highchart yaitu Highstock dan Highmaps.



## D. 1. Implementasi Dasar

Mari kita menginstalnya menggunakan *Composer*

```
| composer require --prefer-dist miloschuman/yii2-highcharts-widget "*"
```

Cara menggunakan cukup *simple*, sebagaimana yang kita dapat di panduannya yaitu cukup pada *view* saja, karena *extension* ini berjenis *widget*.

```
use miloschuman\highcharts\Highcharts;

echo Highcharts::widget([
 'options' => [
 'title' => ['text' => 'Fruit Consumption'],
 'xAxis' => [
 'categories' => ['Apples', 'Bananas', 'Oranges']
],
 'yAxis' => [
 'title' => ['text' => 'Fruit eaten']
],
 'series' => [
 ['name' => 'Jane', 'data' => [1, 0, 4]],
 ['name' => 'John', 'data' => [5, 7, 3]]
]
]
]);
```

Alternatif lain, kita bisa menuliskannya dengan format *JSON*.

```
echo Highcharts::widget([
 'options'=>'{
 "title": { "text": "Fruit Consumption" },
 "xAxis": {
 "categories": ["Apples", "Bananas", "Oranges"]
 },
 "yAxis": {
 "title": { "text": "Fruit eaten" }
 },
 "series": [
 { "name": "Jane", "data": [1, 0, 4] },
 { "name": "John", "data": [5, 7, 3] }
]
 }'
]);
```

Mari kita coba terapkan contoh di atas pada aplikasi kita, dengan membuat sebuah fungsi *action* pada *controller*, sebagai contoh SiteController.

```
public function actionHighchart()
{
 return $this->render('highchart');
```

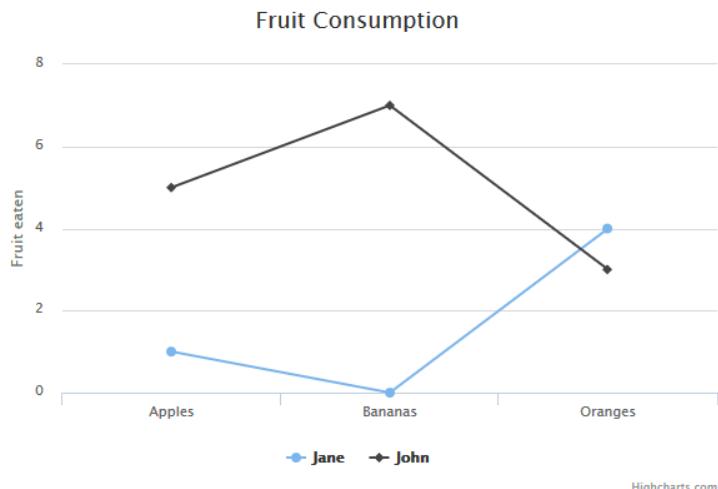
Kemudian kita buat *view* @app/views/site/highchart.php sesuai contoh di atas.

```
<?php
use miloschuman\highcharts\Highcharts;
echo "<h1>Sample Highchart</h1>";
echo Highcharts::widget([
 'options' => [
 'title' => ['text' => 'Fruit Consumption'],
 'xAxis' => [
 'categories' => ['Apples', 'Bananas', 'Oranges']
],
 'yAxis' => [
 'title' => ['text' => 'Fruit eaten']
],
 'series' => [
 ['name' => 'Jane', 'data' => [1, 0, 4]],
 ['name' => 'John', 'data' => [5, 7, 3]]
]
]
]);
```

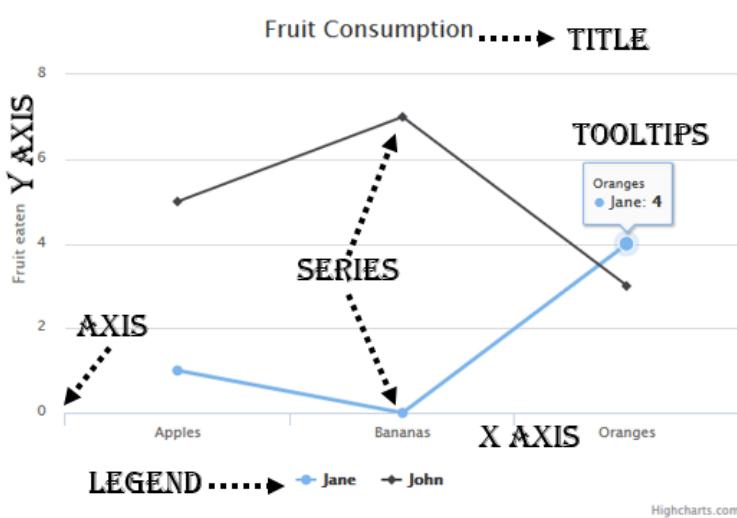
Mari kita lihat hasilnya.

∞ <http://localhost/basic/web/site/highchart>

## Sample Highchart



Berikut ini penjelasan dari *chart* di atas



### Title

Adalah judul *chart*, di samping itu ada juga *subtitle* atau sub judul

```
'options' => [
 'title' => ['text' => 'Judul Chart'],
 'subtitle' => ['text' => 'Sub Judul Chart'],
```

```
[...]
```

## Axis

Adalah informasi tentang sumbu x dan sumbu y atau tergantung tipe dari *chart* yang digunakan, misalnya tipe *pie* tidak memiliki *axis*.

*xAxis* dan *yAxis* masing-masing bisa kita set judulnya

```
'yAxis' => [
 'title' => ['text' => 'Fruit eaten']
],
```

Kita juga bisa mengesetnya menjadi kategori dari data *series*.

```
'xAxis' => [
 'categories' => ['Apples', 'Bananas', 'Oranges']
],
```

Hendaknya kita menyesuaikan dengan jumlah data *series*, sehingga tidak ada yang kosong. Misal jika kita hanya *set* dua kategori yaitu *Apples* dan *Bananas* maka data *series* ke tiga akan berisi *index* (dimulai dari 0)



Kita juga bisa *set type*-nya menjadi kategori sehingga akan menyesuaikan dengan *name series*-nya.

```
'xAxis' => [
 'type' => 'category'
],
```

Selengkapnya <http://www.highcharts.com/docs/chart-concepts/axes>

## Tooltips

Adalah *popup* yang menampilkan data *chart* ketika *mouse* di atas *series*. Kita bisa melakukan *setting* sebagai berikut.

```
echo Highcharts::widget([
 'options' => [
 ...
 'series' => [
 ...
],
 'tooltip' => [
 'backgroundColor' => '#FCFFC5',
 'borderColor' => 'black',
 'borderRadius' => 10,
 'borderWidth' => 3
],
],
]);
```



Selengkapnya <http://www.highcharts.com/docs/chart-concepts/tooltip>

## Legend

Adalah *group* dari data *series* yaitu parameter *name* pada *series*. Secara *default* *legend* akan tampil pada *chart* namun kita bisa menyembunyikannya dengan cara berikut.

```
'legend' => [
 'enabled' => false,
],
```

## Series

Adalah data yang akan ditampilkan pada *chart*. Series mempunyai dua *parameter* dasar yaitu *name* dan *data*

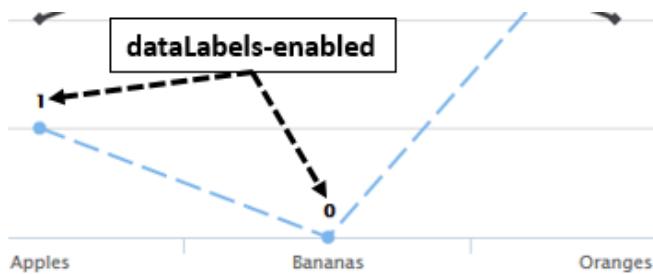
```
series: [
 name: ''
 data: []
]
```

Pada *series* kita bisa mengeset warna, lebar, dan *style chart*.

```
'series' => [
 ['name' => 'Jane', 'data' => [1, 0, 4], 'dashStyle' => 'longdash'],
 ['name' => 'John', 'data' => [5, 7, 3], 'lineWidth' => 3],
 ['name' => 'Mary', 'data' => [2, 4, 1], 'color' => '#00FF00'],
],
```

*dashStyle : dash, dot, longdash*

Kita bisa menampilkan data *series* melalui *plotOptions - line - dataLabels - enabled - true*



```
echo Highcharts::widget([
 'options' => [
 ...
 'plotOptions' => [
 'line' => [
 'dataLabels' => [
 'enabled' => true,
],
],
],
],
 ...
])
```

Sesuaikan *line* dengan jenis *chart*-nya, misal *column*.

Selengkapnya <http://www.highcharts.com/docs/chart-concepts/series>

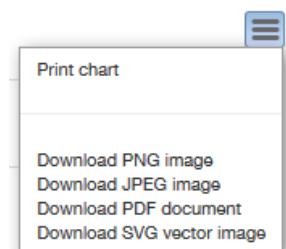
## Export

Kita bisa menampilkan tombol *export*, sehingga *chart* bisa di *export* ke berbagai format *file* yaitu dengan menambahkan *module exporting*.

```
echo Highcharts::widget([
 'scripts' => [
 'modules/exporting',
],
 'options' => [

```

Tombol *export* berada pada bagian kanan atas dari *chart*



## Chart Type

Ada berbagai tipe *chart* yang didukung oleh Highchart, antara lain: *line (default)*, *spline*, *area*, *areaspline*, *column*, *bar*, *pie*, *scatter*, *gauge*, *arearange*, *areasplinerange* dan *columnrange*

Cara mengeset tipe *chart*, sebagai berikut.

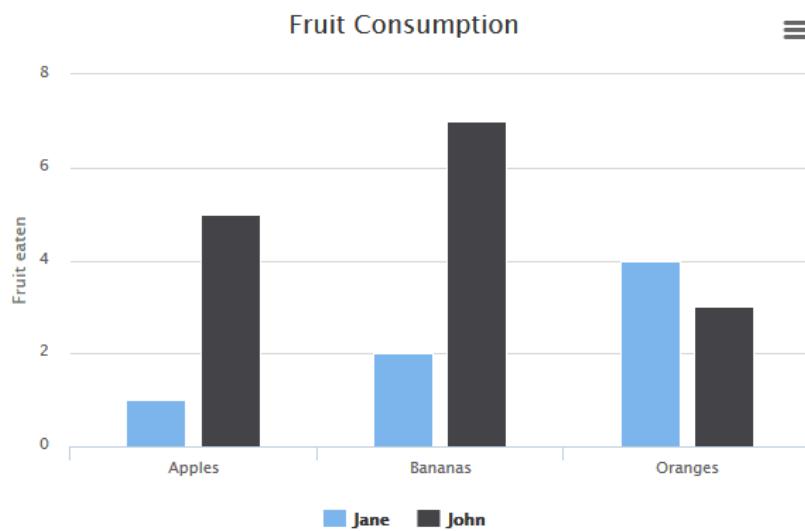
```
echo Highcharts::widget([
 'options' => [
 'chart' => [
 'type' => 'column',
],
 'series' => [

```

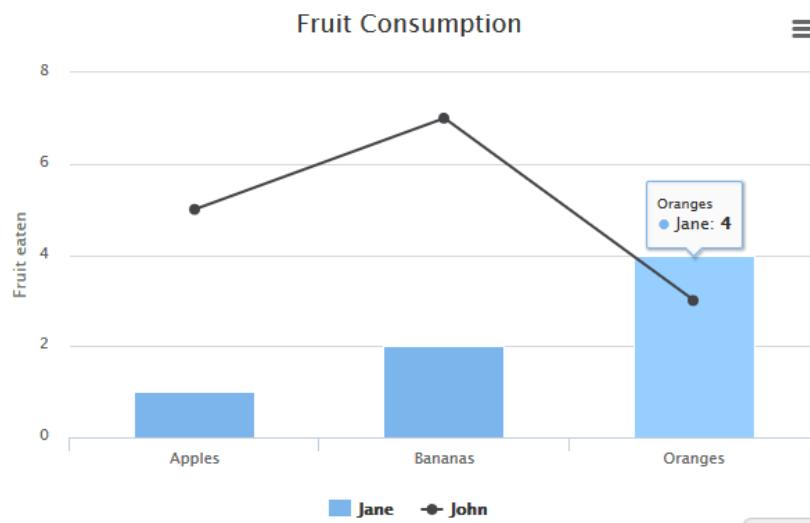
Di samping itu kita juga bisa mengkombinasikan tipe-tipe *chart* tersebut.

```
'series' => [
 ['type' => 'column', 'name' => 'Jane', 'data' => [1, 2, 4]],
 ['type' => 'line', 'name' => 'John', 'data' => [5, 7, 3]]
]
```

Berikut ini *chart* bertipe *column*

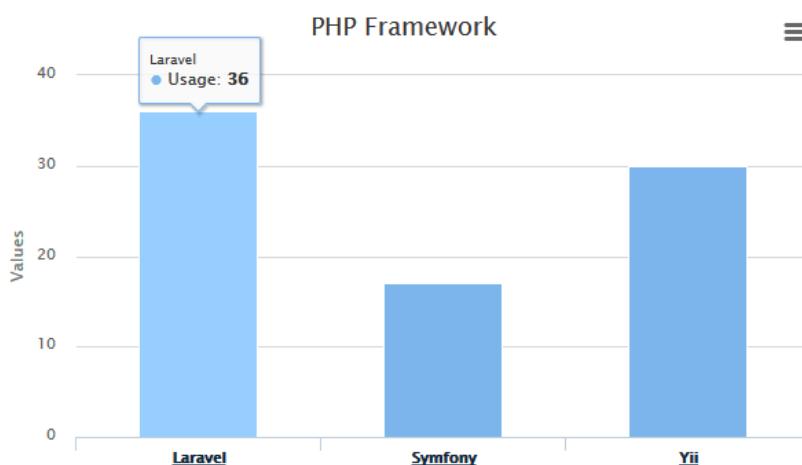


Berikut ini *chart* kombinasi

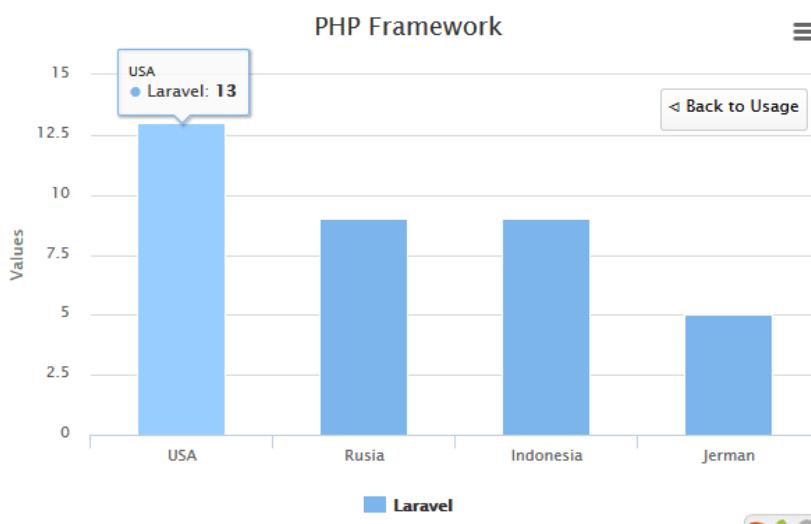


## Drilldown

Highchart juga mendukung sub *chart* atau *drilldown* yang merupakan detail dari sebuah *chart*. Sebagai contoh kasus, kita ingin menampilkan data penggunaan *framework* PHP, maka pada *chart* utama ditampilkan nama-nama *framework*, sebagai berikut.



Kemudian pada masing-masing *framework*, kita ingin agar pengguna bisa melihat detailnya, misalnya dari negara mana saja penggunanya.



Untuk membuat seperti ini, kita perlu menggunakan *modules drilldown*.

```
echo Highcharts::widget([
 'scripts' => [
 'modules/drilldown',
 'modules/exporting',
],
 'options' => [
 'chart' => [
 'type' => 'column',
],
 'title' => ['text' => 'PHP Framework'],
 'xAxis' => [
 'type' => 'category'
],
 'series' => [
 ['name' => 'Usage', 'data' => [
 ['name' => 'Laravel', 'y' => 36, 'drilldown' => 'laravel'],
 ['name' => 'Symfony', 'y' => 17, 'drilldown' => 'symfony'],
 ['name' => 'Yii', 'y' => 30, 'drilldown' => 'yii'],
]],
],
 'drilldown' => [
 'series' => [
 ['id' => 'laravel', 'name' => 'Laravel', 'data' => [['USA', 13], ['Rusia', 9], ['Indonesia', 9], ['Jerman', 5]]],
 ['id' => 'symfony', 'name' => 'Symfony', 'data' => [['USA', 5], ['Rusia', 6], ['Indonesia', 3], ['Jerman', 3]]],
 ['id' => 'yii', 'name' => 'Yii', 'data' => [['USA', 3], ['Rusia', 15], ['Indonesia', 6], ['Jerman', 6]]],
],
],
],
]);
```

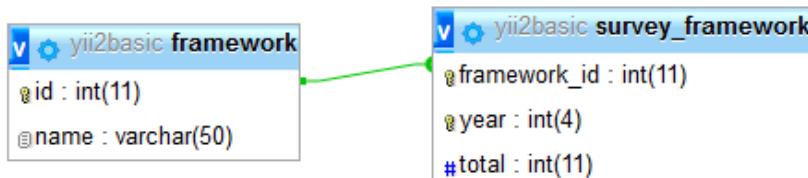
```
[],
],
]);
```

## D. 2. Dynamic Data

### Persiapan Data

Sebagai contoh kasus, kita ingin menampilkan data hasil *survey* penggunaan *framework*, dimana ada dua tabel yaitu table *framework* yang berisi jenis *framework*, dan tabel *survey\_framework* yang berisi data hasil *survey framework* beserta tahun *survey*-nya.

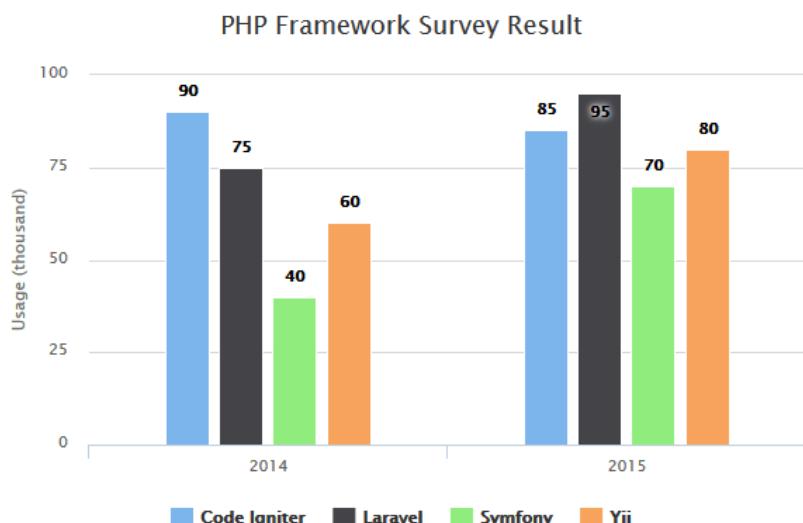
Berikut ini strukturnya.



Kemudian kita isi dengan beberapa data *dummy*, untuk uji coba ini.

framework		survey_framework		
id	name	framework_id	year	total
1	Code Igniter	1	2014	90
2	Laravel	2	2014	75
3	Symfony	3	2014	40
4	Yii	4	2014	60
		1	2015	85
		2	2015	95
		3	2015	70
		4	2015	80

Target tampilan kita adalah sebagai berikut.



Jika data yang digunakan tidak dinamis, kodenya cukup di *view* sebagai berikut.

```

echo Highcharts::widget([
 'options' => [
 'chart' => ['type' => 'column'],
 'title' => ['text' => 'PHP Framework Survey Result'],
 'xAxis' => [
 'categories' => ['2014', '2015']
],
 'yAxis' => [
 'title' => ['text' => 'Usage (thousand)']
]
]
]);
```

```

],
 'series' => [
 ['name' => 'Code Igniter', 'data' => [90,85]],
 ['name' => 'Laravel', 'data' => [75,95]],
 ['name' => 'Symfony', 'data' => [40,70]],
 ['name' => 'Yii', 'data' => [60,80]],
],
 'plotOptions' => [
 'column' => [
 'dataLabels' => [
 'enabled' => true,
],
],
],
],
],
);

```

Dari kode di atas bisa kita tarik kesimpulan bahwa data yang akan dinamis ada dua yaitu pada `xAxis - categories` dan `series`.

### **Query Data**

Untuk mendapatkan data `xAxis - categories` yang berupa daftar tahun `survey`,

```
| ['2014', '2015']
```

kita bisa gunakan `query distinct` dan fungsi `queryColumn()`

```

$db = \Yii::$app->db;
$years = $db->createCommand(
 'SELECT DISTINCT(year) FROM survey_framework
 ORDER BY year ASC')
->queryColumn();
print_r($years);

```

Hasil keluaran dari perintah di atas adalah Array ( [0] => 2014 [1] => 2015 )

Kemudian untuk mendapatkan data `series` yang berupa data `survey`,

```

[, [
 ['name' => 'Code Igniter', 'data' => [90,85]],
 ['name' => 'Laravel', 'data' => [75,95]],
 ['name' => 'Symfony', 'data' => [40,70]],
 ['name' => 'Yii', 'data' => [60,80]],
]
]
```

maka kita bisa gunakan cara kombinasi berikut.

```

$db = \Yii::$app->db;
$frameworks = $db->createCommand(
 'SELECT * FROM framework
 ORDER BY id ASC')
->queryAll();
$series = [];
foreach($frameworks as $framework){
 $results = $db->createCommand(
 'SELECT total FROM survey_framework
 WHERE framework_id='.$framework['id'].'.
 ORDER BY year ASC')
->queryColumn();
 $data = array_map('intval', $results);
 $series[] = [
 'name' => $framework['name'],
 'data' => $data,
];
}
print_r($series);

```

Kode `$data = array_map('intval', $results);` adalah untuk mengkonversi dan memastikan hasil `result`-nya ke dalam tipe `integer`, karena data `series` harus bertipe `integer`.

Berikut ini hasil keluarannya

```
| Array ([0] => Array ([name] => Code Igniter [data] => Array ([0] => 90 [1] => 85)
) [1] => Array ([name] => Laravel [data] => Array ([0] => 75 [1] => 95)) [2] =>
```

```
| Array ([name] => Symfony [data] => Array ([0] => 40 [1] => 70)) [3] => Array (
| [name] => Yii [data] => Array ([0] => 60 [1] => 80)))
```

## Implementasi

Setelah mendapatkan data dari basis data yang sesuai dengan format highchart, maka sekarang bisa kita implementasikan. Mari kita buat fungsi baru pada SiteController yaitu actionDynamicChart

```
public function actionDynamicChart()
{
 $db = \Yii::$app->db;
 $years = $db->createCommand('
 SELECT DISTINCT(year) FROM survey_framework
 ORDER BY year ASC')
 ->queryColumn();

 $frameworks = $db->createCommand('
 SELECT * FROM framework
 ORDER BY id ASC')
 ->queryAll();
 $series = [];
 foreach($frameworks as $framework){
 $results = $db->createCommand('
 SELECT total FROM survey_framework
 WHERE framework_id='.$framework['id'].'.
 ORDER BY year ASC')
 ->queryColumn();
 $data = array_map('intval', $results);
 $series[] = [
 'name' => $framework['name'],
 'data' => $data,
];
 }
}

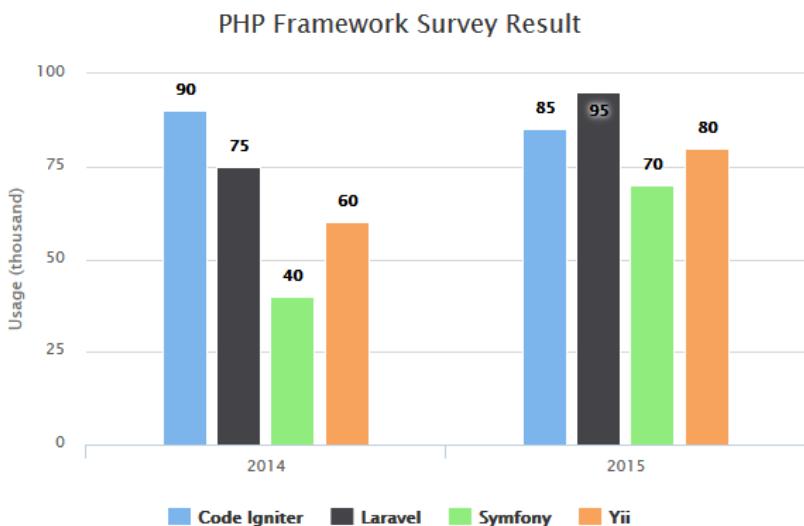
return $this->render('dynamic-chart', [
 'years' => $years,
 'series' => $series,
]);
}
```

Kemudian kita buat view @app/views/site/dynamic-chart.php

```
<?php
use miloschuman\highcharts\Highcharts;
echo "<h1>Dynamic Chart</h1>";
echo Highcharts::widget([
 'options' => [
 'chart' => ['type' => 'column'],
 'title' => ['text' => 'PHP Framework Survey Result'],
 'xAxis' => [
 'categories' => $years
],
 'yAxis' => [
 'title' => ['text' => 'Usage (thousand)']
],
 'series' => $series,
 'plotOptions' => [
 'column' => [
 'dataLabels' => [
 'enabled' => true,
],
],
],
],
]);
});
```

Mari kita lihat hasilnya.

∞ <http://localhost/basic/web/site/dynamic-chart>



Selesai.

Cara lain kita bisa memanfaatkan parameter *callback* sebagaimana yang dicontohkan pada artikel ini.

∞ <https://github.com/miloschuman/yii2-highcharts/blob/master/doc/examples/highstock.md>

## E. Kesimpulan

Metode unggah *file* di Yii pada dasarnya sama dengan metode unggah *file* dengan menggunakan PHP biasa, hanya saja Yii telah membuatkan kita sebuah *class* yang bertugas menangani *file* yang diunggah. Validasi *file*-nya juga bisa dilakukan dengan mudah melalui fungsi *rules* di model.

Adapun untuk impor data khususnya *spreadsheet*, kita bisa menggunakan *library* yang mampu menangani *spreadsheet* seperti PHPExcel dimana *extension* hscstudio/yii2-export telah membungkus *library* tersebut. Demikian juga dengan pelaporan, kita bisa menggunakan *library* yang *powerfull* yaitu OpenTBS (tinybutstrong).

Visualisasi data bisa dengan mudah dibuat di Yii menggunakan *extension* yii2-highchart di mana *extension* ini merupakan *wrapper* dari *library* Highchart. Penulis juga membuat *extension* yang semisal dengan ini yaitu hscstudio/yii2-chart

Pada bab selanjutnya, kita akan belajar tentang *web service*, baik penggunaan maupun pembuatannya.

# 14

## Web Service

Pada bab ini kita akan belajar tentang:

- Definisi *web service*
- Cara menggunakan *web service*
- Cara membuat *web service*
- *Grabing* situs web

### A. Mengenal *Web Service*

#### A. 1. *Definisi*

*Web service* adalah standard yang digunakan untuk pertukaran data antar aplikasi atau sistem. Mengapa perlu standard? karena masing-masing aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada *platform* yang berbeda.

Contoh implementasi dari *web service* antara lain adalah SOAP dan REST.

#### A. 2. *RESTful*

REST adalah singkatan dari *REpresentational State Transfer*. Merupakan standard dalam arsitektur web yang menggunakan Protocol HTTP untuk pertukaran data. Konsep REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.

Cara kerjanya, REST server menyediakan jalur untuk akses *resource* atau data, sedangkan REST *client* melakukan akses *resource* dan kemudian menampilkan atau menggunakannya. *Resource* yang dihasilkan sebenarnya berupa teks, namun formatnya bisa bermacam-macam tergantung keinginan pemrogram aplikasi, umumnya adalah JSON dan XML.

Dalam mengakses sebuah *resource*, REST juga menggunakan konsep URI dimana ada *method* yang digunakan, secara *default* adalah *GET*. Berikut ini *method-method* yang mendukung REST:

- *GET*, cocok untuk *resource* yang hanya perlu dibaca saja (*read only*)
- *PUT*, cocok digunakan untuk membuat/*create resource* baru.
- *DELETE*, cocok digunakan untuk menghapus suatu *resource*.
- *POST*, cocok digunakan untuk memperbarui suatu *resource*.
- *OPTIONS*, cocok digunakan untuk mendapatkan operasi yang didukung pada *resource*.

Penulis menggunakan kata cocok karena implementasinya sebenarnya terserah masing-masing pemrogram aplikasi, itu hanya standarisasi. Ibarat kita di Indonesia harusnya bicara menggunakan bahasa Indonesia yang baik dan benar namun kenyataanya, bahasa kita sering bercampur dengan bahasa daerahnya masing-masing ☺. Tetapi komunikasi masih berjalan dengan lancar bukan? Nah itu yang lebih penting.

*Web services* yang berbasis arsitektur REST kemudian dikenal sebagai RESTful *web services*. Pada panduan ini kita akan fokus membahas tentang *web service* jenis RESTful.

#### A. 3. *Cara Kerja Web Service*

Cara kerja RESTful *web service* sebenarnya sangat sederhana. Berikut ini alur kerjanya:

Mula-mula suatu *client* mengirimkan sebuah data atau *request* melalui HTTP *Request* dan kemudian server merespon melalui HTTP *Response*. Yap, seperti itu saja ☺

Adapun komponen dari HTTP *request* adalah:

- *Verb*, HTTP *method* yang digunakan misalnya *GET*, *POST*, *DELETE*, *PUT* dll.
- *URI*, *Uniform Resource Identifier* (*URI*) untuk mengidentifikasi lokasi *resource* pada server.
- *HTTP Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.

- Request Header, berisi metadata untuk HTTP Request. Contoh, type client/ *web browser*, format yang didukung oleh client, format dari body pesan, setting cache dll.
- Request Body, konten dari data.

Sedangkan komponen dari HTTP *response* adalah:

- *Status/Response Code*, mengindikasikan status server terhadap *resource* yang direquest. misal : 404, artinya *resource* tidak ditemukan dan 200 *response OK*.
- *HTTP Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
- *Response Header*, berisi metadata untuk HTTP *Response*. Contoh, tipe server, panjang konten, tipe konten, waktu *response*, dll
- *Response Body*, konten dari data yang diberikan.

Dalam arsitektur REST, pemrogram aplikasi seharusnya tidak boleh menyimpan *state* atau penanda dari *client* di server. Hal ini disebut sebagai *stateless* atau *statelessness*. Contohnya pada kasus *session*, dimana *session* merupakan penanda *client* yang disimpan di server. Nah *session* pada arsitektur REST tidak diperbolehkan.

Keuntungan dari *stateless* antara lain:

- *Web services* dapat melayani setiap *request* secara *independent*.
- *Web services* tidak perlu menyimpan penanda pengguna, hal ini tentu membuat desain *application* lebih sederhana.
- HTTP merupakan protokol yang bersifat *stateless*.

Sedangkan kerugian dari *stateless* yaitu: *web services* membutuhkan informasi tambahan pada masing-masing *request* untuk menerjemahkan *request* dan *state* dari *client*. Solusi dari masalah ini adalah perlunya *token* untuk setiap *request*.

Target utama dari *stateless* sebenarnya adalah untuk *scale-up* atau meningkatkan *concurrent* akses (akses pada saat bersamaan) terhadap aplikasi. Karena aplikasi tidak perlu menyimpan *state* dari *client* sehingga meningkatkan jumlah *service* terhadap *request* pada satu waktu.

## B. Menggunakan *Web Service*

### B. 1. Pendahuluan

Untuk memanfaatkan data *web service* pada aplikasi web maka kita membutuhkan penghubung atau *tools* agar aplikasi kita bisa berkomunikasi dengan aplikasi penyedia data. Adapun *tools* tersebut adalah yiisoft/yii2-*httpclient*.

*Extension* ini sebenarnya belum *release* untuk *production*, namun berdasarkan uji coba yang dilakukan penulis, sepertinya *extension* ini sudah layak untuk digunakan, silakan lihat di

∞ <https://github.com/yiisoft/yii2-HttpClient>.

---

Alternatif lain yang bisa digunakan adalah <https://github.com/Understeam/yii2-HttpClient>, *extension* ini merupakan *wrapper* dari *library* HTTP *client* Guzzle yang sudah cukup populer.

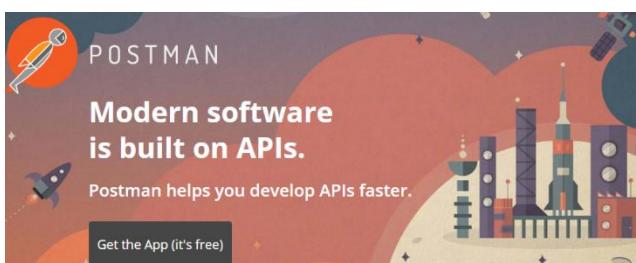
Cara Instalasinya menggunakan perintah berikut

```
| composer require --prefer-dist yiisoft/yii2-HttpClient "dev-master"
```

Instalasi menggunakan versi dev-master karena memang belum ada *release* untuk *extension* ini.

### B. 2. Tools untuk Uji Coba

Untuk sekedar menguji apakah *web service* berjalan atau tidak sesuai dengan yang diharapkan maka sebenarnya kita bisa gunakan *tools* HTTP *client* yaitu Postman - *plugin* dari *web browser* Chrome. <https://www.getpostman.com>



Berikut tampilan antarmuka pengguna dan contoh penggunaannya.

- *Request URL*

GET ▼ http://api.rajaongkir.com/starter/province Params Send

- Menentukan *Method* dari *Request*

GET ▼ http://api.rajaongkir.com/starter/province Params Send

**POST**

PUT  
PATCH  
DELETE

- Menambahkan Parameter Pada *Request*

GET ▼ http://api.rajaongkir.com/starter/province Params Send

Authorization Headers (1) Body Pre-request script Tests

key	a23e91 API KEY DISINI 55405c	X
Header	Value	edit

- Menambahkan Parameter pada *Method* POST

POST ▼ http://api.rajaongkir.com/starter/cost Params Send

Authorization Headers (1) **Body** Pre-request script Tests

form-data x-www-form-urlencoded raw binary

origin	501	Text
destination	114	Text
weight	1700	Text
courier	jne	Text

### B. 3. Mengecek Tarif Pengiriman

Pada studi kasus dalam buku ini yaitu toko daring, kita membutuhkan data berupa tarif pengiriman yang populer sebagai ongkos kirim (ongkir). Kita sebenarnya bisa membuat basis data sendiri untuk ongkos kirim ini, dimana sumber data bisa kita dapatkan dengan mudah di internet. Hanya saja kita perlu memperbaruiinya jika terjadi perubahan ongkir, dan tentu saja ini cukup merepotkan.

Oleh karena itu, kita akan memanfaatkan jasa pihak ketiga untuk mendapatkan info tentang ongkir melalui *web service*. Entah mengapa masing-masing ekspedisi yaitu POS, JNE, TIKI dll tidak ada satupun yang menyediakan *web service* ongkirnya. Padahal itu untuk tujuan mereka sendiri. Mudah-mudahan ada salah satu dari pemrogram aplikasi mereka yang membaca buku ini dan kemudian membuat *web service* resminya.

Pada panduan ini kita akan menggunakan *web service* dari <http://rajaongkir.com>, yaitu penyedia *web service* ongkos kirim gratis\*) untuk jasa ekspedisi besar.

The screenshot shows the RajaOngkir website. At the top, there's a header with links for Beranda, Integrasi, Download, Bantuan, and Akun. Below the header, there's a section titled 'Aplikasi Android' with a sub-section about the Android app. It includes a small image of a smartphone with a stylus. A purple button labeled 'Unduh gratis' is visible.

Adapun jasa ekspedisi yang didukung yaitu: JNE, POS, TIKI, (untuk versi *premium* termasuk juga ESL, RPX, WAHANA, PANDU, PCP, dan ongkir Internasional dll). Selain data ongkir, *web service* dari rajaongkir juga menyediakan data provinsi dan kota di Indonesia (versi premiumnya menyediakan data hingga kecamatan). Lebih lengkapnya silakan baca fitur-fitur yang disediakan di sini <http://rajaongkir.com/dokumentasi>

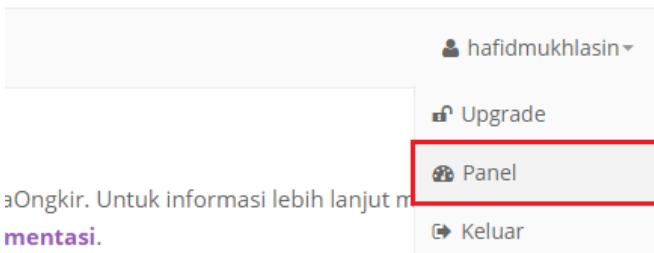
Di samping itu, cara menggunakan *web service*-nya terbilang cukup mudah apalagi didukung dengan dokumentasi API yang jelas. Pada panduan ini kita akan menggunakan versi gratis yaitu *starter*, dokumentasinya bisa anda baca di sini <http://rajaongkir.com/dokumentasi/starter#>

Untuk menggunakan API-nya, maka kita perlu registrasi dahulu dengan mengakses URL <http://rajaongkir.com/akun/daftar>

The registration form has fields for Username (hafidmukhlasin), Email (hafidmukhlasin@gmail.com), Password, and Konfirmasi Password. Below the form is a reCAPTCHA box with a checked checkbox 'Saya bukan robot'. At the bottom is a purple 'Daftar' button.

Setelah proses registrasi selesai, maka kita bisa login.

Kemudian klik menu pojok kanan atas (nama akun kita), maka akan muncul *dropdown* menu, silakan pilih menu Panel



Maka kita akan di bawa ke halaman API key

The screenshot shows a user interface for managing API keys. On the left, there are navigation links: Beranda, Integrasi, Download, and Bantuan. Below these are three buttons: Profil, API Key (which is selected), Upgrade, and Ubah perujuk. The main content area is titled "API Key" and contains the following text: "Gunakan API Key ini untuk menggunakan API RajaOngkir. Untuk informasi lebih lan menggunakan API RajaOngkir, silakan baca [dokumentasi](#)". Below this is a large red-bordered box containing the API key value: "a23eb1^2384.. 7c80af9 31a55405c". A small orange warning message at the bottom of this box reads: "Peringatan: API key Anda berfungsi layaknya username dan password. Jaga baik".

### Mendapatkan Data Propinsi

Method "province" digunakan untuk mendapatkan daftar propinsi yang ada di Indonesia.

- URL yang bisa digunakan.

Method	URL
GET	∞ <a href="http://api.rajaongkir.com/starter/province">http://api.rajaongkir.com/starter/province</a>

- Parameter yang bisa digunakan

Method	Parameter	Wajib	Tipe	Keterangan
GET/HEAD	key	Ya	String	API Key
GET/HEAD	android-key	Tidak	String	Identitas aplikasi Android
GET/HEAD	ios-key	Tidak	String	Identitas aplikasi iOS
GET	id	Tidak	String	ID propinsi

- Contoh Request

∞ <http://api.rajaongkir.com/starter/province?id=12>

Adapun respon dari request di atas sebagai berikut:

- Response Sukses

Berupa JSON sebagai berikut

```
{
 "rajaongkir": {
 "query": {
 "id": "12"
 },
 "status": {
 "code": 200,
 "description": "OK"
 },
 "results": {
 "province_id": "12",
 "province": "Kalimantan Barat"
 }
 }
}
```

- Respon Gagal

```
{
 "rajaongkir": {
 "status": {
 "code": 400,
 "description": "Invalid key."
 }
 }
}
```

## Penjelasan respon

Komponen	Tipe	Keterangan
Id	String	ID propinsi
Code	Int	Code status response
description	String	Penjelasan dari kode status
province_id	String	ID propinsi
province_name	String	Nama propinsi

Berikut ini implementasi di Yii, buat buat TestController @app/controllers/TestController.php

```
<?php
namespace app\controllers;
use Yii;
use yii\web\Controller;
use yii\httpClient\Client;

class TestController extends Controller
{
}
```

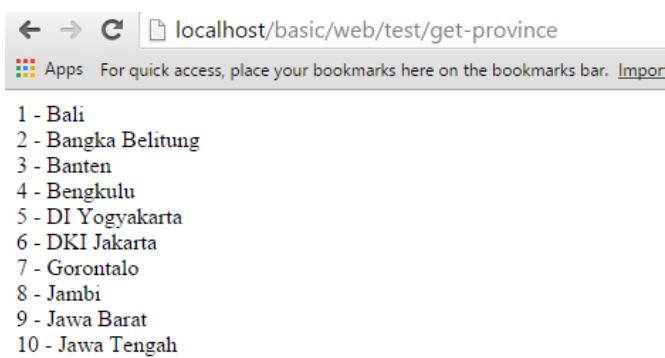
Untuk mendapatkan data *province*, kita bisa menambahkan fungsi sebagai berikut

```
public function actionGetProvince($id=0)
{
 $client = new Client();
 $addUrl=($id>0)?'id='.$id:'';
 $response = $client->createRequest()
 ->setFormat(Client::FORMAT_JSON)
 ->setMethod('get')
 ->setUrl('http://api.rajaongkir.com/starter/province?'.$addUrl)
 ->addHeaders([
 'key' => 'API KEY RAJA ONGKIR MU',
])
 ->send();
 if ($response->isOk) {
 $content = \Yii\helpers\Json::decode($response->content);
 // $content['rajaongkir']['query']
 // $content['rajaongkir']['status']
 $results = $content['rajaongkir']['results'];
 if ($id > 0) {
 if(count($results)>0) {
 echo $results['province_id'] . ' - ';
 echo $results['province'] . '
';
 }
 else{
 echo "blank";
 }
 } else {
 foreach ($results as $provinces) {
 echo $provinces['province_id'] . " - ". $provinces['province'] . "
";
 }
 }
 } else{
 $content = \Yii\helpers\Json::decode($response->content);
 echo $content['rajaongkir']['status']['description'];
 }
}
```

Kita bisa menjalankannya dengan mengakses URL berikut

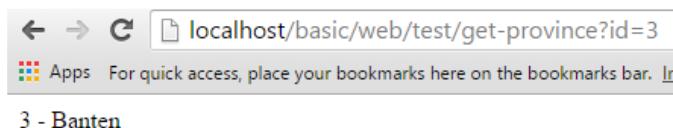
∞ <http://localhost/basic/web/test/get-province>

yang akan mengembalikan daftar nama provinsi seluruh Indonesia



atau untuk mendapatkan nama provinsi berdasarkan id provinsi, kita bisa gunakan URL

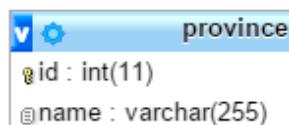
∞ `http://localhost/basic/web/test/get-province?id=ID_PROVINCE`



Hasil keluaran dari *request* ini sebenarnya adalah berformat Json, namun kita bisa meng-convert-nya menjadi *array* dengan menggunakan fungsi *decode* sebagai berikut

```
| $content = \Yii\helpers\Json::decode($response->content);
```

Mumpung gratis, kita bisa ambil data *province* tersebut ke lokal basis data kita, namun sebelumnya buat tabel *province* sebagai berikut:



Lalu kita bisa tambahkan kode berikut ke dalam fungsi di atas untuk menyimpan data *province* daring ke tabel *province* di lokal.

```
| Yii::$app->db->createCommand()->insert('province', [
 'id' => $provinces['province_id'],
 'name' => $provinces['province']
])->execute();
```

### Mendapatkan Data Kota

Method "city" digunakan untuk mendapatkan daftar kota/kabupaten yang ada di Indonesia.

- URL yang bisa digunakan.

Method	URL
GET	∞ <code>http://api.rajaongkir.com/starter/city</code>

- Parameter yang bisa digunakan

Method	Parameter	Wajib	Tipe	Keterangan
GET/HEAD	key	Ya	String	API Key
GET/HEAD	android-key	Tidak	String	Identitas aplikasi Android
GET/HEAD	ios-key	Tidak	String	Identitas aplikasi iOS
GET	id	Tidak	String	ID kota/kabupaten

GET	province	Tidak	String	ID propinsi
-----	----------	-------	--------	-------------

- Contoh Request
- ∞ <http://api.rajaongkir.com/starter/city?id=39&province=5>

Adapun respon dari *request* di atas sebagai berikut:

- Response Sukses

Berupa JSON sebagai berikut

```
{
 "rajaongkir": {
 "query": {
 "province": "5",
 "id": "39"
 },
 "status": {
 "code": 200,
 "description": "OK"
 },
 "results": {
 "city_id": "39",
 "province_id": "5",
 "province": "DI Yogyakarta",
 "type": "Kabupaten",
 "city_name": "Bantul",
 "postal_code": "55700"
 }
 }
}
```

- Respon Gagal

```
{
 "rajaongkir": {
 "status": {
 "code": 400,
 "description": "Invalid key."
 }
 }
}
```

Penjelasan respon

Komponen	Tipe	Keterangan
Id	String	ID kota/kabupaten
Code	Int	Code status <i>response</i>
description	String	Penjelasan dari kode status
province_id	String	ID propinsi
city_id	String	ID kota/kabupaten
province	String	Nama propinsi
Type	String	Jenis Daerah Tingkat II. Berisi "Kota" atau "Kabupaten"
city_name	String	Nama kota/kabupaten
postal_code	String	Kodepos kota/kabupaten

Berikut ini implementasi di Yii untuk mendapatkan data city

```
public function actionGetCity($id=0, $province=0)
{
 $client = new Client();
 $addUrl=($id>0)?'id='.$id.'&':';
 $addUrl.=($province>0)?'province='.$province:'';
 $response = $client->createRequest()
```

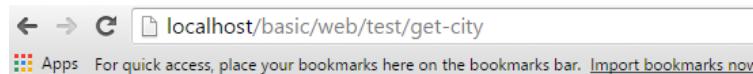
```

->setFormat(Client::FORMAT_JSON)
->setMethod('get')
->setUrl('http://api.rajaongkir.com/starter/city?'. $addUrl)
->addHeaders([
 'key' => 'API KEY RAJA ONGKIRMU',
])
->send();
if ($response->isOk) {
 $content = \Yii\helpers\Json::decode($response->content);
 // $content['rajaongkir']['query']
 // $content['rajaongkir']['status']
 $results = $content['rajaongkir']['results'];
 if($id>0){
 if(count($results)>0) {
 echo "<h1>". $results['province_id'] . " - " . $results['province'] . "</h1>
";
 echo $results['city_id'] . " - " . $results['city_name'] . " - " . $results['
type'] . " - " . $results['postal_code'] . "
";
 }
 else{
 echo 'blank';
 }
 }
 else{
 if(count($results)>0) {
 $last_province = 0;
 foreach ($results as $cities) {
 if ($last_province != $cities['province_id']) {
 echo "<h1> " . $cities['province_id'] . " - " . $cities['provi
nce'] . "</h1>";
 $last_province = $cities['province_id'];
 }
 echo $cities['city_id'] . " - " . $cities['city_name'] . " - " . "
$cities['type'] . " - " . $cities['postal_code'] . "
";
 }
 else{
 echo 'blank';
 }
 }
 }
 else{
 $content = \Yii\helpers\Json::decode($response->content);
 echo $content['rajaongkir']['status']['description'];
 }
}
}

```

Kode di atas hampir sama dengan kode untuk mendapatkan data propinsi hanya saja ada tambahan data kota yang bisa kita tampilkan.

- Mendapatkan data kota seluruh indonesia
- ∞ <http://localhost/basic/web/test/get-city>



## 21 - Nanggroe Aceh Darussalam (NAD)

- 1 - Aceh Barat - Kabupaten - 23681
- 2 - Aceh Barat Daya - Kabupaten - 23764
- 3 - Aceh Besar - Kabupaten - 23951
- 4 - Aceh Jaya - Kabupaten - 23654
- 5 - Aceh Selatan - Kabupaten - 23719
- 6 - Aceh Singkil - Kabupaten - 24785

- Mendapatkan data kota berdasarkan id province
- ∞ [http://localhost/basic/web/test/get-city?province=ID\\_PROVINCE](http://localhost/basic/web/test/get-city?province=ID_PROVINCE)

localhost/basic/web/test/get-city?province=3

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now

### 3 - Banten

106 - Cilegon - Kota - 42417  
 232 - Lebak - Kabupaten - 42319  
 331 - Pandeglang - Kabupaten - 42212  
 402 - Serang - Kabupaten - 42182  
 403 - Serang - Kota - 42111

- Menampilkan data kota berdasarkan id kota
- ∞ [http://localhost/basic/web/test/get-city?id=ID\\_CITY](http://localhost/basic/web/test/get-city?id=ID_CITY)

localhost/basic/web/test/get-city?id=3

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now

### 21 - Nanggroe Aceh Darussalam (NAD)

3 - Aceh Besar - Kabupaten - 23951

Sama dengan data *province*, kita juga bisa mengambil data *city* tersebut ke lokal basis data kita, namun sebelumnya buat tabel *city* sebagai berikut:

**city**

- id : int(11)
- #province\_id : int(11)
- @name : varchar(255)
- ♦type : enum('kabupaten','kota')
- @postal\_code : varchar(10)

Lalu kita bisa tambahkan kode berikut ke dalam fungsi di atas untuk menyimpan data *city* daring ke tabel *city* di lokal.

```
Yii::$app->db->createCommand()->insert('city', [
 'id' => $cities['city_id'],
 'province_id' => $cities['province_id'],
 'name' => $cities['city_name'],
 'type' => strtolower($cities['type']),
 'postal_code' => $cities['postal_code']
])->execute();
```

#### Mendapatkan Data Ongkir

Method “cost” digunakan untuk mengetahui tarif pengiriman (ongkos kirim) dari dan ke kota tujuan tertentu dengan berat tertentu pula.

- URL yang bisa digunakan.

Method	URL		
GET	∞ <a href="http://api.rajaongkir.com/starter/cost">http://api.rajaongkir.com/starter/cost</a>		

- Parameter yang bisa digunakan

Method	Parameter	Wajib	Tipe	Keterangan
POST/HEAD	key	Ya	String	API Key
POST/HEAD	android-key	Tidak	String	Identitas aplikasi Android
POST/HEAD	ios-key	Tidak	String	Identitas aplikasi iOS
POST	origin	Ya	String	ID kota atau kabupaten asal

POST	destination	Ya	String	ID kota atau kabupaten tujuan
POST	weight	Ya	Int	Berat kiriman dalam gram
POST	courier	Ya	String	Kode kurir: jne, pos, tiki.

- Contoh Request

Via CURL

```
curl_setopt_array($curl, array(
 CURLOPT_URL => "http://api.rajaongkir.com/starter/cost",
 CURLOPT_CUSTOMREQUEST => "POST",
 CURLOPT_POSTFIELDS => "origin=501&destination=114&weight=1700&courier=jne",
 CURLOPT_HTTPHEADER => array(
 "content-type: application/x-www-form-urlencoded",
 "key: your-api-key"
),
));
```

Adapun respon dari *request* di atas sebagai berikut:

- Response Sukses

Berupa JSON sebagai berikut

```
{
 "rajaongkir": {
 "query": {
 "origin": "501",
 "destination": "114",
 "weight": 1700,
 "courier": "jne"
 },
 "status": {
 "code": 200,
 "description": "OK"
 },
 "origin_details": {
 "city_id": "501",
 "province_id": "5",
 "province": "DI Yogyakarta",
 "type": "Kota",
 "city_name": "Yogyakarta",
 "postal_code": "55000"
 },
 "destination_details": {
 "city_id": "114",
 "province_id": "1",
 "province": "Bali",
 "type": "Kota",
 "city_name": "Denpasar",
 "postal_code": "80000"
 },
 "results": [
 {
 "code": "jne",
 "name": "Jalur Nugraha Ekakurir (JNE)",
 "costs": [
 {
 "service": "OKE",
 "description": "Ongkos Kirim Ekonomis",
 "cost": [
 {
 "value": 38000,
 "etd": "4-5",
 "note": ""
 }
]
 }]
```

```

 {
 "service": "REG",
 "description": "Layanan Reguler",
 "cost": [
 {
 "value": 44000,
 "etd": "2-3",
 "note": ""
 }
]
 },
 {
 "service": "SPS",
 "description": "Super Speed",
 "cost": [
 {
 "value": 349000,
 "etd": "",
 "note": ""
 }
]
 },
 {
 "service": "YES",
 "description": "Yakin Esok Sampai",
 "cost": [
 {
 "value": 98000,
 "etd": "1-1",
 "note": ""
 }
]
 }
]
}

```

#### - Respon Gagal

```

{
 "rajaongkir": {
 "status": {
 "code": 400,
 "description": "Invalid key."
 }
 }
}

```

Penjelasan respon

Komponen	Tipe	Keterangan
Id	String	ID propinsi
Code	Int	Code status response
description	String	Penjelasan dari kode status
province_id	String	ID propinsi
province_name	String	Nama propinsi
Origin	String	ID kota/kabupaten asal
destination	String	ID kota/kabupaten tujuan
weight	Int	Berat kiriman
courier	String	Kode kurir yang dipakai
Code	Int	Kode status response

<i>description</i>	<i>String</i>	Penjelasan kode status
<i>city_id</i>	<i>String</i>	ID kota atau kabupaten
<i>province_id</i>	<i>String</i>	ID propinsi
<i>province</i>	<i>String</i>	Propinsi dimana kota atau kabupaten berada
<i>Type</i>	<i>String</i>	Jenis Daerah Tingkat II
<i>city_name</i>	<i>String</i>	Nama kota atau kabupaten
<i>postal_code</i>	<i>String</i>	Kodepos kota atau kabupaten
<i>code</i>	<i>String</i>	Bagian dari <i>results</i> yang merupakan kode kurir
<i>name</i>	<i>String</i>	Nama kurir
<i>service</i>	<i>String</i>	Nama layanan yang digunakan dalam pengiriman
<i>description</i>	<i>String</i>	Dekripsi dari layanan pengiriman terkait
<i>value</i>	<i>Int</i>	Tarif pengiriman (ongkos kirim)
<i>etd</i>	<i>String</i>	Perkiraan waktu pengiriman (dalam hari).
<i>note</i>	<i>String</i>	Catatan terkait tarif pengiriman

Untuk mengimplementasikannya di Yii, kita bisa memanfaatkan data *city* dan *province* yang telah kita simpan ke basis data lokal sebagai data input. Namun sebelumnya kita harus buat model ActiveRecord-nya untuk ke dua tabel tersebut, gunakan *tools Gii*.

Mari kita buat fungsi baru pada TestController yaitu *actionCekOngkir*. Oleh karena *method* dari web *service* ini adalah POST maka kita perlu membuat formulir dengan *field* sebagai berikut.

<i>field</i>	<i>type</i>	keterangan
<i>origin</i>	<i>String</i>	ID kota atau kabupaten asal
<i>destination</i>	<i>String</i>	ID kota atau kabupaten tujuan
<i>weight</i>	<i>Int</i>	Berat kiriman dalam gram
<i>courier</i>	<i>String</i>	Kode kurir: jne, pos, tiki.

Untuk itu kita perlu sebuah model untuk memudahkan kita membuat formulir dan validasinya. Yii mempunyai *dynamic model* yang cocok digunakan pada kasus seperti ini.

```
public function actionCekOngkir()
{
 $model = new \yii\base\DynamicModel([
 'origin', 'destination', 'weight', 'courier',
]);
 $model->addRule(['origin', 'destination', 'weight', 'courier'], 'required');
 $model->addRule(['weight'], 'integer');
 $model->addRule(['courier'], 'in', ['range' => ['jne', 'pos', 'tiki']]);
 return $this->render('cek-ongkir', ['model'=>$model]);
}
```

Lalu kita buat *file* untuk *view formnya* @app/views/test/cek-ongkir.php

```
<?php
use yii\helpers\Html;
use yii\widgets\ActiveForm;
?>
<div class="cek-ongkir-form">
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'origin')->textInput() ?>
<?= $form->field($model, 'destination')->textInput() ?>
<?= $form->field($model, 'weight')->textInput(['value'=>1000]) ?>
<?= $form->field($model, 'courier')->dropDownList([
 'jne'=>'JNE',
 'pos'=>'POS',
 'tiki'=>'TIKI'
]) ?>
</div>
```

```

 'tiki'=>'TIKI',
]) ?>
<div class="form-group">
 <?= Html::submitButton('Cek Ongkir', [
 'class' => 'btn btn-success'
]) ?>
</div>
<?php ActiveForm::end(); ?>
</div>

```

Untuk *origin* dan *destination* kita bisa menggunakan data sumber dari basis data lokal yaitu data *city* yang kita peroleh dari RajaOngkir. Sebenarnya kita bisa gunakan dropDownList seperti atribut *courier*, namun agar mempunyai fitur *searching* maka kita bisa gunakan *widget select2* (merupakan paket dari *extension yii2-widgets* yang telah kita instalasi sebelumnya) @app/views/test/cek-ongkir.php

```

$data = ArrayHelper::map(
 City::find()->select(['id', 'name'])
 ->asArray()->all(),
 'id', 'name');

echo $form->field($model, 'origin')->widget(Select2::classname(), [
 'options' => ['placeholder' => 'Select for a city ...'],
 'data'=>$data,
]);
echo $form->field($model, 'destination')->widget(Select2::classname(), [
 'options' => ['placeholder' => 'Select for a city ...'],
 'data'=>$data,
]);

```

Tentunya kita perlu *use* dulu ketiga *library* yang kita gunakan di atas

```

use kartik\widgets\Select2;
use yii\helpers\ArrayHelper;
use app\models\City;

```

Mari kita edit fungsi *actionCekOngkir* pada *TestController* untuk memproses formulir cek ongkir di atas

```

public function actionCekOngkir()
{
 $model = new \yii\base\DynamicModel([
 'origin', 'destination', 'weight', 'courier',
]);
 $model->addRule(['origin', 'destination', 'weight', 'courier'], 'required');
 $model->addRule(['weight'], 'integer');
 $model->addRule(['courier'], 'in', ['range' => ['jne', 'pos', 'tiki']]);
 $results = [];
 if ($model->load(Yii::$app->request->post())) {
 $client = new Client();
 $response = $client->createRequest()
 ->setMethod('post')
 ->setUrl('http://api.rajaongkir.com/starter/cost')
 ->addHeaders([
 'key' => 'API_KEY',
])
 ->setData([
 'origin' => $model->origin,
 'destination' => $model->destination,
 'weight' => $model->weight,
 'courier' => $model->courier,
])
 ->send();
 if ($response->isOk) {
 $results = json_decode($response->content);
 }
 }
 return $this->render('cek-ongkir', [
 'model'=>$model,
 'results'=>$results,
]);
}

```

Langkah selanjutnya adalah menampilkan variabel *\$results* yang dihasilkan ke *view* @app/views/test/cek-ongkir.php

```

if(!empty($results)){
 echo "<table class='table'>";
 foreach($results as $result) {
 //print_r($result->query);
 //print_r($result->status);
 //print_r($result->origin_details);
 //print_r($result->results);
 foreach ($result->results[0]->costs as $costs) {
 echo '<tr><th>service</th><th>: </th><th>' . $costs-
>service . '</th></tr>';
 echo '<tr><td>description</td><td>: </td><td>' . $costs-
>description . '</td></tr>';
 echo '<tr><td>cost</td><td>: </td><td>' . $costs->cost[0]-
>value . '</td></tr>';
 echo '<tr><td>etd</td><td>: </td><td>' . $costs->cost[0]-
>etd. '</td></tr>';
 }
 }
}

```

Memang terlihat agak rumit, karen *array* yang bertumpuk, penulis pun harus mencoba beberapa kali hingga menghasilkan tampilan yang benar.

Mari kita uji coba

**Origin**

**Destination**

**Weight**

**Courier**

**Cek Ongkir**

Hasilnya bisa kita lihat sebagai berikut

<b>service</b>	<b>:</b>	<b>OKE</b>
description	<b>:</b>	Ongkos Kirim Ekonomis
cost	<b>:</b>	16000
etd	<b>:</b>	2
<b>service</b>	<b>:</b>	<b>REQ</b>
description	<b>:</b>	Layanan Reguler
cost	<b>:</b>	18000
etd	<b>:</b>	
<b>service</b>	<b>:</b>	<b>YES</b>
description	<b>:</b>	Yakin Esok Sampai
cost	<b>:</b>	44000
etd	<b>:</b>	1-1

Selesai. Silakan dikembangkan.

## B. 4. Mengecek Resi Pengiriman

Sama dengan ongkir, pada pengecekan resi (bukti pengiriman barang dari ekspedisi) pun juga tidak ada jasa ekspedisi yang membuatkan *web service*-nya secara resmi untuk umum. Oleh karena itu kita juga akan menggunakan jasa pihak ketiga yaitu <http://www.cekresi.co.id>

Sebenarnya cekresi.co.id tidak menyediakan *service API* untuk umum, namun berdasarkan penelusuran penulis, kita bisa menggunakannya.

- URL yang bisa digunakan.

Method	URL
POST	∞ <a href="http://api.cekresi.co.id/cnote.php">http://api.cekresi.co.id/cnote.php</a>

- Parameter yang bisa digunakan

Method	Parameter	Wajib	Tipe	Keterangan
POST	id	Ya	String	No resi

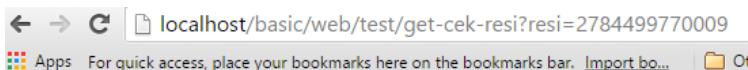
Adapun implementasinya di Yii bisa dilihat sebagai berikut

```
public function actionGetCekResi($resi='2784499770009')
{
 $client = new Client();
 $response = $client->createRequest()
 ->setMethod('post')
 ->setUrl('http://api.cekresi.co.id/cnote.php')
 ->setData(['id' => $resi])
 ->send();
 if ($response->isOk) {
 echo ($response->content);
 }
}
```

Untuk menggunakannya kita bisa akses URL

∞ [http://localhost/basic/web/test/get-cek-resi?resi=NO\\_RESI\\_JNE](http://localhost/basic/web/test/get-cek-resi?resi=NO_RESI_JNE)

Berikut ini hasilnya, yaitu berformat HTML



### Expedisi JNE

Customer Service: (021) 2927 8888, Email : [customercare@jne.co.id](mailto:customercare@jne.co.id)

#### I. Informasi Pengiriman

No Resi	: 2784499770009	
Status	: DELIVERED	
Service	: CTC15	
Dikirim tanggal	: 2015-11-08	
Dikirim oleh	: *SUB AGEN GARUT GARUT	
Dikirim ke	: TRISNO SONJAYA PURWAKARTA.KAB.PURWAKARTA	
JNE Status	: DELIVERED	
<b>POD Detail</b>		
<b>Tanggal</b>	<b>Lokasi</b>	<b>Keterangan</b>
2015-11-08	PURWAKARTA.KAB.PURWAKARTA	DELIVERED

#### Perhatian:

Cekresi.co.id tidak menyediakan *web service* pengecekan resi untuk umum. Informasi yang ada dalam buku ini semata hanya untuk kepentingan ilmiah.

Jika anda ingin menggunakannya untuk tujuan komersial, penulis menyarankan anda menggunakan *service berbayar* seperti RajaOngkir.com atau menghubungi pihak cekresi.co.id. Penulis tidak bertanggung jawab atas penyalahgunaan teknik ini.

## B. 5. Pembayaran menggunakan Payment Gateway

Materi ini sengaja penulis sampaikan sebagai dasar untuk membuat aplikasi toko daring yang akan dibahas pada seri kedua dari buku ini.

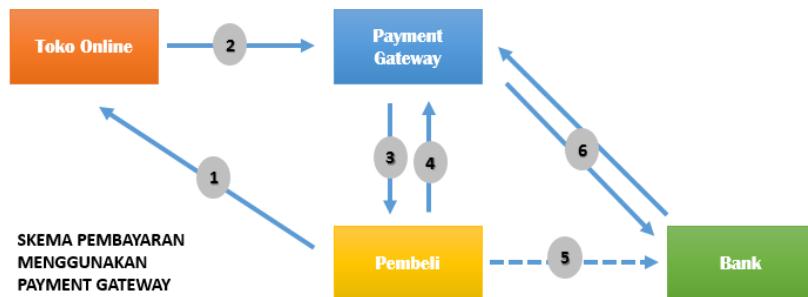
### **Mengenal Payment Gateway**

#### ❖ Apa itu Payment Gateway?

*Payment gateway* adalah perusahaan yang memberikan jasa perantara untuk transaksi pembayaran yang terhubung ke bank.

#### ❖ Bagaimana mekanisme pembayaran menggunakan *Payment Gateway*?

Gambaran sederhana mekanisme pembayaran menggunakan *payment gateway*



1. Mula-mula Pembeli melakukan *checkout* keranjang belanjanya pada toko daring
2. Toko daring mengirimkan data keranjang belanja Pembeli tersebut ke *Payment Gateway*
3. *Payment Gateway* menampilkan formulir pembayaran kepada Pembeli
4. Pembeli melakukan pembayaran langsung melalui *Payment Gateway*, dan atau
5. Pembeli melakukan pembayaran ke Bank (ATM atau Setor Tunai) (opsional)
6. *Payment Gateway* memproses pembayaran Pembeli ke Bank atau secara berkala mengecek data transaksi Pembeli pada Bank.

#### ❖ Apa keuntungan menggunakan *Payment Gateway*

Ada banyak keuntungan jika kita menggunakan jasa *payment gateway*, antara lain:

- Kita tidak perlu melakukan kerjasama dan perjanjian yang mungkin sangat ribet dengan pihak bank untuk mengakses data pembayaran customer toko daring kita
- Kita tidak perlu repot membuat sistem sendiri untuk mengakses data pembayaran *customer* toko daring kita, cukup menggunakan *web service* dari *payment gateway*
- Keamanan transaksi lebih terjamin karena sistem pada *payment gateway* telah teruji dan tersertifikasi keamanan internasional.

#### ❖ Berapa biaya yang kita keluarkan untuk bekerjasama dengan *payment gateway*?

Umumnya tidak ada biaya alias gratis untuk mendaftar dan mengintegrasikan sistem *payment gateway* dengan aplikasi kita. Adapun biasanya biaya dikenakan per transaksi, misalnya setiap transaksi *payment gateway* minta 1% dari nilai suatu transaksi. Sehingga jika ada transaksi Rp. 100.000,- maka biaya yang dikenakan hanya Rp. 1.000,- murah sekali bukan? Yap, murah sekaligus mudah.

#### ❖ Apa ada perusahaan *payment gateway* di Indonesia?

Ada, jika internasional kita mengenal Paypal (<https://paypal.com>), sedangkan untuk level nasional antara lain: Ipaymu - <https://ipaymu.com>, Veritrans - <https://veritrans.co.id>, Firstpay - <http://firstpay.co.id>, Doku - <http://doku.com>, dll

### **Registrasi Payment Gateway**

#### ❖ **Ipaymu**

Salah satu *payment gateway* populer yang digunakan terutama oleh UKM, dan toko daring di Indonesia adalah Ipaymu. Berikut ini beberapa fitur dan kelebihan dari Ipaymu sebagai *payment gateway*:

- Bebas Biaya Setup & Biaya Bulanan

- Transaksi *Online* Aman & Cepat
- Integrasi ATM Bersama & ATM Prima
- Integrasi 137 Bank Indonesia
- Dapat Melakukan Pengiriman Uang Melalui 4000 Kantor Pos
- Menerima Pembayaran *Non Member IPAYMU*
- Mudah Tarik Tunai melalui ATM
- Menerima Pembayaran Kartu Kredit
- Integrasi dengan Paypal
- Integrasi Mudah, *Plug & Play!*
- *Web Base & Mobile Payment*

Adapun fitur Ipaymu sebagai pembayaran secara daring bisa dilihat di sini <https://ipaymu.com/fitur>

Di samping fitur-fitur yang sangat lengkap, ipaymu juga sangat memanjakan pemrogram aplikasi khususnya PHP Programmer, karena dokumentasinya hanya mendukung bahasa pemrograman PHP

∞ <https://ipaymu.com/cara-integrasi-webstore-tingkat-lanjut/>

Sebelum menggunakan Ipaymu, kita harus memiliki akun Ipaymu dulu.

1. Akses <https://ipaymu.com>, tekan tombol Daftar Sekarang



2. Isi formulir pendaftaran

Transaksi Online Lebih Mudah

Email: hafidmukhlasin@gmail.com

Password:

Nama: Hafid Mukhlasin

No. HP (GSM / CDMA): 081328023455

Website (opsional): <http://www.hafidmukhlasin.com>

Captcha: 86095502

Ya, saya telah membaca dan setuju dengan [Persetujuan Pengguna](#) dan [Syarat & Ketentuan](#) yang telah ditentukan untuk menggunakan layanan iPaymu.

**DAFTAR SEKARANG**

3. Setelah menekan tombol DAFTAR SEKARANG, maka Ipaymu akan mengirimkan *email* konfirmasi, silakan tekan tautan konfirmasi yang terkirim ke *email* untuk mengaktifkan akun Ipaymu kita.



## Selamat!

Pendaftaran Anda telah terkirim. Kami mengirimkan email konfirmasi ke alamat email Anda.

Silahkan klik link yang terdapat di dalam email untuk konfirmasi akun iPaymu Anda.

Link konfirmasi hanya bekerja selama 48 jam.

- Jika sudah, maka silakan *login* ke akun Ipaymu

The screenshot shows the iPaymu account dashboard. At the top, there's a navigation bar with links for Akun Saya, Transfer, Penarikan, API Menu Penjual, Mobile Payment, Shopping Cart Tools, and Log Out. Below the navigation bar, the user information is displayed: "Selamat Datang, Hafid Mukhlasin" and "Status Akun: PERSONAL | UNVERIFIED". To the right, it shows "Saldo iPaymu: Rp. 0,00". At the bottom of this section are two buttons: "Detail Akun" and "Verifikasi Akun Anda".

Terlihat bahwa saldo Ipaymu saya masih 0 Ⓛ, yap karena akun memang baru dibuat. Akun ini juga belum terverifikasi, kita bisa memverifikasinya dengan menekan tombol “Verifikasi Akun anda” dan mengunggah beberapa dokumen yang dipersyaratkan.

- Di bawah bagian tersebut kita bisa dapatkan API Key, nah sebagai pemrogram aplikasi inilah yang sangat kita perlukan. Amankan API key itu karena akan kita gunakan pada bagian selanjutnya.

This screenshot shows the same iPaymu account dashboard as above. The "Detail Akun" and "Verifikasi Akun Anda" buttons are visible. A red box highlights the "API Key iPaymu Anda:" field, which contains a long string of characters: "jO2e...XFI4AkYtlMy87u...su26nG.". Below this field are two buttons: "Isi Saldo Sekarang" and "Panduan Integrasi".

- Aktifkan pembayaran dengan Paypal dan Kartu Kredit jika kita ingin menggunakan dua fitur itu.

- Silakan akses <https://my.ipaymu.com/members/merchant.htm>

<b>Layanan Non Member</b> Merchant iPaymu dapat menerima pembayaran dari transfer bank manapun dengan notifikasi otomatis.	<b>Layanan PayPal</b> PayPal dan iPaymu bekerja sama dalam bentuk sistem Single Checkout Platform. Merchant harus memiliki akun PayPal.	<b>Kartu Kredit</b> iPaymu mempermudah merchant untuk menerima pembayaran dengan kartu kredit.
<b>Aktifkan ➔</b>	<b>Aktifkan ➔</b>	<b>Aktifkan ➔</b>

Ikuti panduannya. Selesai.

### ***Uji Coba Payment Gateway***

#### ❖ **ipaymu**

Pada tautan ini <https://ipaymu.com/cara-integrasi-webstore-tingkat-lanjut>, kita bisa jumpai contoh implementasi ipaymu menggunakan PHP. Dengan berbekal API key mari kita uji coba dulu.

- Buat *folder* ipaymu di *localhost*, untuk uji coba
- Buat *file* payment.php

```
<?php
// URL Payment IPAYMU
$url = 'https://my.ipaymu.com/payment.htm';
// Prepare Parameters
```

```

$params = array(
 'key' => 'ganti dengan API KEY kamu', // API Key Penjual
 'action' => 'payment',
 'product' => 'Kaos Yii',
 'price' => '101000', // Total Harga
 'quantity' => 2,
 'comments' => 'Coba-coba aja', // Optional
 'ureturn' => 'http://localhost/ipaymu/return.php',
 'unotify' => 'http://localhost/ipaymu/notify.php',
 'ucancel' => 'http://localhost/ipaymu/cancel.php',
 /* Parameter untuk pembayaran lain menggunakan PayPal
 * -----
 'paypal_email' => 'ganti dengan email paypal kamu',
 'paypal_price' => 1, // Total harga dalam kurs USD
 'invoice_number' => uniqid('INV-'), // Optional
 /* -----
 'format' => 'json' // Format: xml / json. Default: xml
);
$params_string = http_build_query($params);
//open connection
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, count($params));
curl_setopt($ch, CURLOPT_POSTFIELDS, $params_string);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, FALSE);
//execute post
$request = curl_exec($ch);
if ($request === false) {
 echo 'Curl Error: ' . curl_error($ch);
} else {
 $result = json_decode($request, true);
 if(isset($result['url']))
 header('location: ' . $result['url']);
 else {
 echo "Request Error ". $result['Status'] .": ". $result['Keterangan'];
 }
}
//close connection
curl_close($ch);

```

### 3. Buat juga file return.php, notify.php, dan cancel.php

Adapun isi dari ketiga file itu bebas ☺ suka-suka aja.

Penjelasan:

- *Return URL* adalah halaman pada saat transaksi belanja pada toko daring, yang biasanya berisi ucapan terima kasih,  
∞ contoh: <http://www.domainanda.com/terimakasih.html>
- *Notify URL* adalah halaman dibelakang layar (tanpa tampilan) untuk memvalidasi kembali hasil transaksi yang akan dikirim iPaymu menggunakan *method POST* setelah terjadinya pembayaran,  
∞ contoh: <http://www.domainanda.com/notify-ipaymu.php>
- *Cancel URL* adalah halaman apabila *customer* melakukan pembatalan pembelian pada toko daring kita,  
∞ contoh <http://www.domainanda.com/batal.html>

### 4. Setelah itu akses payment.php

∞ <http://localhost/ipaymu/payment.php>

### 5. Maka kita akan dialihkan ke halaman berikut

Pembayaran ke:

**Hafid Mukhlasin**  
PERSONAL

**Detil Transaksi**

Deskripsi	Jumlah	Harga
Kaos Yii Coba-coba aja	2	Rp. 101.000,00
<b>Total</b>		<b>Rp. 202.000,00</b>

**Login (Member iPaymu)**

Jika Anda **sudah memiliki akun iPaymu**, silahkan login untuk melanjutkan transaksi.

Username

Password

Captcha

**Login** **Kembali**

Belum memiliki akun iPaymu? [Daftar sekarang!](#)

**Transfer Bank (Non-Member iPaymu)**

**PayPal**

Halaman inilah yang akan dilihat oleh pengunjung atau pembeli toko kita.

6. Untuk uji coba, pilih saja *Transfer Bank*, dan masukkan saja data fiktif.

**Transfer Bank (Non-Member iPaymu)**

Jika Anda **tidak memiliki akun iPaymu**, Anda tetap dapat melakukan pembayaran melalui transfer ke rekening bank dengan mengisi form berikut terlebih dahulu:

Nama	<input type="text"/>
Email	<input type="text"/>
No. HP / Telepon	<input type="text"/>
Nama Bank	<input type="text" value="-- Pilih --"/>
No. Rekening	<input type="text"/>
Catatan	<input type="text"/>
Captcha	<input type="text"/>
<b>Kirim</b> <b>Kembali</b>	

Setelah diisi lengkap silakan tekan tombol Kirim, maka akan terkirim ke *email* pengguna atau *customer* kita yang berisi cara pembayaran. Adapun halaman web akan dialihkan atau dikembalikan sesuai dengan URL *address* yang telah kita *set* pada parameter 'ureturn' di payment.php.

- Untuk mengecek apakah data pembayaran sudah dilakukan, kita bisa melihat atau memantauanya pada halaman *dashboard* ipaymu kita.

ID	Tanggal	Tipe	Member	Jumlah	Status
41126	01/10/2016 17:22 WIB	Non Member FROM	non member	Rp. 99.990,00	Pending

Data status ini *realtime*, artinya sesaat setelah pengguna melakukan pembayaran maka kita status akan berubah. Ipaymu juga menyediakan API untuk mengakses informasi ini. Dengan format URL:

∞ <https://my.ipaymu.com/api/CekTransaksi.php>

dengan parameter sebagai berikut:

Variabel	Keterangan
key	API Key (diperoleh melalui Menu Akun » Sunting Profil)
id	ID / kode unik transaksi
format (opsional)	Format hasil keluaran dari <i>request</i> . Bisa menggunakan XML ( <i>default</i> ) atau JSON

Contoh *request*:

∞ [https://my.ipaymu.com/api/CekTransaksi.php?key=ganti\\_dengan\\_api\\_key&id=320](https://my.ipaymu.com/api/CekTransaksi.php?key=ganti_dengan_api_key&id=320)

Hasil keluaran parameter:

Jika *request valid*

Variabel	Keterangan
Status	Kode status transaksi
	-1 => Sedang diproses
	0 => Pending
	1 => Berhasil
	2 => Batal
	3 => Refund
Keterangan	Keterangan status transaksi
Pengirim	Username pengirim
Penerima	Username penerima
Nominal	Nominal jumlah transaksi
Waktu	Tanggal dan waktu transaksi
Tipe	Tipe transaksi

Contoh hasil keluarannya (format: XML)

```

1
budi
eka
100000
02/11/2011 02:41 WIB
Berhasil Transfer

```

Jika *request* tidak valid (error)

Variabel	Keterangan
Status	Kode status galat
Keterangan	Keterangan galat

Contoh hasil keluarannya (format: XML)

```
-1005
Transaksi tidak ditemukan
```

## C. Membuat Web Service

Yii menyediakan fitur-fitur untuk mempermudah kita dalam mengimplementasikan RESTful *Web Service* APIs. Berikut ini beberapa fitur RESTful APIs yang didukung Yii:

- *Prototyping* cepat dengan dukungan API untuk ActiveRecord;
- Format respon yang bisa disesuaikan (mendukung JSON dan XML secara *default*);
- Objek yang diproses bisa disesuaikan untuk bidang hasil keluaran yang dipilih;
- Format dari *collection* data dan validasi galatnya lebih baik;
- *Routing* yang efisien dengan pengecekan HTTP *verb* yang tepat;
- Secara *built-in* mendukung *options* dan *head verb*;
- Otentikasi dan otorisasi;
- *Caching* data dan HTTP *caching*;
- Pembatasan akses (*rate limiting*);

Pada panduan ini akan dijabarkan tentang dua cara membuat RESTfull di Yii yaitu cara *instan* dan cara semi manual.

### C. 1. Instant

Yii memang tidak menyediakan *generator* kode Gii untuk membuat *web service*, namun dengan usaha atau koding minimal maka kita sudah bisa menghasilkan fitur RESTful yang lengkap.

Sebagai studi kasus, kita akan gunakan data *employee* yang mana model ActiveRecord-nya telah kita buat pada bab sebelumnya yaitu app\models\Employee.

#### Buat Controller

Caranya buat sebuah controller misal EmployeeRestController pada direktori @app/controllers/

```
<?php
namespace app\controllers;
use yii\rest\ActiveController;
class EmployeeRestController extends ActiveController{
 public $modelClass = 'app\models\Employee';
}
```

Pada kode di atas, *class controller* meng-extends *class yii\rest\ActiveController*, yang mengimplementasikan fungsi-fungsi umum dari RESTful (CRUD). Adapun pendefinisian variabel *modelClass* yang berisi app\models\Employee, supaya controller mengetahui tentang model yang dijadikan dasar untuk mendapatkan dan memanipulasi data.

#### Konfigurasi Routing

Untuk RESTful, kita perlu mengatur konfigurasi *routing*-nya. Pada @app/config/web.php, modifikasi konfigurasi pada *component* UrlManager

```
'urlManager' => [
 'enablePrettyUrl' => true,
 'showScriptName' => false,
 'rules' => [
 ['class' => 'yii\rest\UrlRule', 'controller' => 'employee-rest'],
],
]
```

Sesuaikan nilai dari *controllers* dengan *controller* ID RESTful yang telah dibuat sebelumnya.

### Enable Json Input

Setting ini hanyalah opsional, supaya RESTful API yang kita buat dapat menerima input data berformat JSON. Caranya, pada konfigurasi tambahkan *component request* sebagai berikut :

```
'request' => [
 'parsers' => [
 'application/json' => 'yii\web\JsonParser',
]
]
```

Tanpa konfigurasi ini, maka RESTful kita hanya bisa menerima input data berformat application/x-www-form-urlencoded dan multipart/form-data saja.

### Uji coba

Dengan usaha minimal di atas, kita sudah bisa menghasilkan fungsi RESTful API sebagai berikut:

- GET /employee-rests : daftar *employee* per *page*;
- GET /employee-rests/123 : detail *employee* 123
- POST /employee-rests : buat *employee* baru;
- PATCH /employee-rests/123 dan PUT /employee-rests/123: *update employee* 123;
- DELETE /employee-rests/123: hapus *employee* 123;
- HEAD /employee-rests : menampilkan informasi dari *daftar employee*;
- HEAD /employee-rests/123: menampilkan informasi dari *employee* 123;
- OPTIONS /employee-rests : menampilkan *verb* yang didukung pada URL /employee-rests;
- OPTIONS /employee-rests/123: menampilkan *verb* yang didukung pada URL /employee-rests/123.

Yii akan secara otomatis membuat nama *controller* jadi bentuk *plural*, dari *employee-rest* menjadi *employee-rests*. Namun kita juga bisa mengurnya melalui [[yii\rest\UrlRule:\$pluralize]]. Lihat contoh berikut:

```
'urlManager' => [
 'enablePrettyUrl' => true,
 'showScriptName' => false,
 'rules' => [
 [
 'class' => 'yii\rest\UrlRule',
 'pluralize' => false,
 'controller' => 'employee-rest'
],
],
],
```

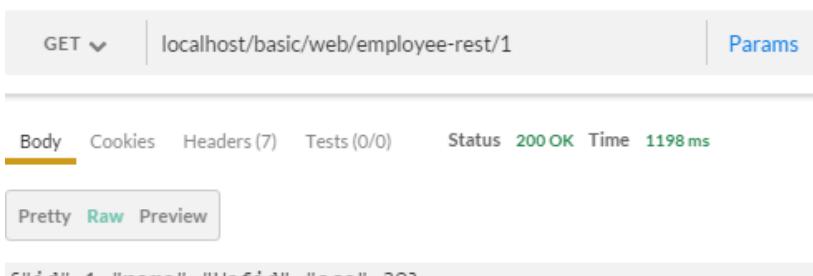
Untuk menguji cobanya kita akan menggunakan *tools Postman*

- Menampilkan *daftar employee*
- ∞ <http://localhost/basic/web/employee-rest>

The screenshot shows the Postman interface with a GET request to `localhost/basic/web/employee-rest`. The response status is `200 OK` and the time taken is `1233 ms`. The `Body` tab is selected, displaying the following JSON data:

```
[{"id":1,"name":"Hafid","age":29}, {"id":2,"name":"Junaidi","age":35}, {"id":3,"name":"Dewi","age":24}, {"id":4,"name":"Agung","age":51}, {"id":5,"name":"Fuad","age":12}, {"id":6,"name":"Susanti","age":22}, {"id":7,"name":"sadasd","age":1}, {"id":8,"name":"sadasdasd","age":12}, {"id":9,"name":"Hafids","age":1}, {"id":10,"name":"Oke","age":1}]
```

- Menampilkan *data employee* dengan ID = 1
- ∞ <http://localhost/basic/web/employee-rest/1>

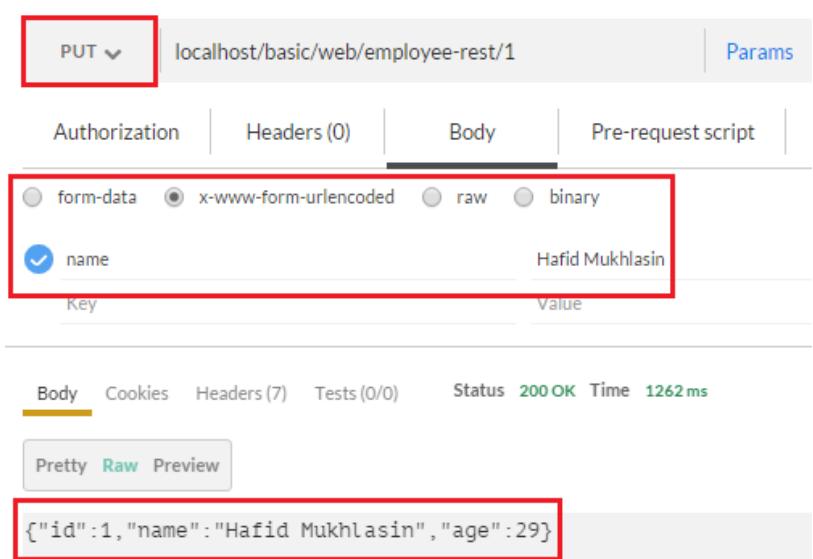


Body Cookies Headers (7) Tests (0/0) Status 200 OK Time 1198 ms

Pretty Raw Preview

```
{"id":1,"name":"Hafid","age":29}
```

- Memperbarui data *employee* untuk field *name* menjadi Hafid Mukhlasin, pada ID 1

PUT localhost/basic/web/employee-rest/1 Params

Authorization Headers (0) Body Pre-request script

form-data  x-www-form-urlencoded  raw  binary

name	Hafid Mukhlasin
Key	Value

Body Cookies Headers (7) Tests (0/0) Status 200 OK Time 1262 ms

Pretty Raw Preview

```
{"id":1,"name":"Hafid Mukhlasin","age":29}
```

Silakan diuji coba untuk action yang lain. Mudah sekali bukan?.

## C. 2. Semi Manual

Di samping cara instan di atas, ada cara lain yang penulis menyebutnya sebagai semi manual. Ya, karena memang tidak manual banget. Pada cara sebelumnya semua fungsi telah dibuat, sedangkan dengan cara ini kita harus membuat masing-masing fungsi yang kita inginkan secara manual. *Class* yang akan kita gunakan adalah [[yii\rest\Controller]].

Sebagai contoh, kita akan membuat RESTful untuk data *Province* maka kita buat sebuah *controller* yang meng-extends [[yii\rest\Controller]]

```
<?php
namespace app\controllers;
use yii\rest\Controller;
class ProvinceRestController extends Controller{}
```

Kemudian tambahkan fungsi *actionIndex* untuk menampilkan daftar *province*

```
public function actionIndex()
{
 $provinces = \app\models\Province::find()->all();
 return [
 'results'=>$provinces,
];
}
```

```
{"results": [{"id": 1, "name": "Bali"}, {"id": 2, "name": "Bangka Belitung"}, {"id": 3, "name": "Bengkulu"}, {"id": 4, "name": "DI Yogyakarta"}, {"id": 5, "name": "DKI Jakarta"}, {"id": 6, "name": "Gorontalo"}, {"id": 7, "name": "Jambi"}, {"id": 8, "name": "Jawa Barat"}, {"id": 9, "name": "Jawa Tengah"}, {"id": 10, "name": "Jawa Timur"}, {"id": 11, "name": "Kalimantan Selatan"}, {"id": 12, "name": "Kalimantan Tengah"}, {"id": 13, "name": "Kalimantan Timur"}, {"id": 14, "name": "Kalimantan Barat"}]
```

Untuk membatasi agar `actionIndex` ini hanya bisa diakses menggunakan *method* GET, maka kita bisa tambahkan fungsi *verb* sebagai berikut.

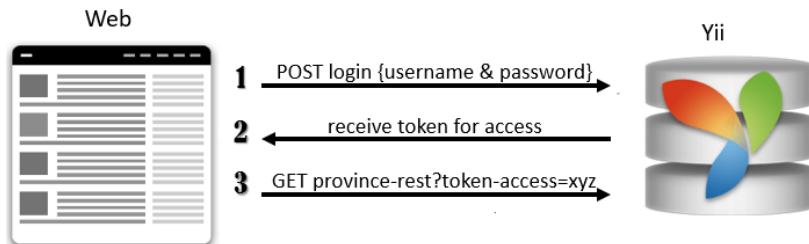
```
protected function verbs()
{
 return [
 'index' => ['GET'],
];
}
```

Tanpa itu maka `actionIndex` akan bisa diakses oleh semua *method*, kan lucu ☺.

### Otentikasi

Kita juga bisa membatasi agar hanya pengguna terdaftar yang boleh mengakses RESTful kita, maka kita bisa terapkan *authentication* yaitu `[[yii\filters\auth\QueryParamAuth]]`. Otentikasi ini menggunakan mekanisme pengiriman *token* (pada *web service* RajaOngkir disebut sebagai API Key) melalui *query parameter* di URL, e.g.,

∞ <https://example.com/users?access-token=xxxxxxxxx>.



Sebagaimana mekanisme yang digambarkan di atas, tentunya diawali harusnya ada mekanisme *login* yang kemudian Yii mengirimkan *token*, atau bisa juga *token* didapat secara manual sebagaimana yang dilakukan RajaOngkir. Kemudian, *token* itulah yang akan menjadi penanda bahwa pengguna *web service* tersebut memang berhak.

Caranya, tambahkan fungsi *behavior* pada *controller*

```
public function behaviors() {
 $behaviors = parent::behaviors();
 $behaviors['authenticator'] = [
 'class' => yii\filters\auth\QueryParamAuth::className(),
];
 return $behaviors;
}
```

Apabila kita uji coba maka RESTful kita akan menganggap bahwa *request* kita *invalid credential* atau *Unauthorized*. Nah oleh karena itu kita perlu tambahkan *token/API Key* pada setiap *request*, namun apa tokennya?

Tentang token ini kita bisa terapkan di Yii dengan mengimplementasikan fungsi `findIdentityByAccessToken` pada class model `app\models\User`.

```
public static function findIdentityByAccessToken($token, $type = null)
{
 //throw new NotSupportedException('"findIdentityByAccessToken" is not implemented');
 return static::findOne(['auth_key' => $token, 'status' => self::STATUS_ACTIVE]);
}
```

Ketika kita menerapkan `class [[yii\filters\auth\QueryParamAuth]]` pada fungsi *behavior* di *controller* REST kita maka secara *default controller* tersebut akan mengecek apakah *request* tersebut diperbolehkan dengan cara mengecek melalui

fungsi `findIdentityByAccessToken` di model `User`. Untuk lebih jelasnya marilah kita mengintip `class [[yii\filters\auth\QueryParamAuth]]`

```
class QueryParamAuth extends AuthMethod
{
 public $tokenParam = 'access-token';

 public function authenticate($user, $request, $response)
 {
 $accessToken = $request->get($this->tokenParam);
 if (is_string($accessToken)) {
 $identity = $user->loginByAccessToken($accessToken, get_class($this));
 if ($identity !== null) {
 return $identity;
 }
 }
 if ($accessToken !== null) {
 $this->handleFailure($response);
 }

 return null;
 }
}
```

Fungsi `loginByAccessToken()` bisa kita lihat pada `class [[yii\web\User]]`,

```
public function loginByAccessToken($token, $type = null)
{
 /* @var $class IdentityInterface */
 $class = $this->identityClass;
 $identity = $class::findIdentityByAccessToken($token, $type);
 if ($identity && $this->login($identity)) {
 return $identity;
 } else {
 return null;
 }
}
```

Yaitu mengakses fungsi `findIdentityByAccessToken`. Oleh karena itulah kita memodifikasi fungsi tersebut. Adapun maksud dari kode berikut

```
| return static::findOne(['auth_key' => $token, 'status' => self::STATUS_ACTIVE]);
```

adalah untuk mengecek adakah pengguna yang memiliki status *active* dan `auth_key` sama dengan `token` dari `request` yang dikirimkan oleh pengguna. Jika terpenuhi atau ada maka request akan diteruskan ke fungsi `action` yang di-request, jika tidak maka request ditolak. Ini hanya contoh, kita bisa menggunakan mekanisme lain misalnya membuat *file* baru pada tabel `user` dsb.

Untuk mengujinya, kita perlu lihat terlebih dulu `auth_key` dari salah satu pengguna, misal username admin dengan `auth_key` 123456

<b>id</b>	<b>username</b>	<b>auth_key</b>	<b>password_hash</b>
1	admin	123456	\$2y\$13\$qsMMC1phi6LureFFqfx
2	test	o34ZhOdP4RTZPw5wPKli4BshBln9u0qe	\$2y\$13\$bAbAP3glFHy4oFjylE6
3	hafid_jmbr@yahoo.com	RPwaSE4b4xN7qp10XbDLdiFSunCxZrt2	\$2y\$13\$Elr9AeN8qB7GtoZHJY0
4	hafid	1x1WuicJze_w3XcyZ3BGAYNAM7Whqh-L	\$2y\$13\$nPC.x3mYVKI2KCrDpa

Berbekal informasi ini mari kita akses lagi menggunakan URL yang sama dengan sebelumnya namun dengan tambahan parameter `access-token`

```

{
 "results": [
 {"id": 1, "name": "Bali"}, {"id": 2, "name": "Bangka Belitung"}, {"id": 3, "name": "Banten"}, {"id": 4, "name": "Bengkulu"}, {"id": 5, "name": "DI Yogyakarta"}, {"id": 6, "name": "Jambi"}, {"id": 7, "name": "Gorontalo"}, {"id": 8, "name": "Jambi"}, {"id": 9, "name": "Jawa Barat"}, {"id": 10, "name": "Jawa Tengah"}, {"id": 11, "name": "Jawa Timur"}, {"id": 12, "name": "Kalimantan Barat"}, {"id": 13, "name": "Kalimantan Selatan"}, {"id": 14, "name": "Kalimantan Tengah"}, {"id": 15, "name": "Kalimantan Timur"}, {"id": 16, "name": "Kepulauan Riau"}, {"id": 17, "name": "Lampung"}, {"id": 18, "name": "Nusa Tenggara Barat"}, {"id": 19, "name": "Nusa Tenggara Timur"}, {"id": 20, "name": "Papua"}, {"id": 21, "name": "Sulawesi Barat"}, {"id": 22, "name": "Sulawesi Selatan"}, {"id": 23, "name": "Sulawesi Tengah"}, {"id": 24, "name": "Sulawesi Tenggara"}, {"id": 25, "name": "Sulawesi Utara"}, {"id": 26, "name": "Sumatera Barat"}, {"id": 27, "name": "Sumatera Selatan"}, {"id": 28, "name": "Sumatera Tengah"}, {"id": 29, "name": "Sumatera Utara"}
]
}

```

{"results": [{"id": 1, "name": "Bali"}, {"id": 2, "name": "Bangka Belitung"}, {"id": 3, "name": "Banten"}, {"id": 4, "name": "Bengkulu"}, {"id": 5, "name": "DI Yogyakarta"}, {"id": 6, "name": "Jambi"}, {"id": 7, "name": "Gorontalo"}, {"id": 8, "name": "Jambi"}, {"id": 9, "name": "Jawa Barat"}, {"id": 10, "name": "Jawa Tengah"}, {"id": 11, "name": "Jawa Timur"}, {"id": 12, "name": "Kalimantan Barat"}, {"id": 13, "name": "Kalimantan Selatan"}, {"id": 14, "name": "Kalimantan Tengah"}, {"id": 15, "name": "Kepulauan Riau"}, {"id": 16, "name": "Lampung"}, {"id": 17, "name": "Nusa Tenggara Barat"}, {"id": 18, "name": "Nusa Tenggara Timur"}, {"id": 19, "name": "Papua"}, {"id": 20, "name": "Sulawesi Barat"}, {"id": 21, "name": "Sulawesi Selatan"}, {"id": 22, "name": "Sulawesi Tengah"}, {"id": 23, "name": "Sulawesi Tenggara"}, {"id": 24, "name": "Sulawesi Utara"}, {"id": 25, "name": "Sumatera Barat"}, {"id": 26, "name": "Sumatera Selatan"}, {"id": 27, "name": "Sumatera Tengah"}, {"id": 28, "name": "Sumatera Utara"}]}

Sukses.

Kita juga bisa mengubah parameter access-token melalui variabel \$tokenParam, @app/controller/ProvinceRestController.php

```

public function behaviors() {
 $behaviors = parent::behaviors();
 $behaviors['authenticator'] = [
 'class' => \yii\filters\auth\QueryParamAuth::className(),
 'tokenParam'=>'key',
];
 return $behaviors;
}

```

Sekarang kita bisa mengaksesnya sebagai berikut

```
∞ localhost/basic/web/province-rest?key=123456
```

Tapi kalau kita ingat bahwa *service* RajaOngkir tidak menggunakan URL untuk mengirimkan *token*, melainkan menggunakan HTTP header. Nah untuk itu kita pun juga bisa membuatnya.

Caranya yaitu dengan membuat class baru yang kodennya mirip dengan *class* QueryParamAuth, pada contoh ini kita buat *class* baru bernama CustomAuth yang kita letakkan di @app/components/CustomAuth.php

```

<?php
namespace app\components;
class CustomAuth extends \yii\filters\auth\AuthMethod
{
 public $tokenParam = 'access-token';

 public function authenticate($user, $request, $response)
 {
 $accessToken = $request->getHeaders()->get($this->tokenParam);
 if (is_string($accessToken)) {
 $identity = $user->loginByAccessToken($accessToken, get_class($this));
 if ($identity !== null) {
 return $identity;
 }
 }
 if ($accessToken !== null) {
 $this->handleFailure($response);
 }

 return null;
 }
}

```

Kemudian pada ProvinceRestController kita ubah *behavior*-nya.

```

public function behaviors() {
 $behaviors = parent::behaviors();
 $behaviors['authenticator'] = [
 'class' => \app\components\CustomAuth::className(),
 'tokenParam'=>'key',
];
}

```

```
 return $behaviors;
```

Kemudian mari kita uji coba.

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** localhost/basic/web/province-rest
- Headers:** (1)
  - key:** 123456
- Status:** 200 OK
- Time:** 1287 ms
- Body:** (Pretty, Raw, Preview)

Wow berhasil lagi..

### *Rate Limiting*

Untuk mencegah terlalu banyaknya akses terhadap RESTful kita maka kita bisa membatasi maksimal akses per penggunanya. Implementasi *rate limiting* di Yii cukup mudah yaitu dengan menggunakan *interface class* [[yii\filters\RateLimitInterface]]. Interface ini membutuhkan implementasi dari tiga fungsi:

1. `getRateLimit()`: mengembalikan maksimal jumlah *request* yang dizinkan serta periode waktunya. Misal [100, 600] artinya 100 *request* dalam 600 detik.
  2. `loadAllowance()`: mengembalikan jumlah *request* sisa yang dizinkan dan UNIX *timestamp* terkait ketika *rate limit* terakhir dicek.
  3. `saveAllowance()`: menyimpan jumlah *request* sisa yang diizinkan dan UNIX *timestamp* saat ini.

Caranya sebagai berikut.

1. Buka `@app/models/User.php`, kemudian tambahkan implementasi dari `class [[yii\filters\RateLimitInterface]]`

```
<?php
namespace app\models;
class User extends \yii\base\Object implements \yii\web\IdentityInterface, \yii\filters\RateLimitInterface
{

```

2. Pada basis data tabel *user* tambahkan dua *field* yaitu *allowance\_updated\_at* dan *allowance* yang keduanya bertipe *integer*.
  3. Tambahkan tiga fungsi bawaan *interface* RateLimitInterface pada model *User*

```
public function getRateLimit($request, $action)
{
 // sesuaikan
 return [1,60]; // contoh maksimal 1 request per 60 detik
}

public function loadAllowance($request, $action)
{
 return [$this->allowance, $this->allowance_updated_at];
}

public function saveAllowance($request, $action, $allowance, $timestamp)
{
 $this->allowance = $allowance;
```

```

 $this->allowance_updated_at = $timestamp;
 $this->save();
 }
}

```

Selesai.

Sekarang waktunya uji coba, pada pengaksesan pertama data muncul sedangkan pada pengaksesan kedua yang dilakukan kurang dari 60 detik dari pengaksesan pertama maka muncul sebagai berikut

The screenshot shows a POST request to `localhost/basic/web/province-rest`. In the Headers tab, there is a key-value pair `key: 123456`. The response status is `429 Too Many Requests` with a time of `218 ms`. The response body is a JSON object:

```
{"name": "Too Many Requests", "message": "Rate limit exceeded.", "code": 0, "status": 429, "type": "yii\\web\\TooManyRequestsHttpException"}
```

Yaitu *request* ditolak karena terlalu banyak *request*.

Kita juga bisa mengimplementasikan *rate limiter* menggunakan penyimpanan lain seperti *cookie*, *local storage*, *session*, *redis*, dll sehingga tidak perlu menambahkan *field* pada tabel *User*.

Berikut ini contoh implementasi menggunakan penyimpanan *cookie* yaitu dengan memodifikasi fungsi `loadAllowance` dan `saveAllowance` pada model *User*.

```

public function loadAllowance($request, $action)
{
 return [$_COOKIE['allowance'], $_COOKIE['allowance_updated_at']];
}

public function saveAllowance($request, $action, $allowance, $timestamp)
{
 setcookie('allowance', $allowance, time() + (86400 * 30), '/');
 setcookie('allowance_updated_at', $timestamp, time() + (86400 * 30), '/');
}

```

Lebih dalam tentang RESTful, silakan mempelajari dari sumber utama

∞ <http://www.yiiframework.com/doc-2.0/guide-rest-quick-start.html>

## D. Grabing Situs Web

### D. 1. Definisi

*Grabing* situs web adalah teknik untuk mendapatkan konten dari suatu halaman situs web. Sebenarnya topik ini sudah diluar topik tentang *web service* namun penulis memasukkan di sini karena masih ada sedikit kaitan antara keduanya ☺.

Pada contoh ini, penulis akan mencoba untuk meng-*grab* data kurs mata uang asing dari situs web Bank Indonesia. Mengapa harus di-*grab*? Karena tidak ada yang menyediakan data dalam bentuk *web service*. Adapun data kurs mata uang yang kita dapat dari tautan berikut adalah data yang mutakhir.

∞ <http://www.bi.go.id/id/moneter/informasi-kurs/transaksi-bi/Default.aspx>

Data sumber berupa tabel HTML, tantangan kita adalah bagaimana mem-*parsing* tabel tersebut ke dalam bentuk yang mudah dibaca sistem, misalnya *array* atau *JSON*.

Mata Uang	Nilai	Kurs Jual	Kurs Beli
AUD	1.00	9,730.82	9,627.69
BND	1.00	9,705.13	9,607.15
CAD	1.00	9,715.94	9,616.51
CHF	1.00	13,891.10	13,740.06
CNY	1.00	2,126.09	2,105.06
DKK	1.00	2,033.34	2,012.94

Data ini nantinya akan kita gunakan untuk transaksi pembayaran menggunakan Paypal. Hal ini dikarenakan Paypal tidak mendukung mata uang rupiah ☹, sehingga kita harus mengkonversinya ke *dollar* Amerika (USD).

Apakah ini legal? menurut apa yang penulis baca, bahwa situs web BI mengizinkan kita mengambil data yang memang ditampilkan pada situs web, adapun caranya tidak disebutkan. Namun yang lebih penting adalah bukan legal atau tidaknya, namun dari sisi teknis atau ilmiah hal ini dimungkinkan.

## D. 2. Implementasi

Untuk melakukannya, PHP menyediakan fungsi CURL atau di Yii kita bisa menggunakan *extension* yang menyediakan HTTP *client* sebagaimana yang telah kita gunakan sebelumnya

Setelah itu kita bisa uji coba misalnya kita buat GrabController

```
@app/controllers/GrabController.php

<?php
namespace app\controllers;
use Yii;
use yii\web\Controller;
use yii\httpClient\Client;

class GrabController extends Controller

 public function actionKurs()
 {
 $kurs = $this->getKurs('USD');
 print_r($kurs);
 }

 public function getKurs($currencyId=' ')
 {
 $client = new Client();
 $response = $client->createRequest()
 ->setMethod('get')
 ->setUrl('http://www.bi.go.id/id/moneter/informasi-kurs/transaksi-
bi/Default.aspx')
 ->send();
 if ($response->isOk) {
 preg_match_all('#<table[^>]+>(.+?)</table>#ims', $response-
>content, $matches);
 $table = $matches[0][3];
 preg_match_all('#<tr>(.+?)</tr>#ims', $table, $matches2);
 foreach ($matches2[0] as $key => $value) {
 preg_match_all('#<td[^>]+>(.+?)</td>#ims', $value, $matches3);
 if(isset($matches3[0][0])){
 $free = trim(strip_tags($value));
 $currency = substr($free, 0, 3);
 $sale = $matches3[0][1];
 $buy = $matches3[0][2];
 if($currencyId==' ' or $currencyId==$currency) {
 $kurs[$currency] = [
 'sale' => $sale,
 'buy' => $buy,
];
 }
 }
 }
 }
 }
}
```

```
 return $kurs;
 }
}
```

Ada dua bagian penting pada kode di atas,

1. Kode berikut berfungsi me-*request* atau mendapatkan halaman web

<http://www.bi.go.id/id/moneter/informasi-kurs/transaksi-bi/Default.aspx>

```
$client = new Client();
$response = $client->createRequest()
 ->setFormat(Client::FORMAT_JSON)
 ->setMethod('get')
 ->setUrl('http://www.bi.go.id/moneter/informasi-kurs/transaksi-
bi/Default.aspx')
 ->send();
if ($response->isOk) {
```

2. Bagian kedua adalah kode berikut.

```

preg_match_all('#<table[^>]+>(.+?)</table>#ims', $response->content, $matches);
$table = $matches[0][3];
preg_match_all('#<tr>(.+?)</tr>#ims', $table, $matches2);
foreach ($matches2[0] as $key => $value) {
 preg_match_all('#<td[^>]+>(.+?)</td>#ims', $value, $matches3);
 if(isset($matches3[0][0])){
 $free = trim(strip_tags($value));
 $currency = substr($free, 0, 3);
 $sale = $matches3[0][1];
 $buy = $matches3[0][2];
 if($currencyId=='' or $currencyId==$currency){
 $kurs[$currency] = [
 'sale' => $sale,
 'buy' => $buy,
];
 }
 }
}

```

Kode di atas berfungsi mem-parsing halaman HTML menggunakan fungsi regex, preg\_match dan preg\_match\_all. Tujuannya untuk mendapatkan tiga data penting yaitu mata uang, harga jual, dan harga beli.

Saatnya uji coba, silakan akses URL

∞ <http://localhost/basic/web/grab/kurs>

Berikut ini hasilnya

```
| Array ([USD] => Array ([sale] => 13,955.00 [buy] => 13,817.00))
```

Dari data tersebut kita bisa gunakan data *sale*. Jadi misalnya harga produk Rp. 100.000,- maka untuk mengkonversi ke mata uang *dollars* cukup dengan membaginya dengan data *sale*. Hasilnya USD 7.17.

## E. Kesimpulan

*Web service* adalah aplikasi web yang memberikan *service* kepada aplikasi lain. Yii melakukan pendekatan yang memudahkan pemrogram aplikasi untuk membuat serta menggunakan *web service*. Jenis *web service* yang didukung oleh Yii adalah RESTful.

Pada bab selanjutnya, kita akan belajar tentang beragam topik yang mungkin sering anda jumpai di dunia nyata. Sengaja digabung dalam satu bab karena hanya merupakan tips dan trik saja.

# 15 Tips dan Trik

**Pada bab ini kita akan belajar tentang:**

- Tips dan trik Yii
- Topik-topik aktual yang belum dibahas pada bab sebelumnya

## A. Kirim E-Mail

Yii menggunakan *library* SwiftMailer, dimana telah dibuatkan *official extension*-nya yaitu yii2-swiftmailer, untuk menangani pengiriman *email*. Extension ini secara *default* telah disertakan saat kita melakukan instalasi Yii.

Pada @app/config/web.php, kita juga telah menjumpai konfigurasinya.

```
'components' => [
 ...
 'mailer' => [
 'class' => 'yii\swiftmailer\Mailer',
 //send all mails to a file by default. You have to set
 //'useFileTransport' to false and configure a transport
 //for the mailer to send real emails.
 'useFileTransport' => true,
],
],
```

Parameter *useFileTransport* yang bernilai *true* akan mengirimkan *email* ke @app/runtime/mail, hal ini hanya untuk uji coba.

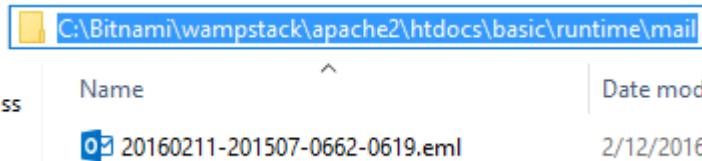
Contoh, pada *controller*, kita bisa tuliskan kode untuk uji coba mengirim *email* sebagai berikut.

```
public function actionEmail()
{
 Yii::$app->mailer->compose()
 ->setFrom('milisstudio@gmail.com')
 ->setTo('hafidmukhsin@gmail.com')
 ->setSubject('Uji coba aja')
 ->setTextBody('Teks yang tampil di body')
 ->setHtmlBody('Contoh text HTML')
 ->send();
}
```

Jika kita akses *actionEmail*

```
∞ http://localhost/basic/web/site/email
```

Maka perintah di atas akan mengirimkan *email* ke direktori @app/runtime/mail



Berikut ini jika dibuka menggunakan *software mail client* bawaan Windows.



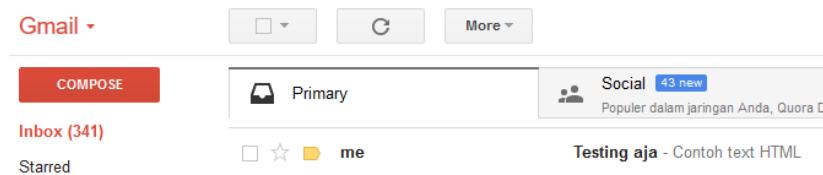
### Contoh text HTML

Untuk mengirimkan *email* ke alamat *email* nyata di internet, maka kita perlu setting *parameter useFileTransport menjadi false* dan mendefinisikan SMTP yang digunakan.

Pada contoh ini kita akan menggunakan SMTP Google, maka konfigurasinya sebagai berikut.

```
'mailer' => [
 'class' => 'yii\swiftmailer\Mailer',
 'useFileTransport' => false,
 'transport'=>[
 'class'=>'Swift_SmtpTransport',
 'host'=>'smtp.gmail.com',
 'username'=>'yourgmail@gmail.com',
 'password'=>'yourpassword',
 'port'=>'587',
 'encryption'=>'tls',
],
],
```

Ganti *yourgmail* dan *yourpassword* dengan *username* dan kata sandi Gmail anda. Setelah itu, silakan uji coba, lalu cek *email*.



Jika tidak terkirim, maka pastikan anda menkonfigurasi pengaturan Gmail pada *mode less secure apps* melalui tautan berikut.

∞ <https://www.google.com/settings/security/lesssecureapps>

Dan tentunya komputer anda harus terkoneksi internet.

## B. Tabular Input

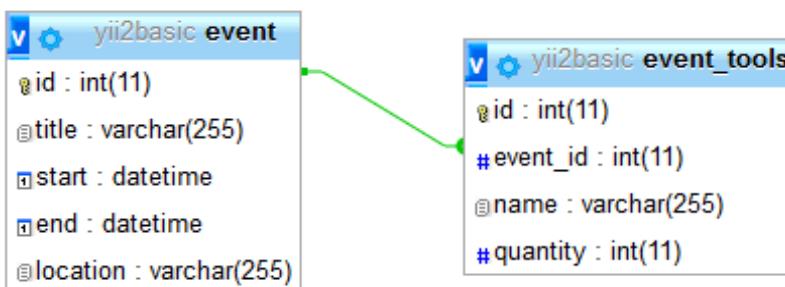
Untuk menangani *tabular input* atau *dynamic form field*, lebih mudahnya adalah dengan menggunakan *extension*. Salah satu *extension* yang menurut penulis cukup bagus adalah *wbraganca/yii2-dynamicform*.

∞ <https://github.com/wbraganca/yii2-dynamicform>

Berikut, cara instalasinya menggunakan Composer.

```
| composer require --prefer-dist wbraganca/yii2-dynamicform ***
```

Sebagai contoh kasus, misalnya kita ingin membuat formulir *input* data nama-nama alat/*tools* yang harus disiapkan pada sebuah kegiatan/*event*.



Relasi tabel *event* ke tabel *event\_tools* adalah *one to many*, atau satu *event* bisa memiliki lebih dari satu *event\_tools*. Hendaknya *constraint* relasinya *cascade on delete & update* sehingga ketika tabel *event* dihapus maka semua akan terhapus.

*Foreign key* (*event\_id*) hendaknya diset *NULL* supaya ketika *validate* tidak gagal. Pastikan *field primary key* menggunakan nama ‘*id*’ bukan ID atau yang lain, supaya kita tidak perlu modifikasi modelnya.

Name	Type	Collation	Attributes	Null	Default	Extra
<b>id</b>	int(11)			No	None	AUTO_INCREMENT
<b>event_id</b>	int(11)			Yes	NULL	
<b>name</b>	varchar(255)	utf8_general_ci		Yes	NULL	
<b>quantity</b>	int(11)			Yes	NULL	

Pertama yang harus kita lakukan adalah meng-*generate* model untuk dua tabel di atas menggunakan *tools Gii*.

## Model Generator

This generator generates an Active Record class for the specified table.

**Table Name**: event

**Model Class**: Event

**Namespace**: app\models

**Base Class**: CActiveRecord

**Model Generator**

This generator generates an Active Record class for the specified table.

**Table Name**: event\_tools

**Model Class**: EventTools

**Namespace**: app\models

Lalu kemudian generate CRUD untuk *model event* saja.

## CRUD Generator

This generator generates a controller and view for the specified model.

**Model Class**: app\models\Event

**Search Model Class**: app\models\EventSearch

**Controller Class**: app\controllers\EventController

Lakukan sedikit perubahan yaitu mengganti *field input start & end* menjadi datepicker menggunakan *widget kartik datepicker*. Widget ini sudah terinstalasi ketika anda melakukan instalasi yii2-mimin.

```
@app\views\event_form.php

<?php $form = ActiveForm::begin(['id' => 'dynamic-form']); ?>

...
<div class="row">
 <div class="col-md-4">
 <?= $form->field($model, 'start') ->widget(\kartik\widgets\DatePicker::classname(), [
 'options' => ['placeholder' => 'Enter start event...'],
 'pluginOptions' => [
 'format' => 'dd-mm-yyyy',
 'language' => 'en',
],
]);
 </div>
</div>
```

```

 'removeButton' => false,
 'pluginOptions' => [
 'autoclose'=>true,
 'format' => 'yyyy-mm-dd',
]
]) ?>
</div>
<div class="col-md-4">
<?= $form->field($model, 'end')-
>widget(\kartik\widgets\DatePicker::classname(), [
 'options' => ['placeholder' => 'Enter end event...'],
 'removeButton' => false,
 'pluginOptions' => [
 'autoclose'=>true,
 'format' => 'yyyy-mm-dd',
]
]) ?>
</div>
</div>
...

```

Tambahkan id formulir menjadi dynamic-form yang akan kita pergunakan nanti.

Mari kita coba uji coba untuk *create event*

∞ <http://localhost/basic/web/event/create>

## Create Event

**Title**

**Start**

 Enter start event...

**End**

 Enter end event...

**Location**

**Create**

Nah pada view formulir *create / update* inilah kita akan menambahkan formulir untuk *input tools* yang diperlukan dengan model *input* yang dinamis.

Namun sebelum itu kita perlu membuat sebuah model untuk menangani formulir *tabular* ini. @app/models/Tabular.php

```

<?php
namespace app\models;

use Yii;
use yii\helpers\ArrayHelper;

class Tabular extends \yii\base\Model
{
 public static function createMultiple($modelClass, $multipleModels = [])
 {
 $model = new $modelClass;
 $formName = $model->formName();
 $post = Yii::$app->request->post($formName);
 $models = [];

 if (! empty($multipleModels)) {
 $keys = array_keys(ArrayHelper::map($multipleModels, 'id', 'id'));
 $multipleModels = array_combine($keys, $multipleModels);
 }

 if ($post && is_array($post)) {
 foreach ($post as $i => $item) {
 if (isset($item['id']) && !empty($item['id']) && isset($multipleModel
s[$item['id']])) {
 $models[] = $multipleModels[$item['id']];
 } else {

```

```

 $models[] = new $modelClass;
 }
}

unset($model, $formName, $post);
return $models;
}
}

```

Kode di atas digunakan untuk meng-create class model dalam bentuk *array* dan membaca data formulir dalam bentuk *array* juga.

Sebelum menambahkan formulir untuk *input tools* pada *form event*, kita perlu sesuaikan dulu fungsi *actionCreate* pada *EventController*.

`@app/controllers/EventController.php`

```

public function actionCreate()
{
 $model = new Event();
 $modelsTools = [new EventTools];

 if ($model->load(Yii::$app->request->post())) {
 // load data from form submit
 $modelsTools = Tabular::createMultiple(EventTools::classname());
 Tabular::loadMultiple($modelsTools, Yii::$app->request->post());
 // validasi data
 $valid = $model->validate();
 $valid = Tabular::validateMultiple($modelsTools) && $valid;
 if ($valid) {
 $transaction = Yii::$app->db->beginTransaction();
 try {
 if ($flag = $model->save(false)) {
 foreach ($modelsTools as $indexTools => $modelTools) {
 if ($flag === false) {
 break;
 }
 $modelTools->event_id = $model->id;
 if (!$flag = $modelTools->save(false)) {
 break;
 }
 }
 }
 }
 if ($flag) {
 $transaction->commit();
 \Yii::$app->session->setFlash('success', 'Input data sukses');
 return $this->redirect(['view', 'id' => $model->id]);
 } else {
 $transaction->rollBack();
 \Yii::$app->session->setFlash('error', 'Input data gagal');
 }
 } catch (\Exception $e) {
 $transaction->rollBack();
 \Yii::$app->session->setFlash('error', 'Input data gagal');
 }
 }
} else {
 return $this->render('create', [
 'model' => $model,
 'modelsTools' => (empty($modelsTools)) ? [new EventTools] : $modelsTools,
]);
}
}

```

Jangan lupa untuk meng-use dua models

```

use app\models\EventTools;
use app\models\Tabular;

```

Perhatikan baik-baik kode pada *actionCreate* di atas. Kode yang paling utama adalah pada kode berikut.

```
$modelsTools = Tabular::createMultiple(EventTools::classname());
Tabular::loadMultiple($modelsTools, Yii::$app->request->post());
```

Kode tersebut merupakan mekanisme untuk menangkap nilai dari data formulir yang berformat *array* ke dalam model *EventTools* dengan menggunakan model *Tabular*.

Karena actionCreate me-render view *create.php*, maka kita perlu tambahkan *parameter* \$modelsTools pada *render* di *view* *create.php* tersebut

```
<?= $this->render('_form', [
 'model' => $model,
 'modelsTools' => $modelsTools,
]) ?>
```

Kemudian kita tambahkan *widget* *DynamicFormWidget* bawaan *extension* tersebut pada *view* *@app/views/event/\_form.php*

```
<?php
...
use wbraganca\dynamicform\DynamicFormWidget;
?>
<div class="event-form">
 <?php $form = ActiveForm::begin(['id' => 'dynamic-form']); ?>
 <?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>
 ...
 <div class="padding-v-md">
 <div class="line line-dashed"></div>
 </div>
 <?php DynamicFormWidget::begin([
 'widgetContainer' => 'dynamicform_wrapper',
 'widgetBody' => '.container-items',
 'widgetItem' => '.tools-item',
 'limit' => 10,
 'min' => 1,
 'insertButton' => '.add-tools',
 'deleteButton' => '.remove-tools',
 'model' => $modelsTools[0],
 'formId' => 'dynamic-form',
 'formFields' => [
 'name',
 'quantity',
],
]); ?>
 <table class="table table-bordered table-striped">
 <thead>
 <tr>
 <th>Name</th>
 <th>Quantity</th>
 <th class="text-center" style="width: 90px;">
 <button type="button" class="add-tools btn btn-success btn-xs"></button>
 </th>
 </tr>
 </thead>
 <tbody class="container-items">
 <?php foreach ($modelsTools as $indexTools => $modelTools) : ?>
 <tr class="tools-item">
 <td class="vcenter">
 <?= $form->field($modelTools, "[{$indexTools}]name") ->label(false)->textInput(['maxlength' => true]) ?>
 </td>
 <td class="vcenter">
 <?= $form->field($modelTools, "[{$indexTools}]quantity") ->label(false)->textInput() ?>
 </td>
 <td class="text-center vcenter" style="width: 90px; vertical-align: middle;">
 <button type="button" class="remove-tools btn btn-danger btn-xs"></button>
 </td>
 </tr>
 <?php endforeach; ?>
 </tbody>
```

```

 </table>
 <?php DynamicFormWidget::end(); ?>
 <div class="form-group">
 <?= Html::submitButton($model->isNewRecord ? 'Create' : 'Update', ['class' => $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
 </div>
 <?php ActiveForm::end(); ?>
 </div>

```

Mari kita cek lagi dan coba lagi.

**Title**

**Start**      **End**

	2016-02-12		2016-02-12 00:00
--	------------	--	------------------

**Location**

Name	Quantity	Action
Notebook	3	
Printer	1	

**Update**

Kita bisa tambahkan *tools* baru dengan menekan tombol + di tabel pojok kanan atau tombol – untuk menghapus *tools*.

**Input data sukses**

## Workshop Yii Framework Batch I

**Update** **Delete**

ID	1
...	...

Cek di basis data tabel event\_tools, dan data masuk.

event_id	name	quantity
1	Laptop	3
1	Printer	1

Oke, untuk actionCreate selesai.

Kemudian kita perlu sesuaikan untuk actionUpdate.

```

public function actionUpdate($id)
{
 $model = $this->findModel($id);
 $modelsTools = $model->eventTools;
 if ($model->load(Yii::$app->request->post())) {
 $oldToolsIDs = ArrayHelper::map($modelsTools, 'id', 'id');
 $modelsTools = Tabular::createMultiple(EventTools::classname(), $modelsTools);
 }
 Tabular::loadMultiple($modelsTools, Yii::$app->request->post());
 $deletedToolsIDs = array_diff($oldToolsIDs, array_filter(ArrayHelper::map($modelsTools, 'id', 'id')));
 // validate models
 $valid = $model->validate();
 $valid = Tabular::validateMultiple($modelsTools) && $valid;
 if ($valid) {
 $transaction = Yii::$app->db->beginTransaction();

```

```

try {
 if ($flag = $model->save(false)) {
 if (! empty($deletedToolsIDs)) {
 EventTools::deleteAll(['id' => $deletedToolsIDs]);
 }

 foreach ($modelsTools as $indexTools => $modelTools) {
 if ($flag === false) {
 break;
 }
 $modelTools->event_id = $model->id;
 if (!($flag = $modelTools->save(false))) {
 break;
 }
 }
 }
 if ($flag) {
 $transaction->commit();
 \Yii::$app->session->setFlash('success', 'Update data sukses');
 return $this->redirect(['view', 'id' => $model->id]);
 } else {
 \Yii::$app->session->setFlash('error', 'Update data gagal');
 $transaction->rollBack();
 }
} catch (\Exception $e) {
 \Yii::$app->session->setFlash('error', 'Update data gagal');
 $transaction->rollBack();
}
} else {
 return $this->render('update', [
 'model' => $model,
 'modelsTools' => (empty($modelsTools)) ? [new EventTools] : $modelsTools,
]);
}
}
}

```

Karena menggunakan *class ArrayHelper*, maka kita perlu *use* dulu

```
| use yii\helpers\ArrayHelper;
```

Lalu pada *view update* juga perlu kita edit bagian *render*-nya.

```
@app/views/event/update.php
```

```
<?= $this->render('_form', [
 'model' => $model,
 'modelsTools' => $modelsTools,
]) ?>
```

Mari kita *test*

```
∞ http://localhost/basic/web/event/update?id=1
```

Title

Start

End

Location

Name	Quantity	
Notebook	3	
Printer	1	

**Update**

Kita juga bisa sesuaikan untuk fungsi actionView agar bisa menampilkan data *tools*-nya.

```
public function actionView($id)
{
 $model = $this->findModel($id);
 $modelsTools = $model->eventTools;

 return $this->render('view', [
 'model' => $model,
 'modelsTools' => (empty($modelsTools)) ? [new EventTools] : $modelsTools,
]);
}
```

Lalu pada *view* @app/views/event/view.php

```
<h1>Tools</h1>
<table class="table table-bordered table-striped">
 <thead>
 <tr>
 <th>Name</th>
 <th>Quantity</th>
 </tr>
 </thead>
 <tbody>
 <?php foreach ($modelsTools as $indexTools => $modelTools): ?>
 <tr>
 <td><?= $modelTools->name ?></td>
 <td><?= $modelTools->quantity ?></td>
 </tr>
 <?php endforeach; ?>
 </tbody>
</table>
```

Sehingga hasilnya, tabel daftar *tools* akan muncul di bawah tabel *detail event*.

		<a href="#">Update</a>	<a href="#">Delete</a>
<b>ID</b>	1		
<b>Title</b>	Workshop Yii Framework Batch I		
<b>Start</b>	2016-02-12 00:00:00		
<b>End</b>	2016-02-12 00:00:00		
<b>Location</b>	Jakarta		

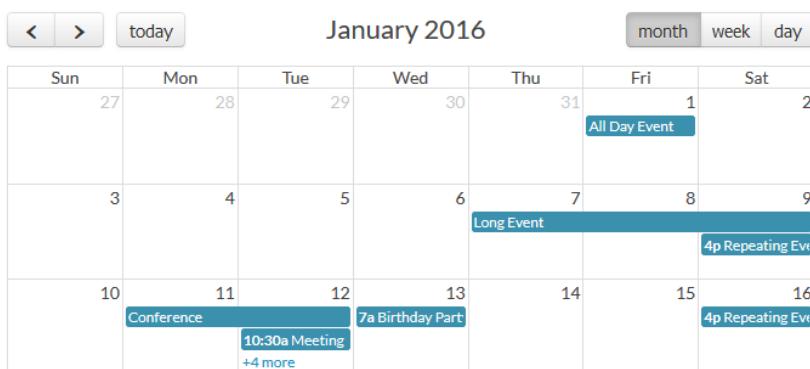
## Tools

Name	Quantity
Notebook	3
Printer	1

Silakan dikembangkan.

## C. Event Calendar

*Event Calendar* adalah tampilan dalam bentuk kalender yang berisi kegiatan tertentu, biasanya untuk penjadwalan. Salah satu *library* yang cukup populer untuk menangani hal ini adalah <http://fullcalendar.io/>



Ada beberapa *extension* Yii yang mem-wrap *library* ini, salah satunya adalah <https://github.com/hscstudio/yii2-fullcalendar>

```
| composer require -prefer-dist hscstudio/yii2-fullcalendar "*",
```

Setelah melakukan instalasi menggunakan *Composer*, kita akan mulai dengan menggunakan studi kasus tabel *event* yang telah kita buat sebelumnya.

Pada `app\controllers\EventController`, kita tambahkan `actionCalendar` untuk menampilkan tampilan *calendar*.

```
public function actionCalendar()
{
 return $this->render('calendar');
}
```

Buat `view calendar.php` pada direktori `@app/views/event/calendar.php`

```
<?php
use yii\helpers\Html;
use yii\helpers\Url;

$this->title = 'Event Calendar';
$this->params['breadcrumbs'][] = ['label' => 'Events', 'url' => ['index']];
?>
<div>
 <h1><?= Html::encode($this->title) ?></h1>
 <?php $eventUrl = Url::to(['event-calendar']); ?>
 <?= hscstudio\calendar\FullCalendar::widget([
 'options'=>[
 'id'=>'calendar',
 'header'=>[
 'left'=>'prev,next today',
 'center'=>'title',
 'right'=>'month,agendaWeek,agendaDay',
],
 'editable'=> true,
 'eventLimit'=>true, // allow "more" link when too many events
 'events' => [
 'url' => $eventUrl,
],
],
]) ?>
</div>
```

Untuk data *event*, kita perlu buatkan fungsi sesuai URL yang ditunjuk oleh `events-url` yaitu fungsi `actionEventCalendar` pada `EventController`

```
public function actionEventCalendar($start=NULL,$end=NULL,$_=NULL) {
 \Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;

 $model= \app\models\Event::find()->all();
 if(!empty($start) and !empty($end)) {
 $model= \app\models\Event::find()
 ->where(['>=' , 'start',date('Y-m-d 00:00:01',strtotime($start))])
 ->andWhere(['<=' , 'end',date('Y-m-d 23:59:59',strtotime($end))])
 ->all();
 }

 $events = [];
}
```

```

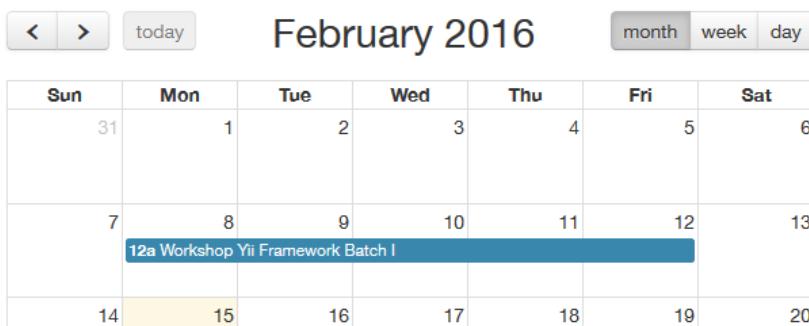
foreach ($model as $event) {
 $events[]=[
 'title'=>$event->title,
 'start'=>date('Y-m-d 00:00:01', strtotime($event->start)),
 'end'=>date('Y-m-d 23:59:59', strtotime($event->end)),
 //'color'=>'#CC0000',
 //'allDay'=>true,
 //'url'=>'http://anyurl.com'
];
}
return $events;
}

```

Mari kita *test*

∞ <http://localhost/basic/web/event/calendar>

## Event Calendar



Selesai.

## D. Mode Maintenance / Luring

Adakalanya situs web kita sedang proses *maintenance*, sehingga kita ingin membuat situs web kita ber-*mode maintenance* / luring. Yii menyediakan fitur untuk membuat situs web kita ber-*mode maintenance*, yaitu dengan menggunakan parameter `catchAll`

```

'catchAll' => [
 'site/maintenance',
],

```

Parameter ini akan melewati semua *request* untuk diarahkan ke *route* yang didefinisikan tersebut atau dalam hal ini SiteController dengan fungsi actionMaintenance.

```

public function actionMaintenance()
{
 return $this->render('maintenance');
}

```

Tentu kita harus membuat *view* untuk tampilan halaman *maintenance* di `@app/views/site/maintenance`

```

<h1>The site is currently under maintenance</h1>
<p>We apologize for inconvenience. Please come back later.</p>

```

Ketika kita mengakses URL apapun maka akan menampilkan sebagai berikut

Testing Doang

## The site is currently under maintenance

We apologize for inconvenience. Please come back later.

Di samping itu, kita bisa membuatnya lebih dinamis, misalnya dengan menggunakan basis data. Sehingga *administrator* cukup melakukan konfigurasi melalui basis data, untuk membuat mode *maintenance* bekerja.

Mari kita buat basis data tabel *setting*, dengan tiga *field* yaitu

yii2basic setting		
name	value	note
name : varchar(100)		
value : varchar(255)		
note : varchar(255)		

Kemudian tambahkan isinya sebagai berikut

name	value	note
maintenance	yes	

Kita bisa manfaatkan *event on beforeRequest* atau *on beforeAction*

@app/config/web.php

```
'on beforeRequest' => function () {
 $db = Yii::$app->db;
 $maintenance = $db->createCommand("SELECT * FROM setting WHERE name='maintenance'")->queryOne();
 if($maintenance['value']=='yes'){
 Yii::$app->catchAll = ['site/maintenance'];
 }
},
```

Cukup dengan mengubah nilai dari *key maintenance* menjadi *yes* pada basis data tabel *setting*, maka situs web atau aplikasi akan ber-*mode maintenance*..

Supaya lebih rapi, kita juga bisa memindahkan kode di atas pada *bootstrap*.

## E. Manipulasi Image

Yii menyediakan *extension official* untuk memanipulasi *image* yaitu yii2-imagine (<https://github.com/yiisoft/yii2-imagine>) yang merupakan *wrapper* dari *library Imagine* (<http://imagine.readthedocs.org>)

Pada bab “Bekerja dengan Extension” kita telah belajar cara menginstalnya. Adapun *image* yang didukung adalah jpg, jpeg, gif, png, wbmp dan xbm

Fitur yang dimiliki oleh *Imagine* terbagi menjadi tiga.

- *Image manipulation* seperti *resize*, *crop*, dll.
- *Drawing API* – untuk membuat bentuk dasar dan *chart*, menulis teks pada *image*.
- Fungsi *masking* yaitu untuk membuat mengaplikasikan *image black&white* atau *grayscale* sebagai *mask*, dsb.

### E. 1. Thumbnail Image

Untuk mengubah ukuran dimensi suatu *image* atau untuk membuat *thumbnail*, maka kita bisa gunakan cara sebagai berikut.

```
use \yii\imagine\Image;
$image = Image::thumbnail('@app/uploads/btn1-120x90.png', 60, 45);
$image->save(Yii::getAlias('@app/uploads/thumb-btn1-120x90.png'), ['quality' => 50]);
```

### E. 2. Crop Image

Fungsinya untuk meng-*crop image*

```
use \yii\imagine\Image;
// crop ($filename,$width,$height, array $start = [0, 0])
$image = Image::crop('@app/uploads/btn1-120x90.png', 60, 45, [5,5]);
$image->save(Yii::getAlias('@app/uploads/crop-btn1-120x90.png'), ['quality' => 50]);
```



### E. 3. Draw Text

Fungsinya untuk menulis suatu teks ke *image*.

```
$font = Yii::getAlias('@vendor') . '/imagine/imagine/tests/Imagine/Fixtures/font/Arial.ttf';
$image = Image::text('@app/uploads/btn1-120x90.png', 'TESTING', $font, [5,5], [
 'size' => 12, 'color' => '000', 'angle' => 0
]);
$image->save(Yii::getAlias('@app/uploads/text-btn1-120x90.png'), ['quality' => 50]);
```

Kita bebas menentukan jenis *font* sendiri, sayangnya harus disebutkan *path* lengkapnya. Berikut ini contohnya.



### E. 4. Watermark

Fungsinya untuk memberikan *watermark* atau menumpuk dua *image*.

```
$base = '@app/uploads/med-rect-300x250.png';
$watermark = '@app/uploads/golang.png';
$image = Image::watermark($base, $watermark, [25,25]);
$image->save(Yii::getAlias('@app/uploads/watermark.png'), ['quality' => 50]);
```



### E. 5. Grayscale

Berfungsi untuk membuat warna menjadi *grayscale* (hitam putih)

```
$imagine = Image::getImagine();
$image = $imagine->open(Yii::getAlias('@app') . '/uploads/golang.png');
$image->effects()->grayscale();
$image->save(Yii::getAlias('@app/uploads/grayscale.png'), ['quality' => 50]);
```

## F. Internationalization

*Internationalization* (I18N) mengacu pada proses desain aplikasi yang dapat mengadaptasi berbagai bahasa dan negara tanpa membutuhkan perubahan *coding*. Untuk aplikasi berbasis web, fitur ini sangat penting, sebab pengunjung web bisa berasal dari berbagai bahasa dan negara.

Yii menyediakan fitur I18N seperti penerjemahan pesan, penerjemahan tampilan/*view*, format waktu dan nomor.

### F. 1. Pengaturan Bahasa

Yii mempunyai dua jenis pengaturan bahasa, yaitu *source language* dan *target language*. *Source language* adalah bahasa yang digunakan untuk menulis teks pada aplikasi, sedangkan *target language* adalah bahasa yang akan ditampilkan pada pengguna aplikasi.

Pengaturan bahasa bisa kita lakukan pada konfigurasi @app/config/web.php

```
return [
 // set target language to be Indonesia
 'language' => 'id-ID',
 // set source language to be English
 'sourceLanguage' => 'en-US',
```

```
|];
```

Secara *default*, bahasa sumber / sourceLanguage adalah bahasa Inggris Amerika Serikat, yaitu en-US. en mengacu pada bahasa (*English*), sedangkan US mengacu pada negara (*United States*). *language* adalah bahasa target, dimana id adalah bahasa Indonesia, dan ID adalah negara Indonesia. Aturan penulisan kode bahasa dan negara ini sesuai dengan standard di <http://www.loc.gov/standards/iso639-2/> untuk bahasa dan untuk negara bisa dilihat di sini <http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html>.

Apa efek dari pengaturan bahasa ini?

Efek yang langsung terlihat adalah pada pesan *default* aplikasi. Misalnya sebelum diset bahasanya, maka galat yang muncul sebagai berikut.

#### Title

Title cannot be blank.

Ketika telah diset maka akan berubah menjadi sebagai berikut.

#### Title

Title tidak boleh kosong.

#### Informasi

Yii mendukung penerjemahan pesan galat internal kedalam 26 bahasa dunia, termasuk Bahasa Indonesia

## F. 2. Penerjemahan Pesan

Penerjemahan pesan adalah proses menerjemahkan pesan teks pada aplikasi dari satu bahasa (biasanya bahasa sumber/sourceLanguage) ke bahasa lain (biasanya bahasa target/language). Fitur ini melakukan penerjemahan dengan membandingkan teks yang akan diterjemahkan apakah sesuai dengan “kamus” terjemahan. Jika teks terjemahan ditemukan, maka fitur ini mengembalikan hasil terjemahan, namun jika tidak ditemukan maka akan mengembalikan teks asli.

Berikut ini langkah-langkah untuk menggunakan fitur ini.

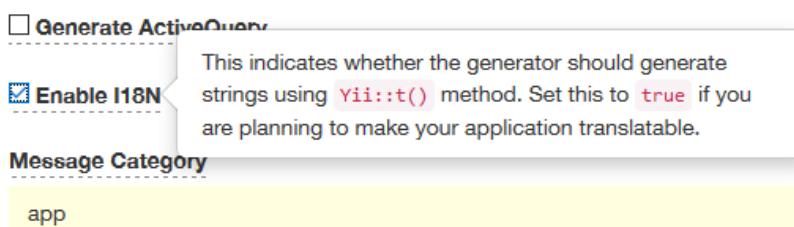
#### Fungsi Yii::t()

Pertama, kita harus membungkus semua teks yang perlu diterjemahkan menggunakan fungsi Yii::t(). Berikut ini contohnya.

```
| echo \Yii::t('app', 'This is a string to translate!');
```

Fungsi ini menggunakan dua parameter. Parameter pertama merupakan grup atau kategori pesan (pada contoh di atas adalah app), sedangkan parameter kedua adalah pesan teks yang akan diterjemahkan.

Tools Gii juga mendukung penggunaan fungsi ini, dengan cara meng-checklist Enable I18N.



Maka teks pada kode yang dihasilkan oleh Gii akan dibungkus dengan fungsi Yii::t()

#### Konfigurasi

Fungsi Yii::t() akan memanggil komponen i18n untuk melakukan proses penerjemahan. Komponen ini bisa dikonfigurasi pada @app/config/web.php

```
| 'components' => [
 // ...
 'i18n' => [
```

```
'translations' => [
 'app*' => [
 'class' => 'yii\\i18n\\PhpMessageSource',
 //'basePath' => '@app/messages',
 //'sourceLanguage' => 'en-US',
 'fileMap' => [
 'app' => 'app.php',
 'app/error' => 'error.php',
],
],
],
],
],
```

Format app\* menunjukkan bahwa semua kategori pesan yang namanya diawali dengan app harus diterjemahkan sesuai dengan konfigurasi tersebut. *Class* [[yii\i18n\PhpMessageSource]] menggunakan *file PHP* untuk menyimpan terjemahan pesan.

Secara *default*, nama *file* pesan harus sama dengan nama kategori. Namun, kita dapat mengkonfigurasinya melalui parameter `fileMap` untuk memetakan kategori untuk *file PHP*.

Pada contoh di atas, kategori app atau galat dipetakan ke file PHP @app/messages/id-ID/error.php (dengan asumsi id-ID adalah bahasa target / language). Tanpa konfigurasi ini, maka kategori akan dipetakan ke @app/messages/id-ID/app/error.php, sebagai gantinya.

Di samping menyimpan terjemahan pada file PHP, kita juga bisa menyimpannya pada basis data menggunakan class [[yii\i18n\DbMessageSource]]. Jika kita menggunakan basis data, maka kita harus menciptakan tabel *message* dengan format sebagai berikut.

```
CREATE TABLE source_message (
 id INTEGER PRIMARY KEY AUTO_INCREMENT,
 category VARCHAR(32),
 message TEXT
);

CREATE TABLE message (
 id INTEGER,
 language VARCHAR(16),
 translation TEXT,
 PRIMARY KEY (id, language),
 CONSTRAINT fk_message_source_message FOREIGN KEY (id)
 REFERENCES source_message (id) ON DELETE CASCADE ON UPDATE RESTRICT
);
```

## *Membuat Kamus*

Penerjemahan pesan membutuhkan kamus untuk melakukan penerjemahan. Sebagaimana definisi dari konfigurasi i18n di atas, maka kita perlu buat kamus untuk semua bahasa yang akan kita dukung untuk aplikasi kita. Misal aplikasi kita akan mendukung bahasa Indonesia, maka kita buat file kamus pada lokasi `@app/messages/id-ID/app.php` dan untuk pesan galat di `@app/messages/id-ID/app.php`

Bagaimana struktur atau isi dari *file kamus*?

Strukturnya hanya berupa *return array* sebagai berikut.

@app/messages/id-ID/app.php

```
<?php
return [
 'This is a string to translate!' => 'Ini adalah teks untuk diterjemahkan!',
 'Only example' => 'Hanya contoh',
 //... dst
];

```

Untuk mengujinya, kita bisa akses *view* atau *action* yang mengandung fungsi Yii::t() maka akan diterjemahkan dengan svarat teks tersebut terdapat pada *file kamus*.

Berikut ini contoh fungsi pada SiteController yang menerapkan penerjemahan pesan

```
public function actionTranslation()
{
 echo \Yii::t('app', 'This is a string to translate!');
}
```

```

 echo "
";
 echo \Yii::t('app', 'Only example');
}

```

Maka ketika diakses, teks yang muncul adalah teks hasil terjemahan.



**Ini adalah teks untuk diterjemahkan!**

Hanya contoh

Bagaimana dengan format kamus jika kita menggunakan media basis data?

Tabel source\_message

id	category	message
1	app	This is a string to translate!
2	app	Only example

Tabel message

id	language	translation
1	id-ID	Ini adalah teks untuk diterjemahkan!
2	id-ID	Hanya contoh
1	ru-RU	Это строка для перевода!
1	zh-CN	这是将一个字符串！

#### Perhatian.

Data kamus ini bersifat *case sensitif*.

## F. 3. Pemformatan Pesan

Pemformatan pesan memungkinkan kita menggunakan parameter dinamis dalam penerjemahan. Penulisan parameter adalah dibungkus dengan kurung kurawal buka { dan tutup }.

### Pemformatan Teks

Berikut ini contohnya.

```

<?php
public function actionTranslation()
{
 $username = 'Hafid';
 echo \Yii::t('app', 'Hello, {username}, how are you?', [
 'username' => $username,
]);
 echo "
";
 $username = 'Budi';
 echo \Yii::t('app', 'Hello, {username}, how are you?', [
 'username' => $username,
]);
}

```

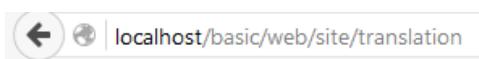
Adapun pada file kamus kita definisikan sebagai berikut.

```

<?php
return [
 //...
 'Hello, {username}, how are you?' => 'Halo, {username}, apa kabar?',
];

```

Maka ketika kita akses akan nampak sebagai berikut.



Halo, Hafid, apa kabar?

Halo, Budi, apa kabar?

## Pemformatan Angka

Berikut ini contohnya.

```
$sum = 42;
echo \Yii::t('app', 'Balance: {0,number}', $sum);
echo "
";
$sum = 42;
echo \Yii::t('app', 'Balance: {0,number,currency}', $sum);
```

Data kamus sebagai berikut

```
<?php
return [
 ..
 'Balance: {0,number}' => 'Saldo: {0,number}',
 'Balance: {0,number,currency}' => 'Saldo: {0,number,currency}',
];
```

Hasilnya

Saldo: 42  
Saldo: Rp42

## Pemformatan Tanggal

Berikut ini contohnya.

```
echo \Yii::t('app', 'Today is {0,date}', time())."
";
echo \Yii::t('app', 'Today is {0,date,short}', time())."
";
echo \Yii::t('app', 'Today is {0,date,yyyy-MM-dd}', time());
```

Data kamus sebagai berikut

```
<?php
return [
 ..
 'Today is {0,date}' => 'Hari ini {0,date}',
 'Today is {0,date,short}' => 'Hari ini {0,date,short}',
 'Today is {0,date,yyyy-MM-dd}' => 'Hari ini {0,date,yyyy-MM-dd}',
];
```

Hasilnya

Hari ini 14 Feb 2016  
Hari ini 14/02/16  
Hari ini 2016-02-14

## Pemformatan Waktu

Berikut ini contohnya.

```
echo \Yii::t('app', 'It is {0,time}', time())."
";
echo \Yii::t('app', 'It is {0,time,short}', time())."
";
echo \Yii::t('app', 'It is {0,date,HH:mm}', time());
```

Data kamus sebagai berikut

```
<?php
return [
 ..
 'It is {0,time}' => 'Sekarang pukul {0,time}',
 'It is {0,time,short}' => 'Sekarang pukul {0,time,short}',
 'It is {0,time,HH:mm}' => 'Sekarang pukul {0,time,HH:mm}',
];
```

Hasilnya

Sekarang pukul 09.22.57  
Sekarang pukul 09.22  
It is 09:22

## Pemformatan Terbilang

Berikut ini contohnya.

```
echo \Yii::t('app', '{n,number} is spelled as {n,spellout}', ['n' => 42])."
";
echo \Yii::t('app', 'I am {n,spellout,%spellout-ordinal} agent', ['n' => 47]);
```

Data kamus sebagai berikut

```
<?php
return [
// ..
'{n,number} is spelled as {n,spellout}' => '{n,number} dieja sebagai {n,spellout}',
'I am {n,spellout,%spellout-
ordinal} agent' => 'Saya adalah agen {n,spellout,%spellout-ordinal}',
];
```

Hasilnya

42 di eja sebagai empat puluh dua  
Saya adalah agen keempat puluh tujuh

## Pemformatan Ordinal

Berikut ini contohnya.

```
echo \Yii::t('app', 'You are the {n,ordinal} visitor here!', ['n' => 42]);
```

Data kamus sebagai berikut

```
<?php
return [
// ..
'You are the {n,ordinal} visitor here!' => 'Kamu adalah pengunjung {n,ordinal} di
sinis!',];
];
```

Hasilnya

Kamu adalah pengunjung ke42 disini!

Selengkapnya silakan buka guide resmi Yii tentang i18n <http://www.yiiframework.com/doc-2.0/guide-tutorial-i18n.html>

## G. Caching

*Caching* adalah cara yang cukup efektif untuk meningkatkan kinerja aplikasi web. Dengan menyimpan data yang relatif statis dalam *cache* (penampung bisa berupa *file*, *memory*, dan basis data) dan jika dibutuhkan maka tinggal mengambilnya dari *cache*. Hal ini tentu akan menghemat waktu dibandingkan dengan meng-generate data dari awal.

---

*Caching* cocok digunakan jika data aplikasi kita besar.

*Caching* dapat diterapkan baik di sisi server maupun *client*. Di sisi server, *cache* dapat digunakan untuk menyimpan data-data yang diambil dari basis data dan menyimpan sebagian atau seluruh halaman web. Di sisi *client*, *caching* dapat digunakan untuk menyimpan konten halaman dalam *cache web browser*.

Yii mendukung semua mekanisme *caching* ini:

- Data *caching*
- Fragment *caching*
- Page *caching*
- HTTP *caching*

### G. 1. Data Caching

Data *caching* adalah tentang menyimpan variabel PHP pada *cache*. *Caching* ini juga merupakan dasar dari jenis *caching* lain yaitu *query caching* dan *page caching*.

## Cache Storage

Secara *default*, *cache* di Yii telah siap digunakan tanpa perlu konfigurasi apapun yaitu menggunakan *cache storage* berupa *file* melalui *class* [[yii\caching\FileCache]]. Lihat konfigurasinya di @app/config/web.php

```
return [
 // ...
 'components' => [
 'cache' => [
 'class' => 'yii\caching\FileCache',
],
],
];
```

FileCache ini akan menyimpan data *cache* pada suatu *file* yang diletakkan pada direktori @app/runtime/cache. Pada aplikasi yang membutuhkan performa tinggi tidak disarankan menggunakan FileCache.

Yii mendukung berbagai *cache storage* antara lain:

- [[yii\caching\DbCache]],

*Cache* ini menggunakan tabel basis data untuk menyimpan data *cache*, untuk menggunakannya maka kita perlu membuat tabel ‘cache’ pada basis data, dengan struktur sebagai berikut.

```
CREATE TABLE cache (
 id char(128) NOT NULL PRIMARY KEY,
 expire int(11),
 data BLOB
);
```

Pada aplikasi yang membutuhkan performa tinggi, minimal kita harus menggunakan *storage* jenis ini.

- [[yii\caching\ApcCache]],

*Cache* ini menyimpan data *cache* pada *memory* dengan menggunakan *extension PHP APC* (Alternative PHP Cache). Oleh karena tidak di-*bundle* dengan paket PHP maka kita perlu melakukan instalasi APC dan mengkonfigurasinya. Silakan bereksplorasi sendiri jika ingin mencoba untuk menggunakan *storage* jenis ini, karena tidak dibahas pada buku ini.

*Cache* jenis ini cocok jika kita hanya menggunakan satu server tanpa *load balancer*.

- [[yii\caching\MemCache]],

*Cache* ini menyimpan data *cache* pada *memory* dengan menggunakan *extension PHP memcahe* atau *memcached*. Oleh karena tidak di-*bundle* dengan paket PHP maka kita perlu melakukan instalasi *library memcache/memcached* dan mengkonfigurasinya. Silakan bereksplorasi sendiri jika ingin mencoba untuk menggunakan *storage* jenis ini, karena tidak dibahas pada buku ini.

∞ <http://www.hafidmukhsin.com/2014/11/16/yii2-handle-large-data-with-filecache-dan-memcache/>

*Cache* jenis ini cocok jika kita menggunakan *multi server* dengan *load balancer*.

- [[yii\redis\Cache]].

*Cache* ini menyimpan data *cache* pada *memory* menggunakan redis (<http://redis.io>). Untuk aplikasi yang membutuhkan performa tinggi, direkomendasikan untuk menggunakan *cache* jenis ini.

Silakan bereksplorasi sendiri jika ingin mencoba untuk menggunakan *storage* jenis ini, karena tidak dibahas pada buku ini.

Tip:

Kita dizinkan menggunakan beberapa *cache storage* sekaligus pada satu aplikasi. Strategi yang umum digunakan adalah menggunakan *storage* berbasis *memory* untuk menyimpan data yang kecil namun sering digunakan (contoh: data statistik), dan menggunakan *storage* berbasis *file* atau basis data untuk menyimpan data yang besar dan jarang digunakan (contoh konten halaman).

Contoh konfigurasi jika menggunakan lebih dari satu *storage cache*.

```
return [
 // ...
 'components' => [
 'cache' => [
 'class' => 'yii\caching\FileCache',
],
 'cache2' => [

```

```

 'class' => 'yii\caching\MemCache',
 // konfigurasi dsb
],
],
];

```

Pada panduan ini, kita akan menggunakan *storage cache* berjenis *file* atau FileCache.

### Implementasi

Implementasi atau penggunaan *cache* pada Yii cukup mudah, hampir sama dengan penggunaan *session*.

```

$data = $cache->get($key);
if ($data === false) {
 $cache->set($key, $data);
}

```

Misalnya kita mempunyai *variabel* yang ingin kita simpan di *cache*, maka kita harus tentukan dulu *key* atau ID dari *variabel* tersebut (bebas). Mekanismenya, setiap kali membutuhkan data tersebut maka kita harus mengecek ada tidaknya data *cache* dengan ID tersebut, jika data ditemukan maka dibaca dari *cache*, jika tidak maka *variabel* tersebut disimpan di *cache* dengan ID tertentu.

Sebagai contoh kita buat fungsi baru pada SiteController yaitu actionDataCache

```

public function actionDataCache()
{
 $var1 = "Teks ini akan disimpan di cache";
 $cache = \Yii::$app->cache;
 $data = $cache->get('var1');
 if ($data === false) {
 $cache->set('var1', $var1);
 $data = $var1;
 }
 echo $data;
}

```

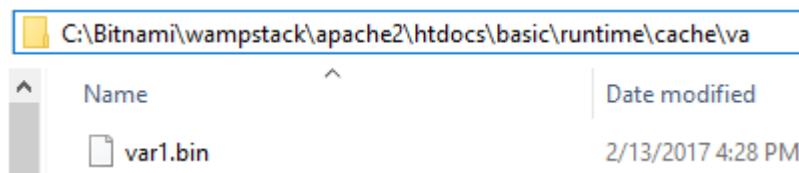
Mari kita test



**Teks ini akan di simpan di cache**

Untuk menguji coba apakah *cache* bekerja, maka kita coba ganti nilai dari \$var1, kemudian diakses kembali, maka teks yang akan muncul tetap sama dengan teks sebelumnya.

Mari kita lihat apa yang sebenarnya terjadi. Buka *folder cache* di @app/runtime/cache, maka kita akan jumpai sebuah *folder* lagi yang di-generate secara acak oleh Yii, di dalam *folder* itu akan ada *file* dengan nama sesuai dengan nama yang sama dengan *key* yaitu var1.bin



Isinya sebagai berikut

var1.bin	source.php
1 a:2:{i:0;s:32:"Teks ini akan di simpan di cache";i:1;N;}	

Kita juga bisa mengakses *cache* menggunakan metode *array*.

```

public function actionDataCache()
{
 $var2 = "Teks ini akan disimpan di cache dengan metode array";
 $cache = \Yii::$app->cache;
 $data = $cache['var2'];
 if ($data === false) {
 $cache['var2'] = $var2;
 $data = $var2;
 }
}

```

```

 echo $data;
}

```



**Teks ini akan di simpan di cache dengan metode array**

### **Cache Expiration**

Sampai kapan atau berapa lama *cache* ini akan disimpan di server? Jawabannya adalah selama-lamanya, atau tidak akan kadaluarsa kecuali kita menghapus *file cache* di server. Hal ini disebabkan karena kita tidak mengkonfigurasi waktu kadaluarsanya.

```

public function actionDataCache()
{
 $var3 = "Teks ini akan disimpan di cache dalam waktu 60 detik";
 $cache = \Yii::$app->cache;
 $data = $cache->get('var3');
 if ($data === false) {
 $cache->set('var3', $var3, 60);
 $data = $var3;
 }
 echo $data;
}

```



**Teks ini akan di simpan di cache dalam waktu 60 detik**

60 detik kemudian jika kita ubah nilai dari *variabel* \$var3 maka nilai dari \$data akan berubah.

### **Cache Dependency**

Di samping waktu *expire*, kita bisa juga menggunakan *dependency* untuk menentukan “umur” dari suatu *cache*. Artinya selama *dependency* belum berubah maka *cache* tidak akan diubah.

Sebagai contoh kita akan menggunakan *class* [[*yii\caching\FileDependency*]] yaitu *dependency* berdasarkan *file* tertentu.

```

public function actionDataCache()
{
 $var4 = "Teks ini akan disimpan di cache tergantung dari file dependency.txt";
 $cache = \Yii::$app->cache;
 $data = $cache->get('var4');
 if ($data === false) {
 $dependency = new \yii\caching\FileDependency([
 'fileName' => '@runtime/cache/dependency.txt',
]);
 $cache->set('var4', $var4, 0, $dependency);
 $data = $var4;
 }
 echo $data;
}

```

Tentu kita harus buat *file* teks pada @app/runtime/cache/dependency.txt

Mari kita uji coba.



**Teks ini akan di simpan di cache tergantung dari file dependency.txt**

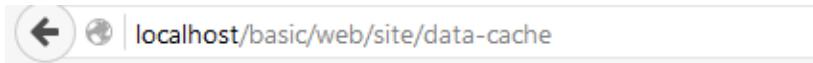
Selama kita tidak mengubah *file* dependency.txt maka *cache* tidak akan dibuang. Mari kita lihat seperti apa *file cache*-nya

```
a:2:{i:0;s:70:"Teks ini akan di simpan di cache tergantung dari file dependency.txt";i:1;O:26:"yii\caching\FileDependency":3:{s:8:"fileName";s:29:"@runtime/cache/dependency.txt";s:4:"data";i:1455444409;s:8:"reusable";b:0;}}
```

Kita juga bisa menggunakan *dependency caching* berdasarkan hasil *query* dari basis data. Selama *query* tidak menghasilkan *result* yang berbeda maka data *cache* tetap tidak akan *expire*.

```
public function actionDataCache()
{
 $var5 = "Teks ini akan disimpan di cache tergantung dari db dependency";
 $cache = \Yii::$app->cache;
 $data = $cache->get('var5');
 if ($data === false) {
 $dependency = new \yii\caching\DbDependency([
 'sql'=> 'SELECT count(id) FROM category',
]);
 $cache->set('var5', $var5, 0, $dependency);
 $data = $var5;
 }
 echo $data;
}
```

Artinya, selama jumlah *category* tidak berubah, maka *cache* juga tidak akan diperbarui.



**Teks ini akan di simpan di cache tergantung dari db dependency**

Berikut ini *file cache*-nya

```
a:2:{i:0;s:62:"Teks ini akan di simpan di cache tergantung dari db dependency";i:1;O:24:"yii\caching\DbDependency":5:{s:2:"db";s:2:"db";s:3:"sql";s:30:"SELECT count(id) FROM category";s:6:"params";a:0:{}s:4:"data";a:1:{s:9:"count(id)";s:2:"26";}s:8:"reusable";b:0;}}
```

### Query Caching

Hasil dari *query* juga bisa di *cache*. Berikut ini contoh penerapannya pada *single query*

```
public function actionQueryCaching()
{
 $db = \Yii::$app->db;
 $result = $db->createCommand('SELECT * FROM category WHERE id=1')->cache(1)->queryOne();
 echo $result['title'];
}
```

Sedangkan berikut ini contoh penerapan pada *multiple query*

```
public function actionQueryCaching()
{
 $db = \Yii::$app->db;
 $result = $db->cache(function ($db) {
 return $db->createCommand('SELECT * FROM category WHERE id=1')->queryOne();
 });
 echo $result['title'];
}
```



## Buku Tulis2

Dengan cara ini, maka meskipun *title category* dengan id 1 diubah di basis data maka pada aplikasi kita akan tetap menampilkan hasil yang sebelumnya.

Kita juga bisa menerapkannya pada ActiveRecord

```
$result = Category::getDb() -> cache(function ($db) {
 return Category::find() -> where(['id' => 1]) -> one();
});
```

*Expiration* dan *dependency* juga bisa diterapkan pada *query caching*.

Pada *single query* sebagai berikut

```
public function actionQueryCaching()
{
 $db = \Yii::$app->db;
 $expiration = 60;
 $dependency = new \yii\caching\DbDependency([
 'sql'=> 'SELECT count(id) FROM category',
]);
 $result = $db->createCommand('SELECT * FROM category WHERE id=1')
 ->cache($expiration, $dependency)->queryOne();
 echo $result['title'];
}
```

sedangkan pada *multiple query* sebagai berikut.

```
public function actionQueryCaching()
{
 $db = \Yii::$app->db;
 $expiration = 60;
 $dependency = new \yii\caching\DbDependency([
 'sql'=> 'SELECT count(id) FROM category',
]);
 $result = $db->cache(function ($db) {
 return $db->createCommand('SELECT * FROM category WHERE id=1')->queryOne();
 }, $expiration, $dependency);
 echo $result['title'];
}
```

*Query caching* tidak akan bekerja jika digunakan untuk menyimpan *resource handle* misal data berformat *blob*. Beberapa *storage cache* memiliki batasan maksimal ukuran masing-masing *cache*, sebagai contoh pada Memcache maksimal adalah 1 MB.

## G. 2. Fragment Caching

*Fragment caching* adalah menyimpan sebagian halaman *web* pada *cache*. Misal digunakan untuk menyimpan halaman yang berisi grafik *chart*, atau menyimpan halaman berisi tabel *Gridview*. Intinya *fragment caching* digunakan untuk meng-*cache* suatu *view*.

Berikut ini *format*-nya

```
if ($this->beginCache($id)) {
 // ... generate content here ...
 // Gridview atau Chart dsb

 $this->endCache();
}
```

\$id adalah *key*. Kita bisa meletakkannya pada *view*.

*Expiration* dan *dependency* juga bisa diterapkan di sini.

```
if ($this->beginCache($id, [
 'duration' => $duration,
 'dependency' => $dependency,
])) {
 // ... generate content here ...
 // Gridview atau Chart dsb
}
```

```

 $this->endCache();
}
```

Di samping itu kita juga bisa menerapkan *dependency* lain yaitu *variation* dan *toggle*

```

if ($this->beginCache($id, ['variations' => [Yii::$app->language]])) {
 // ... generate content here ...
 $this->endCache();
}
```

Contoh *toggle* menggunakan *enabled*

```

if ($this->beginCache($id, ['enabled' => Yii::$app->request->isGet])) {
 // ... generate content here ...
 $this->endCache();
}
```

### G. 3. Page Caching

*Page caching* hampir sama dengan *fragment caching*, hanya saja *caching* ini menyimpan seluruh halaman web bukan sebagian. Cara penggunaanya juga bukan di *view* melainkan pada *controller*.

Lihat contoh berikut.

```

public function behaviors()
{
 return [
 [
 'class' => 'yii\filters\PageCache',
 'only' => ['index'],
 'duration' => 60,
 'variations' => [
 \Yii::$app->language,
],
 'dependency' => [
 'class' => 'yii\caching\DbDependency',
 'sql' => 'SELECT COUNT(*) FROM category',
],
],
];
}
```

Contoh di atas hanya akan meng-*cache* halaman index yang didefinisikan pada parameter *only*.

### G. 4. HTTP Caching

*HTTP caching* adalah *caching* di sisi *client* menggunakan *cache* pada *web browser*. Yii menggunakan *class* [[*yii\filters\HttpCache*]] sebagai *filter* dari *controller* sebagaimana *page caching*. *Cache* ini hanya mendukung *request* GET atau HEAD saja.

Berikut ini contoh menggunakan *dependency Last Modified*

```

public function behaviors()
{
 return [
 [
 'class' => 'yii\filters\HttpCache',
 'only' => ['index'],
 'lastModified' => function ($action, $params) {
 $q = new \yii\db\Query();
 return $q->from('category')->max('updated_at');
 },
],
];
}
```

Selama nilai dari *update\_at* terakhir pada tabel *category* tidak berubah maka setiap *request* akan tetap menggunakan *cache* di *web browser*.

Berikut ini menggunakan *dependency tag Header*

```

public function behaviors()
{
 return [
 [
 'class' => 'yii\filters\HttpCache',
 'only' => ['view'],
 'etagSeed' => function ($action, $params) {
 $category = $this->findModel(\Yii::$app->request->get('id'));
 return serialize([$category->title, $category->description]);
 },
],
];
}

```

Selama nilai dari judul dan *description* pada tabel *category* tidak berubah maka setiap *request* akan tetap menggunakan *cache* di *web browser*.

## G. 5. Kesimpulan

*Caching* memang cukup efektif untuk meningkatkan performa aplikasi, namun yang perlu diperhatikan bahwa sebaiknya ketika *development* aplikasi kita tidak mengaktifkan *cache*. Hal ini dikarenakan pada saat *development* aplikasi pastinya sering melakukan perubahan dan uji coba, bisa jadi karena adanya *cache* maka perubahan yang kita lakukan tidak nampak karena data telah di *cache*.

Oleh karena itu, Yii menyediakan *class* penyimpanan *cache* [[*yii\caching\DummyCache*]]. Yaitu komponen *cache* yang hanya digunakan untuk *development* saja, komponen ini tidak benar-benar melakukan proses *cache*. Kemudian jika aplikasi hendak di-*launching* ke mode *production* maka kita bisa ganti menjadi komponen *cache* yang lain.

## H. Performance Tuning

Ada banyak faktor yang mempengaruhi performa dari aplikasi web kita. Sebagian menyangkut *environment* dari server, sebagian terkait dengan *coding* kita dan sebagian lagi terkait dengan *framework* Yii.

Berikut ini beberapa hal yang bisa kita lakukan untuk mengoptimalkan performa aplikasi web kita.

### H. 1. Optimizing Server Environment

PHP versi 7 telah mencapai versi *stable* dimana memiliki performa yang lebih baik daripada versi 5.6 kebawah. Oleh karena itu sebaiknya kita menggunakan PHP versi ini, apalagi Yii telah mendukung PHP versi 7 juga.

Aktifkan PHP Opcache yang akan meng-*compile* kode PHP kita menjadi bytecode dan menyimpannya dalam *cache* sehingga akan lebih cepat diakses.

Berikut ini setting PHP Opcache pada *php.ini* yang direkomendasikan

```

opcache.memory_consumption=128
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=4000
opcache.revalidate_freq=60
opcache.fast_shutdown=1
opcache.enable_cli=1

```

Gunakan *web server* NginX yang memiliki performa lebih tinggi dibandingkan dengan *web server* Apache. Instalasi dan konfigurasi PHP FPM. Untuk konfigurasinya silakan googling ya ☺

### H. 2. Menonaktifkan Mode Debug

Pada *mode debug* maka Yii akan meng-generate semua informasi tentang *debugging* aplikasi dan hal ini membutuhkan waktu. Oleh karena itu pada saat *production* sebaiknya kita menonaktifkannya.

Caranya, pada *entry script*, @app/web/index.php. Hapus atau *comment* baris berikut.

```

// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

```

atau bisa juga diubah menjadi sebagai berikut

```

defined('YII_DEBUG') or define('YII_DEBUG', false);
defined('YII_ENV') or define('YII_ENV', 'prod');

```

---

Jika kita menggunakan *application template advanced* maka kita bisa jalankan perintah *init*.

---

### H. 3. Menggunakan Teknik Caching

Telah dibahas pada bagian sebelumnya.

### H. 4. Mengaktifkan Schema Caching

*Schema caching* adalah fitur *caching* khusus yang harus diaktifkan jika kita menggunakan ActiveRecord. Sebagaimana yang telah kita ketahui, ActiveRecord bisa mendeteksi skema tabel secara otomatis tanpa kita perlu mendefinisikannya.

ActiveRecord memperoleh informasi ini dengan menjalankan *query SQL ekstra*. Dengan mengaktifkan *schema caching*, maka informasi skema yang diambil akan disimpan dalam *cache* dan akan digunakan kembali jika ada permintaan berikutnya.

Berikut ini cara mengaktifkan *schema caching* yaitu pada konfigurasi @app/config/web.php

```
return [
 // ...
 'components' => [
 // ...
 'cache' => [
 'class' => 'yii\caching\FileCache',
],
 'db' => [
 'class' => 'yii\db\Connection',
 'dsn' => 'mysql:host=localhost;dbname=db',
 'username' => 'root',
 'password' => '',
 'enableSchemaCache' => true,
 'schemaCacheDuration' => 3600,
 'schemaCache' => 'cache',
],
],
];
```

### H. 5. Mengoptimalkan Session Storage

Salah satu cara untuk meningkatkan performa aplikasi adalah dengan memindahkan media penyimpanan *session* ke basis data. *Session* secara *default* akan menyimpan data pada sebuah *file*, jika aplikasi memiliki banyak pengunjung, maka proses *read/write* data *session* ke *file* akan membutuhkan waktu yang lama. Oleh karena itu disarankan kita memindahkannya ke basis data.

Caranya dengan mengkonfigurasinya pada @app/config/web.php

```
return [
 // ...
 'components' => [
 'session' => [
 'class' => 'yii\web\DbSession',
 // Set the following if you want to use DB component other than
 // default 'db'.
 // 'db' => 'mydb',

 // To override default session table, set the following
 // 'sessionTable' => 'my_session',
],
],
];
```

Kemudian kita buat tabel *session* pada basis data untuk menyimpan data *session*. Adapun struktur tabelnya sebagai berikut

```
CREATE TABLE session (
 id CHAR(40) NOT NULL PRIMARY KEY,
 expire INTEGER,
 data BLOB
)
```

Jika kita melakukan instalasi Redis pada server, maka sangat direkomendasikan untuk menyimpan *session* pada Redis dengan menggunakan *class* [[*yii\redis\Session*]].

## H. 6. Mengoptimalkan Composer Autoloader

Secara *default*, *Composer Autoloader* akan secara otomatis *me-load* semua class yang diletakkan pada direktori terdaftar. Namun sayangnya hal itu membutuhkan usaha yang tidak sedikit terutama jika *class library* yang digunakan banyak. *Autoloader* menggunakan fungsi PHP *file\_exists* untuk menemukan *file class* dimaksud pada daftar *namespace*. Oleh karenanya kita bisa optimalkan dengan menjalankan perintah berikut pada CMD.

```
| composer dumpautoload -o
```

Perintah tersebut akan membuat Composer meng-*generate* “class map” yaitu *array* yang berisi daftar lokasi dari semua *class*. Cara ini akan meningkatkan performa hingga lebih dari 20%.

```
∞ http://mouf-php.com/optimizing-composer-autoloader-performance
```

## I. Kesimpulan

Sebenarnya masih banyak tips-triks yang bisa digali lagi pada Yii terutama yang berkaitan dengan *library* pihak ketiga. Hal ini dikarenakan begitu mudahnya integrasi suatu *library* dengan Yii.

Pada bab selanjutnya sekaligus bab terakhir dari buku ini, kita akan belajar tentang bagaimana men-*deploy* atau melakukan instalasi aplikasi kita ke server sehingga bisa diakses oleh siapapun yang mempunyai koneksi internet.

# 16 Deploying Aplikasi

Pada bab ini kita akan belajar tentang:

- *Hosting & Virtual Private Server*
- Implementasi *deploy* aplikasi ke server

*Deploy* aplikasi maksudnya adalah melakukan instalasi aplikasi *web* kita ke server sehingga bisa diakses oleh siapapun yang terkoneksi dengan internet. Prosesnya meliputi: unggah kode sumber ke server, impor basis data, dan konfigurasinya.

## A. Definisi

Ada beberapa istilah dari layanan server yang umum kita jumpai. Oleh karena itu penulis merasa bahwa istilah-istilah tersebut perlu dibahas terlebih dahulu supaya kita mempunyai satu *frame* pemahaman yang sama.

### A. 1. *Hosting*

*Hosting* merupakan jenis layanan dimana kita bisa men-*deploy* aplikasi kita ke server, baik itu kode sumber dan basis datanya. Dengan menggunakan atau menyewa *hosting* maka kita tidak akan direpotkan dengan persiapan *environment* server seperti instalasi OS, instalasi perangkat lunak pendukung, konfigurasi dsb sebab semua telah disiapkan oleh penyedia layanan ini. Sehingga kita cukup fokus pada aplikasi kita saja.

Layanan *hosting* memiliki beberapa jenis.

#### 1. *Shared Hosting*

*Shared hosting* adalah layanan *hosting* yang umum digunakan yaitu server yang melayani atau digunakan oleh banyak situs web. Sehingga *resource* (Disk, CPU, dan RAM) yang umumnya berupa satu server fisik ini digunakan secara bersama-sama oleh semua situs web yang di-*hosting* pada server tersebut.

Layanan ini biasanya hanya menyewakan *space* untuk aplikasi kita, dimana kita bisa menggunakan nama *domain*-nya sendiri. Di samping itu pada layanan ini menyediakan fitur antar muka untuk manajemen aplikasi (umumnya cpanel) seperti fitur unggah atau unduh *file* kode sumber serta basis datanya. Adapun OS, dan *tools-tools* pendukung (web server, PHP, basis data) sudah ditentukan oleh penyedia jasanya.

#### 2. *Dedicated Hosting*

Layanan ini hampir sama dengan layanan *shared hosting*, hanya saja satu server fisik digunakan oleh satu situs web atau pengguna sehingga *resource*-nya tidak dibagi-bagi.

#### 3. *Cloud Hosting*

Layanan ini sebenarnya sama dengan *shared hosting* hanya saja poin yang ditekankan pada layanan ini adalah fleksibilitas, dan skalabilitas. Hal ini dikarenakan, umumnya *provider* layanan ini memiliki beberapa data center yang terpisah yang dihubungkan dengan teknologi virtualisasi.

## A. 2. *Virtual Private Server (VPS)*

Jenis layanan ini menyediakan server kosong dimana kita harus melakukan instalasi sendiri mulai dari sistem operasi hingga *tools-tools* yang diperlukan oleh aplikasi kita. VPS juga memiliki beberapa jenis.

#### 1. *Cloud Server*

Layanan ini mengizinkan kita untuk membuat sendiri virtual server, melakukan instalasi OS dan *tools* yang diperlukan, melakukan pengaturan melalui SSH dsb. Adapun virtualisasinya tersebut berbasis *cloud*, yang *resource* dari severnya bisa terdistribusi dan bisa juga *sharing*.

## 2. Dedicated Server

Layanan ini hampir sama dengan *cloud server*, hanya saja satu pengguna akan mendapatkan satu server fisik sendiri sehingga *resource*-nya tidak di-*sharing*. *Dedicated server* bisa juga digolongkan menjadi jenis layanan tersendiri.

## B. Implementasi

Pada bagian ini kita akan belajar men-*deploy* aplikasi kita ke server daring. Adapun server yang akan digunakan pada panduan ini adalah server yang berjenis *shared hosting* dan VPS.

### B. 1. Deploy di Shared Hosting

*Shared hosting* merupakan jenis layanan dengan banyak limitasi, yang mana kita tidak bisa mengakses konfigurasi tertentu pada server. Sebagai contoh: pada *hosting* yang penulis sewa, tidak tersedia fitur untuk mengakses SSH. (mungkin ada *web hosting* lain yang menyediakan fitur SSH namun tentunya kemampuannya akan dibatasi)

Panduan *deploy* ke *shared hosting* ini mengikuti atau merujuk pada praktik terbaik yang dicontohkan oleh Yii melalui panduan resminya yaitu dengan mengubah nama *folder web* menjadi **www** atau **public\_html** sesuai dengan *web root* dari *hosting* kita. Hal ini dikarenakan, umumnya pada *shared hosting*, kita tidak diperbolehkan untuk menentukan sendiri lokasi *web root*.

Karena pada umumnya, *shared hosting* menggunakan *tools* CPanel untuk mengelola *hosting* kita, maka panduan ini akan mencontohkan langkah-langkah *deploy* menggunakan CPanel

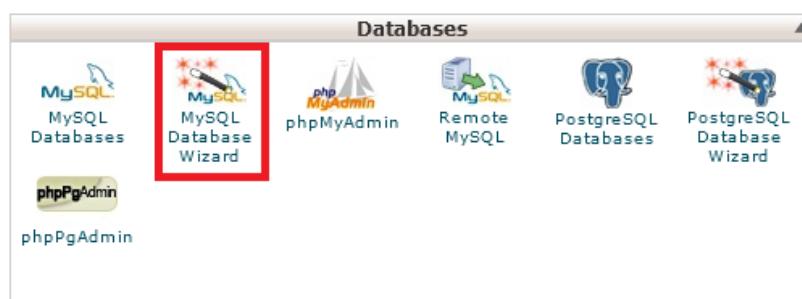


Berikut ini langkah-langkah untuk *deploy* aplikasi.

#### Buat basis data aplikasi kita

Pertama kita perlu mempersiapkan database aplikasi kita.

1. Melalui cPanel kita bisa menggunakan *MySQL Database Wizard*



2. Buat basis data baru, pada *field* nama basis data, masukkan nama basis data aplikasi kita misal yii2basic

---

#### Catatan.

Pada *shared hosting* umumnya nama basis data diawali dengan *username hosting* kita

**MySQL® Database Wizard**

[Video Tutorial](#)

MySQL Databases allow you to store a large amount of information in an easy to access manner, are not easily read by humans. MySQL databases are required by many web applications including content management systems, and others. To use a database, you'll need to create it. Only MySQL users (not other users) that have privileges to access a database can read from or write to that database.

**Step 1: Create A Database**

New Database: hafidmuk\_yii2basic

[Next Step](#)

3. Buat pengguna untuk basis data yang baru kita buat.

**Step 2: Create Database Users**

Username: hafidmuk\_hafid

NOTE: Usernames cannot contain more than seven characters.

Password:

Reenter Password:

Strength ([Why?](#)): [Password Generator](#)

[Create User](#)

[← Go Back](#) | [← Go Back to the Main MySQL Databases Interface](#)

4. Setelah itu pilih *All Privileges*, sehingga pengguna yang baru kita buat bisa memiliki hak akses penuh (semua perintah SQL). Hal ini dikarenakan kita akan gunakan untuk mengimpor basis data. Adapun setelah basis data diimporkan kita bisa edit *privileges* sesuai kebutuhan, misalnya *Select*, *Create*, *Update*, dan *Delete*.

**Step 3: Add a User to the Database**

User: hafidmuk\_hafid  
Database: hafidmuk\_yii2basic

<input checked="" type="checkbox"/> ALL PRIVILEGES	
<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> ALTER ROUTINE
<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> CREATE ROUTINE
<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> CREATE VIEW
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP
<input checked="" type="checkbox"/> EVENT	<input checked="" type="checkbox"/> EXECUTE
<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> INSERT
<input checked="" type="checkbox"/> LOCK TABLES	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> SHOW VIEW
<input checked="" type="checkbox"/> TRIGGER	<input checked="" type="checkbox"/> UPDATE

[Next Step](#)

[← Go Back](#) | [← Go Back to the Main MySQL Databases Interface](#)

Jika berhasil maka kita akan menjumpai pesan sebagai berikut..

**Step 4: Complete the Task**

"hafidmuk\_hafid" now has privileges on the database "hafidmuk\_yii2basic".

Langkah selanjutnya adalah mengimpor basis data aplikasi menggunakan PHPMyAdmin.

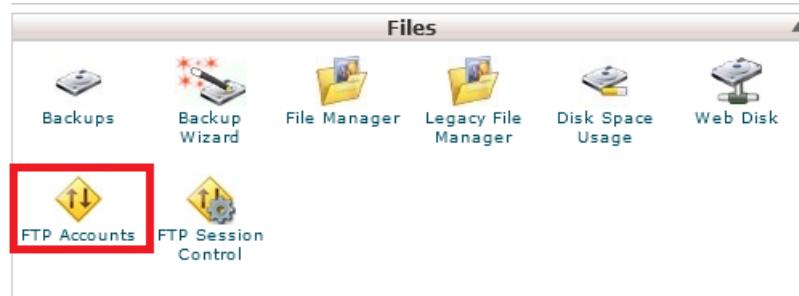


Penulis tidak akan menjelaskan lagi bagaimana cara mengimpor database ☺

### **Unggah kode sumber aplikasi ke server**

Setelah basis data aplikasi siap, maka langkah selanjutnya adalah mengunggah kode sumber aplikasi. Tools yang digunakan untuk mengunggah kode sumber adalah FTP. Namun kita juga bisa menggunakan *File Manager*. Sebelum kita bisa menggunakan FTP, kita harus membuat terlebih dahulu akun FTP pada *hosting* kita.

1. Pada CPanel pilih FTP *Accounts* pada panel *Files*



2. Umumnya kita telah dibuatkan akun FTP sesuai *username hosting* kita dan kata sandi yang sama dengan CPanel, tekan tombol *Configure FTP*

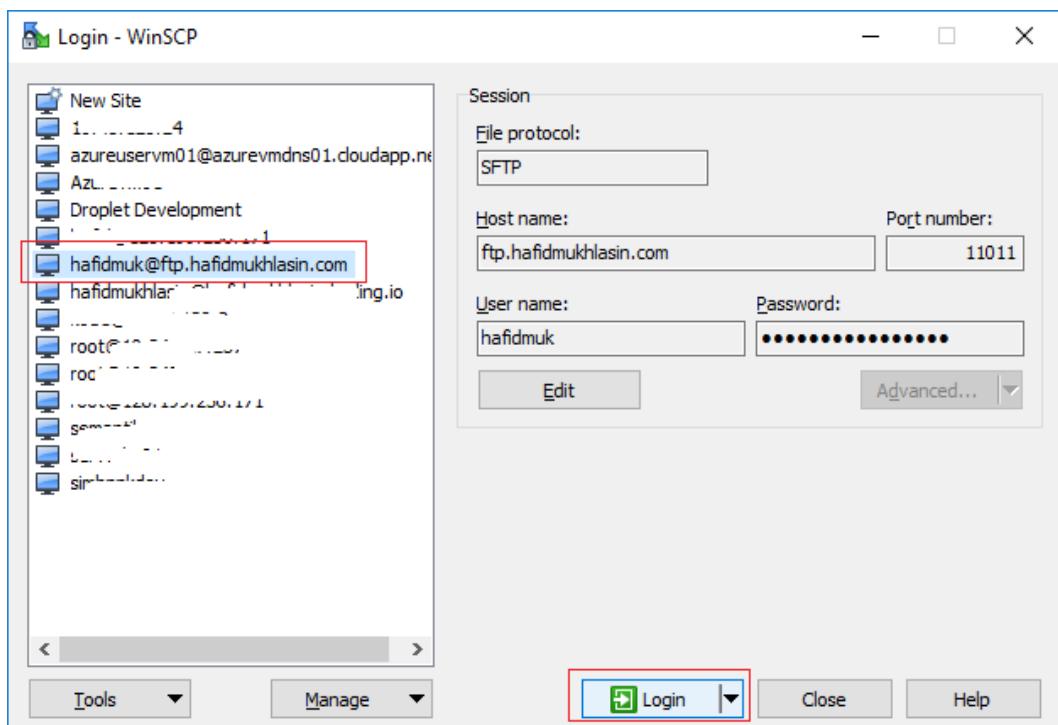
Special FTP Accounts (Why?)				
Type	Username	Path	Usage / Quota	Actions
User	hafidmuk	/home/hafidmuk	0 / ∞ MB	NA Configure FTP Client
<b>Manual Settings</b>				
FTP Username: <b>hafidmuk</b> FTP Server: <b>ftp.hafidmukhlasin.com</b> FTP & explicit FTPS port: <b>21</b> SFTP port: <b>110</b>				

3. Setelah kita mendapatkan akun FTP, maka langkah selanjutnya adalah unggah kode sumber ke server dengan menggunakan *tools* FTP misalnya: WinSCP, FileZilla, Cyberduck (Mac OS X), dll.

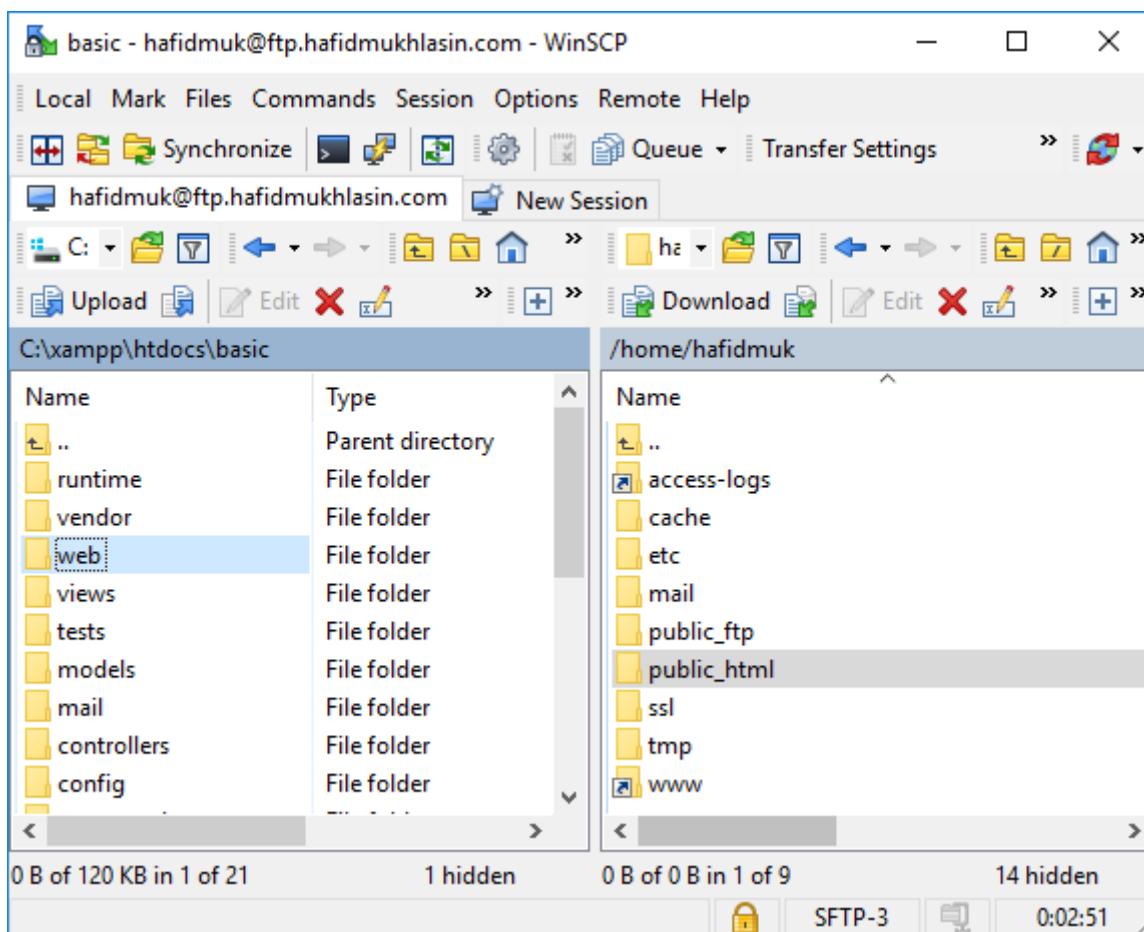
#### **Catatan.**

Pada panduan ini, penulis menggunakan WinSCP.

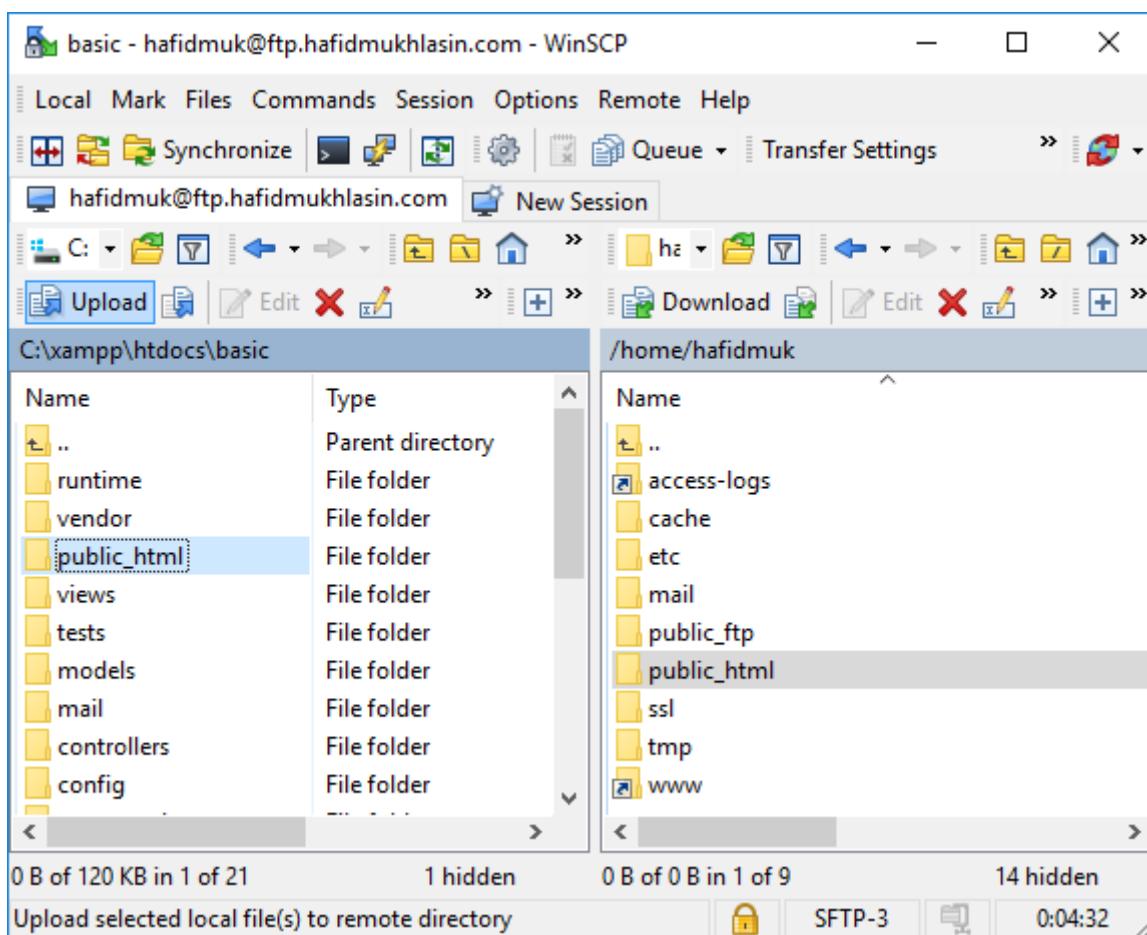
4. Pada jendela login di *tools* FTP kita (misalnya: WinSCP), klik *New Site*, lalu pilih protokol SFTP, dan lengkapil *hostname, port, username* dan kata sandi sesuai dengan data *hosting* anda.



5. Tekan tombol *Save* untuk menyimpan data *login* ini dan tekan tombol *login*. Maka akan muncul *file explorer* yang terdiri dari dua kolom sebagai berikut.



- Kolom kiri adalah direktori aplikasi kita di lokal komputer kita sedangkan kolom kanan adalah *root* direktori di server.
- Nah sebelum kita unggah kode sumber aplikasi kita, maka kita perlu mengubah nama *folder web* di lokal komputer kita menjadi **public\_html** (adapun *folder www* pada *hosting* penulis hanya tautan ke *folder public\_html*) sebagaimana yang dijelaskan sebelumnya.
  - Setelah nama folder diubah maka kita bisa mengunggah semua *file & folder* di kolom kiri, ke *root directory* server di kanan, dengan cara menyeleksi semua *file & folder* di kolom kiri kemudian tekan tombol **upload**.



Selesai

### Cek Requirements

Untuk mengecek kebutuhan minimal Yii di *hosting*, Yii telah menyediakan *file requirements.php* yang terletak pada @app/requirements.php. Mari kita salin untuk sementara *file* ini di direktori public\_html sehingga bisa diakses melalui *web browser*

∞ <http://namadomainmu.com/requirements.php>

Pastikan yang kita butuhkan terpenuhi. Silakan merujuk ke bab awal buku ini tentang instalasi.

---

#### Perhatian:

Jangan lupa untuk menghapus kembali *file* ini dari server.

---

### Konfigurasi

Karena Yii membutuhkan hak akses *write* untuk memodifikasi *folder* runtime, maka kita bisa *set* agar *folder* runtime, /web/assets bisa *write*. Pada FTP kita bisa klik kanan *folder* yang ingin kita set sebagai *writable*.

Setelah basis data dan *file* terunggah, maka kita perlu melakukan konfigurasi.

1. Edit *file* koneksi basis data di server yaitu @app/config/db.php (contoh: /home/hafidmuk/config/db.php) sesuaikan dengan konfigurasi basis data pada server
2. Edit *entry script* di server yaitu @app/public\_html/index.php (/home/hafidmuk/web/index.php), *comment* dua baris pertama.

```
<?php
// comment out the following two lines when deployed to production
// defined('YII_DEBUG') or define('YII_DEBUG', true);
// defined('YII_ENV') or define('YII_ENV', 'dev');
```

Selesai.

## B. 2. Deploy di VPS

Sebagai contoh, penulis akan menggunakan Digital Ocean yaitu salah satu *cloud hosting provider* yang cukup populer, untuk men-deploy project kita. Kelebihan dari Digital Ocean ini adalah memiliki tampilan antar muka yang sederhana serta mendukung dan didukung banyak *tools* sehingga memudahkan pemrogram aplikasi atau siapapun untuk men-deploy aplikasinya.

Ada banyak cara, *tools* dan skenario yang bisa dilakukan untuk men-deploy aplikasi ke VPS. Pada panduan ini kita akan menggunakan *tools* Git dan Composer. Berikut ini skenarionya.

1. Buat Git *repository* kita (bisa di server sendiri atau publik misal Github)
2. Dari lokal CMD, *push project* kita ke server *repository*
3. Dari server *console* (Putty), *clone project* kita dari server *repository*
4. Jalankan perintah composer install
5. Lakukan konfigurasi

Demikian langkah singkatnya, adapun langkah panjangnya akan dibahas berikut ini.

### Persiapan Tools di Server

Asumsinya kita telah melakukan instalasi Apache, PHP, & MySQL di server atau *droplet* (istilah VM di Digital Ocean).

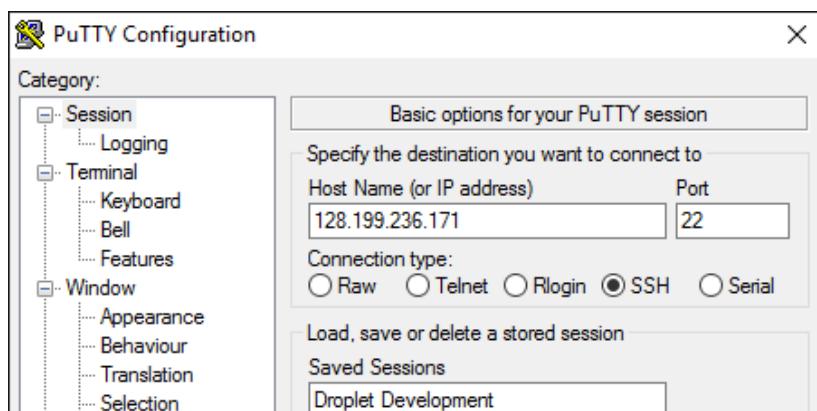
#### Catatan.

Pada panduan ini, IP publik dari *droplet* yang digunakan oleh penulis adalah 128.199.236.17, sesuaikan dengan IP server anda.

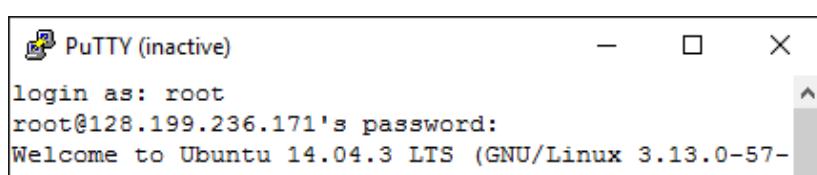
Pastikan kita telah melakukan instalasi *tools* Git dan Composer di server. Ikuti petunjuk instalasi sebagaimana yang telah dibahas pada bab awal buku ini.

*Tools* yang akan digunakan untuk mengakses server adalah Putty (<http://www.putty.org>). *Tools* ini cukup populer dikalangan *web administrator*, ukurannya kecil, tampilannya sederhana namun cukup *powerfull*.

Ketika menjalankan Putty, masukkan IP *address* server dan *port*-nya serta pilih *connection type*-nya SSH.



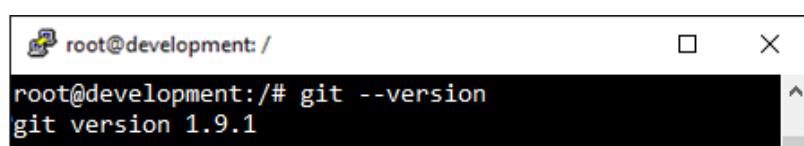
Kemudian tekan tombol *Open*, maka kita akan dibawa pada jendela CMD Putty. Kita akan diminta *username* dan kata sandi server.



Defaultnya, *user*: *root*, kata sandi sesuai dengan kata sandi *login* DO.

Mari kita uji coba untuk memastikan Git telah terinstalasi di server

```
| git --version
```



Demikian juga untuk Composer.

```
| composer -v
```

```
root@development: /root
root@development:/# composer -v
Composer version 1.1.2 2016-05-31 19:48:11
```

Semua panduan instalasinya telah dibahas pada bab awal buku ini, silahkan merujuk ke sana.

### Buat Git Repository (repo) di Server

Langkah selanjutnya adalah kita perlu membuat direktori *repo* Git di server. Sebagai contoh, pada panduan ini, penulis membuat *repo* pada direktori /basic.git

```
mkdir -p /basic.git
cd /basic.git
git init --bare
```

```
root@development: /basic.git
root@development:/# mkdir -p /basic.git
root@development:/# cd /basic.git
root@development:/basic.git# git init --bare
Initialized empty Git repository in /basic.git/
root@development:/basic.git#
```

#### Catatan:

Sudah menjadi kesepakatan bahwa penamaan *folder* repositori Git menggunakan akhiran *git*, meskipun kita tidak harus mengikutinya.

Setelah membaca dari tautan berikut, ada perintah yang perlu kita jalankan

```
∞ http://stackoverflow.com/questions/11117823/git-push-error-refusing-to-update-checked-out-branch
| git config --bool core.bare true
```

```
root@development: /basic.git
root@development:/basic.git# git config --bool
core.bare true
```

Dengan konfigurasi tersebut maka URL untuk repositori kita adalah

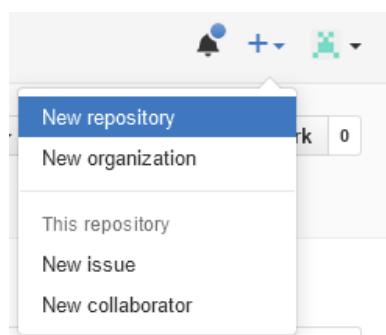
```
∞ ssh://root@128.199.236.171/basic.git
```

Sehingga kita pun bisa menggunakan perintah git biasa pada *repo* ini layaknya di Github. Misal.

```
| git clone ssh://root@128.199.236.171/basic.git
```

Tentu saja pada *real life* anda harus membuat akun pengguna tersendiri untuk Git, jadi tidak menggunakan *root* ☺

Alternatif lain, kita bisa membuat *repo* Git pada Github.



Pada panduan ini, penulis membuat *repo* Git di Github dengan URL

```
| ∞ https://github.com/hscstudio/basic
```

Sehingga kita nanti bisa *cloning project* dengan perintah

```
| git clone https://github.com/hscstudio/basic.git
```

#### **Perhatian.**

Jadi ada dua pilihan pada panduan ini, apakah anda ingin menggunakan github sebagai repo ataukah membuat repo sendiri di server. Pada intinya sama saja.

### **Menghubungkan Project Lokal ke Server Repo**

Karena kita sudah mempunyai *project* di lokal komputer kita maka langkah selanjutnya kita cukup menghubungkannya dengan *repo* di server

Asumsinya, lokasi direktori *project* pada komputer penulis ada di

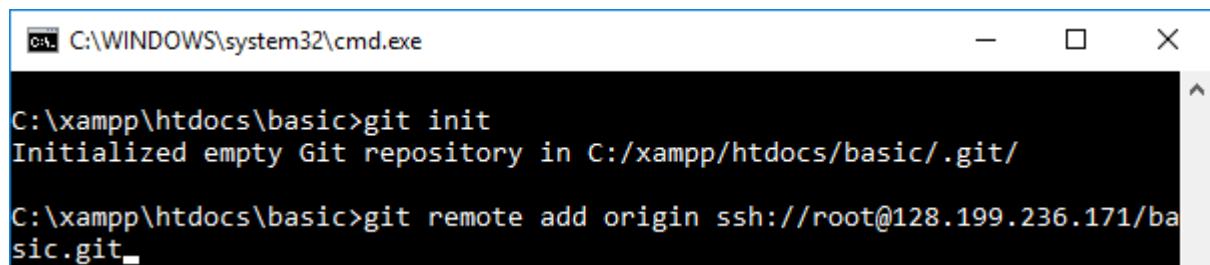
```
C:\xampp\htdocs\basic\
```

Maka melalui CMD di lokal komputer kita bisa jalankan perintah berikut

```
| git init
| git remote add origin ssh://root@128.199.236.171/basic.git
```

Jika kita menggunakan Github maka kodennya menjadi seperti berikut:

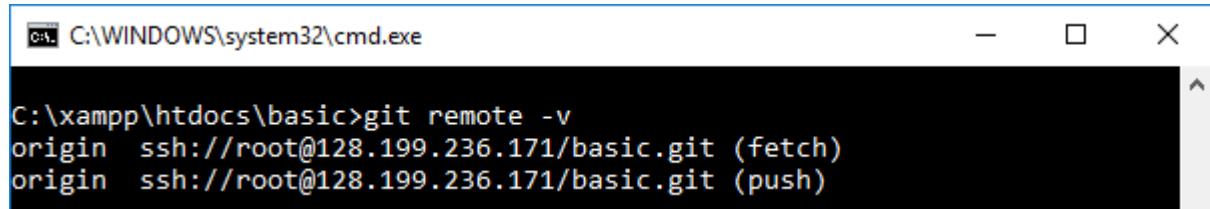
```
git remote add origin https://github.com/hscstudio/basic.git
```



```
C:\Windows\system32\cmd.exe
C:\xampp\htdocs\basic>git init
Initialized empty Git repository in C:/xampp/htdocs/basic/.git/
C:\xampp\htdocs\basic>git remote add origin ssh://root@128.199.236.171/basic.git.
```

Kita bisa mengecek *remote repo* Git dengan menggunakan perintah

```
| git remote -v
```



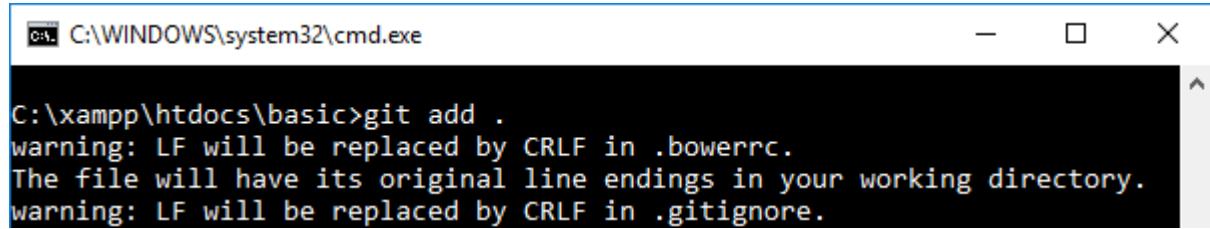
```
C:\Windows\system32\cmd.exe
C:\xampp\htdocs\basic>git remote -v
origin ssh://root@128.199.236.171/basic.git (fetch)
origin ssh://root@128.199.236.171/basic.git (push)
```

Untuk menghapus *remote repo* kita bisa gunakan perintah berikut.

```
| git remote rm origin
```

Sebelum kita *push project* kita ke server Git maka kita harus *commit* dulu semua *file project* kita.

```
| git add .
| git commit -m "init"
```



```
C:\Windows\system32\cmd.exe
C:\xampp\htdocs\basic>git add .
warning: LF will be replaced by CRLF in .bowerrc.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in .gitignore.
```

```
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in yii.bat.
The file will have its original line endings in your working directory.

C:\xampp\htdocs\basic>_
```

```
C:\Windows\system32\cmd.exe

C:\xampp\htdocs\basic>git commit -m "init"
[master (root-commit) 47d7c90] init
warning: LF will be replaced by CRLF in .bowerrc.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in .gitignore.

create mode 100644 web/index.php
create mode 100644 web/robots.txt
create mode 100644 yii
create mode 100644 yii.bat

C:\xampp\htdocs\basic>_
```

Jika sudah di-*commit*, maka kita bisa *pull* menggunakan perintah

```
| git push origin master
```

```
C:\Windows\system32\cmd.exe

C:\xampp\htdocs\basic>git push origin master
git push origin master
root@128.199.236.171's password:
Counting objects: 92, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (76/75), done.
Writing objects: 100% (92/92), 30.72 KiB | 0 bytes/s, done.
Total 92 (delta 6). reused 0 (delta 0)
To ssh://root@128.199.236.171/basic.git
 * [new branch] master -> master

C:\xampp\htdocs\basic>_
```

### *Clone Repo ke Root Web Server*

Setelah *project* terunggah ke *repo Git* maka langkah selanjutnya adalah meng-*clone* *repo* tersebut ke *root web server* di mana pada server penulis terletak di /var/www/html. *Tools* yang digunakan untuk melakukan proses ini adalah Putty. Lakukan *git clone URL repo* di server dengan menjalankan perintah berikut

```
| cd /var/www/html
| git clone ssh://root@128.199.236.171/basic.git
```

```
root@development:/# cd /var/www/html
root@development:/var/www/html# git clone ssh://root@128.199.236.171/basic.git
Cloning into 'basic'...
root@128.199.236.171's password:
remote: Counting objects: 92, done.
remote: Compressing objects: 100% (76/76) done.
Receiving objects: 100% (92/92), 30.72 KiB | 0
bytes/s, done.
Resolving deltas: 100% (6/6), done.
remote: Total 92 (delta 6), reused 0 (delta 0)
Checking connectivity... done.
```

Selanjutnya, mari kita cek apakah *project basic* kita telah terunggah di *web root*. Masuk ke *folder basic* dan gunakan perintah *dir*.

```
| cd basic
| dir
```

```
root@development:/var/www/html/basic# cd basic
root@development:/var/www/html/basic# dir
assets controllers requirements.php yii
commands LICENSE.md runtime yii.bat
composer.json mail tests
composer.lock models views
config README.md web
root@development:/var/www/html/basic#
```

Ya, gambar yang berisi daftar *file* dan *folder* pada direktori /var/www/html/basic di atas menunjukkan bahwa *project* kita telah berhasil terunggah ke server.

#### Catatan.

*cd (change directory)* adalah perintah untuk berpindah ke suatu direktori. Anda seharusnya belajar tentang dasar-dasar perintah di Linux karena pada buku ini tidak akan dijelaskan secara mendetail.

### Perintah Composer Install pada Server

Pada saat kita melakukan *git push* maka ada *file* atau *folder* yang dikecualikan sehingga tidak ikut terunggah yaitu

```
/vendor
```

Oleh karena itu kita perlu menggunakan Composer berikut untuk melakukan instalasi semua *library* yang diperlukan dalam *project* kita. Jalankan perintah berikut.

```
| composer global require "fxp/composer-asset-plugin:1.1.1"
```

Pada saat panduan ini dibuat Composer Asset Plugin masih versi 1.1.1, silahkan disesuaikan dengan versi terakhir.

```
root@development:/var/www/html# composer global require "fxp/composer-asset-plugin:1.1.1"
Changed current directory to /root/.composer
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Installing fxp/composer-asset-plugin (v1.1.1)
 Downloading: 100%
Writing lock file
Generating autoload files
```

Setelah Composer Asset Plugin terinstalasi dengan baik maka selanjutnya kita menginstalasi *library* lain yang dibutuhkan dalam *project* ini yaitu dengan menggunakan perintah berikut.

```
| composer install
```

```
root@development: /var/www/html/basic
root@development:/var/www/html# cd basic
root@development:/var/www/html/basic# composer
install
Loading composer repositories with package
information
Installing dependecies (including require-dev) from
lock files
- Installing yiisoft/yii2-composer (2.0.3)
 Downloading: 100%
- Installing bower-asset/jquery (2.1.4)
 Downloading: 100%
```

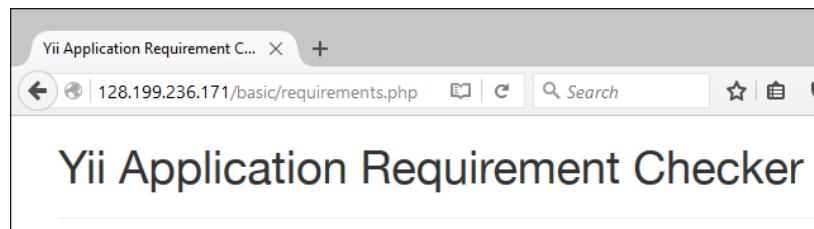
### **Check Requirements**

Untuk memastikan bahwa *project* dapat berjalan dengan baik di server, maka kita perlu mengecek kebutuhan minimum untuk *project* kita dengan cara mengakses URL berikut

```
∞ http://URL_VPS/basic/requirements.php
```

atau pada *droplet* penulis

```
∞ http://128.199.236.171/basic/requirements.php
```



Pada *droplet* penulis, ada beberapa *warning* yaitu: Intl extension, ICU version, PDO MySQL extension, Memcache extension, APC extension, ImageMagick PHP extension with PNG support, Expose PHP.

Mari kita bereskan satu persatu

1. *Warning* Intl Extension & ICU Version

Untuk mengatasi *warning* ini, maka kita perlu menginstalasi *extension* Intl yang memang belum ada.

```
| sudo apt-get install php5-intl
```

2. PDO MySQL extension

Untuk mengatasi *warning* ini, maka kita perlu menginstalasi *extension* MySQL yang memang belum ada.

```
| sudo apt-get install php5-mysql
```

Sedangkan untuk basis data SQLite kita bisa gunakan *extension* php5-sqlite dan untuk basisi data PostgreSQL *extension*-nya adalah php5-pgsql.

3. Memcached & APC Extension

Untuk mengatasi *warning* ini, maka kita perlu menginstalasi *extension* memcached.

```
| apt-get install php5-memcached
| sudo apt-get install php-apc
```

4. ImageMagick

Untuk mengatasi *warning* ini, maka kita perlu menginstalasi *extension* imagick.

```
| sudo apt-get install php5-imagick
| sudo php5enmod imagick
```

5. Expose PHP

Untuk mengatasi warning ini dengan cara mengubah nilai dari variabel expose\_php pada php.ini yang semula *On* menjadi *Off*

```
| vi /etc/php5/apache2/php.ini
```

Setelah semua diperbarui jangan lupa untuk me-restart server

```
| sudo service apache2 restart
```

Selesai, dan silahkan dicek kembali *requirements*-nya.

### **Membuat Basis Data di Server**

Berhubung kita tidak menginstalasi *tools* phpmyadmin ke server ini maka kita bisa gunakan console untuk membuat basis data. Pada console, jalankan perintah berikut.

```
| mysql -u root -p
| create database yii2basic;
```

Masukkan *password* MySQL sesuai dengan *password* ketika anda menginstalasi MySQL di server.

The screenshot shows a terminal window titled "root@development: /". The user has run the command "mysql -u root -p" and entered their password. They then run "create database yii2basic;". The terminal output includes the MySQL monitor welcome message, copyright information, and a note about trademarks. It ends with the confirmation "Query OK, 1 row affected (0.00 sec)".

```
root@development:/# mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or
\g.
Your MySQL connection id is 11
Server version: 5.6.46-0ubuntu0.14.04.2 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its
affiliates. All rights reserved.

Oracle is a registered trademark of Oracle
Corporation and/or its
affiliates. Other names may be trademarks of their
respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the
current input statement.

mysql> create database yii2basic;
Query OK, 1 row affected (0.00 sec)
```

Pada contoh ini, kita membuat basis data dengan nama "yii2basic".

### **Konfigurasi**

Setelah project berhasil diunggah dan basis data berhasil dibuat maka langkah selanjutnya adalah melakukan konfigurasi terhadap server berkaitan dengan *project* kita.

#### **Set Privileges Folder**

Karena Yii membutuhkan *privileges write* untuk memodifikasi *folder runtime*, maka kita bisa *set* agar *folder runtime*, */web/assets* bisa *write*. Untuk lebih mudahnya, kita bisa jalankan perintah berikut.

```
| chmod -R 777 /var/www/html/basic/runtime
| chmod -R 777 /var/www/html/basic/web/assets
```

Namun tentu saja cara di atas tidak disarankan, untuk keamanan kita bisa gunakan alternatif berikut.

```
| chown www-data /var/www/html/basic/runtime
| chown www-data /var/www/html/basic/web/assets
```

---

#### **Catatan:**

www-data adalah *user* Apache di Ubuntu, untuk OS atau distro Linux lain silakan disesuaikan

---

```
128.199.236.171 - PuTTY
root@development:/# chown www-data /var/www/html/basic/*
/runtime
root@development:/# chown www-data /var/www/html/basic/*
/web/assets
root@development:/#
```

### Konfigurasi Koneksi Basis Data

Agar aplikasi Yii kita dapat terhubung ke basis data maka pastikan konfigurasinya benar. Edit file koneksi basis data di server yaitu @app/config/db.php (contoh: /var/www/html/basic/config/db.php, sesuaikan dengan letak konfigurasi basis data pada server anda).

Untuk melakukan pengeditan, maka kita akan menggunakan tools editor sederhana namun cukup powerful yang merupakan editor bawaan Linux berbasis CMD yaitu vi. Jalankan perintah berikut

```
| cd /var/www/html/basic
| vi config/db.php
```

```
128.199.236.171 - PuTTY
root@development:/var/www/html/basic# vi config/db.php ^

<?php

return [
 'class' => 'yii\db\Connection',
 'dsn' => 'mysql:host=localhost;dbname=yii2basic',
 'username' => 'root',
 'password' => '123456',
 'charset' => 'utf8',
];
```

### Panduan Menggunakan Vi

Berikut ini perintah-perintah yang bisa digunakan pada vi editor

- Tekan tombol INSERT/INS pada keyboard, untuk mengubah mode read menjadi write, sehingga ada bisa mengedit. Anda juga bisa melakukan salin tempel teks dari luar, dengan cara mengklik kanan mouse pada area tujuan kemudian tempel.
- Tekan tombol ESCAPE/ESC pada keyboard, untuk mengubah mode tulis menjadi baca
- Pada mode baca, tekan tombol slash (/) diikuti dengan teks tertentu untuk melakukan pencarian teks pada file yang dibuka tersebut.
- Pada mode baca tekan titik dua (: ) diikuti dengan huruf Q untuk keluar dari file. (Q = quite).
- Pada mode baca tekan titik dua (: ) diikuti dengan huruf WQ! untuk menyimpan perubahan dan keluar dari file (W= Write, != force)

Jika konfigurasi basis data sudah kita sesuaikan, maka langkah selanjutnya adalah menjalankan migrasi basis data.

```
| yii migrate/up
```

Atau jika kita mempunyai file SQL maka kita bisa mengimportnya menggunakan perintah berikut

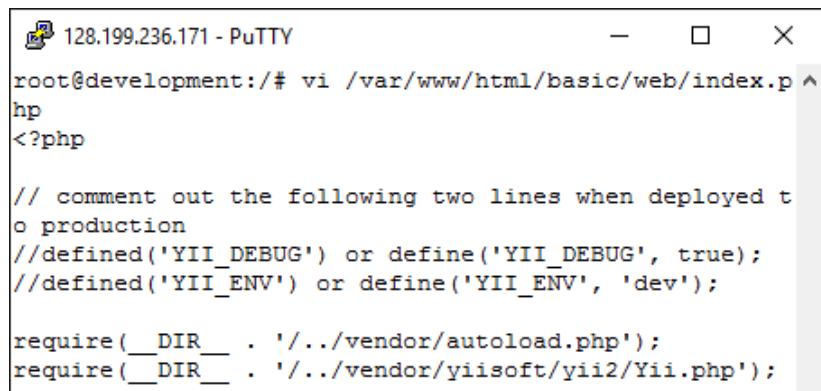
```
| mysql -u root -p yii2basic < /var/www/html/basic/file.sql
```

Silakan disesuaikan dengan lokasi file SQL anda dan nama basis datanya.

### Konfigurasi Entry Script

Edit file entry script di server yaitu file @app/web/index.php (/var/www/html/basic/web/index.php), comment dua baris pertama.

```
| vi /var/www/html/basic/web/index.php
```



```
root@development:/# vi /var/www/html/basic/web/index.php
hp
<?php

// comment out the following two lines when deployed to production
//defined('YII_DEBUG') or define('YII_DEBUG', true);
//defined('YII_ENV') or define('YII_ENV', 'dev');

require(__DIR__ . '/../vendor/autoload.php');
require(__DIR__ . '/../vendor/yiisoft/yii2/Yii.php');
```

Langkah ini dimaksudkan untuk menonaktifkan mode *debug* dan mengeset *environment* menjadi *production*.

### Konfigurasi Web Root

Langkah selanjutnya adalah memindahkan *web root* dari yang sebelumnya `/var/www/html` menjadi `/var/www/html/basic/web`. Hal ini dimaksudkan agar ketika kita mengakses URL *web* kita (`http://URL_VPS/`) maka kita langsung mendapatkan tampilan situs web, tanpa perlu menambahkan `/basic/web` atau `/web` dibelakang URL.

Pada CMD Putty, kita jalankan perintah berikut.

```
| vi /etc/apache2/sites-available/000-default.conf
```



```
#ServerName www.example.com
ServerName 128.199.236.171
ServerAdmin hafidmukhlasin@gmail.com
DocumentRoot /var/www/html/basic/web
```

Pada `ServerName` kita tentukan IP *address* (public) dari server kita kemudian `DocumentRoot` dengan `/var/www/html/basic/web`. Tentunya jika kita telah mempunyai nama domain maka ubah `ServerName` menjadi nama *domain* kita (`www.example.com`)

Jika kita menggunakan *Pretty URL* maka kita perlu *set* pada *file apache.conf* yaitu dengan menambahkan atau mengedit jika sudah ada pada bagian berikut.

```
<Directory /var/www/html/basic/web>
Options +FollowSymLinks
IndexIgnore /*

RewriteEngine on

if a directory or a file exists, use it directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

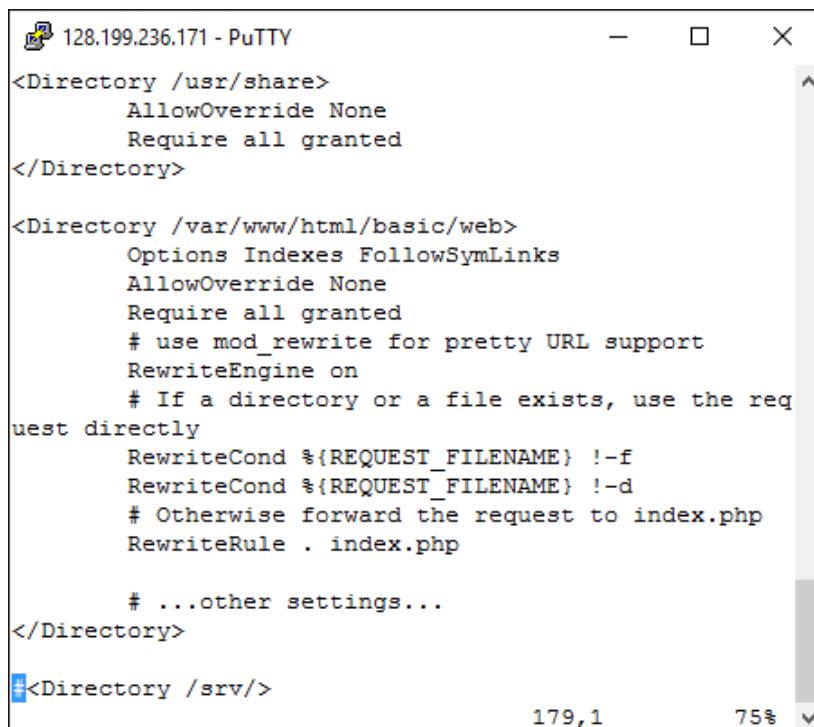
otherwise forward it to index.php
RewriteRule . index.php
</Directory>
```

Konfigurasi ini sebagaimana yang direkomendasikan oleh Yii

∞ <http://www.yiiframework.com/doc-2.0/guide-tutorial-shared-hosting.html#add-extras-for-webserver>

Mari kita buka `apache.conf` menggunakan *editor vi*.

```
| vi /etc/apache2/apache.conf
```



```

128.199.236.171 - PuTTY
<Directory /usr/share>
 AllowOverride None
 Require all granted
</Directory>

<Directory /var/www/html/basic/web>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
 # use mod_rewrite for pretty URL support
 RewriteEngine on
 # If a directory or a file exists, use the request directly
 RewriteCond %{REQUEST_FILENAME} !-f
 RewriteCond %{REQUEST_FILENAME} !-d
 # Otherwise forward the request to index.php
 RewriteRule . index.php

 # ...other settings...
</Directory>

#<Directory /srv/>

```

179,1 75% ▼

Jangan lupa untuk *me-restart* server Apache, sehingga seluruh konfigurasi kita akan di *build* ulang.

| sudo service apache2 restart

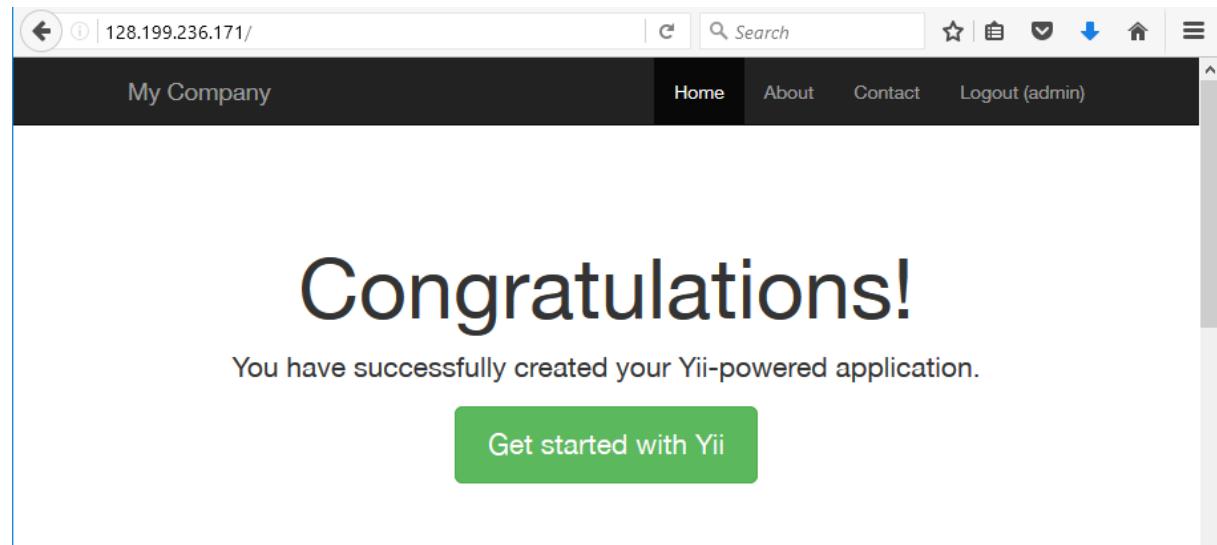
Selengkapnya tentang hal ini bisa kita lihat pada tautan berikut.

∞ <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>

### ***Uji Coba***

Setelah melakukan semua langkah di atas, maka untuk selanjutnya kita bisa mengakses situs web kita langsung menggunakan *domain* atau IP-nya.

∞ <http://128.199.236.171/>



Berhasil!

### ***Whats Next***

Apa yang akan kita lakukan setelah ini?. Ada dua *file* yang seharusnya tidak perlu diubah lagi yaitu

/config/db.php

/web/index.php

Hal ini karena kedua file tersebut berisi konfigurasi *production* yang berbeda dengan konfigurasi *development*, seperti *setting* basis data yang berbeda. Oleh karena itu untuk mencegah terjadinya perubahan karena ter-*replace* ketika proses sinkronisasi project maka kita perlu menambahkan kedua file di atas ke dalam daftar file yang dikecualikan untuk diperbarui yaitu pada file *.gitignore*.

```
| /config/db.php
| /web/index.php
```

Sebuah aplikasi tentu akan terus berkembang, misalnya penambahan fitur, perbaikan tampilan, perubahan konten dsb, dimana hal itu memerlukan perubahan kode sumber aplikasi. Tapi ingat bahwa kode yang kita edit bukanlah yang di server melainkan kode yang berada di lokal komputer kita atau komputer *development*. Perubahan tersebut perlu kita *test* dahulu untuk memastikan bebas galat sebelum diunggah ke server *production*. Editing kode secara langsung di server sebaiknya tidak dilakukan karena sangat beresiko.

Langkah yang perlu kita lakukan setelah melakukan perubahan kode di lokal komputer adalah dengan meng-*commit* perubahan tersebut serta mengirimkannya atau mem-*push* ke Git *repository*.

Berikut ini perintah yang bisa kita jalankan di CMD lokal.

```
| git add .
| git commit -m "init"
| git push origin master
```

Jika perintah di atas berhasil lalu berarti kode perubahan kita terakhir telah masuk ke *repo* Git kita, maka langkah selanjutnya adalah menarik atau meng-copy kode dari *repo* Git ke *web root* aplikasi kita di server. Caranya adalah dengan menjalankan perintah *git pull* di server melalui CMD Putty.

```
| git pull
```

```
root@development: /var/www/html/basic
root@development:/var/www/html/basic# git pull
root@128.199.236.171's password:
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (4/4), done
remote: Total 4 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done
From ssh://128.199.236.171/basic
 479bca5..edafad7 master -> origin/master
Updating 479bca5..edafad7
Fast-forward
 .gitignore | 3 +++
 web/.htaccess | 7
 2 files changed, 3 insertions(+), 7 deletions(-)
 delete mode 100644 web/.htaccess
root@development:/var/www/html/basic#
```

Pada contoh di atas diketahui bahwa ada 2 file diubah, 3 file ditambahkan, dan 7 file dihapus. Pastikan data tersebut sesuai dengan perubahan yang kita lakukan di lokal komputer.

Jadi pada intinya, ketika terjadi perubahan kode sumber lagi maka cukup dengan menjalankan perintah *git push* di CMD lokal komputer kemudian menjalankan perintah *git pull* di CMD Putty server. Sehingga tidak perlu mengulangi langkah awal lagi. Lebih mudah bukan?

#### **Perhatian**

Ketika terjadi *update* atau penambahan *library* maka di samping menjalankan *git pull*, kita juga perlu menjalankan perintah *composer install* pada server

## C. Kesimpulan

Kita bisa men-deploy aplikasi Yii kita pada *shared hosting* maupun VPS dengan menggunakan berbagai *tools* dan skenario. *Tools* yang digunakan pada panduan ini adalah FTP untuk *shared hosting* sedangkan Git & Composer digunakan untuk VPS.

Materi dalam buku ini hanyalah panduan dasar. Oleh karena itu untuk mendalami pengaturan server yang efisien dan *secure*, tentu anda harus mencari sumber lain yang pembahasannya lebih mendalam.

Pekerjaan *deploy* aplikasi yang dilakukan secara berulang kali ini, kemudian ada pihak yang berinisiatif membuatkan *tools* untuk mengotomatisasinya. Ada banyak *tools* untuk mengotomatisasi *deploy* aplikasi, salah satunya adalah Travis CI yang juga terintegrasi dengan *code testing*.

## Daftar Pustaka

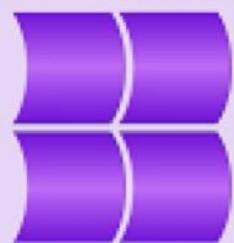
- Mukhlasin, H. 2016. www.hafidmukhlasin.com [daring]. Available at: <http://www.hafidmukhlasin.com> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.yiiframework.com [daring]. Available at: <http://www.yiiframework.com> [Diakses 30 Juli 2016]
- \_\_\_\_\_. 2016. www.github.com/yiisoft/yii2 [daring]. Available at: <http://github.com/yiisoft/yii2> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.getcomposer.org [daring]. Available at: <http://getcomposer.org> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.packagist.org [daring]. Available at: <http://packagist.org> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.codeception.org [daring]. Available at: <http://codeception.org> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.highchart.org [daring]. Available at: <http://highchart.org> [Diakses 14 Februari 2016]
- \_\_\_\_\_. 2016. www.digitalocean.org [daring]. Available at: <http://digitalocean.org> [Diakses 14 Februari 2016]

- Persiapan Lingkungan Kerja
- Bekerja dengan Composer
- Pengenalan Yii Framework
- Model View Controller
- Bekerja dengan Basis Data
- Gii: Code Generator
- Ajax dan Pjax
- Layout dan Module Aplikasi
- Bekerja dengan Extension
- Code Testing
- Otentikasi Pengguna
- Otorisasi Pengguna
- Unggah, Impor & Pelaporan
- Web Service
- Tips dan Trik
- Deploying Aplikasi

ISBN 978-602-1018-12-5



9 786021 018125



BUKU BAIK