

Panduan Definitif Untuk Yii 2.0

<http://www.yiiframework.com/doc/guide>

Qiang Xue,
Alexander Makarov,
Carsten Brandt,
Klimov Paul,
and
many contributors from the Yii community

Bahasa Indonesia translation provided by:
Misbahul D. Munir,
Muhammad Cahya,
Seto Kuslaksono,
Novrian Y.F.

This tutorial is released under the [Terms of Yii Documentation](#).

Copyright 2014 Yii Software LLC. All Rights Reserved.

Daftar Isi

1	Pengantar	1
1.1	Apa Itu Yii	1
1.2	Upgrade dari Versi 1.1	3
2	Mulai	15
2.1	Instalasi Yii	15
2.2	Menjalankan Aplikasi	21
2.3	Katakan Hello	25
2.4	Bekerja dengan Form	28
2.5	Bekerja dengan Database	35
2.6	Membuat Kode menggunakan Gii	41
2.7	Menatap ke Depan	48
3	Struktur Aplikasi	51
3.1	Tinjauan	51
3.2	Skrip Masuk	52
3.3	Aplikasi	54
3.4	Komponen Aplikasi	67
4	Penanganan Permintaan	79
5	Konsep Pokok	89
6	Bekerja dengan Database	99
7	Mendapatkan Data dari Pengguna	105
8	Menampilkan Data	111
9	Keamanan	119
10	Caching	127
11	Layanan Web RESTful	133

12 Alat Pengembangan	143
13 Pengujian	145
14 Topik Khusus	153
15 Widget	163
16 Alat Bantu	165

Bab 1

Pengantar

1.1 Apa Itu Yii

Yii merupakan kerangka kerja berbasis komponen yang berkecepatan tinggi (high performance) yang digunakan untuk kasus pengembangan yang cepat pada aplikasi Web Modern. Disebut Yii (diucapkan **Yee** atau [ji:]) yang berarti “sederhana dan berevolusi” dalam bahasa Cina. Bisa juga dianggap sebagai (akronim) singkatan dari **Yes It Is (Ya, Itu Dia)!**

1.1.1 Yii Paling Cocok Digunakan untuk Apa?

Yii adalah kerangka kerja pemrograman web yg umum(general), ini berarti kita bisa menggunakannya untuk membangun berbagai macam aplikasi berbasis web menggunakan PHP. Karena merupakan arsitektur berbasis komponen yang mana memiliki dukungan caching yang canggih, Yii sangat cocok untuk pengembangan aplikasi skala besar seperti web portal, forum, CMS, e-commerce, REST API dan lain sebagainya.

1.1.2 Bagaimana jika Yii Dibandingkan dengan Frameworks lain?

Jika Anda sudah familiar dengan framework lain, Anda mungkin akan bersyukur ketika membandingkan Yii dengan yang lain:

- Seperti kebanyakan PHP framework, Yii mengimplementasikan pola arsitektur MVC (Model-View-Controller) dan mempromosikan kode organisasi berdasarkan pola itu.
- Yii mengambil filosofi bahwa kode harus ditulis dengan cara sederhana namun elegan. Yii tidak akan pernah mencoba untuk mendesain berlebihan terutama untuk mengikuti beberapa pola desain secara ketat.
- Yii adalah fullstack framework yang menyediakan banyak fitur teruji dan siap pakai seperti: query builder dan ActiveRecord baik untuk

relasional maupun NoSQL database; dukungan pengembangan REST API; dukungan caching berlapis dan masih banyak lagi.

- Yii sangat extensible. Anda dapat menyesuaikan atau mengganti hampir setiap bagian dari kode inti Yii. Anda juga bisa mengambil keuntungan dari arsitektur ekstensi Yii yang solid untuk menggunakan atau mengembangkan ekstensi untuk disebarluaskan kembali.
- Kinerja tinggi (High performance) selalu menjadi tujuan utama dari Yii.

Yii tidak dikerjakan oleh satu orang, Yii didukung oleh tim pengembang inti yang kuat¹, serta komunitas besar profesional yang terus memberikan kontribusi bagi pengembangan Yii. Tim pengembang Yii terus mengamati perkembangan tren terbaru Web, pada penerapan terbaik (best practices) serta fitur yang ditemukan dalam framework dan proyek lain. Penerapan terbaik yang paling relevan dan fitur yang ditemukan di tempat lain secara teratur dimasukkan ke dalam kerangka inti dan menampilkannya melalui antarmuka yang sederhana dan elegan.

1.1.3 Versi Yii

Yii saat ini memiliki dua versi utama yang tersedia: 1.1 dan 2.0. Versi 1.1 adalah generasi lama dan sekarang dalam mode pemeliharaan. Versi 2.0 adalah penulisan ulang lengkap dari Yii, mengadopsi teknologi dan protokol terbaru, termasuk composer, PSR, namespace, trait, dan sebagainya. Versi 2.0 merupakan generasi framework yang sekarang dan terus menerima upaya pengembangan selama beberapa tahun ke depan. Panduan ini terutama tentang versi 2.0.

1.1.4 Persyaratan dan Prasyarat

Yii 2.0 memerlukan PHP 7.3.0 atau versi lebih tinggi. Anda dapat menemukan persyaratan yang lebih rinci untuk setiap fitur dengan menjalankan pengecek persyaratan yang diikutsertakan dalam setiap rilis Yii.

Menggunakan Yii memerlukan pengetahuan dasar tentang pemrograman berorientasi objek (OOP), mengingat Yii adalah framework berbasis OOP murni. Yii 2.0 juga memanfaatkan fitur terbaru dari PHP, seperti namespace² dan traits³. Memahami konsep-konsep ini akan membantu Anda lebih mudah memahami Yii 2.0.

¹<https://www.yiiframework.com/team>

²<https://www.php.net/manual/en/language.namespaces.php>

³<https://www.php.net/manual/en/language.oop5.traits.php>

1.2 Upgrade dari Versi 1.1

Ada banyak perbedaan antara versi 1.1 dan 2.0 karena Yii Framework benar-benar ditulis ulang di versi 2.0. Akibatnya, upgrade dari versi 1.1 tidak mudah seperti upgrade untuk versi minor. Dalam panduan ini Anda akan menemukan perbedaan utama antara dua versi.

Jika Anda belum pernah menggunakan Yii 1.1 sebelumnya, Anda dapat dengan aman melewati bagian ini dan menuju ke “[Persiapan](#)”.

Harap dicatat bahwa Yii 2.0 memperkenalkan lebih banyak fitur baru dari yang tercakup dalam ringkasan ini. Sangat dianjurkan Anda membaca keseluruhan panduan definitif untuk mempelajari hal tersebut. Ada kemungkinan bahwa beberapa fitur yang sebelumnya harus anda kembangkan sendiri, kini telah menjadi bagian dari kode inti.

1.2.1 Instalasi

Yii 2.0 sepenuhnya menggunakan composer⁴, yaitu dependency manager yang sudah diakui oleh PHP.

- Instalasi dari kerangka inti serta ekstensi, ditangani melalui Composer.
- Silakan merujuk ke bagian [Instalasi Yii](#) untuk belajar cara menginstal Yii 2.0.
- Jika ingin membuat ekstensi baru, atau mengubah/memperbarui ekstensi 1.1 yang telah Anda buat ke ekstensi 2.0 (agar kompatibel), silakan merujuk pada panduan Membuat Ekstensi.

1.2.2 Persyaratan PHP

Yii 2.0 membutuhkan PHP 5.4 atau versi yang lebih tinggi, ini karena ada perubahan besar atas PHP versi 5.2 yang sebelumnya dibutuhkan oleh Yii 1.1. Akibatnya, ada banyak perbedaan pada tingkat bahasa yang harus Anda perhatikan. Dibawah ini ringkasan perubahan utama mengenai PHP tersebut:

- Namespaces⁵.
- Anonymous fungsi⁶.
- Sintaks array pendek [... elemen ...] digunakan sebagai pengganti array (... elemen ...).
- Tags echo pendek <= digunakan dalam tampilan file. Ini aman digunakan mulai dari PHP 5.4.
- Class SPL dan interface⁷.
- Late Static Bindings⁸.

⁴<https://getcomposer.org/>

⁵<https://www.php.net/manual/en/language.namespaces.php>

⁶<https://www.php.net/manual/en/functions.anonymous.php>

⁷<https://www.php.net/manual/en/book.spl.php>

⁸<https://www.php.net/manual/en/language.oop5.late-static-bindings.php>

- Tanggal dan Waktu⁹.
- Traits¹⁰.
- Intl¹¹. Yii 2.0 menggunakan ekstensi PHP intl untuk mendukung fitur internasionalisasi.

1.2.3 Namespace

Perubahan yang paling terlihat jelas dalam Yii 2.0 adalah penggunaan namespace. Hampir setiap kelas inti menggunakan namespace, misalnya, `yii\web\Request`. Awalan yang sebelumnya menggunakan huruf “C” tidak lagi digunakan dalam nama kelas. Skema penamaan sekarang mengikuti struktur direktori. Misalnya, `yii\web\Request` mengindikasikan bahwa file kelasnya adalah `web/Request.php` sesuai dengan lokasi folder framework Yii yang ada.

(Anda dapat menggunakan setiap kelas inti tanpa menyertakannya secara eksplisit berkat Yii class loader.)

1.2.4 Komponen dan Object

Yii 2.0 membagi kelas `CComponent` di 1.1 menjadi dua kelas: `yii\base\BaseObject` dan `yii\base\Component`. Class `BaseObject` adalah class dasar ringan yang memungkinkan mendefinisikan objek properti melalui getter dan setter. Class `Component` adalah perluasan(`extends`) dari `BaseObject` dengan dukungan `Event` dan `Behavior`.

Jika class Anda tidak memerlukan fitur event atau behavior, Anda harus mempertimbangkan menggunakan `BaseObject` sebagai class dasar. Hal ini biasanya terjadi untuk class yang mewakili struktur data dasar.

1.2.5 Konfigurasi objek

Class `BaseObject` memperkenalkan cara seragam untuk mengkonfigurasi objek. Setiap class turunan dari `BaseObject` harus menyatakan konstruktor (jika diperlukan) dengan cara berikut agar dapat dikonfigurasi dengan benar:

```
class MyClass extends \yii\base\BaseObject
{
    public function __construct($param1, $param2, $config = [])
    {
        // ... inisialisasi sebelum konfigurasi diterapkan

        parent::__construct($config);
    }

    public function init()
```

⁹<https://www.php.net/manual/en/book.datetime.php>

¹⁰<https://www.php.net/manual/en/language.oop5.traits.php>

¹¹<https://www.php.net/manual/en/book.intl.php>

```

{
    parent::init();

    // ... inisialisasi setelah konfigurasi diterapkan
}
}

```

Dalam contoh diatas, parameter terakhir dari konstruktor harus mengambil array konfigurasi (`$config`) yang berisi pasangan nama-nilai [`key => value`] untuk meng-inisialisasi properti pada akhir konstruktor. Anda dapat mengganti method `init()` untuk melakukan inisialisasi yang harus dilakukan setelah konfigurasi diterapkan.

Dengan mengikuti ketentuan ini, Anda akan dapat membuat dan mengkonfigurasi objek baru menggunakan array konfigurasi:

```

$object = Yii::createObject([
    'class' => 'MyClass',
    'property1' => 'abc',
    'property2' => 'cde',
], [$param1, $param2]);

```

Rincian lebih lanjut mengenai konfigurasi dapat ditemukan pada bagian [Konfigurasi](#).

1.2.6 Event

Di Yii 1, event dibuat dengan mendefinisikan method `on` (misalnya, `onBeforeSave`). Di Yii 2, Anda sekarang dapat menggunakan semua nama sebagai event. Untuk memicu suatu event, Anda dapat melakukannya dengan memanggil method `trigger()`:

```

$event = new \yii\base\Event;
$component->trigger($eventName, $event);

```

Untuk melekatkan/memasang handler pada suatu event, dapat dilakukan dengan menggunakan method `on()`:

```

$component->on($eventName, $handler);
// Untuk mematikannya dapat dilakukan dengan cara:
// $component->off($eventName, $handler);

```

Ada banyak pengembangan dari fitur event. Untuk lebih jelasnya, silakan lihat pada bagian [Event](#).

1.2.7 Path Alias

Yii 2.0 memperluas penggunaan alias path baik untuk file/direktori maupun URL. Yii 2.0 juga sekarang mensyaratkan nama alias dimulai dengan karakter `@`. Misalnya, alias `@yii` mengacu pada direktori instalasi Yii. Alias path

didukung di sebagian besar tempat di kode inti Yii. Misalnya, `yii\caching\FileCache::$cachePath` dapat mengambil baik alias path maupun direktori normal.

Sebuah alias juga terkait erat dengan namespace kelas. Disarankan alias didefinisikan untuk setiap akar namespace, sehingga memungkinkan Anda untuk menggunakan autoloader class Yii tanpa konfigurasi lebih lanjut. Misalnya, karena `@yii` mengacu pada direktori instalasi Yii, class seperti `yii\web\Request` dapat otomatis diambil. Jika Anda menggunakan librari pihak ketiga seperti Zend Framework. Anda dapat menentukan alias path `@Zend` yang mengacu pada direktori instalasi framework direktori. Setelah Anda selesai melakukannya, Yii akan dapat menload setiap class dalam librari Zend Framework.

Lebih jauh tentang alias path dapat pelajari pada bagian [Alias](#).

1.2.8 View

Perubahan yang paling signifikan tentang view di Yii 2 adalah bahwa variabel khusus `$this` dalam sebuah view tidak lagi mengacu controller saat ini atau widget. Sebaliknya, `$this` sekarang mengacu pada objek *view*, konsep baru yang diperkenalkan di 2.0. Objek *view* adalah `yii\web\View`, yang merupakan bagian view dari pola MVC. Jika Anda ingin mengakses controller atau widget di tampilan, Anda dapat menggunakan `$this->context`.

Untuk membuat tampilan parsial dalam view lain, Anda menggunakan `$this->render()`, tidak lagi `$this->renderPartial()`. Panggilan untuk `render` juga sekarang harus secara eksplisit *di-echo*-kan, mengingat method `render()` sekarang mengembalikan nilai yang dirender, bukan langsung menampilkannya. Sebagai contoh:

```
echo $this->render('_item', ['item' => $item]);
```

Selain menggunakan PHP sebagai bahasa template utama, Yii 2.0 juga dilengkapi dengan dukungan resmi dua *template engine* populer: Smarty dan Twig. *Template engine* Prado tidak lagi didukung. Untuk menggunakan mesin template ini, Anda perlu mengkonfigurasi komponen aplikasi *view* dengan menetapkan properti `View::$renderers`. Silakan merujuk ke bagian [Template Engine](#) untuk lebih jelasnya.

1.2.9 Model

Yii 2.0 menggunakan `yii\base\Model` sebagai model dasar, mirip dengan `CModel` di 1.1. class `CFormModel` telah dibuang seluruhnya. Sebaliknya, di Yii 2 Anda harus memperluas `yii\base\Model` untuk membuat class model formulir.

Yii 2.0 memperkenalkan metode baru yang disebut `yii\base\Model::scenario()` untuk menyatakan skenario yang didukung, dan untuk menunjukkan di mana skenario atribut perlu divalidasi serta atribut yang dapat dianggap sebagai aman atau tidak dll Sebagai contoh:

```
public function scenarios()
{
    return [
        'backend' => ['email', 'role'],
        'frontend' => ['email', '!role'],
    ];
}
```

Dalam contoh di atas, dua skenario dinyatakan: `backend` dan `frontend`. Untuk skenario `backend`, baik atribut `email` maupun `role` aman dan dapat diassign secara masal. Untuk skenario `frontend`, `email` dapat diassign secara masal sementara `role` tidak bisa. Kedua `email` dan `role` harus divalidasi sesuai aturan.

Method `rules()` ini masih digunakan untuk menyatakan aturan validasi. Perhatikan bahwa dengan dikenalkannya `yii\base\Model::scenario()` sekarang tidak ada lagi validator `unsafe`.

Dalam kebanyakan kasus, Anda tidak perlu menimpa `yii\base\Model::scenario()` jika method `rules()` sepenuhnya telah menentukan skenario yang akan ada dan jika tidak ada kebutuhan untuk menyatakan atribut `unsafe`.

Untuk mempelajari lebih lanjut tentang model, silakan merujuk ke bagian [Model](#).

1.2.10 Controller

Yii 2.0 menggunakan `yii\web\Controller` sebagai kelas dasar controller, yang mirip dengan `CController` di Yii 1.1. `yii\base\Action` adalah kelas dasar untuk kelas action.

Dampak paling nyata dari perubahan ini pada kode Anda adalah bahwa aksi kontroler harus mengembalikan nilai konten alih-alih menampilkannya:

```
public function actionView($id)
{
    $model = \app\models\Post::findOne($id);
    if ($model) {
        return $this->render('view', ['model' => $model]);
    } else {
        throw new \yii\web\NotFoundException;
    }
}
```

Silakan merujuk ke bagian [Controller](#) untuk rincian lebih lanjut tentang controller.

1.2.11 widget

Yii 2.0 menggunakan `yii\base\Widget` sebagai kelas dasar widget, mirip dengan `CWidget` di Yii 1.1.

Untuk mendapatkan dukungan yang lebih baik untuk kerangka di IDE, Yii 2.0 memperkenalkan sintaks baru untuk menggunakan widget. Metode statis `begin()`, `end()`, dan `widget()` mulai diperkenalkan, yang akan digunakan seperti:

```
use yii\widgets\Menu;
use yii\widgets\ActiveForm;

// Perlu diingat bahwa "echo" tetap diperlukan untuk menampilkan hasilnya
echo Menu::widget(['items' => $items]);

// Passing an array to initialize the object properties
$form = ActiveForm::begin([
    'options' => ['class' => 'form-horizontal'],
    'fieldConfig' => ['inputOptions' => ['class' => 'input-xlarge']],
]);
... form input fields here ...
ActiveForm::end();
```

Silakan merujuk ke bagian [Widget](#) untuk lebih jelasnya.

1.2.12 Tema

Tema benar-benar bekerja secara berbeda di 2.0. Kini tema telah berdasarkan mekanisme pemetaan path yang memetakan file sumber ke file tema. Misalnya, jika peta path untuk tema adalah `['/web/views' => '/web/themes/basic']`, maka versi tema dari file view `/web/views/site/index.php` akan menjadi `/web/themes/basic/site/index`. Untuk alasan ini, tema sekarang bisa diterapkan untuk setiap file view, bahkan view diberikan di luar controller ataupun widget.

Juga, tidak ada lagi komponen `CThemeManager`. Sebaliknya, `theme` adalah properti dikonfigurasi dari komponen `view` pada aplikasi.

Silakan merujuk ke bagian [Theming](#) untuk lebih jelasnya.

1.2.13 Aplikasi konsol (CLI)

Aplikasi konsol sekarang diatur sebagai controller seperti pada aplikasi Web. kontroler konsol harus diperluas dari `yii\console\Controller`, mirip dengan `CConsoleCommand` di 1.1.

Untuk menjalankan perintah konsol, menggunakan `yii <route>`, di mana `<route>` adalah rute kontroler (Misalnya `sitemap/index`). Argumen anonim tambahan dilewatkan sebagai parameter ke action controller yang sesuai, sedangkan argumen bernama diurai menurut deklarasi pada `yii\console\Controller::options()`.

Yii 2.0 mendukung pembuatan informasi bantuan command secara otomatis berdasarkan blok komentar.

Silakan lihat bagian [Console Commands](#) untuk lebih jelasnya.

1.2.14 I18N

Yii 2.0 menghilangkan formater tanggal dan angka terpasang bagian dari PECL modul intl PHP¹².

Penterjemahan pesan sekarang dilakukan melalui komponen aplikasi `i18n`. Komponen ini mengelola satu set sumber pesan, yang memungkinkan Anda untuk menggunakan pesan yang berbeda sumber berdasarkan kategori pesan.

Silakan merujuk ke bagian [Internasionalisasi](#) untuk rincian lebih lanjut.

1.2.15 Action Filter

Action Filter sekarang diimplementasikan melalui behavior. Untuk membuat baru, filter diperluas dari `yii\base\ActionFilter`. Untuk menggunakan filter, pasang Kelas filter untuk controller sebagai behavior. Misalnya, untuk menggunakan filter `yii\filters\AccessControl`, Anda harus mengikuti kode berikut di kontroler:

```
public function behaviors()
{
    return [
        'access' => [
            'class' => 'yii\filters\AccessControl',
            'rules' => [
                ['allow' => true, 'actions' => ['admin'], 'roles' => ['@']],
            ],
        ],
    ];
}
```

Silakan merujuk ke bagian [Filtering](#) untuk lebih jelasnya.

1.2.16 Aset

Yii 2.0 memperkenalkan konsep baru yang disebut *bundel aset* yang menggantikan konsep paket script di Yii 1.1.

Bundel aset adalah kumpulan file asset (misalnya file JavaScript, file CSS, file gambar, dll) dalam direktori. Setiap bundel aset direpresentasikan sebagai kelas turunan dari `yii\web\AssetBundle`. Dengan mendaftarkan bundel aset melalui `yii\web\AssetBundle::register()`, Anda membuat aset dalam bundel diakses melalui Web. Tidak seperti di Yii 1, halaman yang mendaftarkan bundel akan secara otomatis berisi referensi ke JavaScript dan file CSS yang ditentukan dalam bundel itu.

Silakan merujuk ke bagian [Managing Aset](#) untuk lebih jelasnya.

¹²<https://pecl.php.net/package/intl>

1.2.17 Helper

Yii 2.0 memperkenalkan banyak helper umum untuk digunakan, termasuk.

- yii\helpers\Html
- yii\helpers\ArrayHelper
- yii\helpers\StringHelper
- yii\helpers\FileHelper
- yii\helpers\Json

Silakan lihat bagian [Tinjauan Helper](#) untuk lebih jelasnya.

1.2.18 Formulir

Yii 2.0 memperkenalkan konsep *field* untuk membangun formulir menggunakan yii\widgets\ActiveForm. Field adalah wadah yang terdiri dari label, masukan, pesan kesalahan, dan atau teks petunjuk. Field diwakili sebagai objek ActiveForm. Menggunakan field, Anda dapat membangun formulir yang lebih bersih dari sebelumnya:

```
<?php $form = yii\widgets\ActiveForm::begin(); ?>
    <?= $form->field($model, 'username') ?>
    <?= $form->field($model, 'password')->passwordInput() ?>
    <div class="form-group">
        <?= Html::submitButton('Login') ?>
    </div>
<?php yii\widgets\ActiveForm::end(); ?>
```

Silakan merujuk ke bagian [Membuat Formulir](#) untuk lebih jelasnya.

1.2.19 Query Builder

Dalam 1.1, query builder itu tersebar di antara beberapa kelas, termasuk CDbCommand, CDbCriteria, dan CDbCommandBuilder. Yii 2.0 merepresentasikan sebuah query DB sebagai objek Query yang dapat berubah menjadi sebuah pernyataan SQL dengan bantuan QueryBuilder. Sebagai contoh:

```
$query = new \yii\db\Query();
$query->select('id, name')
    ->from('user')
    ->limit(10);

$command = $query->createCommand();
$sql = $command->sql;
$rows = $command->queryAll();
```

Yang terbaik dari semua itu adalah, query builder juga dapat digunakan ketika bekerja dengan [Active Record](#).

Silakan lihat bagian [Query Builder](#) untuk lebih jelasnya.

1.2.20 Active Record

Yii 2.0 memperkenalkan banyak perubahan [Active Record](#). Dua yang paling jelas melibatkan query builder dan penanganan permintaan relasional.

Kelas `CDbCriteria` di 1.1 digantikan oleh `yii\db\ActiveQuery` di Yii 2. Karena kelas tersebut adalah perluasan dari `yii\db\Query`, dengan demikian mewarisi semua metode query builder. Anda bisa memanggil `yii\db\ActiveRecord::find()` untuk mulai membangun query:

```
// Untuk mengambil semua customer yang *active* diurutkan sesuai ID:
$customers = Customer::find()
    ->where(['status' => $active])
    ->orderBy('id')
    ->all();
```

Untuk menyatakan suatu relasi, hanya dengan menentukan metod getter yang mengembalikan sebuah objek `ActiveQuery`. Nama properti yang didefinisikan oleh getter akan menjadi nama relasi. Misalnya, kode berikut mendeklarasikan sebuah relasi `orders` (di 1.1, Anda akan harus menyatakan relasi di tempat `relations()`):

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

Sekarang Anda dapat menggunakan `$customer->orders` untuk mengakses pesanan pelanggan dari tabel terkait. Anda juga dapat menggunakan kode berikut untuk melakukan permintaan relasi secara cepat dengan kondisi permintaan yang disesuaikan:

```
$orders = $customer->getOrders()->andWhere('status=1')->all();
```

Ketika ingin memuat relasi, Yii 2.0 melakukannya secara berbeda dari 1.1. Secara khusus, di 1.1 query JOIN akan dibuat untuk memilih data utama dan data relasi. Di Yii 2.0, dua pernyataan SQL dijalankan tanpa menggunakan JOIN: pernyataan pertama membawa kembali data utama dan yang kedua membawa kembali data relasi dengan menyaring sesuai kunci primer dari data utama.

Alih-alih mengembalikan objek `ActiveRecord`, Anda mungkin ingin menyambung dengan `asArray()` ketika membangun query untuk mendapatkan sejumlah besar data. Hal ini akan menyebabkan hasil query dikembalikan sebagai array, yang dapat secara signifikan mengurangi waktu CPU yang dibutuhkan dan memori jika terdapat sejumlah besar data. Sebagai contoh:

```
$customers = Customer::find()->asArray()->all();
```

Perubahan lain adalah bahwa Anda tidak dapat menentukan nilai default atribut melalui properti publik lagi. Jika Anda membutuhkan mereka, Anda harus mengatur mereka dalam metode init kelas record Anda.

```
public function init()
{
    parent::init();
    $this->status = self::STATUS_NEW;
}
```

Ada beberapa masalah dengan override konstruktor dari kelas ActiveRecord di 1.1. Ini tidak lagi hadir di versi 2.0. Perhatikan bahwa ketika menambahkan parameter ke constructor Anda mungkin harus mengganti `yii\db\ActiveRecord::instantiate()`.

Ada banyak perubahan lain dan perangkat tambahan untuk Rekaman Aktif. Silakan merujuk ke bagian [Rekaman Aktif](#) untuk rincian lebih lanjut.

1.2.21 Active Record Behaviors

Dalam 2.0, kami telah membuang kelas behavior dasar `CActiveRecordBehavior`. Jika Anda ingin membuat behavior Active Record, Anda akan harus memperluasnya langsung dari `yii\base\Behavior`. Jika kelas behavior perlu menanggapi beberapa event dari pemilik, Anda harus mengganti method `events()` seperti berikut ini,

```
namespace app\components;

use yii\db\ActiveRecord;
use yii\base\Behavior;

class MyBehavior extends Behavior
{
    // ...

    public function events()
    {
        return [
            ActiveRecord::EVENT_BEFORE_VALIDATE => 'beforeValidate',
        ];
    }

    public function beforeValidate($event)
    {
        // ...
    }
}
```

1.2.22 Pengguna dan IdentityInterface

Kelas `CWebUser` di 1.1 kini digantikan oleh `yii\web\User`, dan sekarang tidak ada lagi Kelas `CUserIdentity`. Sebaliknya, Anda harus menerapkan `yii\web\IdentityInterface` yang jauh lebih mudah untuk digunakan. Template proyek lanjutan memberikan contoh seperti itu.

Silakan merujuk ke bagian [Otentikasi](#), [Otorisasi](#), dan [Template Proyek Lanjutan](#)¹³ untuk lebih jelasnya.

1.2.23 Manajemen URL

Manajemen URL di Yii 2 mirip dengan yang di 1.1. Tambahan utamanya adalah, sekarang manajemen URL mendukung opsional parameter. Misalnya, jika Anda memiliki aturan dinyatakan sebagai berikut, maka akan cocok baik dengan `post/popular` maupun `post/1/popular`. Dalam 1.1, Anda akan harus menggunakan dua aturan untuk mencapai tujuan yang sama.

```
[
    'pattern' => 'post/<page:\d+>/<tag>',
    'route' => 'post/index',
    'defaults' => ['page' => 1],
]
```

Silakan merujuk ke bagian [docs manajer Url](#) untuk lebih jelasnya.

Perubahan penting dalam konvensi penamaan untuk rute adalah bahwa nama-nama camelcase dari controller dan action sekarang dikonversi menjadi huruf kecil di mana setiap kata dipisahkan oleh hypen, misal controller `id` untuk `CamelCaseController` akan menjadi `camel-case`. Lihat bagian tentang [Kontroler ID dan Action ID](#) untuk lebih jelasnya.

1.2.24 Menggunakan Yii 1.1 dan 2.x bersama-sama

Jika Anda memiliki warisan kode Yii 1.1 yang ingin Anda gunakan bersama-sama dengan Yii 2.0, silakan lihat bagian [Menggunakan Yii 1.1 dan 2.0 Bersama](#).

¹³<https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/README.md>

Bab 2

Mulai

2.1 Instalasi Yii

Anda dapat menginstal Yii dalam dua cara, menggunakan Composer¹ paket manager atau dengan mengunduh file arsip. Yang pertama adalah cara yang lebih disukai, karena memungkinkan Anda untuk menginstal *ekstensi* baru atau memperbarui Yii dengan hanya menjalankan *command line*.

Hasil instalasi standar Yii baik framework maupun template proyek keduanya akan terunduh dan terpasang. Sebuah template proyek adalah proyek Yii yang menerapkan beberapa fitur dasar, seperti login, formulir kontak, dll. Kode diatur dalam cara yang direkomendasikan. Oleh karena itu, dapat berfungsi sebagai titik awal yang baik untuk proyek-proyek Anda. Dalam hal ini dan beberapa bagian berikutnya, kita akan menjelaskan cara menginstal Yii dengan apa yang disebut *Template Proyek Dasar* dan bagaimana menerapkan fitur baru di atas template ini. Yii juga menyediakan template lain yang disebut yang *Template Proyek Lanjutan*² yang lebih baik digunakan dalam lingkungan pengembangan tim untuk mengembangkan aplikasi dengan beberapa tingkatan.

Info: Template Proyek Dasar ini cocok untuk mengembangkan 90 persen dari aplikasi Web. Ini berbeda dari Template Proyek Lanjutan terutama dalam bagaimana kode mereka diatur. Jika Anda baru untuk Yii, kami sangat merekomendasikan Anda tetap pada Template Proyek Dasar untuk kesederhanaan dan fungsi yang cukup.

¹<https://getcomposer.org/>

²<https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/README.md>

2.1.1 Menginstal melalui Komposer

Jika Anda belum memiliki Composer terinstal, Anda dapat melakukannya dengan mengikuti petunjuk di [getcomposer.org] (<https://getcomposer.org/download/>). Pada Linux dan Mac OS X, Anda akan menjalankan perintah berikut:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

Pada Windows, Anda akan mengunduh dan menjalankan Composer-Setup.exe³.

Silakan merujuk ke Dokumentasi Composer⁴ jika Anda menemukan masalah atau ingin mempelajari lebih lanjut tentang penggunaan Composer.

Jika Composer sudah terinstal sebelumnya, pastikan Anda menggunakan versi terbaru. Anda dapat memperbarui Komposer dengan menjalankan `composer self-update`.

Dengan Komposer diinstal, Anda dapat menginstal Yii dengan menjalankan perintah berikut di bawah folder yang terakses web:

```
composer global require "fxp/composer-asset-plugin:~1.4.1"
composer create-project --prefer-dist yiisoft/yii2-app-basic basic
```

Perintah pertama menginstal komposer aset Plugin⁵ yang memungkinkan mengelola bower dan paket npm melalui Composer. Anda hanya perlu menjalankan perintah ini sekali untuk semua. Perintah kedua menginstal Yii dalam sebuah direktori bernama `basic`. Anda dapat memilih nama direktori yang berbeda jika Anda ingin.

Catatan: Selama instalasi, Composer dapat meminta login Github Anda. Ini normal karena Komposer Perlu mendapatkan cukup API rate-limit untuk mengambil informasi paket dari Github. Untuk lebih jelasnya, Silahkan lihat Documentation Composer⁶.

Tips: Jika Anda ingin menginstal versi pengembangan terbaru dari Yii, Anda dapat menggunakan perintah berikut sebagai gantinya, Yang menambahkan opsi stabilitas⁷:

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

Perhatikan bahwa versi pengembangan dari Yii tidak boleh digunakan untuk produksi karena kemungkinan dapat *merusak* kode Anda yang sedang berjalan.

³<https://getcomposer.org/Composer-Setup.exe>

⁴<https://getcomposer.org/doc/>

⁵<https://github.com/fxp/composer-asset-plugin>

⁶<https://getcomposer.org/doc/articles/troubleshooting.md#api-rate-limit-and-oauth-tokens>

⁷<https://getcomposer.org/doc/04-schema.md#minimum-stability>

2.1.2 Instalasi dari file Arsip

Instalasi Yii dari file arsip melibatkan tiga langkah:

1. Download file arsip dari [yiiframework.com](https://www.yiiframework.com)⁸.
2. Uraikan file yang didownload ke folder yang bisa diakses web.
3. Memodifikasi `config/web.php` dengan memasukkan kunci rahasia untuk `cookieValidationKey`. (Ini dilakukan secara otomatis jika Anda menginstal Yii menggunakan Composer):

```
// !!! Isikan nilai key jika kosong - ini diperlukan oleh cookie
validation
'cookieValidationKey' => 'enter your secret key here',
```

2.1.3 Pilihan Instalasi lainnya

Petunjuk instalasi di atas menunjukkan cara menginstal Yii, yang juga menciptakan aplikasi Web dasar yang bekerja di luar kotak. Pendekatan ini adalah titik awal yang baik untuk sebagian besar proyek, baik kecil atau besar. Hal ini terutama cocok jika Anda hanya mulai belajar Yii.

Tetapi ada pilihan instalasi lain yang tersedia:

- Jika Anda hanya ingin menginstal kerangka inti dan ingin membangun seluruh aplikasi dari awal, Anda dapat mengikuti petunjuk seperti yang dijelaskan dalam [Membangun Aplikasi dari Scratch](#).
- Jika Anda ingin memulai dengan aplikasi yang lebih canggih, lebih cocok untuk tim lingkungan pengembangan, Anda dapat mempertimbangkan memasang [Template Lanjutan Proyek] (<https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/README.md>).

2.1.4 Memverifikasi Instalasi

Setelah instalasi selesai, baik mengkonfigurasi web server Anda (lihat bagian berikutnya) atau menggunakan [Built-in web server PHP] (<https://www.php.net/manual/en/features.commandline.webserver.php>) dengan menjalankan berikut konsol perintah sementara dalam proyek `web` direktori:

```
php yii serve
```

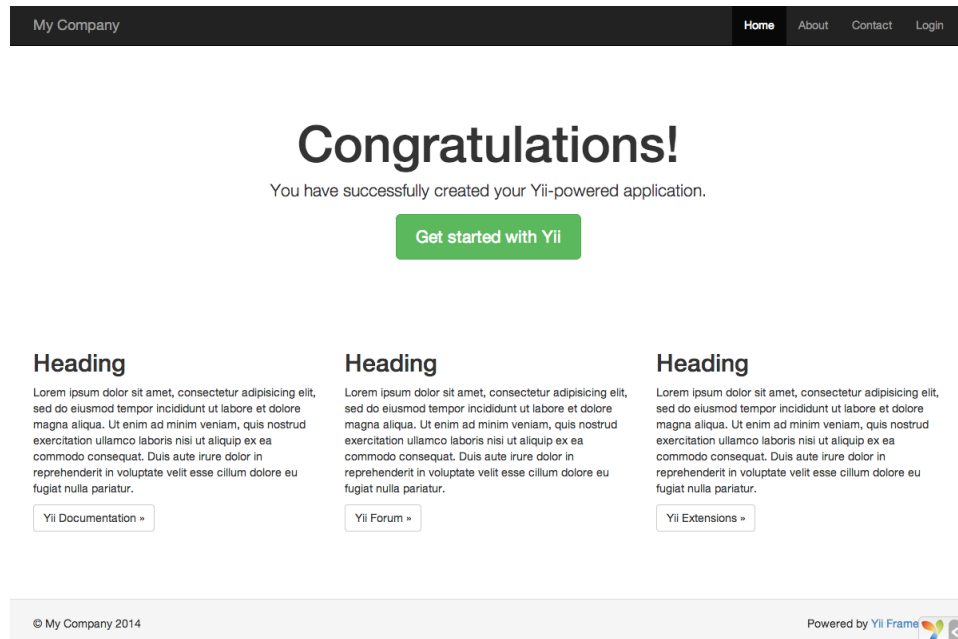
Catatan: Secara default HTTP-server akan mendengarkan port 8080. Namun jika port yang sudah digunakan atau Anda ingin melayani beberapa aplikasi dengan cara ini, Anda mungkin ingin menentukan port apa yang harus digunakan. Cukup tambahkan argumen `-port`:

⁸<https://www.yiiframework.com/download/>

```
php yii serve --port = 8888
```

Anda dapat menggunakan browser untuk mengakses aplikasi Yii yang diinstal dengan URL berikut:

<http://localhost:8080/>



Anda seharusnya melihat halaman “Congratulations!” di browser Anda. Jika tidak, periksa apakah instalasi PHP Anda memenuhi persyaratan Yii. Anda dapat memeriksa apakah persyaratan minimumnya cocok dengan menggunakan salah satu pendekatan berikut:

- Copy `/requirements.php` ke `/web/requirements.php` kemudian gunakan browser untuk mengakses melalui `http://localhost/requirements.php`
- Jalankan perintah berikut:

```
`bash` cd basic` php requirements.php`
```

Anda harus mengkonfigurasi instalasi PHP Anda sehingga memenuhi persyaratan minimal Yii. Yang paling penting, Anda harus memiliki PHP versi 5.4 atau lebih. Anda juga harus menginstal PDO PHP Ekstensi⁹ dan driver database yang sesuai (seperti `pdo_mysql` untuk database MySQL), jika aplikasi Anda membutuhkan database.

2.1.5 Konfigurasi Web Server

Info: Anda dapat melewati seksi ini untuk saat ini jika Anda hanya menguji sebuah Yii dengan niat penggelaran itu untuk server produksi.

⁹<https://www.php.net/manual/en/pdo.installation.php>

Aplikasi yang diinstal sesuai dengan petunjuk di atas seharusnya bekerja dengan baik pada Apache HTTP server¹⁰ atau Nginx HTTP server¹¹, pada Windows, Mac OS X, atau Linux yang menjalankan PHP 5.4 atau lebih tinggi. Yii 2.0 juga kompatibel dengan facebook HHVM¹². Namun, ada beberapa kasus di mana HHVM berperilaku berbeda dari PHP asli, sehingga Anda harus mengambil beberapa perlakuan ekstra ketika menggunakan HHVM.

Pada server produksi, Anda mungkin ingin mengkonfigurasi server Web Anda sehingga aplikasi dapat diakses melalui URL `https://www.example.com/index.php` bukannya `https://www.example.com/dasar/web/index.php`. konfigurasi seperti itu membutuhkan root dokumen server Web Anda menunjuk ke folder `basic/web`. Anda mungkin juga ingin menyembunyikan `index.php` dari URL, seperti yang dijelaskan pada bagian [Routing dan Penciptaan URL](#). Dalam bagian ini, Anda akan belajar bagaimana untuk mengkonfigurasi Apache atau Nginx server Anda untuk mencapai tujuan tersebut.

Info: Dengan menetapkan `basic/web` sebagai akar dokumen, Anda juga mencegah pengguna akhir mengakses kode private aplikasi Anda dan file data sensitif yang disimpan dalam direktori sejajar dari `basic/web`. Mencegah akses ke folder lainnya adalah sebuah peningkatan keamanan.

Info: Jika aplikasi Anda akan berjalan di lingkungan shared hosting di mana Anda tidak memiliki izin untuk memodifikasi konfigurasi server Web-nya, Anda mungkin masih menyesuaikan struktur aplikasi Anda untuk keamanan yang lebih baik. Silakan merujuk ke yang lebih baik. Lihat bagian [Shared Hosting Lingkungan](#) untuk rincian lebih lanjut.

Konfigurasi Apache yang Direkomendasikan

Gunakan konfigurasi berikut di file `httpd.conf` Apache atau dalam konfigurasi virtual host. Perhatikan bahwa Anda harus mengganti `path/to/basic/web` dengan path `dasar/web` yang sebenarnya.

```
# Set document root to be "basic/web"
DocumentRoot "path/to/basic/web"

<Directory "path/to/basic/web">
    # use mod_rewrite for pretty URL support
    RewriteEngine on
    # If a directory or a file exists, use the request directly
    RewriteCond %{REQUEST_FILENAME} !-f
```

¹⁰<https://httpd.apache.org/>

¹¹<https://nginx.org/>

¹²<https://hhvm.com/>

```

RewriteCond %{REQUEST_FILENAME} !-d
# Otherwise forward the request to index.php
RewriteRule . index.php

# ...other settings...
</Directory>

```

Konfigurasi Nginx yang Direkomendasikan

Untuk menggunakan Nginx¹³, Anda harus menginstal PHP sebagai FPM SAPI¹⁴. Anda dapat menggunakan konfigurasi Nginx berikut, menggantikan path/to/basic/web dengan path yang sebenarnya untuk basic/web dan mysite.test dengan hostname yang sebenarnya untuk server.

```

server {
    charset utf-8;
    client_max_body_size 128M;

    listen 80; ## listen for ipv4
    #listen [::]:80 default_server ipv6only=on; ## listen for ipv6

    server_name mysite.test;
    root        /path/to/basic/web;
    index       index.php;

    access_log  /path/to/basic/log/access.log;
    error_log   /path/to/basic/log/error.log;

    location / {
        # Redirect everything that isn't a real file to index.php
        try_files $uri $uri/ /index.php$is_args$args;
    }

    # uncomment to avoid processing of calls to non-existing static files by
    # Yii
    #location ~ \.(js|css|png|jpg|gif|swf|ico|pdf|mov|fla|zip|rar)$ {
    #    try_files $uri =404;
    #}
    #error_page 404 /404.html;

    location ~ /\.php$ {
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_pass 127.0.0.1:9000;
        #fastcgi_pass unix:/var/run/php5-fpm.sock;
        try_files $uri =404;
    }

    location ~ /\. (ht|svn|git) {
        deny all;
    }
}

```

¹³<https://wiki.nginx.org/>

¹⁴<https://www.php.net/install.fpm>

```
}
}
```

Bila menggunakan konfigurasi ini, Anda juga harus menetapkan `cgi.fix_pathinfo=0` di file `php.ini` untuk menghindari banyak panggilan `stat()` sistem yang tidak perlu.

Sekalian catat bahwa ketika menjalankan server HTTPS, Anda perlu menambahkan `fastcgi_param HTTPS on;` sehingga Yii benar dapat mendeteksi jika sambungan aman.

2.2 Menjalankan Aplikasi

Setelah menginstal Yii, Anda memiliki aplikasi Yii yang dapat diakses melalui URL `https://hostname/basic/web/index.php` atau `https://hostname/index.php`, tergantung pada konfigurasi Anda. Bagian ini akan memperkenalkan fungsi built-in aplikasi, bagaimana kode ini disusun, dan bagaimana aplikasi menangani permintaan secara umum.

Info: Untuk mempermudah, selama tutorial “Mulai”, itu diasumsikan bahwa Anda telah menetapkan `basic/web` sebagai root dokumen server Web Anda, dan URL dikonfigurasi untuk mengakses aplikasi Anda untuk menjadi `https://hostname/index.php` atau sesuatu yang serupa. Untuk kebutuhan Anda, silakan menyesuaikan URL sesuai deskripsi kami. Perhatikan bahwa tidak seperti framework itu sendiri, setelah template proyek diinstal, itu semua milikmu. Anda bebas untuk menambah atau menghapus kode dan memodifikasi keseluruhannya sesuai yang Anda butuhkan.

2.2.1 Fungsi

Aplikasi dasar diinstal berisi empat halaman:

- Homepage, ditampilkan saat Anda mengakses URL `https://hostname/index.php`,
- Halaman “About”,
- Halaman “Contact”, yang menampilkan formulir kontak yang memungkinkan pengguna akhir untuk menghubungi Anda melalui email,
- Dan halaman “Login”, yang menampilkan form login yang dapat digunakan untuk otentikasi pengguna akhir. Cobalah masuk dengan “admin/admin”, dan Anda akan menemukan item “Login” di menu utama akan berubah menjadi “Logout”.

Halaman ini berbagi header umum dan footer. header berisi menu bar utama untuk memungkinkan navigasi antara halaman yang berbeda.

Anda juga harus melihat toolbar di bagian bawah jendela browser. Ini adalah debugger tool¹⁵ yang disediakan oleh Yii untuk merekam dan me-

¹⁵<https://github.com/yiisoft/yii2-debug/blob/master/docs/guide/README.md>

nampilkan banyak informasi debug, seperti log pesan, status respon, query database berjalan, dan sebagainya.

Selain itu untuk aplikasi web, ada script konsol yang disebut `yii`, yang terletak di direktori aplikasi dasar. Script ini dapat digunakan untuk menjalankan aplikasi background dan tugas pemeliharaan untuk aplikasi, yang diuraikan di bagian [Console Application](#).

2.2.2 Struktur aplikasi

Direktori yang paling penting dan file dalam aplikasi Anda (dengan asumsi direktori root aplikasi adalah `basic`):

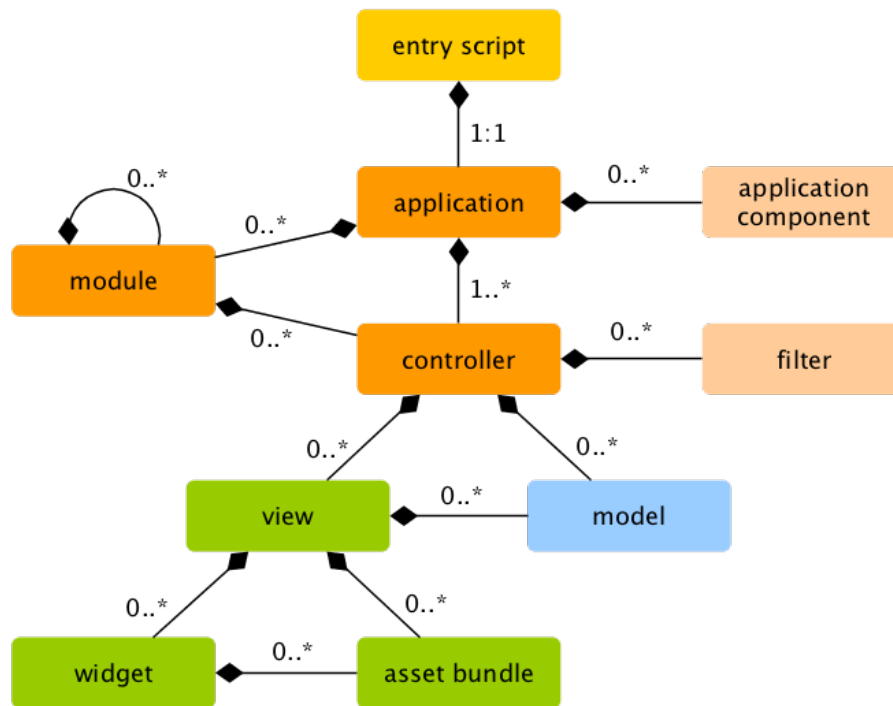
<code>basic/</code>	path aplikasi dasar
<code>composer.json</code>	digunakan oleh Composer, package information
<code>config/</code>	berisi konfigurasi aplikasi dan yang lain
<code>console.php</code>	konfigurasi aplikasi konsol
<code>web.php</code>	konfigurasi aplikasi web
<code>commands/</code>	contains console command classes
<code>controllers/</code>	contains controller classes
<code>models/</code>	contains model classes
<code>runtime/</code>	contains files generated by Yii during runtime, such as logs and cache files
<code>vendor/</code>	contains the installed Composer packages, including the Yii framework itself
<code>views/</code>	contains view files
<code>web/</code>	application Web root, contains Web accessible files
<code>assets/</code>	contains published asset files (javascript and css)
by Yii	
<code>index.php</code>	the entry (or bootstrap) script for the application
<code>yii</code>	the Yii console command execution script

Secara umum, file dalam aplikasi dapat dibagi menjadi dua jenis: mereka yang di bawah `basic/web` dan mereka yang di bawah direktori lain. Yang pertama dapat langsung diakses melalui HTTP (yaitu, di browser), sedangkan yang kedua tidak dapat dan tidak seharusnya boleh.

Yii mengimplementasikan pola arsitektur model-view-controller (MVC)¹⁶, yang tercermin dalam organisasi direktori di atas. Direktori `models` berisi semua [Model kelas](#), direktori `views` berisi semua [view script] `structure-views.md`), dan direktori `controllers` mengandung semua [kelas kontroler](#).

Diagram berikut memperlihatkan struktur statis dari sebuah aplikasi.

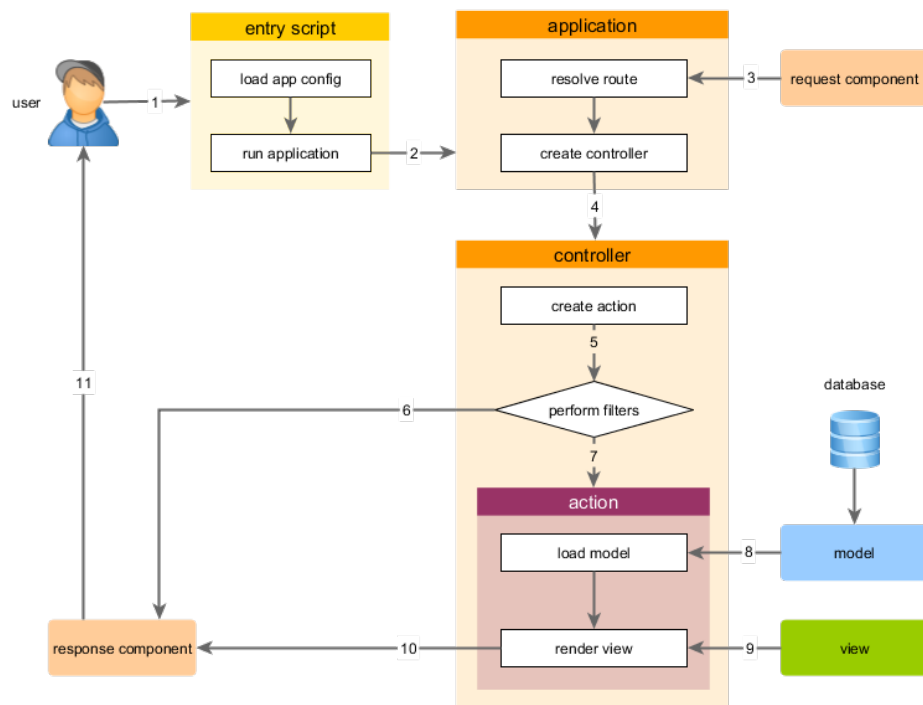
¹⁶<https://wikipedia.org/wiki/Model-view-controller>



Setiap aplikasi memiliki naskah entri `web/index.php` yang merupakan satu-satunya PHP skrip dalam aplikasi yang dapat diakses web. Naskah entri mengambil permintaan masuk dan menciptakan **aplikasi** untuk menanganinya. **Aplikasi** menyelesaikan permintaan dengan bantuan **komponennya**, dan mengirimkan permintaan ke elemen MVC. **Widget** digunakan dalam **view** untuk membantu membangun elemen antarmuka pengguna yang kompleks dan dinamis.

2.2.3 Daur Hidup Request

Diagram berikut menunjukkan bagaimana aplikasi menangani permintaan.



1. Pengguna membuat permintaan ke **skrip entri** `web/index.php`.
2. Naskah entri memuat **konfigurasi** aplikasi dan menciptakan **aplikasi** untuk menangani permintaan.
3. Aplikasi menyelesaikan **route** yang diminta dengan bantuan **komponen request** aplikasi.
4. Aplikasi ini menciptakan **kontroler** untuk menangani permintaan.
5. Controller menciptakan **action** dan melakukan filter untuk action.
6. Jika filter gagal, aksi dibatalkan.
7. Jika semua filter lulus, aksi dieksekusi.
8. Action memuat model data, mungkin dari database.
9. Aksi menyiapkan view, menyediakannya dengan model data.
10. Hasilnya diberikan dikembalikan ke komponen aplikasi **respon**.
11. Komponen respon mengirimkan hasil yang diberikan ke browser pengguna.

2.3 Katakan Hello

Bagian ini menjelaskan cara membuat halaman “Hello” baru dalam aplikasi Anda. Untuk mencapai tujuan ini, Anda akan membuat action dan sebuah `view`:

- Aplikasi ini akan mengirimkan permintaan halaman ke `action`.
- Dan `action` pada gilirannya akan membuat tampilan yang menunjukkan kata “Hello” kepada pengguna akhir.

Melalui tutorial ini, Anda akan belajar tiga hal:

1. Cara membuat `action` untuk menanggapi permintaan,
2. Cara membuat `view` untuk menyusun konten respon, dan
3. bagaimana aplikasi mengirimkan permintaan ke `action`.

2.3.1 Membuat Action

Untuk tugas “Hello”, Anda akan membuat `action say` yang membaca parameter `message` dari request dan menampilkan pesan bahwa kembali ke pengguna. Jika request tidak memberikan parameter `message`, aksi akan menampilkan pesan “Hello”.

Info: Action adalah objek yang pengguna akhir dapat langsung merujuk ke eksekusi. Action dikelompokkan berdasarkan `controllers`. Hasil eksekusi `action` adalah respon yang pengguna akhir akan terima.

Action harus dinyatakan di `controllers`. Untuk mempermudah, Anda mungkin mendeklarasikan `action say` di `SiteController` yang ada. kontroler ini didefinisikan dalam file kelas `controllers/SiteController.php`. Berikut adalah awal dari `action` baru:

```
<?php

namespace app\controllers;

use yii\web\Controller;

class SiteController extends Controller
{
    // ...existing code...

    public function actionSay($message = 'Hello')
    {
        return $this->render('say', ['message' => $message]);
    }
}
```

Pada kode di atas, action `say` didefinisikan sebagai metode bernama `actionSay` di kelas `SiteController`. Yii menggunakan awalan `action` untuk membedakan metode action dari metode non-action dalam kelas controller. Nama setelah awalan `action` peta untuk ID tindakan ini.

Untuk sampai pada penamaan action, Anda harus memahami bagaimana Yii memperlakukan ID action. ID action selalu direferensikan dalam huruf kecil. Jika ID tindakan membutuhkan beberapa kata, mereka akan digabungkan dengan tanda hubung (Mis, `create-comment`). nama metode aksi yang dipetakan ke ID tindakan diperoleh dengan menghapus tanda hubung apapun dari ID, mengkapitalkan huruf pertama di setiap kata, dan awalan string yang dihasilkan dengan `action`. Sebagai contoh, ID action `create-comment` sesuai dengan nama method action `actionCreateComment`.

Metode action dalam contoh kita mengambil parameter `$message`, yang nilai defaultnya adalah `"Hello"` (persis dengan cara yang sama Anda menetapkan nilai default untuk fungsi atau metode apapun argumen di PHP). Ketika aplikasi menerima permintaan dan menentukan bahwa action `say` bertanggung jawab untuk penanganan request, aplikasi akan mengisi parameter ini dengan parameter bernama sama yang ditemukan dalam request. Dengan kata lain, jika permintaan mencakup a parameter `message` dengan nilai `"Goodbye"`, maka variabel `$message` dalam aksi akan ditugaskan nilai itu.

Dalam metode action, `render()` dipanggil untuk membuat sebuah [view](#) dari file bernama `say`. Parameter `message` juga diteruskan ke view sehingga dapat digunakan di sana. Hasil render dikembalikan dengan metode tindakan. Hasil yang akan diterima oleh aplikasi dan ditampilkan kepada pengguna akhir di browser (sebagai bagian dari halaman HTML yang lengkap).

2.3.2 Membuat View

[View](#) adalah skrip yang Anda tulis untuk menghasilkan konten respon. Untuk “Hello” tugas, Anda akan membuat view `say` yang mencetak parameter `message` yang diterima dari metode aksi:

```
<?php
use yii\helpers\Html;
?>
<?= Html::encode($message) ?>
```

View `say` harus disimpan dalam file `views/site/say.php`. Ketika metode `render()` disebut dalam tindakan, itu akan mencari file PHP bernama `views/ControllerID/ViewName.php`.

Perhatikan bahwa dalam kode di atas, parameter `message` adalah di-HTML-encoded sebelum dicetak. Hal ini diperlukan karena sebagai parameter yang berasal dari pengguna akhir, sangat rentan terhadap serangan Cross-site scripting (XSS)¹⁷ dengan melekatkan kode JavaScript berbahaya dalam parameter.

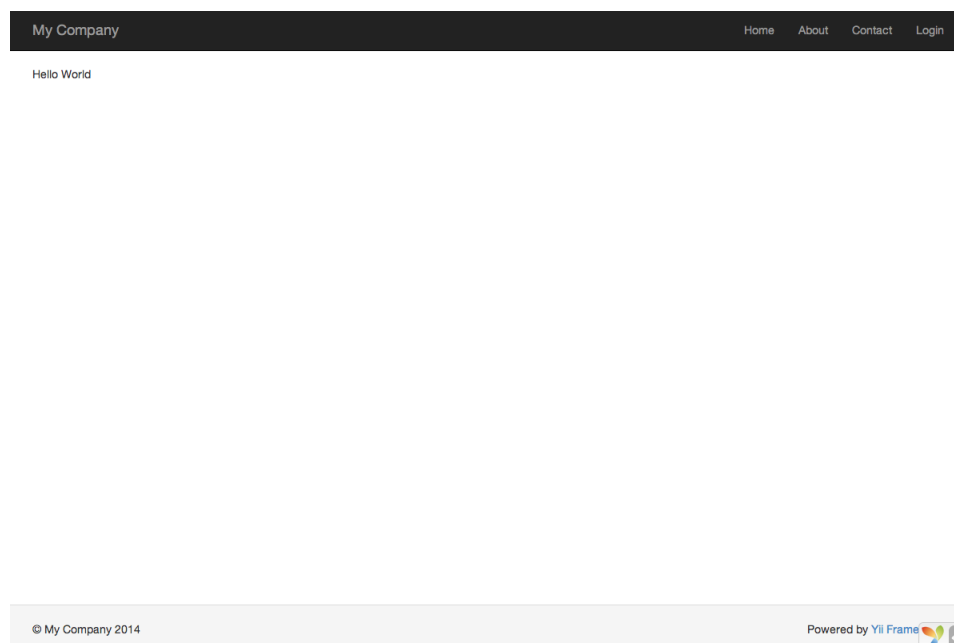
¹⁷https://en.wikipedia.org/wiki/Cross-site_scripting

Tentu, Anda dapat menempatkan lebih banyak konten di view `say`. konten dapat terdiri dari tag HTML, teks biasa, dan bahkan pernyataan PHP. Nyatanya, view `say` hanyalah sebuah script PHP yang dijalankan oleh metode `render()`. Isi dicetak oleh skrip view akan dikembalikan ke aplikasi sebagai hasil respon ini. Aplikasi ini pada gilirannya akan mengeluarkan hasil ini kepada pengguna akhir.

2.3.3 Trying it Out

Setelah membuat action dan view, Anda dapat mengakses halaman baru dengan mengakses URL berikut:

```
https://hostname/index.php?r=site%2Fsay&message=Hello+World
```



URL ini akan menghasilkan halaman yang menampilkan “Hello World”. Halaman yang berbagi header dan footer yang sama dengan halaman aplikasi lainnya.

Jika Anda menghilangkan parameter `message` dalam URL, Anda akan melihat tampilan halaman “Hello”. Hal ini karena `message` dilewatkan sebagai parameter untuk metode `actionSay()`, dan ketika itu dihilangkan, nilai default “Hello” akan digunakan sebagai gantinya.

Info: Halaman baru berbagi header dan footer yang sama dengan halaman lain karena metode `render()` otomatis akan menanamkan hasil view `say` kedalam apa yang disebut layout yang dalam hal ini Kasus terletak di `views/layouts/main.php`.

Parameter `r` di URL di atas memerlukan penjelasan lebih lanjut. Ini adalah singkatan dari *route*, sebuah ID unik aplikasi yang mengacu pada action. format rute ini adalah `ControllerID/ActionID`. Ketika aplikasi menerima permintaan, itu akan memeriksa parameter ini, menggunakan bagian `ControllerID` untuk menentukan kontroler kelas harus dipakai untuk menangani permintaan. Kemudian, controller akan menggunakan bagian `ActionID` untuk menentukan action yang harus dipakai untuk melakukan pekerjaan yang sebenarnya. Dalam contoh kasus ini, rute `site/say` akan diselesaikan dengan kontroler kelas `SiteController` dan action `say`. Sebagai hasilnya, metode `SiteController::actionSay()` akan dipanggil untuk menangani permintaan.

Info: Seperti action, kontroler juga memiliki ID yang unik mengidentifikasi mereka dalam sebuah aplikasi. ID kontroler menggunakan aturan penamaan yang sama seperti ID tindakan. nama kelas controller yang berasal dari kontroler ID dengan menghapus tanda hubung dari ID, memanfaatkan huruf pertama di setiap kata, dan suffixing string yang dihasilkan dengan kata `Controller`. Misalnya, controller ID `post-comment` berkorespondensi dengan nama kelas controller `PostCommentController`.

2.3.4 Ringkasan

Pada bagian ini, Anda telah menyentuh controller dan melihat bagian dari pola arsitektur MVC. Anda menciptakan sebuah action sebagai bagian dari controller untuk menangani permintaan khusus. Dan Anda juga menciptakan view untuk menulis konten respon ini. Dalam contoh sederhana ini, tidak ada model yang terlibat. Satu-satunya data yang digunakan adalah parameter `message`.

Anda juga telah belajar tentang rute di Yii, yang bertindak sebagai jembatan antara permintaan pengguna dan tindakan controller.

Pada bagian berikutnya, Anda akan belajar cara membuat model, dan menambahkan halaman baru yang berisi bentuk HTML.

2.4 Bekerja dengan Form

Bagian ini memaparkan bagaimana membuat halaman dengan form untuk mengambil data dari pengguna. Halaman akan menampilkan form dengan input field Nama dan Email. Setelah mendapatkan dua data dari pengguna, halaman akan menampilkan kembali data yang diinput pada form sebagai konfirmasi.

Untuk mencapai tujuan, disamping membuat sebuah *action*, dan dua *view*, anda juga harus membuat *model*.

Sepanjang tutorial ini, anda akan mempelajari bagaimana cara untuk:

- Membuat sebuah `model` sebagai representasi data yang diinput oleh pengguna melalui form,
- Membuat *rules* untuk memvalidasi data yang telah diinput.
- Membuat form HTML di dalam `view`.

2.4.1 Membuat Model

Data yang akan diambil dari pengguna akan direpresentasikan oleh class model `EntryForm` sebagaimana ditunjukkan di bawah dan di simpan pada file `models/EntryForm.php`. Silahkan membaca bagian [Class Autoloading](#) untuk penjelasan lengkap mengenai penamaan file class.

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;

class EntryForm extends Model
{
    public $name;
    public $email;

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

Class di *extends* dari `yii\base\Model`, class standar yang disediakan oleh Yii, yang secara umum digunakan untuk representasi data dari form.

Info: `yii\base\Model` digunakan sebagai *parent* untuk class model yang tidak berhubungan dengan database. `yii\db\ActiveRecord` normalnya digunakan sebagai *parent* untuk class model yang berhubungan dengan tabel di database.

Class `EntryForm` terdiri dari dua *public property*, `name` dan `email`, dimana akan digunakan untuk menyimpan data yang diinput oleh pengguna. Class ini juga terdapat *method* yang dinamakan `rules()`, yang akan mengembalikan (*return*) sejumlah pengaturan (*rules*) untuk memvalidasi data. Pengaturan validasi (*Validation Rules*) yang di deklarasikan harus mendeskripsikan bahwa

- kedua field, yaitu `name` and `email` wajib di input
- data `email` harus merupakan alamat email yang valid

Jika anda memiliki objek `EntryForm` yang sudah mengandung data yang di input oleh pengguna, anda boleh memanggil method `validate()` untuk melaksanakan validasi data. Kegagalan validasi data akan menentukan (*set*) property `hasErrors` menjadi `true`, dan anda dapat mengetahui pesan kegagalan validasi melalui `errors`.

```
<?php
$model = new EntryForm();
$model->name = 'Qiang';
$model->email = 'bad';
if ($model->validate()) {
    // Valid!
} else {
    // Tidak Valid!
    // Panggil $model->getErrors()
}
```

2.4.2 Membuat Action

Selanjutnya, anda harus membuat *entry action* pada controller *site* yang akan memanfaatkan model yang baru saja dibuat. Proses membuat dan menggunakan *action* dijelaskan pada bagian [Mengatakan Hello](#).

```
<?php

namespace app\controllers;

use Yii;
use yii\web\Controller;
use app\models\EntryForm;

class SiteController extends Controller
{
    // ...kode lain...

    public function actionEntry()
    {
        $model = new EntryForm();

        if ($model->load(Yii::$app->request->post()) && $model->validate())
        {
            // data yang valid diperoleh pada $model

            // lakukan sesuatu terhadap $model di sini ...

            return $this->render('entry-confirm', ['model' => $model]);
        } else {
            // menampilkan form pada halaman, ada atau tidaknya kegagalan
            // validasi tidak masalah
            return $this->render('entry', ['model' => $model]);
        }
    }
}
```

Pertama-tama, *action* membuat objek `EntryForm`. Kemudian objek tersebut membangun model menggunakan data dari `$_POST`, yang disediakan oleh Yii dengan method `yii\web\Request::post()`. Jika model berhasil dibuat (misal, jika pengguna telah mengirim form HTML), *action* akan memanggil method `validate()` untuk memastikan data yang di input tersebut valid.

Info : *Expression* `Yii::$app` adalah representasi dari objek *aplikasi*, dimana objek tersebut adalah *singleton* yang bebas diakses secara global. Objek tersebut juga merupakan *service locator* yang menyediakan *components* seperti `request`, `response`, `db`, dll. untuk mendukung pekerjaan yang spesifik. Pada kode di atas, *component request* dari objek aplikasi digunakan untuk mengakses data `$_POST`.

Jika tidak ada error, *action* akan me-render *view* bernama `entry-confirm` untuk menginformasikan ke pengguna bahwa pengiriman data tersebut berhasil. Jika tidak ada data yang dikirim atau data tersebut tidak valid, *view entry* yang akan di *render*, dimana form HTML akan ditampilkan, beserta informasi kegagalan pengiriman form tersebut.

Catatan: Pada contoh sederhana ini kita hanya me-render halaman konfirmasi jika data yang dikirim tersebut valid. Pada prakteknya, anda harus pertimbangkan untuk menggunakan `refresh()` atau `redirect()` untuk mencegah permasalahan pengiriman form¹⁸.

2.4.3 Membuat View

Terakhir, buatlah dua file *view* dengan nama `entry-confirm` dan `entry`. *View* ini akan di-render oleh *action entry*, yang sebelumnya dibahas.

View entry-confirm hanya menampilkan data nama dan email. File *view* tersebut harus di simpan di `views/site/entry-confirm.php`.

```
<?php
use yii\helpers\Html;
?>
<p>You have entered the following information:</p>

<ul>
    <li><label>Name</label>: <?= Html::encode($model->name) ?></li>
    <li><label>Email</label>: <?= Html::encode($model->email) ?></li>
</ul>
```

View entry akan menampilkan form HTML. File *view* tersebut harus di simpan di `views/site/entry.php`.

¹⁸<https://en.wikipedia.org/wiki/Post/Redirect/Get>

```

<?php
use yii\helpers\Html;
use yii\widgets\ActiveForm;
?>
<?php $form = ActiveForm::begin(); ?>

    <?= $form->field($model, 'name') ?>

    <?= $form->field($model, 'email') ?>

    <div class="form-group">
        <?= Html::submitButton('Submit', ['class' => 'btn btn-primary']) ?>
    </div>

<?php ActiveForm::end(); ?>

```

View ini menggunakan *widget* yaitu `ActiveForm` untuk membangun form HTML. *Method* `begin()` dan `end()` dari *widget* masing-masing berfungsi untuk me-render tag pembuka dan penutup dari form tag. Diantara dua method tersebut, akan dibuat field input oleh method `field()`. Input field yang pertama diperuntukkan untuk data “name”, dan yang kedua diperuntukkan untuk data “email”. Setelah field input, *method* `yii\helpers\Html::submitButton()` akan dipanggil untuk me-render tombol pengiriman data.

2.4.4 Mari kita uji

Untuk melihat bagaimana prosesnya, gunakan browser anda untuk mengakses URL ini :

<https://hostname/index.php?r=site%2Fentry>

Anda akan melihat halaman yang menampilkan form dengan dua field input. Dibagian atas dari semua input field, ada label yang menginformasikan data yang mana yang akan diinput. Jika anda menekan tombol pengiriman data tanpa menginput apapun, atau anda tidak menginput email address yang tidak valid, anda akan melihat pesan kegagalan yang di tampilkan di bagian bawah field input yang bermasalah.

The screenshot shows a web form for 'My Company'. The header is dark grey with the company name on the left and navigation links (Home, About, Contact, Login) on the right. The form has two input fields: 'Name' and 'Email'. Both fields have a red border and a red error message below them: 'Name cannot be blank.' and 'Email cannot be blank.' respectively. A blue 'Submit' button is located below the email field. The footer is light grey and contains the copyright notice '© My Company 2014' on the left and 'Powered by Yii Frame' with a logo on the right.

Setelah menginput nama dan alamat email yang benar dan menekan tombol kirim, anda akan melihat halaman baru yang menampilkan data yang barusan anda input.

The screenshot shows the result of submitting the form. The header is the same as the previous screenshot. The main content area displays the message 'You have entered the following information:' followed by a bulleted list of the submitted data: 'Name: Qiang Xue' and 'Email: tester@example.com'. The footer is the same as the previous screenshot.

Penjelasan

Anda mungkin bertanya-tanya bagaimana form HTML bekerja dibelakang layar, sepertinya tampak ajaib karna form tersebut mampu menampilkan

label di setiap field input dan menampilkan pesan kegagalan jika anda tidak menginput data dengan benar tanpa me-*reload* halaman.

Betul, validasi data sebenarnya dilakukan di sisi klien menggunakan Javascript, dan selanjutnya dilakukan lagi di sisi server menggunakan PHP. `yii\widgets\ActiveForm` cukup cerdas untuk menerjemahkan pengaturan validasi yang anda deklarasikan pada class `EntryForm`, kemudian merubahnya menjadi kode Javascript, dan menggunakan Javascript untuk melakukan validasi data. Jika saja anda menonaktifkan Javascript pada browser anda, validasi tetap akan dilakukan di sisi server, seperti yang ditunjukkan pada *method* `actionEntry`. Hal ini memastikan bahwa data akan divalidasi dalam segala kondisi.

Perhatian: Validasi melalui sisi klien akan membuat pengalaman pengguna lebih baik. Validasi di sisi server harus selalu dilakukan, walaupun validasi melalui sisi klien digunakan atau tidak.

Label untuk field input dibuat oleh *method* `field()`, menggunakan nama *property* dari model. Contoh, label `Name` akan dibuat untuk *property* `name`.

Anda boleh memodifikasi label di dalam view menggunakan kode seperti di bawah ini:

```
<?= $form->field($model, 'name')->label('Your Name') ?>
<?= $form->field($model, 'email')->label('Your Email') ?>
```

Info: Yii menyediakan banyak *widget* untuk membantu anda dalam membangun *view* yang kompleks dan dinamis. Sebentar lagi anda akan mengetahui, bahwa menulis *widget* juga sangat mudah. Anda mungkin akan mengganti sebagian besar dari kode *view* anda menjadi *widget-widget* yang mampu digunakan ulang untuk menyederhanakan penulisan *view* ke depannya.

2.4.5 Rangkuman

Pada bagian kali ini, anda telah mengetahui semua bagian dari pola arsitektur MVC. Anda sudah mempelajari bagaimana untuk membuat class model sebagai representasi data pengguna dan memvalidasinya.

Anda juga mempelajari bagaimana mengambil data dari pengguna dan bagaimana menampilkan kembali data tersebut ke browser. Pekerjaan seperti ini biasanya memakan waktu lama pada saat mengembangkan aplikasi, tetapi Yii menyediakan *widget* yang bermanfaat yang akan membuat pekerjaan ini menjadi lebih mudah.

Di bagian selanjutnya, anda akan mempelajari bagaimana untuk bekerja dengan database, dimana hal tersebut hampir sangat dibutuhkan pada setiap aplikasi.

2.5 Bekerja dengan Database

Bagian ini akan memaparkan bagaimana membuat halaman yang menampilkan daftar data negara yang diambil dari tabel `country` pada database. Untuk menyelesaikan tugas ini, anda akan melakukan konfigurasi koneksi ke database, membuat class `ActiveRecord`, membuat `action`, dan membuat `view`.

Sepanjang tutorial ini, anda akan mempelajari bagaimana cara untuk:

- konfigurasi koneksi ke database,
- membuat class `ActiveRecord`,
- mengambil (*query*) data menggunakan class `ActiveRecord`,
- menampilkan data ke view dengan halaman per halaman.

Sebagai catatan untuk menyelesaikan bagian ini, anda harus memiliki pengetahuan dan pengalaman dasar dalam menggunakan database. Secara khusus, anda harus mengetahui cara membuat database, dan cara menjalankan perintah SQL menggunakan aplikasi klien database.

2.5.1 Menyiapkan Database

Untuk memulai, buatlah database dengan nama `yii2basic`, yang akan digunakan untuk mengambil data dalam aplikasi anda. Anda bisa membuat database SQLite, MySQL, PostgreSQL, MSSQL, atau Oracle, dimana Yii mendukung banyak aplikasi database. Untuk memudahkan, database yang digunakan adalah MySQL.

Selanjutnya, buat tabel dengan nama `country` pada database, dan *insert* beberapa data sampel. Anda bisa menjalankan perintah SQL dibawah untuk memudahkan:

```
CREATE TABLE `country` (
  `code` CHAR(2) NOT NULL PRIMARY KEY,
  `name` CHAR(52) NOT NULL,
  `population` INT(11) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `country` VALUES ('AU','Australia',24016400);
INSERT INTO `country` VALUES ('BR','Brazil',205722000);
INSERT INTO `country` VALUES ('CA','Canada',35985751);
INSERT INTO `country` VALUES ('CN','China',1375210000);
INSERT INTO `country` VALUES ('DE','Germany',81459000);
INSERT INTO `country` VALUES ('FR','France',64513242);
INSERT INTO `country` VALUES ('GB','United Kingdom',65097000);
INSERT INTO `country` VALUES ('IN','India',1285400000);
INSERT INTO `country` VALUES ('RU','Russia',146519759);
INSERT INTO `country` VALUES ('US','United States',322976000);
```

Hingga saat ini, anda memiliki database bernama `yii2basic`, dan didalamnya terdapat tabel `country` dengan tiga kolom, berisi 10 baris data.

2.5.2 Konfigurasi Koneksi Database

Sebelum melanjutkan, pastikan anda memasang ekstensi PHP PDO¹⁹ dan driver PDO untuk database yang anda gunakan (misal, `pdo_mysql` untuk MySQL). Ini adalah kebutuhan mendasar jika aplikasi anda menggunakan *relational database*.

Jika sudah terpasang, buka file `config/db.php` dan sesuaikan parameter yang sesuai untuk database anda. Secara default, isi file konfigurasi tersebut adalah:

```
<?php

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

File `config/db.php` adalah tipikal konfigurasi yang menggunakan file. File konfigurasi seperti ini menentukan parameter-parameter yang dibutuhkan untuk membuat dan menginisialisasi objek `yii\db\Connection`, dimana anda dapat menjalankan perintah SQL dengan database yang dituju.

Konfigurasi koneksi database di atas dapat diakses pada kode aplikasi melalui *expression* `Yii::$app->db`.

Info: File `config/db.php` akan di *include* oleh konfigurasi aplikasi utama `config/web.php`, yang berfungsi sebagai konfigurasi untuk inisialisasi objek aplikasi. Untuk penjelasan lebih lengkap, silahkan lihat bagian Konfigurasi.

Jika anda membutuhkan dukungan database yang tidak didukung oleh Yii, silahkan cek *extensions* di bawah ini:

- Informix²⁰
- IBM DB2²¹
- Firebird²²

2.5.3 Membuat Active Record

Untuk mengambil data di tabel `country`, buat class turunan *Active Record* dengan nama `Country`, dan simpan pada file `models/Country.php`.

¹⁹<https://www.php.net/manual/en/book.pdo.php>

²⁰<https://github.com/edgardmessias/yii2-informix>

²¹<https://github.com/edgardmessias/yii2-ibm-db2>

²²<https://github.com/edgardmessias/yii2-firebird>

```
<?php

namespace app\models;

use yii\db\ActiveRecord;

class Country extends ActiveRecord
{
}
```

Class Country di *extends* dari `yii\db\ActiveRecord`. Anda tidak perlu untuk menulis kode di dalamnya! Hanya dengan kode di atas, Yii akan mengetahui nama tabel yang dimaksud dari nama *class* tersebut.

Info: Jika nama *class* tidak sesuai dengan nama tabel, anda dapat meng-*override* method `yii\db\ActiveRecord::tableName()` untuk menentukan nama tabel secara eksplisit.

Menggunakan class Country, anda bisa memanipulasi data pada tabel country dengan mudah, sebagaimana yang ditunjukkan pada kode di bawah ini:

```
use app\models\Country;

// mengambil semua negara dari tabel country, dan mengurutkan berdasarkan
"name" (nama)
$countries = Country::find()->orderBy('name')->all();

// mengambil negara yang memiliki primary key "US"
$country = Country::findOne('US');

// menampilkan "United States"
echo $country->name;

// Mengganti nama negara menjadi "U.S.A." dan menyimpan ke database
$country->name = 'U.S.A.';
$country->save();
```

Info: *Active Record* adalah cara yang efektif untuk mengakses dan memanipulasi data dari database secara *object-oriented*. Anda bisa mengetahui lebih banyak lagi pada bagian [Active Record](#). Sebagai alternatif, anda mungkin berinteraksi dengan database menggunakan metode data akses yang lebih mendasar yang disebut [Data Access Objects](#).

2.5.4 Membuat Action

Untuk menampilkan data negara ke pengguna, anda harus membuat *action*. Dibanding menempatkan *action* baru ini pada *controller site* seperti yang sudah anda lakukan pada bagian sebelumnya, sekarang ini ada baiknya membuat spesifik *controller* untuk semua *action* yang berhubungan dengan data

negara. Namakan *controller* baru ini dengan `CountryController`, dan buat *action* `index` pada controller tersebut, seperti yang ditunjukkan di bawah ini.

```
<?php

namespace app\controllers;

use yii\web\Controller;
use yii\data\Pagination;
use app\models\Country;

class CountryController extends Controller
{
    public function actionIndex()
    {
        $query = Country::find();

        $pagination = new Pagination([
            'defaultPageSize' => 5,
            'totalCount' => $query->count(),
        ]);

        $countries = $query->orderBy('name')
            ->offset($pagination->offset)
            ->limit($pagination->limit)
            ->all();

        return $this->render('index', [
            'countries' => $countries,
            'pagination' => $pagination,
        ]);
    }
}
```

Simpan kode di atas pada file `controllers/CountryController.php`.

Action `index` memanggil `Country::find()`. *Method Active Record* ini membuat *query* ke database dan mengambil semua data negara dari tabel `country`. Untuk membatasi jumlah negara yang didapatkan pada setiap pengambilan data, *query* tersebut dipecah menjadi halaman per halaman dengan bantuan dari objek `yii\data\Pagination`. Objek `Pagination` diperuntukkan untuk dua tujuan:

- Menentukan klausa `offset` dan `limit` pada perintah SQL yang digunakan untuk *query* agar mengambil hanya satu halaman data dalam sekali perintah (pada umumnya akan mengambil 5 baris dalam satu halaman).
- Digunakan pada *view* untuk menampilkan tombol halaman yang terdiri dari tombol-tombol nomor halaman, yang selanjutnya akan dijelaskan pada sub bagian berikutnya.

Di akhir kode, *action* `index` me-render *view* dengan nama `index`, dan mengirimkan data negara beserta dengan informasi halaman dari data tersebut.

2.5.5 Membuat View

Di dalam folder `views`, pertama-tama buatlah sub-folder dengan nama `country`. Folder ini akan digunakan untuk menyimpan semua *view* yang akan di *render* oleh *controller* `country`. Di dalam folder `views/country`, buatlah file dengan nama `index.php` berisi kode di bawah ini:

```
<?php
use yii\helpers\Html;
use yii\widgets\LinkPager;
?>
<h1>Countries</h1>
<ul>
<?php foreach ($countries as $country): ?>
    <li>
        <? = Html::encode("{ $country->name } ({ $country->code })") ?> :
        <? = $country->population ?>
    </li>
<?php endforeach; ?>
</ul>

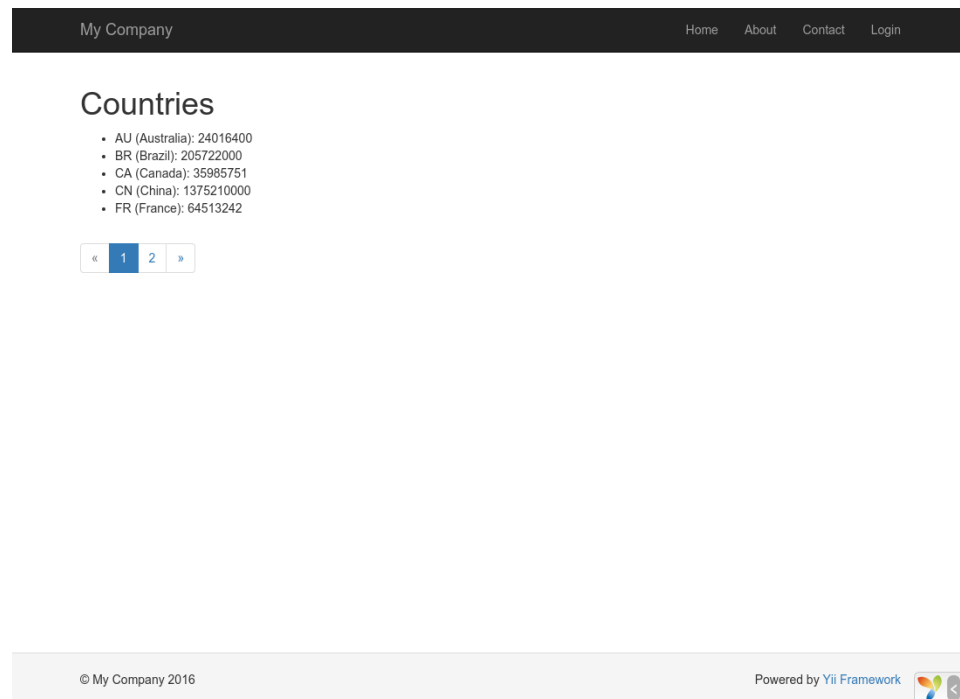
<? = LinkPager::widget(['pagination' => $pagination]) ?>
```

Terkait dengan tampilan data negara, *view* ini terdiri dari dua bagian. Bagian pertama, dilakukan perulangan (*looping*) pada data negara yang tersedia dan di-*render* sebagai *unordered list* HTML. Bagian kedua, *widget* `yii\widgets\LinkPager` di-*render* menggunakan informasi halaman (*pagination*) yang dikirimkan dari *action*. *Widget* `LinkPager` menampilkan tombol-tombol halaman. Mengklik pada salah satu tombol tersebut akan melakukan pengambilan data negara terkait dengan halaman yang diklik.

2.5.6 Mari Kita Coba

Untuk melihat bagaimana kode-kode di atas bekerja, gunakan browser anda untuk mengakses URL ini:

<https://hostname/index.php?r=country%2Findex>



Awalnya, anda akan melihat sebuah halaman yang menampilkan 5 negara. Dibawah daftar negara tersebut, anda akan melihat tombol halaman yang berjumlah empat tombol. Jika anda mengklik tombol “2”, anda akan melihat halaman tersebut menampilkan 5 negara lain pada database: halaman kedua pada record. Silahkan melakukan observasi secara perlahan-lahan dan anda akan mengetahui bahwa URL pada browser juga akan berganti menjadi

`https://hostname/index.php?r=country%2Findex&page=2`

Di belakang layar, **Pagination** menyediakan semua kebutuhan untuk memecah data menjadi halaman per halaman:

- Pertama-tama, **Pagination** menampilkan halaman pertama, dimana menjalankan perintah **SELECT** pada tabel **country** dengan klausa **LIMIT 5 OFFSET 0**. Hasilnya, 5 negara pertama akan diambil dan ditampilkan.
- **Widget LinkPager** me-render tombol halaman menggunakan URL yang dibentuk oleh method **Pagination**. URL tersebut mengandung *query string page*, yang merupakan representasi dari nomor halaman.
- Jika anda mengklik tombol halaman “2”, sebuah *request* yang mengarah ke *route country/index* akan dijalankan hingga selesai. **Pagination** membaca *query string page* dari URL dan kemudian menentukan halaman sekarang adalah halaman 2. Query data negara yang baru mengandung klausa **LIMIT 5 OFFSET 5** dan mengambil 5 data negara selanjutnya untuk kemudian ditampilkan.

2.5.7 Rangkuman

Pada bagian ini, anda mempelajari bagaimana bekerja dengan database. Anda juga mempelajari bagaimana cara mengambil dan membagi data dengan halaman per halaman dengan bantuan `yii\data\Pagination` dan `yii\widgets\LinkPager`.

Di bagian selanjutnya, anda akan mempelajari bagaimana menggunakan *generator* kode yang disebut Gii²³, untuk membantu anda mengimplementasikan fitur-fitur umum pada aplikasi secara instan, seperti operasi *Create-Read-Update-Delete* (CRUD) untuk bekerja dengan data yang terdapat pada tabel di sebuah database. Sebenarnya, kode-kode yang barusan anda tulis, semuanya bisa di *generate* secara otomatis oleh Yii menggunakan tool Gii.

2.6 Membuat Kode menggunakan Gii

Bagian ini akan menjelaskan bagaimana cara menggunakan Gii²⁴ untuk membuat kode secara otomatis yang mengimplementasikan fitur-fitur yang bersifat umum dalam sebuah web site. Menggunakan Gii untuk membuat kode sederhana menginput informasi yang sesuai per satu instruksi seperti yang diterangkan pada halaman web Gii.

Sepanjang bagian ini, anda akan mempelajari bagaimana cara untuk:

- Mengaktifkan Gii pada aplikasi anda,
- Menggunakan Gii untuk membuat *class ActiveRecord*
- Menggunakan Gii untuk membuat kode yang mengoperasikan CRUD untuk database,
- Memodifikasi kode yang sudah dibuat oleh Gii.

2.6.1 Memulai Gii

Gii²⁵ telah disediakan oleh Yii sebagai *module*. Anda dapat mengaktifkan Gii dengan mengatur konfigurasi Gii pada properti `modules` dari objek aplikasi. Tergantung bagaimana anda mengatur aplikasi anda, kode di bawah ini sudah disediakan pada file konfigurasi `config/web.php`:

```
$config = [ ... ];

if (YII_ENV_DEV) {
    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = [
        'class' => 'yii\gii\Module',
    ];
}
```

²³<https://github.com/yiisoft/yii2-gii/blob/master/docs/guide/README.md>

²⁴<https://github.com/yiisoft/yii2-gii/blob/master/docs/guide/README.md>

²⁵<https://github.com/yiisoft/yii2-gii/blob/master/docs/guide/README.md>

Konfigurasi di atas menyatakan bahwa, ketika mode development environment aktif, maka aplikasi harus mengikutkan module yang bernama `gii`, dimana objek tersebut merupakan class `yii\gii\Module`.

Jika anda melihat *entry script* `web/index.php` pada aplikasi anda, anda akan menemukan baris dibawah ini, yang menyatakan secara eksplisit bahwa `YII_ENV_DEV` sama dengan `true`.

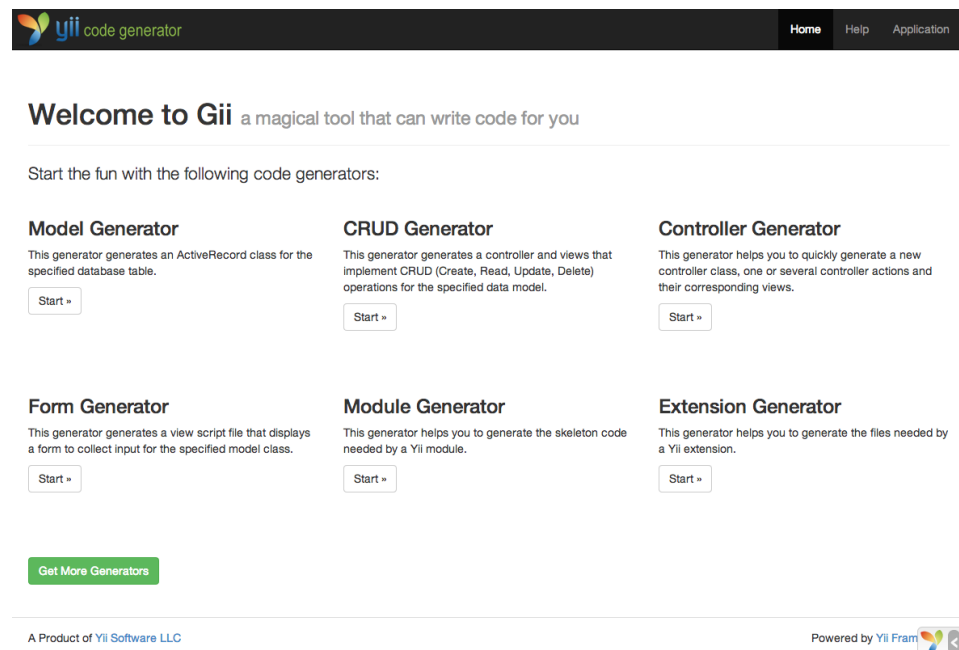
```
defined('YII_ENV') or define('YII_ENV', 'dev');
```

Karna baris tersebut, aplikasi anda harusnya sudah berada pada mode *development*, dan secara otomatis mengaktifkan Gii karena konfigurasi sebelumnya. Anda dapat mengakses Gii melalui URL di bawah ini:

`https://hostname/index.php?r=gii`

Catatan: Jika anda mengakses Gii melalui komputer diluar komputer localhost anda, secara default akses tidak akan diperbolehkan karna alasan keamanan. Anda dapat mengatur Gii untuk menambah alamat IP yang di perbolehkan seperti ini,

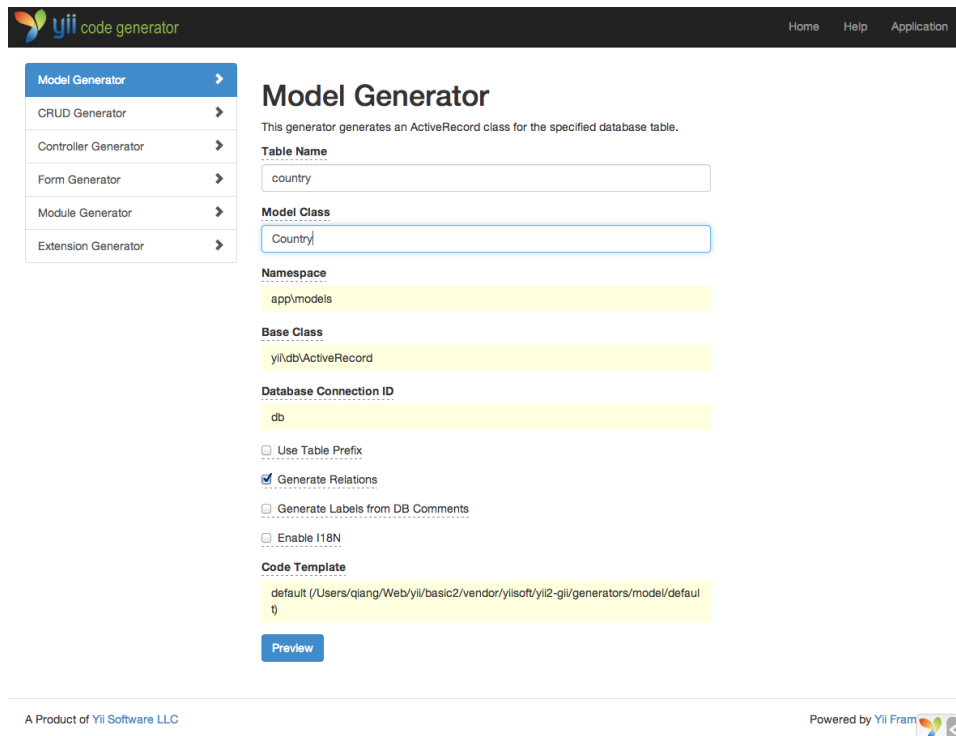
```
'gii' => [
    'class' => 'yii\gii\Module',
    'allowedIPs' => ['127.0.0.1', '::1', '192.168.0.*',
        '192.168.178.20'] // adjust this to your needs
],
```



2.6.2 Membuat class *Active Record*

Untuk menggunakan Gii dalam membuat class Active Record, pilih “Model Generator” (dengan cara mengklik link pada halaman index Gii). Kemudian isi form dengan data berikut:

- Table Name: `country`
- Model Class: `Country`



The screenshot displays the Yii2 Gii Model Generator web interface. On the left, a sidebar lists various generators: Model Generator, CRUD Generator, Controller Generator, Form Generator, Module Generator, and Extension Generator. The 'Model Generator' is selected. The main content area, titled 'Model Generator', explains that it generates an ActiveRecord class. It contains several input fields: 'Table Name' with the value 'country', 'Model Class' with 'Country', 'Namespace' with 'app\models', 'Base Class' with 'yii\db\ActiveRecord', and 'Database Connection ID' with 'db'. There are also checkboxes for 'Use Table Prefix' (unchecked), 'Generate Relations' (checked), 'Generate Labels from DB Comments' (unchecked), and 'Enable I18N' (unchecked). A 'Code Template' field shows the default path. A 'Preview' button is located at the bottom of the form. The footer indicates it is a product of Yii Software LLC and powered by Yii Framework.

Selanjutnya, klik pada tombol “Preview”. Anda akan melihat `models/Country.php` pada daftar class yang akan dibuat. Anda bisa mengklik nama dari class tersebut untuk melihat isi kodenya.

Pada saat menggunakan Gii, jika anda sudah membuat file dengan nama yang sama sebelumnya dan akan menyimpannya, klik tombol `diff` disebelah nama file untuk melihat perbedaan antara kode yang akan dibuat dengan kode yang ada saat ini.

Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name
country

Model Class
Country

Namespace
app\models

Base Class
yii\db\ActiveRecord

Database Connection ID
db

☐ Use Table Prefix

☒ Generate Relations

☐ Generate Labels from DB Comments

☐ Enable I18N

Code Template
default (/Users/qiang/Web/yii/basic2/vendor/yiisoft/yii2-gii/generators/model/default)

[Preview](#) [Generate](#)

Click on the above **Generate** button to generate the files selected below:

Code File	Action
models/Country.php	overwrite

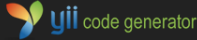
Jika akan menimpa file yang sudah ada, centang kotak di sebelah tulisan “overwrite” dan kemudian klik tombol “Generate”. Jika anda membuat file baru, anda cukup mengklik tombol “Generate”.

Selanjutnya, anda akan melihat halaman konfirmasi yang memberitahui bahwa kode berhasil dibuat. Jika sebelumnya anda sudah mempunyai file yang sama, anda juga akan melihat pesan yang memberitahukan bahwa file tersebut sudah ditimpa dengan file yang baru.

2.6.3 Membuat CRUD

CRUD adalah *Create*, *Read*, *Update*, dan *Delete*, yang merepresentasikan empat tugas umum yang melibatkan website secara umum. Untuk membuat *CRUD* menggunakan Gii, pilih tombol “CRUD Generator” (dengan cara mengklik pada halaman index Gii). Untuk contoh “negara”, isi form yang ditampilkan dengan data berikut:

- Model Class: `app\models\Country`
- Search Model Class: `app\models\CountrySearch`
- Controller Class: `app\controllers\CountryController`

 Home Help Application

Model Generator >

CRUD Generator >

Controller Generator >

Form Generator >

Module Generator >

Extension Generator >

CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

Model Class

Search Model Class

Controller Class

View Path

Base Controller Class

yii\web\Controller

Widget Used in Index Page

GridView

☐ **Enable I18N**

Code Template

default (C:\dev\yii2\extensions\yii\generators\crud\default)

[Preview](#)

A Product of [Yii Software LLC](#)

Powered by [Yii Framework](#)

Selanjutnya, klik tombol “Preview”. Anda akan melihat daftar file-file yang akan dibuat, seperti gambar dibawah ini.

Base Controller Class
yii\web\Controller

Widget Used in Index Page
GridView

☐ **Enable I18N**

Code Template
default (C:\dev\yii2\extensions\gii\generators\crud\default)

Click on the above **Generate** button to generate the files selected below:

☒ **Create** ☐ **Unchanged** ☒ **Overwrite**

Code File	Action	
controllers/CountryController.php	create	☒
models/CountrySearch.php	create	☒
views/country/_form.php	create	☒
views/country/_search.php	create	☒
views/country/create.php	create	☒
views/country/index.php	create	☒
views/country/update.php	create	☒
views/country/view.php	create	☒

Jika anda sebelumnya sudah membuat file `controllers/CountryController.php` dan `views/country/index.php` (pada bagian bekerja dengan database), centang kotak “overwrite” untuk menimpa file tersebut. (File pada bagian bekerja dengan database tidak memiliki dukungan CRUD secara penuh.)

2.6.4 Mari kita coba

Untuk melihat bagaimana proses kerjanya, gunakan browser anda untuk mengakses URL dibawah ini:

`https://hostname/index.php?r=country%2Findex`

Anda akan melihat tabel data yang menampilkan negara dari tabel pada database. Anda dapat mengurutkan tabel, atau memfilter dengan menginput pencarian filter pada baris judul kolom.

Disetiap negara yang tampil pada tabel, anda dapat memilih apakah akan melihat (*view*) detail, memperbaharui (*update*), atau menghapus (*delete*) data tersebut, anda juga dapat mengklik tombol “Create Country” yang berada di atas tabel tersebut untuk menampilkan form untuk membuat data negara yang baru.

My Company

HomeAboutContactLogin

Home / Countries

Countries

Create Country

Showing 1-10 of 10 items.

#	Code	Name	Population	
1	AU	Australia	18886000	  
2	BR	Brazil	170115000	  
3	CA	Canada	1147000	  
4	CN	China	1277558000	  
5	DE	Germany	82164700	  
6	FR	France	59225700	  
7	GB	United Kingdom	59623400	  
8	IN	India	1013662000	  
9	RU	Russia	146934000	  
10	US	United States	278357000	  

<1>

© My Company 2014

Powered by Yii Frame

My Company

HomeAboutContactLogin

Home / Countries / United States / Update

Update Country: United States

Code

US

Name

United States

Population

278357000

Update

© My Company 2014

Powered by Yii Frame

Dibawah ini adalah daftar file yang dihasilkan oleh Gii, mungkin saja anda ingin melakukan observasi bagaimana fitur-fitur ini di implementasikan, atau melakukan modifikasi terhadap file-file yang dihasilkan:

- Controller: `controllers/CountryController.php`

- Model: `models/Country.php` dan `models/CountrySearch.php`
- View: `views/country/*.php`

Info: Gii di desain agar mudah di modifikasi, dan dikembangkan. Menggunakan Gii dapat membuat pengembangan aplikasi anda menjadi lebih cepat. Untuk informasi lebih lanjut, silahkan melihat bagian Gii²⁶.

2.6.5 Penutup

Pada bagian ini, anda telah mengetahui bagaimana cara menggunakan Gii untuk membuat kode yang mengimplementasikan fungsi CRUD secara lengkap dalam mengelola data yang tersimpan pada database.

2.7 Menatap ke Depan

Jika anda membaca sepanjang bab “Mulai”, sekarang anda sudah membuat aplikasi dengan Yii. Pada proses ini, anda sudah mempelajari bagaimana mengimplementasikan fitur-fitur umum yang dibutuhkan, seperti mengambil data dari pengguna melalui form HTML, mengambil data dari database, dan menampilkan data dengan halaman per halaman. Anda juga sudah mempelajari bagaimana menggunakan Gii²⁷ untuk membuat kode secara otomatis. Menggunakan Gii dalam membuat kode, mengubah tugas-tugas pengembangan web yang cukup banyak menjadi satu tugas sederhana, sesederhana mengisi form.

Bagian ini akan merangkum bacaan Yii yang tersedia untuk membantu anda menjadi lebih produktif dalam menggunakan framework ini.

- Dokumentasi
 - Panduan Definitif²⁸: Sesuai dengan judulnya, panduan ini merincikan bagaimana Yii seharusnya bekerja dan menyediakan petunjuk umum tentang menggunakan Yii. Panduan ini sangat-sangat penting, dan panduan ini yang harus anda baca sebelum menulis kode Yii.
 - Referensi Class²⁹: Ini menjelaskan bagaimana menggunakan semua class yang disediakan oleh Yii. Pada umumnya anda akan menggunakan ini ketika anda sedang menulis kode dan ingin memahami bagaimana penggunaan *class*, *method*, *property*. Sebaiknya anda membaca referensi class ini ketika anda memiliki pemahaman dasar tentang seluruh bagian framework.
 - Artikel Wiki³⁰: Artikel wiki ditulis oleh para pengguna Yii ber-

²⁶<https://github.com/yiisoft/yii2-gii/blob/master/docs/guide/README.md>

²⁷<https://github.com/yiisoft/yii2-gii/blob/master/docs/guide/README.md>

²⁸<https://www.yiiframework.com/doc-2.0/guide-README.html>

²⁹<https://www.yiiframework.com/doc-2.0/index.html>

³⁰<https://www.yiiframework.com/wiki/?tag=yii2>

dasarkan pengalaman pribadi masing-masing. Kebanyakan dari artikel ini ditulis seperti layaknya panduan memasak, dan menunjukkan bagaimana menyelesaikan beberapa masalah dengan menggunakan Yii. Walaupun kualitas artikel-artikel ini mungkin tidak selengkap Panduan Definitif, tetapi artikel ini terkadang lebih bermanfaat karna membahas topik yang cukup luas dan mungkin mampu menyediakan solusi-solusi yang sederhana.

- Buku³¹
- Extensions³²: Yang harus dibanggakan adalah Yii memiliki ribuan library extension yang dibuat oleh pengguna yang dapat dipasang di aplikasi anda dengan mudah, dan akan membuat pengembangan aplikasi anda lebih mudah dan cepat.
- Komunitas
 - Forum: <https://forum.yiiframework.com/>
 - IRC: Kanal #yii di Libera (<ircs://irc.libera.chat:6697/yii>)
 - Gitter: <https://gitter.im/yiisoft/yii2>
 - GitHub: <https://github.com/yiisoft/yii2>
 - Facebook: <https://www.facebook.com/groups/yiitalk/>
 - Twitter: <https://twitter.com/yiiframework>
 - LinkedIn: <https://www.linkedin.com/groups/yii-framework-1483367>
 - Stackoverflow: <https://stackoverflow.com/questions/tagged/yii2>

³¹<https://www.yiiframework.com/books>

³²<https://www.yiiframework.com/extensions/>

Bab 3

Struktur Aplikasi

3.1 Tinjauan

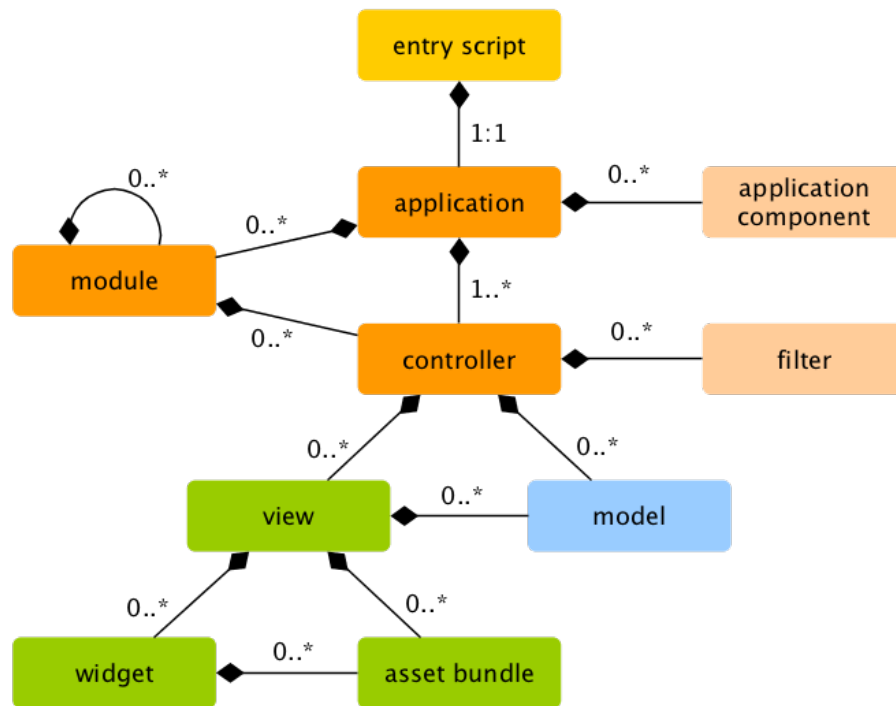
Aplikasi Yii diorganisir berdasarkan pola arsitektur model-view-controller (MVC)¹. **Model** merepresentasikan data, pengaturan dan proses bisnis; **view** adalah output yang merepresentasikan model; dan **controller** mengelola input dan merubahnya menjadi perintah-perintah untuk **model** dan **view**.

Selain MVC, aplikasi Yii juga memiliki entitas berikut:

- **entry scripts**: Ini adalah skrip PHP yang diakses secara langsung oleh pengguna. Ini bertugas untuk memulai siklus penanganan *request*.
- **applications**: Ini adalah objek yang dapat diakses secara global, yang mengelola *component* aplikasi dan mengaturnya untuk memenuhi sebuah *request*.
- **application components**: Ini adalah objek-objek yang didaftarkan pada objek *application* dan menyediakan beragam layanan untuk memenuhi *request*.
- **modules**: Ini adalah paket (*package*) mandiri yang berisikan MVC lengkap. Sebuah aplikasi boleh diistilahkan sebagai module-module yang telah diorganisir.
- **filters**: Ini merepresentasikan kode yang mutlak untuk dijalankan sebelum dan sesudah penanganan dari tiap-tiap *request* yang dikelola oleh *controller*.
- **widgets**: Ini adalah objek-objek yang dapat ditanam kedalam **views**. Ini dapat mengandung logika *controller* dan dapat digunakan berulang-ulang pada *view* yang berbeda.

Diagram dibawah ini menunjukkan struktur statis dari sebuah aplikasi:

¹<https://id.wikipedia.org/wiki/MVC>



3.2 Skrip Masuk

Skrip masuk adalah langkah pertama pada proses *bootstrap* aplikasi. Dalam sebuah aplikasi (apakah itu aplikasi web atau aplikasi konsol) memiliki satu skrip masuk. Pengguna mengirim *request* ke skrip masuk dimana skrip tersebut membangun objek aplikasi dan meneruskan *request* ke objek tersebut.

Skrip masuk untuk aplikasi web harus disimpan pada direktori yang dapat diakses dari web sehingga dapat di akses oleh pengguna. Secara umum, skrip tersebut diberi nama `index.php`, tetapi boleh menggunakan nama lain, selama *web server* bisa mengakses skrip tersebut.

Skrip masuk untuk aplikasi konsol pada umumnya disimpan di dalam *base path* dari objek aplikasi dan diberi nama `yii` (dengan suffix `.php`). Skrip tersebut harus memiliki akses *execute* sehingga pengguna dapat menjalankan aplikasi konsol menggunakan perintah `./yii <route> [argument] [option]`.

Skrip masuk umumnya mengerjakan tugas berikut ini:

- Menentukan *global constant*;
- Mendaftarkan autoloader Composer²;
- Memasukkan file *class Yii*;
- Mengambil konfigurasi aplikasi, dan memuatnya;
- Membuat dan mengatur objek *application*;

²<https://getcomposer.org/doc/01-basic-usage.md#autoloading>

- Memanggil `yii\base\Application::run()` untuk memproses *request* yang diterima;

3.2.1 Aplikasi Web

Kode berikut ini adalah kode yang terdapat pada skrip masuk *Template Proyek Dasar*.

```
<?php

defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

// mendaftarkan autoloader Composer
require __DIR__ . '/../vendor/autoload.php';

// memasukkan file class Yii
require __DIR__ . '/../vendor/yiisoft/yii2/Yii.php';

// Mengambil konfigurasi aplikasi
$config = require __DIR__ . '/../config/web.php';

// Membuat, mengkonfigurasi, dan menjalankan aplikasi
(new yii\web\Application($config))->run();
```

3.2.2 Aplikasi Konsol

Demikian juga dengan aplikasi konsol, kode berikut ini adalah kode yang terdapat pada skrip masuk aplikasi konsol :

```
#!/usr/bin/env php
<?php
/**
 * Yii console bootstrap file.
 *
 * @link https://www.yiiframework.com/
 * @copyright Copyright (c) 2008 Yii Software LLC
 * @license https://www.yiiframework.com/license/
 */

defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

// mendaftarkan autoloader composer
require __DIR__ . '/vendor/autoload.php';

// memasukkan file class Yii
require __DIR__ . '/vendor/yiisoft/yii2/Yii.php';

// Mengambil konfigurasi aplikasi
$config = require __DIR__ . '/config/console.php';
```

```
$application = new yii\console\Application($config);
$exitCode = $application->run();
exit($exitCode);
```

3.2.3 Menentukan *Constant*

Skrip masuk adalah file yang tepat untuk menentukan *global constant*. Yii mengenali tiga *constant* berikut ini:

- **YII_DEBUG**: untuk menentukan apakah aplikasi sedang dalam mode *debug*. Pada saat mode *debug*, aplikasi akan menyimpan informasi log lebih banyak, dan akan menampilkan detail error urutan pemanggilan (*error call stack*) jika ada *exception* yang di-*throw*. Alasan inilah, kenapa mode *debug* sebaiknya digunakan pada tahap pengembangan. Nilai *default* dari **YII_DEBUG** adalah **false**.
- **YII_ENV**: untuk menentukan pada mode *environment* manakah aplikasi ini dijalankan. *Constant* ini akan dijelaskan lebih lanjut di bagian Konfigurasi. Nilai *default* dari **YII_ENV** adalah **prod**, yang berarti aplikasi sedang dijalankan pada *production environment*.
- **YII_ENABLE_ERROR_HANDLER**: untuk menentukan apakah akan mengaktifkan penanganan eror yang disediakan oleh Yii. Nilai *default* dari *constant* ini adalah **true**.

Untuk menentukan *constant*, kita biasanya menggunakan kode berikut ini:

```
defined('YII_DEBUG') or define('YII_DEBUG', true);
```

kode di atas memiliki tujuan yang sama dengan kode berikut ini:

```
if (!defined('YII_DEBUG')) {
    define('YII_DEBUG', true);
}
```

Jelas, kode yang pertama lah yang lebih ringkas dan lebih mudah untuk dimengerti.

Penentuan *constant* sebaiknya ditulis di baris-baris awal pada skrip masuk sehingga akan berfungsi ketika file PHP lain akan dimasukkan (*include*).

3.3 Aplikasi

Aplikasi (*Application*) adalah objek yang mengelola semua struktur dan siklus dari sistem aplikasi Yii. Setiap aplikasi sistem Yii mengandung satu objek aplikasi yang dibuat dalam **skrip masuk** dan mampu diakses secara global melalui *expression* `\Yii::$app`.

Info: Jika kami mengatakan “sebuah aplikasi”, itu bisa diartikan sebagai sebuah objek aplikasi atau sebuah sistem aplikasi, tergantung bagaimana konteksnya.

Terdapat dua tipe aplikasi: Aplikasi Web dan Aplikasi Konsol. Sesuai dengan namanya, yang pertama bertujuan untuk menangani *web request*, sedangkan yang kedua menangani *request* perintah pada konsol.

3.3.1 Konfigurasi Aplikasi

Ketika `skrip masuk` membuat objek aplikasi, objek ini akan mengambil dan memuat sebuah `array konfigurasi` dan menerapkannya pada objek aplikasi seperti berikut ini:

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

// memuat konfigurasi aplikasi
$config = require __DIR__ . '/../config/web.php';

// membuat objek aplikasi & menerapkan konfigurasi
(new yii\web\Application($config))->run();
```

Seperti layaknya `konfigurasi` normal, konfigurasi aplikasi menentukan bagaimana proses inisialisasi *property* dari objek aplikasi. Karena konfigurasi aplikasi pada umumnya sangat kompleks, oleh karena itu konfigurasi tersebut di simpan dalam file konfigurasi, seperti file `web.php` pada contoh di atas.

3.3.2 Property Aplikasi

Terdapat cukup banyak *property* aplikasi penting yang harus anda atur dalam konfigurasi aplikasi. *Property* ini secara khusus menjelaskan *environment* yang sedang dijalankan oleh aplikasi. Sebagai contoh, aplikasi ingin mengetahui bagaimana cara memuat `controller`, dimana seharusnya aplikasi menyimpan file-file yang bersifat sementara (*temporary files*), dll. Kami akan meringkas *property* tersebut dibawah ini:

Property Wajib

Dalam aplikasi apapun, anda harus menentukan setidaknya dua *property*: `id` dan `basePath`.

`id` *Property* `id` menentukan ID unik yang membedakan objek aplikasi dengan yang lainnya. Ini pada umumnya digunakan secara programatik. Walaupun hal ini bukanlah sebuah keharusan, karena persoalan pertukaran informasi, anda sangat direkomendasikan hanya menggunakan karakter alfanumerik ketika menentukan ID dari sebuah aplikasi.

basePath *Property* **basePath** menentukan direktori *root* dari sebuah aplikasi. Yaitu direktori yang menyimpan semua sumber kode aplikasi sistem, dan aksesnya diproteksi dari luar. Didalam direktori ini, anda akan melihat sub-direktori seperti **models**, **views**, dan **controllers** yang menyimpan sumber kode dari pola MVC.

Anda dapat menentukan *property* **basePath** menggunakan *directory path* atau *path alias*. Kedua bentuk ini, direktori yang dimaksud harus benar-benar ada, jika tidak maka sebuah *exception* akan di-throw. *Path* akan dinormalkan dengan memanggil *function* **realpath()**.

Property **basePath** pada umumnya digunakan untuk mengambil *path* penting lainnya (contoh *runtime path*). Karna itulah *alias path* yang dinamakan **@app** disediakan untuk merepresentasikan *path* ini. *Path-path* lainnya boleh dipanggil menggunakan alias ini (contoh: **@app/runtime** untuk merujuk ke direktori runtime).

Property Penting

Property yang dijelaskan di sub-bagian ini cenderung harus di tentukan karena mereka digunakan secara berbeda di lintas aplikasi.

Alias *Property* ini memungkinkan anda untuk menentukan seperangkat *alias* dalam bentuk *array*. *Array Key* merupakan nama alias, dan *Array Value* adalah definisi path yang dimaksud. Sebagai contoh:

```
[
    'aliases' => [
        '@nama1' => 'path/menuju/ke/path1',
        '@nama2' => 'path/menuju/ke/path2',
    ],
]
```

Karna tersedianya *property* ini, anda bisa menentukan beberapa alias pada konfigurasi aplikasi dibanding dengan memanggil *method* **Yii::setAlias()**.

bootstrap *Property* ini merupakan *property* yang bermanfaat. *Property* ini memungkinkan anda untuk menentukan *component* berbentuk *array* yang harus dijalankan dalam **proses bootstrap**. Sebagai contoh, jika anda memerintahkan sebuah *module* untuk merubah *pengaturan URL*, anda dapat menyusun ID-nya sebagai elemen dari *property* ini.

Setiap *component* yang terdaftar pada *property* ini dapat ditentukan berdasarkan salah satu dari format berikut ini:

- ID dari *Component* aplikasi yang ditentukan melalui component,
- ID dari *module* yang ditentukan melalui module,
- Nama *class*,
- Konfigurasi *array*,

- *anonymous function* yang membuat dan mengembalikan (*return*) sebuah *component*.

Sebagai contoh:

```
[
    'bootstrap' => [
        // Component ID atau Module ID
        'demo',

        // Nama Class
        'app\components\Profiler',

        // Konfigurasi dalam bentuk array
        [
            'class' => 'app\components\Profiler',
            'level' => 3,
        ],

        // anonymous function
        function () {
            return new app\components\Profiler();
        }
    ],
]
```

Info: Jika ID *module* tersebut sama dengan ID *component* aplikasi, *component* aplikasi tersebut yang akan dipakai pada saat proses *bootstrap*. Jika anda ingin menggunakan *module*, anda dapat menentukannya melalui *anonymous function* seperti berikut ini:

```
[
    function () {
        return Yii::$app->getModule('user');
    },
]
```

Sepanjang proses *bootstrap*, setiap *component* akan dibuat objeknya. Jika *class component* mengimplementasikan *method interface yii\base\BootstrapInterface*, *method bootstrap()* dari class tersebut juga akan dipanggil.

Salah satu contoh praktis lainnya adalah konfigurasi aplikasi untuk **Template Proyek Dasar**, dimana *module debug* dan *gii* ditentukan sebagai *component bootstrap* ketika aplikasi sedang dijalankan dalam mode pengembangan:

```
if (YII_ENV_DEV) {
    // penyesuaian konfigurasi untuk environment 'dev'
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = 'yii\debug\Module';

    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = 'yii\gii\Module';
}
```

Catatan: Menentukan terlalu banyak *component* pada `bootstrap` akan menurunkan performa dari aplikasi anda, dikarenakan *component* yang sama tersebut harus dijalankan dalam setiap *request*. Jadi gunakanlah *component bootstrap* dengan bijaksana.

catchAll *Property* ini hanya dikenali oleh `Web applications`. *Property* ini menentukan sebuah *action* dari *controller* yang ditugaskan menangani semua *request* dari pengguna. *Property* ini biasanya digunakan ketika aplikasi dalam mode pemeliharaan (*maintenance*) yang mengarahkan semua *request* menuju satu *action*.

Konfigurasinya yaitu sebuah *array* dimana elemen pertama menentukan *route* dari *action*. Element lainnya (*sepasang key-value*) menentukan parameter yang akan diteruskan ke *action*. Sebagai contoh:

```
[
    'catchAll' => [
        'offline/notice',
        'param1' => 'value1',
        'param2' => 'value2',
    ],
]
```

Info: Panel *Debug* pada *development environment* tidak akan berfungsi ketika *property* ini diisi.

components *Property* ini adalah salah satu *property* yang sangat penting. *Property* ini memperbolehkan anda mendaftarkan beberapa *component* yang disebut *component aplikasi* yang bisa anda gunakan di tempat lain. Sebagai contoh:

```
[
    'components' => [
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
    ],
]
```

Setiap *component* aplikasi ditentukan dengan sepasang *key-value* ke dalam *array*. *Key* merepresentasikan ID *component*, dimana *value* merepresentasikan nama class dari *component* atau konfigurasi *array*.

Anda dapat mendaftarkan *component* apapun ke dalam objek aplikasi, dan nantinya *component* tersebut dapat diakses secara global menggunakan *expression* `\Yii::$app->componentID`.

Harap membaca bagian *Component Aplikasi* untuk penjelasan lebih lanjut.

controllerMap *Property* ini memperbolehkan anda untuk melakukan *mapping* sebuah ID *controller* ke class *controller* yang anda inginkan. Secara default, Yii melakukan mapping ID *controller* ke class *controller* berdasarkan kaidah yang ditentukan (Contoh: ID `post` akan di *mapping* ke `app\controllers\PostController`). Dengan menentukan *property* ini, anda diperbolehkan untuk tidak mengikuti kaidah untuk spesifik *controller*. Pada contoh dibawah ini, `account` akan di *mapping* ke `app\controllers\UserController`, sedangkan `article` akan di *mapping* ke `app\controllers\PostController`.

```
[
    'controllerMap' => [
        'account' => 'app\controllers\UserController',
        'article' => [
            'class' => 'app\controllers\PostController',
            'enableCsrfValidation' => false,
        ],
    ],
]
```

Key array dari *property* ini merepresentasikan ID *controller*, sedangkan *value* merepresentasikan nama `_class` yang dimaksud atau konfigurasi *array*.

controllerNamespace *Property* ini menentukan *namespace* default dimana class *controller* tersebut harus dicari. Default ke `app\controllers`. Jika ID *controller* adalah `post`, secara kaidah, nama class *controller*-nya (tanpa *namespace*) adalah `PostController`, dan `app\controllers\PostController` adalah nama class lengkapnya (*Fully Qualified Class Name*).

class *controller* juga boleh disimpan dalam sub-direktori dari direktori yang dimaksud *namespace* ini. Sebagai contoh, jika ada ID *controller* `admin/post`, nama class lengkap yang dimaksud adalah `app\controllers\admin\PostController`.

Sangatlah penting bahwa nama class lengkap dari *controller* tersebut bisa di-autoload dan *namespace* dari class *controller* anda cocok dengan nilai dari *property* ini. Jika tidak, anda akan melihat error “Halaman tidak ditemukan” ketika mengakses aplikasi.

Jika saja anda tidak ingin mengikut kaidah-kaidah yang dijelaskan di atas, anda boleh menentukan *property* `controllerMap`.

language *Property* ini menentukan bahasa apa yang seharusnya ditampilkan pada konten aplikasi ke pengguna. Nilai default dari *property* ini adalah `en`, yang merupakan Bahasa Inggris. Anda harus menentukan *property* ini jika aplikasi anda menyediakan konten dalam berbagai bahasa.

Nilai dari *property* ini menentukan banyak aspek dari *internasionalisasi*, termasuk penerjemahan pesan, format tanggal, format penomoran, dll. Sebagai contoh, *widget yii\jui\DatePicker* akan menggunakan *property* ini secara *default* untuk menentukan bahasa apa yang digunakan pada kalender yang ditampilkan dan bagaimana format tanggal pada kalender tersebut.

Disarankan agar anda menentukan bahasa dalam format Tag Bahasa IETF³. Sebagai contoh, *en* berarti Bahasa Inggris, sedangkan *en-US* berarti Bahasa Inggris yang digunakan di Amerika Serikat.

Informasi selengkapnya mengenai *property* ini dapat dipelajari di bagian *Internasionalisasi*.

modules *Property* ini menentukan *module* apa yang akan digunakan oleh aplikasi.

Property ini ditentukan menggunakan *array* dari *class class modul* atau konfigurasi *array* dimana *array key* merupakan ID dari *module* tersebut. Berikut contohnya:

```
[
    'modules' => [
        // modul "booking" dengan class module yang ditentukan
        'booking' => 'app\modules\booking\BookingModule',

        // modul "comment" yang ditentukan menggunakan konfigurasi array
        'comment' => [
            'class' => 'app\modules\comment\CommentModule',
            'db' => 'db',
        ],
    ],
]
```

Silahkan melihat bagian *Modules* untuk informasi lebih lanjut.

name *Property* ini menentukan nama aplikasi yang bisa ditampilkan ke pengguna. Berbeda dengan *property id*, yang mengharuskan nilainya unik, nilai dari *property* ini secara umum bertujuan untuk keperluan tampilan saja; tidak perlu unik.

Anda tidak perlu menentukan *property* ini jika memang tidak ada kode anda yang akan menggunakannya.

params *Property* ini menentukan parameter berbentuk *array* yang bisa diakses secara global oleh aplikasi. Dibanding menuliskan secara manual angka dan *string* di kode anda, merupakan hal yang bagus jika anda menentukan hal tersebut sebagai parameter-parameter aplikasi di satu tempat yang sama, dan menggunakannya pada tempat dimana dia dibutuhkan. Sebagai

³https://en.wikipedia.org/wiki/IETF_language_tag

contoh, anda mungkin menentukan ukuran *thumbnail* sebagai parameter seperti contoh dibawah ini:

```
[
    'params' => [
        'thumbnail.size' => [128, 128],
    ],
]
```

Kemudian, pada kode dimana anda akan menggunakan ukuran tersebut, anda cukup menggunakannya seperti kode dibawah ini:

```
$size = \Yii::$app->params['thumbnail.size'];
$width = \Yii::$app->params['thumbnail.size'][0];
```

Jika di suatu hari anda memutuskan untuk mengganti ukuran *thumbnail* tersebut, anda cukup menggantinya di konfigurasi aplikasi; anda tidak perlu mengganti di semua kode dimana anda menggunakannya.

sourceLanguage *Property* ini menentukan bahasa apa yang digunakan dalam menulis kode aplikasi. Nilai default-nya adalah `'en-US'`, yang berarti Bahasa Inggris (Amerika Serikat). Anda sebaiknya menentukan *property* ini jika teks pada kode anda bukanlah Bahasa Inggris.

Seperti layaknya *property* language, anda seharusnya menentukan *property* ini dalam format Tag Bahasa IETF⁴. Sebagai contoh, `en` berarti Bahasa Inggris, sedangkan `en-US` berarti Bahasa Inggris (Amerika Serikat).

Untuk informasi lebih lanjut mengenai *property* ini bisa anda pelajari pada bagian [Internasionalisasi](#).

timeZone *Property* ini disediakan sebagai cara alternatif untuk menentukan zona waktu default dari *PHP runtime*. Dengan menentukan *property* ini, pada dasarnya anda memanggil *function* PHP `date_default_timezone_set()`⁵. Sebagai contoh:

```
[
    'timeZone' => 'America/Los_Angeles',
]
```

version *Property* ini menentukan versi dari aplikasi anda. Secara default nilainya adalah `'1.0'`. Anda tidak harus menentukan *property* ini jika tidak ada kode anda yang akan menggunakannya.

⁴https://en.wikipedia.org/wiki/IETF_language_tag

⁵<https://www.php.net/manual/en/function.date-default-timezone-set.php>

Property yang Bermanfaat

Property yang dijelaskan pada sub-bagian ini tidak secara umum digunakan karena nilai default-nya sudah ditentukan berdasarkan kaidah-kaidah yang umum digunakan. Tetapi anda boleh menentukannya sendiri jikalau anda tidak ingin mengikuti kaidah-kaidah tersebut.

charset *Property* ini menentukan *charset* yang digunakan oleh aplikasi. Nilai default-nya adalah 'UTF-8', dimana harus digunakan sebisa mungkin pada kebanyakan aplikasi, kecuali anda sedang membangun sistem lama yang banyak menggunakan data yang tidak termasuk dalam *Unicode*.

defaultRoute *Property* ini menentukan *route* yang harus aplikasi gunakan ketika sebuah *request* tidak memiliki *route*. *Route* dapat terdiri dari ID *child module*, ID *controller*, dan/atau ID *action*. Sebagai contoh, `help`, `post/create`, atau `admin/post/create`. Jika ID *action* tidak diberikan, maka *property* ini akan mengambil nilai default yang ditentukan di `yii\base\Controller::$defaultAction`.

Untuk aplikasi Web, nilai default dari *property* ini adalah 'site', yang berarti *controller* `SiteController` dan default *action*-nya yang akan digunakan. Hasilnya, jika anda mengakses aplikasi tanpa menentukan *route* yang spesifik, maka akan menampilkan output dari `app\controllers\SiteController::actionIndex()`.

Untuk aplikasi konsol, nilai default-nya adalah 'help', yang berarti akan menggunakan `yii\console\controllers\HelpController::actionIndex()` sebagai perintah utamanya. Hasilnya, jika anda menjalankan perintah `yii` tanpa memasukkan argumen, maka akan menampilkan informasi bantuan penggunaan.

extensions *Property* ini menentukan daftar dari *extension* yang terpasang dan digunakan oleh aplikasi. Secara default, akan mengambil *array* yang dikembalikan oleh file `@vendor/yiisoft/extensions.php`. File `extensions.php` dibuat dan dikelola secara otomatis jika anda menggunakan Composer⁶ untuk memasang *extensions*. Secara umum, anda tidak perlu menentukan *property* ini.

Dalam kasus khusus jika anda ingin mengelola *extension* secara manual, anda boleh menentukan *property* ini seperti kode dibawah ini:

```
[
    'extensions' => [
        [
            'name' => 'extension name',
            'version' => 'version number',
            'bootstrap' => 'BootstrapClassName', // Tidak wajib, bisa juga
            berupa konfigurasi array
        ]
    ]
]
```

⁶<https://getcomposer.org>

```

        'alias' => [ // Tidak Wajib
            '@alias1' => 'to/path1',
            '@alias2' => 'to/path2',
        ],
    ],
    // ... extension lain yang ditentukan seperti kode di atas ...
],
]

```

Seperti yang anda lihat, *property* ini menerima spesifikasi *extension* dalam bentuk *array*. Setiap *extension* ditentukan dengan *array* yang terdiri dari elemen *name* dan *version*. Jika *extension* harus dijalankan ketika proses *bootstrap*, elemen *bootstrap* dapat dispesifikasikan dengan nama *class bootstrap*-nya atau konfigurasi *array*. *Extension* juga dapat menentukan beberapa *alias*.

layout *Property* ini menentukan nama dari default layout yang akan digunakan ketika me-render sebuah *view*. Nilai default-nya adalah *'main'*, yang berarti akan menggunakan file layout *main.php* yang disimpan di layout path. Jika kedua dari layout path dan view path mengambil nilai default, maka representasi file layoutnya adalah *path alias @app/views/layouts/main.php*.

Anda dapat menentukan nilai *property* ini menjadi *false* jika anda ingin menonaktifkan layout secara default, tetapi anda seharusnya tidak melakukannya.

layoutPath *Property* ini menentukan path dimana sistem akan mencari file layout. Nilai default-nya adalah sub-direktori *layouts* di dalam view path. Jika view path mengambil nilai defaultnya, maka path layout defaultnya adalah path alias *@app/views/layouts*.

Anda dapat menentukannya sebagai direktori atau path *alias*.

runtimePath *Property* ini menentukan dimana path file yang bersifat sementara, seperti file *log* dan *cache*. Nilai default-nya adalah direktori yang direpresentasikan oleh alias *@app/runtime*.

Anda dapat menentukan nilainya dengan direktori atau path *alias*. Sebagai catatan, *path runtime* wajib memiliki akses tulis (*writable*) oleh *web server* yang menjalankan aplikasi. Dan path tersebut sebaiknya diproteksi aksesnya dari pengguna, karena file yang bersifat sementara di dalamnya mungkin mengandung informasi sensitif.

Untuk menyederhanakan akses ke path ini, Yii sudah menentukan path alias dengan nama *@runtime*.

viewPath *Property* ini menentukan direktori *root* dimana file-file *view* akan disimpan. Nilai default-nya adalah direktori yang di representasikan oleh alias `@app/views`. Anda dapat menentukan nilainya dengan direktori atau path [alias](#).

vendorPath *Property* ini menentukan direktori *vendor* yang di kelola oleh Composer⁷. Direktori ini akan menyimpan semua *library* pihak ketiga yang digunakan oleh aplikasi anda, termasuk *Yii framework*. Nilai default-nya adalah direktori yang di representasikan oleh alias `@app/vendor`.

Anda dapat menentukan nilai *property* ini dengan direktori atau path [alias](#). Jika anda mengganti nilai *property* ini, pastikan anda juga menyesuaikan konfigurasi Composer.

Untuk memudahkan akses ke path ini, Yii sudah menentukan path alias dengan nama `@vendor`.

enableCoreCommands *Property* ini hanya dikenali oleh *console applications*. *Property* ini menentukan apakah perintah inti yang dibawa oleh rilisn Yii harus diaktifkan. Nilai default-nya adalah `true`.

3.3.3 Event Aplikasi

Sebuah objek *aplikasi* menjalankan beberapa *event* sepanjang siklus penanganan *request*. Anda dapat menempelkan penanganan *event* untuk *event-event* ini di dalam konfigurasi aplikasi seperti di bawah ini:

```
[
    'on beforeRequest' => function ($event) {
        // ...
    },
]
```

Penggunaan dari sintaks `on eventName` akan dijelaskan pada bagian Konfigurasi.

Sebagai alternatif, anda dapat menempelkan penanganan *event* ke dalam proses `bootstrap` setelah objek aplikasi telah dibuat. Sebagai contoh:

```
\Yii::$app->on(\yii\base\Application::EVENT_BEFORE_REQUEST, function
($event) {
    // ...
});
```

EVENT_BEFORE_REQUEST

Event ini dijalankan *sebelum* objek aplikasi menangani sebuah *request*. Nama *event*-nya adalah `beforeRequest`.

⁷<https://getcomposer.org>

Ketika *event* ini dijalankan, objek aplikasi sudah dibuat dan di inisialisasi. Jadi waktu ini merupakan waktu yang tepat untuk memasukkan kode anda melalui mekanisme *event* untuk mengintervensi penanganan *request*. Sebagai contoh, di penanganan *event*, anda dapat menentukan *property* `yii\base\Application::$language` secara dinamis berdasarkan parameter tertentu.

EVENT_AFTER_REQUEST

Event ini dijalankan *setelah* objek aplikasi menyelesaikan penanganan sebuah *request* tetapi *sebelum* mengirimkan *response*.

Ketika *event* ini dijalankan, proses penanganan *request* sudah selesai dan anda dapat menggunakan kesempatan untuk melakukan beberapa tugas untuk memodifikasi *request* atau *response*.

Sebagai catatan, *component response* juga menjalankan beberapa *event* pada saat mengirim isi *response* ke pengguna. *Event* tersebut akan dijalankan *setelah event* ini.

EVENT_BEFORE_ACTION

Event ini dijalankan *sebelum* semua *action* dari *controller* diproses. Nama *event*-nya adalah `beforeAction`.

Parameter *event* merupakan objek dari `yii\base\ActionEvent`. Sebuah penanganan *event* boleh menentukan *property* `yii\base\ActionEvent::$isValid` menjadi `false` untuk memberhentikan proses jalannya *action*. Sebagai contoh:

```
[
    'on beforeAction' => function ($event) {
        if (kondisi tertentu) {
            $event->isValid = false;
        } else {
        }
    },
]
```

Sebagai catatan, *event beforeAction* yang sama juga dijalankan oleh *module* dan *controller*. *Event* pada objek aplikasi yang menjalankan *event* ini untuk pertama kali, dilanjutkan oleh *module* (jika ada), dan terakhir oleh *controller*. Jika sebuah penanganan *event* menentukan *property* `yii\base\ActionEvent::$isValid` menjadi `false`, semua *event* selanjutnya TIDAK akan dijalankan.

EVENT_AFTER_ACTION

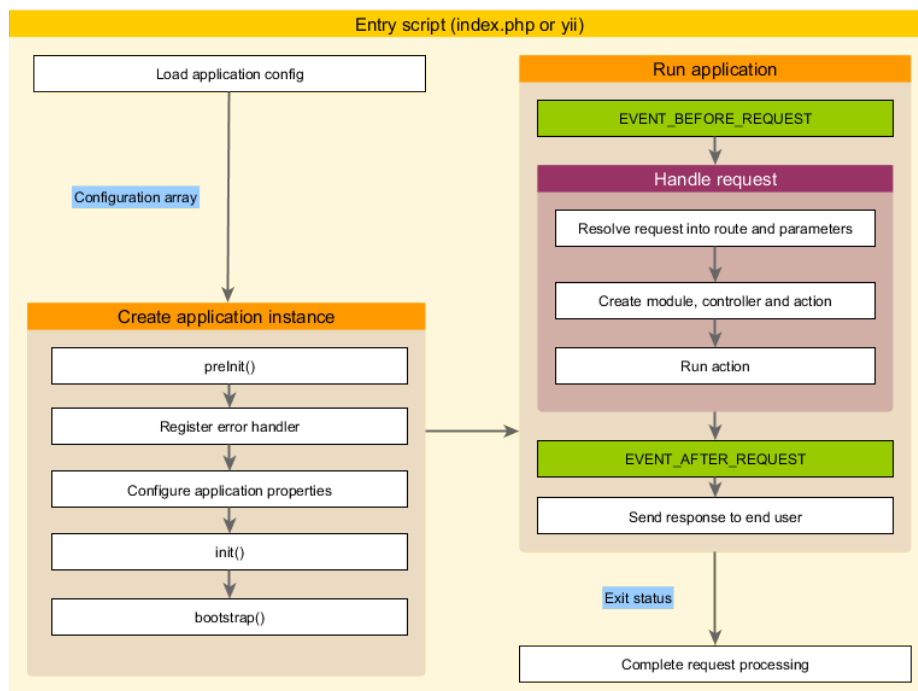
Event ini dijalankan *setelah* menjalankan seluruh *action* dari *controller*. Nama *event*-nya adalah `afterAction`.

Parameter *event* adalah objek dari `yii\base\ActionEvent`. Menggunakan *property* `yii\base\ActionEvent::$result`, *method* penanganan *event* dapat mengakses atau merubah hasil dari *action*. Sebagai contoh:

```
[
    'on afterAction' => function ($event) {
        if (kondisi tertentu) {
            // rubah nilai dari $event->result
        } else {
        }
    },
]
```

Sebagai catatan, *event* `afterAction` yang sama juga dijalankan oleh *module* dan *controllers*. Objek-objek ini menjalankan *event* ini sama seperti `beforeAction`, hanya saja urutannya merupakan kebalikan dari urutan di *event* `beforeAction`. *Controller* adalah objek pertama yang menjalankan *event* ini, setelah itu *module* (jika ada), dan terakhir di level aplikasi.

3.3.4 Application Lifecycle



Ketika skrip masuk sedang dijalankan untuk menangani sebuah *request*, aplikasi akan melewati proses siklus dibawah ini:

1. Skrip masuk mengambil konfigurasi aplikasi dalam bentuk array.

2. Skrip masuk membuat objek aplikasi:
 - `preInit()` dipanggil, dimana akan mengatur beberapa *property* aplikasi yang sangat penting seperti `basePath`.
 - Mendaftarkan **penanganan error**.
 - Mengatur *property* aplikasi.
 - `init()` dipanggil, yang selanjutnya memanggil `bootstrap()` untuk menjalankan proses *bootstrap component*.
3. Skrip masuk memanggil `yii\base\Application::run()` untuk menjalankan aplikasi:
 - Menjalankan *event* `EVENT_BEFORE_REQUEST`.
 - Menangani *request*: memproses *request* menjadi *route* dan parameter-parameternya; membuat objek *module*, *controller*, dan *action* yang dispesifikasikan oleh *route*; dan menjalankan *action*.
 - Menjalankan *event* `EVENT_AFTER_REQUEST`.
 - Mengirim *response* ke pengguna.
4. Skrip masuk mendapatkan *status exit* dari aplikasi dan menyelesaikan proses penanganan *request*.

3.4 Komponen Aplikasi

Objek Aplikasi (*Application*) adalah *service locators*. Objek ini menampung seperangkat apa yang kita sebut sebagai *komponen aplikasi* yang menyediakan berbagai layanan untuk menangani proses *request*. Sebagai contoh, *component* `urlManager` bertanggung jawab untuk menentukan *route* dari *request* menuju *controller* yang sesuai; *component* `db` menyediakan layanan terkait database; dan sebagainya.

Setiap *component* aplikasi memiliki sebuah ID yang mengidentifikasi dirinya secara unik dengan *component* aplikasi lainnya di dalam aplikasi yang sama. Anda dapat mengakses *component* aplikasi melalui *expression* berikut ini:

```
\Yii::$app->componentID
```

Sebagai contoh, anda dapat menggunakan `\Yii::$app->db` untuk mengambil koneksi ke DB, dan `\Yii::$app->cache` untuk mengambil *cache* utama yang terdaftar dalam aplikasi.

Sebuah *component* aplikasi dibuat pertama kali pada saat objek tersebut pertama diakses menggunakan *expression* di atas. Pengaksesan berikutnya akan mengembalikan objek *component* yang sama.

Component aplikasi bisa merupakan objek apa saja. Anda dapat mendaftarkannya dengan mengatur *property* `yii\base\Application::$components` pada konfigurasi aplikasi. Sebagai contoh,

```
[
    'components' => [
        // mendaftarkan component "cache" menggunakan nama class
        'cache' => 'yii\caching\ApcCache',

        // mendaftarkan component "db" menggunakan konfigurasi array
        'db' => [
            'class' => 'yii\db\Connection',
            'dsn' => 'mysql:host=localhost;dbname=demo',
            'username' => 'root',
            'password' => '',
        ],

        // mendaftarkan component "search" menggunakan anonymous function
        'search' => function () {
            return new app\components\SolrService;
        },
    ],
]
```

Info: Walaupun anda dapat mendaftarkan *component* aplikasi sebanyak yang anda inginkan, anda harus bijaksana dalam melakukan hal ini. *Component* aplikasi seperti layaknya variabel global. Menggunakan *component* aplikasi yang terlalu banyak dapat berpotensi membuat kode anda menjadi rumit untuk diuji-coba dan dikelola. Dalam banyak kasus, anda cukup membuat *component* lokal dan menggunakannya pada saat diperlukan.

3.4.1 Bootstrap Components

Seperti yang disebutkan di atas, sebuah *component* aplikasi akan dibuat ketika *component* diakses pertama kali. Jika tidak diakses sepanjang *request* diproses, objek tersebut tidak akan dibuat. Terkadang, anda ingin membuat objek *component* aplikasi tersebut untuk setiap *request*, walaupun *component* tersebut tidak diakses secara eksplisit. Untuk melakukannya, anda dapat memasukkan ID *component* tersebut ke *property bootstrap* dari objek *Application*.

Sebagai contoh, konfigurasi aplikasi di bawah ini memastikan bahwa objek *component log* akan selalu dibuat disetiap *request*:

```
[
    'bootstrap' => [
        'log',
    ],
    'components' => [
        'log' => [
            // Konfigurasi untuk component "log"
        ],
    ],
]
```

3.4.2 *Component* Aplikasi Inti

Yii menentukan seperangkat *component* aplikasi inti dengan ID tetap dan konfigurasi default. Sebagai contoh, *component request* digunakan untuk memperoleh informasi tentang *request* dari pengguna dan merubahnya menjadi *route*. *Component db* merepresentasikan sebuah koneksi ke database yang bisa anda gunakan untuk menjalankan *query* ke database. Dengan bantuan *component* inti inilah maka aplikasi Yii bisa menangani *request* dari pengguna.

Dibawah ini adalah daftar dari *component* aplikasi inti. Anda dapat mengatur dan memodifikasinya seperti *component* aplikasi pada umumnya. Ketika anda mengatur *component* aplikasi inti, jika anda tidak mendefinisikan *class*-nya, maka *class* default yang akan digunakan.

- **assetManager**: mengatur bundel aset (*asset bundles*) dan publikasi aset (*asset publishing*). Harap melihat bagian [Pengelolaan Aset](#) untuk informasi lebih lanjut.
- **db**: merepresentasikan sebuah koneksi database yang bisa anda gunakan untuk melakukan *query* ke database. Sebagai catatan, ketika anda mengatur *component* ini, anda harus menentukan nama *class* dari *component* dan *property* lain dari *component* yang dibutuhkan, seperti `yii\db\Connection::$dsn`. Harap melihat bagian [Data Access Objects](#) untuk informasi lebih lanjut.
- **errorHandler**: menangani error PHP dan *exception*. Harap melihat bagian [Menangani Error](#) untuk informasi lebih lanjut.
- **formatter**: memformat data ketika data tersebut ditampilkan ke pengguna. Sebagai contoh, sebuah angka mungkin ditampilkan menggunakan separator ribuan, dan tanggal mungkin diformat dalam format panjang. Harap melihat bagian [Memformat Data](#) untuk informasi lebih lanjut.
- **i18n**: mendukung penerjemahan dan format pesan (*message*). Harap melihat bagian [Internasionalisasi](#) untuk informasi lebih lanjut.
- **log**: mengelola target log. Harap melihat bagian [Log](#) untuk informasi lebih lanjut.
- **yii\swiftmailer\Mailer**: mendukung pembuatan dan pengiriman email. Harap melihat bagian [Mail](#) untuk informasi lebih lanjut.
- **response**: merepresentasikan *response* yang dikirimkan ke pengguna. Harap melihat bagian [Response](#) untuk informasi lebih lanjut.
- **request**: merepresentasikan *request* yang diterima dari pengguna. Harap melihat bagian [Request](#) untuk informasi lebih lanjut.
- **session**: merepresentasikan informasi *session*. *Component* ini hanya tersedia pada objek Aplikasi Web. Harap melihat bagian [Session dan Cookie](#) untuk informasi lebih lanjut.
- **urlManager**: mendukung penguraian dan pembuatan URL. Harap melihat bagian [Route dan Pembuatan URL](#) untuk informasi lebih lanjut.

- **user**: merepresentasikan informasi otentikasi dari pengguna. *Component* ini hanya tersedia pada objek **Aplikasi Web**. Harap melihat bagian **Otentikasi** untuk informasi lebih lanjut.
- **view**: mendukung proses *render view*. Harap melihat bagian **View** untuk informasi lebih lanjut.

Error: not existing file: structure-controllers.md

Error: not existing file: structure-models.md

Error: not existing file: structure-views.md

Error: not existing file: structure-modules.md

Error: not existing file: structure-filters.md

Error: not existing file: structure-widgets.md

Error: not existing file: structure-assets.md

Error: not existing file: structure-extensions.md

Bab 4

Penanganan Permintaan

Error: not existing file: runtime-overview.md

Error: not existing file: runtime-bootstrapping.md

Error: not existing file: runtime-routing.md

Error: not existing file: runtime-requests.md

Error: not existing file: runtime-responses.md

Error: not existing file: runtime-sessions-cookies.md

Error: not existing file: runtime-handling-errors.md

Error: not existing file: runtime-logging.md

Bab 5

Konsep Pokok

Error: not existing file: concept-components.md

Error: not existing file: concept-properties.md

Error: not existing file: concept-events.md

Error: not existing file: concept-behaviors.md

Error: not existing file: concept-configurations.md

Error: not existing file: concept-aliases.md

Error: not existing file: concept-autoloading.md

Error: not existing file: concept-service-locator.md

Error: not existing file: concept-di-container.md

Bab 6

Bekerja dengan Database

Error: not existing file: db-dao.md

Error: not existing file: db-query-builder.md

Error: not existing file: db-active-record.md

Error: not existing file: db-migrations.md

Bab 7

Mendapatkan Data dari Pengguna

Error: not existing file: input-forms.md

Error: not existing file: input-validation.md

Error: not existing file: input-file-upload.md

Error: not existing file: input-tabular-input.md

Error: not existing file: input-multiple-models.md

Bab 8

Menampilkan Data

Error: not existing file: output-formatting.md

Error: not existing file: output-pagination.md

Error: not existing file: output-sorting.md

Error: not existing file: output-data-providers.md

Error: not existing file: output-data-widgets.md

Error: not existing file: output-client-scripts.md

Error: not existing file: output-theming.md

Bab 9

Keamanan

Error: not existing file: security-overview.md

Error: not existing file: security-authentication.md

Error: not existing file: security-authorization.md

Error: not existing file: security-passwords.md

Error: not existing file: security-cryptography.md

Error: not existing file: security-best-practices.md

Bab 10

Caching

Error: not existing file: caching-overview.md

Error: not existing file: caching-data.md

Error: not existing file: caching-fragment.md

Error: not existing file: caching-page.md

Error: not existing file: caching-http.md

Bab 11

Layanan Web RESTful

Error: not existing file: rest-quick-start.md

Error: not existing file: rest-resources.md

Error: not existing file: rest-controllers.md

Error: not existing file: rest-routing.md

Error: not existing file: rest-response-formatting.md

Error: not existing file: rest-authentication.md

Error: not existing file: rest-rate-limiting.md

Error: not existing file: rest-versioning.md

Error: not existing file: rest-error-handling.md

Bab 12

Alat Pengembangan

Bab 13

Pengujian

Error: not existing file: test-overview.md

Error: not existing file: test-environment-setup.md

Error: not existing file: test-unit.md

Error: not existing file: test-functional.md

Error: not existing file: test-acceptance.md

Error: not existing file: test-fixtures.md

Bab 14

Topik Khusus

Error: not existing file: tutorial-start-from-scratch.md

Error: not existing file: tutorial-console.md

Error: not existing file: tutorial-core-validators.md

Error: not existing file: tutorial-i18n.md

Error: not existing file: tutorial-mailing.md

Error: not existing file: tutorial-performance-tuning.md

Error: not existing file: tutorial-shared-hosting.md

Error: not existing file: tutorial-template-engines.md

Error: not existing file: tutorial-yii-integration.md

Bab 15

Widget

Bab 16

Alat Bantu

Error: not existing file: helper-overview.md

Error: not existing file: helper-array.md

Error: not existing file: helper-html.md

Error: not existing file: helper-url.md