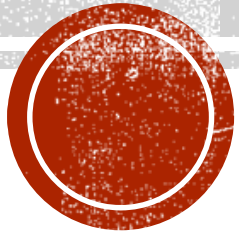
Yii2: Getting Data from Users

Oleh: Ahmad Syauqi Ahsan



Membuat Forms

- Cara utama untuk membuat form dalam Yii adalah melalui **`yii\widgets\ActiveForm`**.
- Pada banyak kasus, form yang ditampilkan ke pengguna mempunyai relasi dengan Model,
  - Model yang terhubung dengan table di database (berbasis **`yii\db\ActiveRecord`**)
  - Model yang berdiri sendiri (berbasis **`yii\base\Model`**) seperti model untuk Login Form.
- Semua yang berada diantara **`ActiveForm::begin()`** dan **`ActiveForm::end()`** (pada kode PHP), akan diletakkan didalam tag html **`<form>`** (pada file html yang di-generate).
- Untuk membuat elemen pada form (beserta label dan validasi javascript-nya) dapat dilakukan dengan memanggil metode **`ActiveForm::field()`**.
- Kode php yang menggunakan **`ActiveForm::field()`** akan menghasilkan file html yang berisi elemen <form>, <label>, <input>, dan tag-tag lainnya sesuai template yang digunakan.
- Atribut "name" pada elemen <input> akan di-generate secara otomatis dengan pola "*nama_model[nama_atribut]*". Misal: LoginForm[username]

Membuat Forms (2)

```php
// The LoginForm Model
<?php
  class LoginForm extends \yii\base\Model
  {
    public $username;
    public $password;
    public $email;

    public function rules()
    {
      return [
      // define validation rules here
      ];
    }
  }
```

```php
// The View
<?php
  use yii\helpers\Html;
  use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['id' => 'login-form',
  'options' => ['class' => 'form-horizontal'],]) ?>

  // adding a hint and a customized label
  <?= $form->field($model, 'username')->textInput()
      ->hint('Please enter your name')->label('Name') ?>
  // a password input with asterisk
  <?= $form->field($model, 'password')->passwordInput() ?>
 // creating a HTML5 email input element
  <?= $form->field($model, 'email')->input('email') ?>

  <div class="form-group">
    <div class="col-lg-offset-1 col-lg-11">
      <?= Html::submitButton('Login', [
          'class' => 'btn btn-primary'])?>
    </div>
  </div>

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

Validasi Input

- Aturan dasar: jangan percaya input dari pengguna → selalu lakukan validasi
- Input dari pengguna yang dimasukkan melalui Model dapat divalidasi dengan memanggil metode **`yii\base\Model::validate()`**.
- Metode diatas akan mengembalikan nilai "true" jika proses validasi berhasil, dan akan mengembalikan pesan error jika tidak.
- Jika memungkinkan, secara default, validasi input ini dilakukan di sisi client (menggunakan javascript).

```
// in Controller
// ...
public function actionContact()
{
  $model = new \app\models\ContactForm();

  // populate model attributes with user inputs
  if ($model->load(\Yii::$app->request->post()))
  {
    if ($model->validate()) {
      // all inputs are valid
    } else {
      // validation failed: $errors is an array
      // containing error messages
      $errors = $model->errors;
    }
  } else
  // if no user inputs, render view contact
  {
    return $this->render('contact', [
      'model' => $model
    ]);
  }
}
```

Deklarasi Rules

- Untuk membuat metode `yii\base\Model::validate()` dapat bekerja, anda perlu mendeklarasikan rule untuk setiap atribut yang akan divalidasi.
- Rule untuk validasi dapat dibuat dengan meng-override metode `yii\base\Model::rules()`.
- Metode ini harus mengembalikan nilai dalam bentuk array dari rule-rule yang dibuat.
- Setiap rule minimal memiliki atribut dimana rule tersebut diberlakukan dan tipe dari validator (misal: required, email, string, etc).
- Setiap rule dapat digunakan untuk mengatur satu atau beberapa atribut sekaligus.
- Keyword: ***Core Validators***

```
// inside the Model
// ...
public function rules()
{
  return [
    // the email attribute should be a valid email
    // address
    ['email', 'email'],

    // the name, email and birthdate
    // attributes are required
    [['name', 'email', 'birthdate'], 'required'],

    // the city attribute should be string and not
    // exceeds 50 characters in length
    [['city'], 'string', 'max' => 50],

  ];
}
// ..
```

Custom Error Messages

- Hampir semua tipe validator memiliki pesan error masing-masing.
- Untuk mengkustomisasi/mengubah pesan error dapat dilakukan dengan menambahkan property “message” ketika mendeklarasikan sebuah rule.
- Beberapa validator mendukung property tambahan yang dapat digunakan untuk menampilkan pesan error yang lebih detil.

```
// inside the Model
// ...
public function rules()
{
  return [
    // custom error message if the “name” field
    // is empty
    ['name', 'required‘,
     ‘message’ => ‘Anda harus mengisi nama’],

    // more precise custom error message
    ['payGrade', 'integer',
      'min' => 5,
      'max' => 10,
      'tooBig'=> 'Nilai terlalu besar‘
    ],
  ];
}
// ..
```

Conditional Validation

- Validasi juga dapat dilakukan hanya jika kondisi tertentu terpenuhi ← Conditional Validation.
- Hal diatas dapat dilakukan dengan cara mengisi property ‘when’ dengan fungsi yang berisi kondisi yang diinginkan.
- Jika anda menginginkan validasi juga dilakukan di sisi client, gunakan property ‘whenClient’ yang berisi string yang merepresentasikan fungsi JavaScript.
- Fungsi JavaScript ini harus mengembalikan nilai apakah akan melakukan validasi atau tidak.
- Lihat contoh disamping

```
// inside the Model
// ...
public function rules()
{
  return [
    // 'country' attribute should not be empty
    ['country', 'required'],

    // 'state' attribute should not be empty
    // only when 'country' attribute is USA
    ['state', 'required',
      'when' => function ($model) {
         return $model->country == 'USA';},
       'enableClientValidation' => false
    ],

    // 'state2' attribute should not be empty
    // only when 'country' attribute is USA
    ['state', 'required',
      'when' => function ($model) {
         return $model->country == 'USA';},
      'whenClient' => "function (attribute, value)
      {
        return $('#contactform-country').val() == 'USA';
      }"
    ],
  ];
}
// ..
```

Data Filtering

- Pada banyak kasus, masukan dari user harus di-filter terlebih dahulu.
- Sebagai contoh:
  - Menghapus spasi kosong sebelum dan sesudah nama user
  - Mengganti atribut yang kosong (tidak diisi) dengan nilai tertentu
- Validation rule juga dapat digunakan untuk mengatasi hal ini

```
// inside the Model
// ...
public function rules()
{
  return [
    // value of 'username' and 'email' attributes
    // will be trimmed before save
    [['username', 'email'], 'trim'],

    // set "username" and "email" as null if they
    // are empty
    [['username', 'email'], 'default'],

    // set "level" to be 1 if it is empty
    ['level', 'default', 'value' => 1],

    // assign "from" and "to" with a date 3 days
    // and 6 days from today, if they are empty
    [['from', 'to'], 'default',
      'value' => function ($model, $attribute) {
        return date('Y-m-d', strtotime($attribute
        === 'to' ? '+3 days' : '+6 days'));
    }],

  ];
}
// ..
```

Uploading Single File

Untuk membuat form yang menerima file upload, ikuti langkah2 berikut:

- Membuat/mengubah Model agar dapat menangani file upload.
- Membuat/mengubah View yang akan menerima file upload
- Membuat/mengubah Controller yang akan menghubungkan View dengan Model

Uploading Single File - Model -

- Tambahkan library class **`yii\web\UploadedFile`**
- Tambahkan sebuah variable/atribut baru untuk menerima file pada Model. (misal: $file1)
- Tambahkan rule yang menyatakan bahwa atribut $file1 diatas adalah sebuah file
- Anda juga dapat menambahkan validasi, misal untuk memastikan bahwa yang dapat diupload hanya file dengan ekstensi .jpg dan .gif saja.

```php
<?php
namespace app\models;

use yii\base\Model;
use yii\web\UploadedFile;

/**
* UploadForm is the model behind the upload form.
*/
class UploadForm extends Model
{
  /**
  * @var UploadedFile file attribute
  */
  public $file1;

  /**
  * @return array the validation rules.
  */
  public function rules()
  {
    return [
      [['file1'], 'file', 'extensions' => 'gif,
        jpg'],
    ];
  }
}
```

Uploading Single File - View -

- Pastikan untuk menambahkan parameter `['options' => ['enctype' => 'multipart/form-data']]` pada `ActiveForm::begin` agar file upload dapat dilakukan.
- `->fileInput()` akan membuat form input yang akan menerima file>

```
<?php
  use yii\widgets\ActiveForm;
?>

<?php
   $form = ActiveForm::begin(['options' =>
     ['enctype' => 'multipart/form-data']])
?>

  <?= $form->field($model, 'file1')->fileInput() ?>
  <button>Submit</button>

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

Uploading Single File - Controller -

- Controller bertugas untuk menghubungkan antara View dengan Model.
- Gunakan class **`yii\web\UploadedFile`**
- File akan disimpan di server ketika metode **`saveAs()`** dipanggil.

```
use yii\web\Controller;
use app\models\UploadForm;
use yii\web\UploadedFile;

class SiteController extends Controller
{
  public function actionUpload()
  {
    $model = new UploadForm();
    if (Yii::$app->request->isPost) {
      $model->file1 = UploadedFile::getInstance(
        $model, 'file1');
      if ($model->file1 && $model->validate()) {
        $model->file1->saveAs('uploads/' . $model
          ->file1->baseName . '.' .
          $model->file1->extension);
      }
    }
  return $this->render('upload', ['model' =>
    $model]);
}
```

Uploading Multiple File

Untuk membuat form untuk meng-upload beberapa file bersamaan, anda perlu melakukan beberapa perubahan:

- Pada Model: tambahkan property `'maxFiles' => 5`.
- Pada View:
  - Ubah nama atribut dari model menjadi bentuk array ('`file1`' menjadi '`file1[]`').
  - Tambahkan property `['multiple' => true]` pada metode `->fileInput()`.
- Pada Controller:
  - Ganti metode `getInstance()` menjadi `getInstances()`.
  - Gunakan `foreach()` untuk menyimpan semua file

Uploading Multiple File - Model-View-Controller -

```
use yii\web\Controller;
use app\models\UploadForm;
use yii\web\UploadedFile;

class SiteController extends Controller
{
  public function actionUpload()
  {
    $model = new UploadForm();
    if (Yii::$app->request->isPost) {
      $model->file1 = UploadedFile::getInstances(
        $model, 'file1');
      if ($model->file1 && $model->validate()) {
        foreach($model->file1 as $file) {
          $file->saveAs('uploads/' . $file->baseName
            . '.' . $model->file->extension);
        }
      }
    }
  return $this->render('upload', ['model' =>
    $model]);
}
```

```
<?php
namespace app\models;

use yii\base\Model;
use yii\web\UploadedFile;

class UploadForm extends Model
{
  public $file1;

  public function rules()
  {
    return [
      [['file1'], 'file', 'maxFiles' => 5
        'extensions' => 'gif, jpg'],
    ];
  }
}
```

```
<?php use yii\widgets\ActiveForm; ?>

<?php
   $form = ActiveForm::begin(['options' =>
     ['enctype' => 'multipart/form-data']])
?>

  <?= $form->field($model, 'file1')->fileInput(
    ['multiple' => true]) ?>
  <button>Submit</button>

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

Latihan

- Tambahkan kemampuan untuk mengupload gambar pada fungsi "create" dan "update" dari **Item** ! (aplikasi backend)
- Tampilkan daftar produk pada halaman utama dari aplikasi frontend!
- Tambahkan tombol untuk beli produk pada setiap item produk!
  Pastikan hanya user yang sudah login yang bias membeli produk.

16

Terima Kasih