Pequeño manual de Yii2

David León

Published with GitBook

Tabla de contenido

Pequeño manual de Yii2

Este proyecto ha sido creado con la finalidad de que se haga mas sencillo empezar a trabajar con Yii 2 y despejar cualquier duda que se tenga con el framework.

Esta obra es un **trabajo en progreso** por lo que su contenido está sujeto a cambios constantemente.

Requerimientos

Antes de poder instalar Yii 2 necesitas tener instalado lo siguiente:

1 - CURL

Primero debes instalar curl en windows, para esto haz click en el enlace que aparece más abajo, descarga el ejecutable y sigue las instrucciones de instalación.

Descargar curl

2 - COMPOSER

Después de instalar curl seguimos con la instalación de Composer, para esto posicionate en la carpeta donde vas a instalar los proyectos de Yii 2 y ejecuta el siguiente comando en el terminal:

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

3 - PLUGIN DE COMPOSER PARA YII 2

El tercer paso es instalar el plugin de composer para la instalación de Yii y sus extenciones. Al igual que composer, el plugin se debe instalar mediante el terminal con el siguiente commando.

```
php composer.phar global require "fxp/composer-asset-plugin:~1.1.1"
```

4 - SERVIDOR APACHE

Debes tener un servidor php instalado y ejecutandose en el sistema, yo utilizo XAMPP para windows.

Instalación

Antes de instalar Yii 2 debes tener ciertos programas instalados que he descrito en los Requerimientos.

Se tienen dos plantillas de Yii 2 disponibles al momento de crear un proyecto nuevo:

- Básico

```
php composer.phar create-project yiisoft/yii2-app-basic basic 2.0.6
```

- Avanzado

```
php composer.phar create-project yiisoft/yii2-app-advanced advanced 2.0.6
```

Ambas plantillas se instalan mediante el terminal con su comando respectivo colocados arriba.

Proyectos existentes

Si ya se tiene un proyecto creado y este ha sido cloneado de un repositorio en Github se deben actualizar/instalar todas sus dependencias y extenciones ya que estos no se encuentran en el repositorio debido a que la carpeta "Vendor" es ignorada por git por razones de tamaño.

Para actualizar/instalar las dependencias debes realizar los siguientes pasos:

1. Instalar composer en la raíz del proyecto

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

2. Actualizar/instalar las dependencias de Yii

   ```
   composer.phar update
   ```

   Si aparecen errores después de ejecutar este último comando verifica que estén instalados todos los requerimientos.

Gii

Gii es una herramienta para generación de código, para usarla debes modificar la configuración de la aplicación de la siguiente manera:

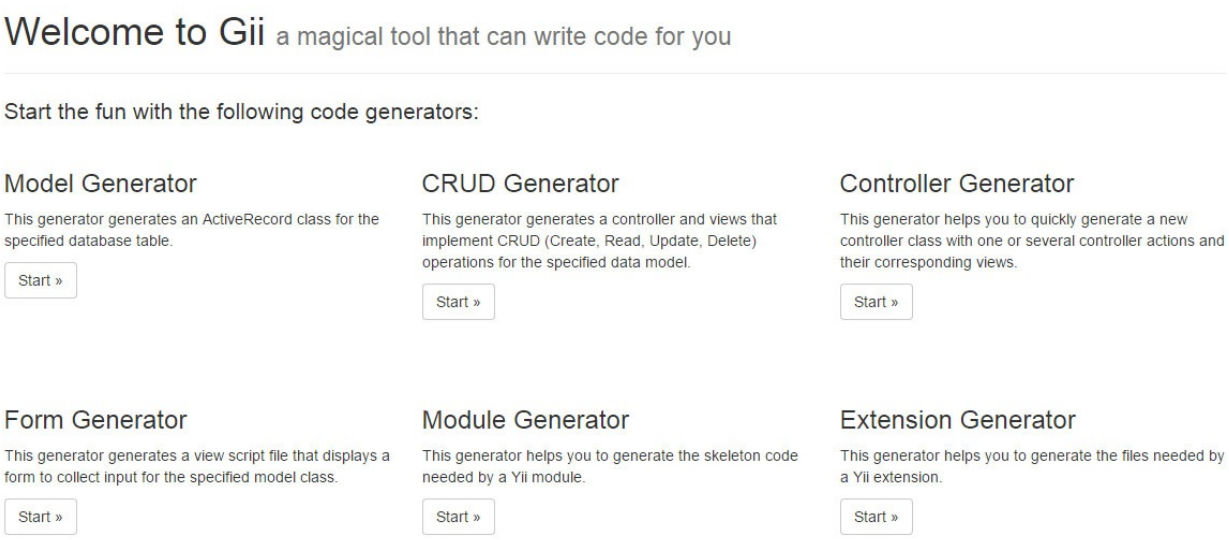
```
return array(
    ......
    'modules'=>array(
        'gii'=>array(
            'class'=>'system.gii.GiiModule',
            'password'=>'coloque una contraseña aquí',
            // 'ipFilters'=>array(...a list of IPs...),
            // 'newFileMode'=>0666,
            // 'newDirMode'=>0777,
        ),
    ),
);
```

Si la aplicación utiliza URL's del tipo *path-format* (configurado con el UrlManager) podrás acceder a Gii a través de `http://aplicacion.com/gii`.

En Gii puedes crear lo siguiente:

- Modelos
- CRUD (crear, leer, actualizar y borrar)
- Controladores
- Formularios
- Módulos
- Extensiones

Al acceder a la herramienta verás la siguiente pantalla:


Welcome to Gii a magical tool that can write code for you
Start the fun with the following code generators:
Model Generator
This generator generates an ActiveRecord class for the specified database table.
Start »
CRUD Generator
This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.
Start »
Controller Generator
This generator helps you to quickly generate a new controller class with one or several controller actions and their corresponding views.
Start »
Form Generator
This generator generates a view script file that displays a form to collect input for the specified model class.
Start »
Module Generator
This generator helps you to generate the skeleton code needed by a Yii module.
Start »
Extension Generator
This generator helps you to generate the files needed by a Yii extension.
Start »

Base de datos

Yii te hace la vida fácil al momento de conectarte a una base de datos, en la carpeta *config* encontrarás el archivo *db.php* que contiene lo siguiente:

```
<?php

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

Si deseas utilizar URLs del tipo *path-format* deberás modificar la configuración de la aplicación de la siguiente manera:

```
'urlManager' => [
    'class' => 'yii\web\UrlManager',
    'enablePrettyUrl' => true,
    'showScriptName' => false,
    'rules' => [
        // las reglas van aquí
    ],
],
```

El segundo paso es crear un archivo *.htaccess* en la carpeta web y colocar dentro de este archivo lo siguiente:

```
<IfModule mod_rewrite.c>
RewriteEngine on

# If a directory or a file exists, use it directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# Otherwise forward it to index.php
RewriteRule . index.php
</IfModule>
```

Así podrás acceder a las diferentes páginas de la aplicación a través de `http://proyecto.com/ruta-a-la/vista`, esto a su vez es muy importante para el SEO y hace amigable el sitio web para los buscadores como por ejemplo Google.

Reglas de acceso

Una forma simple de agregar control de acceso en Yii 2 es agregando la siguiente función pública en el respectivo controlador:

```
public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'actions' => ['logout', 'index','create','delete','update','view'
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }
```

De esta manera las acciones agregadas en el *array* "actions" podrán ser visitadas sólo por los usuarios que hayan iniciado sesión.

Si deseas configurar RBAC (Role-based access control) en la aplicación te recomiendo visitar este enlace.

Advanced Template

Creación de enlaces entre frontend y backend y viceversa

Después de la definición de `$params` en *frontend/config/main.php* (viceversa en backend), escribir lo siguiente:

```
$urlManagerBackEnd = require(__DIR__ . '/../../backend/config/urlManager.php');
$urlManager = require(__DIR__ . '/urlManager.php');
```

después dentro del array *components*

```
'components' => [
    ...
    'urlManager' => $urlManager,
    'urlManagerBackEnd' => $urlManagerBackEnd,
],
```

crear un archivo llamado *urlManager.php* en *frontend/config* con el siguiente contenido:

```
<?php
return [
    'class' => 'yii\web\urlManager',
    'baseUrl' => '//midominio.com',
    'showScriptName' => false,
    'enablePrettyUrl' => true
];
```

Igualmente en *backend/config* crear el archivo urlManager.php con el siguiente contenido:

```
<?php
return [
    'class' => 'yii\web\urlManager',
    'baseUrl' => '//admin.midominio.com',
    'showScriptName' => false,
    'enablePrettyUrl' => true
];
```

y para crear un enlace simplemente haz lo siguiente:

En *backend*:

```
<a href="<?php echo Yii::$app->urlManagerFrontend->createUrl('/'); ?>">Ir a frontpage</a>
```

En *frontend*:

```
<a href="<?php echo Yii::$app->urlManagerBackend->createUrl('/'); ?>">Ir a backend</a>
```

Y si quieres colocar un enlace en un menú debe ser de esta manera:

```
$menuItems[] = ['label'=>'frontend', 'url'=>\Yii::$app->urlManagerFrontend->baseUrl];
$menuItems[] = ['label'=>'frontend', 'url'=>\Yii::$app->urlManagerBackend->baseUrl];
```

Helpers

Al momento de crear la estructura HTML de una vista, Yii nos hace el trabajo más fácil con una serie de métodos para generar código rápidamente.

La forma de escribir un helper es el siguiente:

```
<?= Html::tag('p', Html::encode($user->name), ['class' => 'username']) ?>
```

El primer argumento es el nombre del helper, el segundo es el contenido, el *Html::encode* es necesario debido a que el contenido no es codificado automáticamente para permitir usar HTML cuando sea necesario. El tercer argumento es un *array* de opciones HTML, es decir, atributos como por ejemplo *class* o *id*.

El código de arriba generará el siguiente HTML:

```
<p class="username">Steven</p>
```

Formularios

El fomulario se puede iniciar con el método *beginForm()* de la siguiente manera:

```
<?= Html::beginForm(['order/update', 'id' => $id], 'post', ['enctype' => 'multipart/form-
```

El primer argumento es el *url* en el que se enviará el formulario. El segundo es el método que se utilizará (ejemplo: *GET* y *POST*), el tercero es un *array* de opciones para el tag *form*, en este caso estoy cambiando la forma de codificación de data durante el *post request* a *multipart/form-data*. Esto es requerido para poder enviar archivos en un formulario.

Para cerrar el formulario se utiliza el siguiente helper:

```
<?= Html::endForm() ?>
```

Botones

Los helpers para los botones son los siguientes:

```
<?= Html::button('Press me!', ['class' => 'teaser']) ?>
<?= Html::submitButton('Submit', ['class' => 'submit']) ?>
<?= Html::resetButton('Reset', ['class' => 'reset']) ?>
```

El primer argumento es el título del botón y el segundo son las opciones.

Inputs

```
type, input name, input value, options
<?= Html::input('text', 'username', $user->name, ['class' => $username]) ?>

type, model, model attribute name, options
<?= Html::activeInput('text', $user, 'name', ['class' => $username]) ?>
```

Existen dos tipos de helpers para el campo *input*, los que empiezan con la palabra *active*, estos se utilizan para los inputs que toman data de un modelo y atributo en específico y los normales que se especifica directamente.

Radio y Checkbox

```
<?= Html::radio('agree', true, ['label' => 'I agree']);
<?= Html::activeRadio($model, 'agree', ['class' => 'agreement'])

<?= Html::checkbox('agree', true, ['label' => 'I agree']);
<?= Html::activeCheckbox($model, 'agree', ['class' => 'agreement'])
```

Dropdown list y Listbox

```
<?= Html::dropDownList('list', $currentUserId, ArrayHelper::map($userModels, 'id', 'name'
<?= Html::activeDropDownList($users, 'id', ArrayHelper::map($userModels, 'id', 'name')) ?

<?= Html::listBox('list', $currentUserId, ArrayHelper::map($userModels, 'id', 'name')) ?>
<?= Html::activeListBox($users, 'id', ArrayHelper::map($userModels, 'id', 'name')) ?>
```

Si deseas tener multiples opciones seleccionables el *checkbox list* es una buena opción:

```
<?= Html::checkboxList('roles', [16, 42], ArrayHelper::map($roleModels, 'id', 'name')) ?>
<?= Html::activeCheckboxList($user, 'role', ArrayHelper::map($roleModels, 'id', 'name'))
```

Sino, usa *radio list*:

```
<?= Html::radioList('roles', [16, 42], ArrayHelper::map($roleModels, 'id', 'name')) ?>
<?= Html::activeRadioList($user, 'role', ArrayHelper::map($roleModels, 'id', 'name')) ?>
```

Etiquetas y errores

```
<?= Html::label('User name', 'username', ['class' => 'label username']) ?>
<?= Html::activeLabel($user, 'username', ['class' => 'label username'])
```

Para mostrar errores individualmente puedes usar el siguiente helper:

```
<?= Html::error($post, 'title', ['class' => 'error']) ?>
```

Enlaces o Hyperlinks

Existe un método para generar enlaces y es el siguiente:

```
<?= Html::a('Profile', ['user/view', 'id' => $id], ['class' => 'profile-link']) ?>
```

El primer argumento es el título, no está codificado por lo que si estás utilizando data obtenida de *usuario* debes decodificarla usando *Html::encode()*. El segundo argumento es lo que será el *href* y el tercer argumento es un *array* de propiedades del tag.

Para generar un enlace *mailto* se puede hacer de la siguiente manera:

```
<?= Html::mailto('Contact us', 'admin@example.com') ?>
```

Imágenes

Para generar tag de imágenes puede utilizar el siguiente helper:

```
<?= Html::img('@web/images/logo.png', ['alt' => 'My logo']) ?>
```

Esto genera el siguiente HTML:

```
<img src="http://example.com/images/logo.png" alt="My logo" />
```

Listas

Listas no ordenadas se pueden generar de la siguiente manera:

```
<?= Html::ul($posts, ['item' => function($item, $index) {
    return Html::tag(
        'li',
        $this->render('post', ['item' => $item]),
        ['class' => 'post']
    );
}]) ?>
```

Recursos

En esta sección colocaré cualquier recurso indispensable para Yii 2 que encuentre (Plugins, libros, etc).

Documentación

- La Guía Definitiva de Yii 2 (Inglés)

Comunidad

- Yii Framework Community (Inglés)
- Yii Framework en Stackoverflow (Inglés)
- Blog de Yii (Español)

Plugins

- Krajee Yii Extensions (Inglés)
- Yii2 Icons (Inglés)

Libros

- The Book of Yii (Inglés)

Tutoriales

- Tutsplus: Programming with Yii2: Getting Started (Inglés)