EBook Gratis

APRENDIZAJE yii2

Free unaffiliated eBook created from
Stack Overflow contributors.

#yii2

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: yii2

It is an unofficial and free yii2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official yii2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con yii2

Observaciones

Yii es un marco de programación web genérico, lo que significa que puede usarse para desarrollar todo tipo de aplicaciones web utilizando PHP. Debido a su arquitectura basada en componentes y su sofisticado soporte de almacenamiento en caché, es especialmente adecuado para el desarrollo de aplicaciones a gran escala como portales, foros, sistemas de gestión de contenido (CMS), proyectos de comercio electrónico, servicios web RESTful, etc.

Versiones

Versión	Fecha de lanzamiento
2.0.12	2017-06-05
2.0.11	2017-02-01
2.0.10	2016-10-20
2.0.9	2016-07-11
2.0.8	2016-04-28
2.0.7	2016-02-14
2.0.6	2015-08-06
2.0.5	2015-07-11
2.0.4	2015-05-10
2.0.3	2015-03-01
2.0.2	2015-01-11
2.0.1	2014-12-07
2.0.0	2014-10-12

Examples

Instalación o configuración

Yii2 se puede instalar de dos maneras. Son

1. Instalación a través de Composer
2. Instalación desde un archivo comprimido

Instalación a través de Composer

Instalando compositor

Si aún no tiene Composer instalado, puede hacerlo siguiendo las instrucciones en getcomposer.org . En Linux y Mac OS X, ejecutará los siguientes comandos:

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

Para Windows, solo descargue e instale composer-setup.exe Es posible que tenga que configurar el token de acceso a la API de github para superar el límite de velocidad de la API de Github.

Instalando Yii

Con Composer instalado, puede instalar Yii ejecutando los siguientes comandos en una carpeta accesible desde la Web:

```
composer global require "fxp/composer-asset-plugin:^1.2.0"
composer create-project --prefer-dist yiisoft/yii2-app-basic basic
```

luego ejecute el siguiente comando para instalar Yii2 con la plantilla básica.

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

Para instalar Yii2 con ejecución avanzada de plantillas.

```
composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-advanced advanced
cd advanced
php init
```

Luego, cree una nueva base de datos y ajuste la configuración de los componentes ['db'] en common / config / main-local.php en consecuencia. a continuación, ejecute el siguiente comando para

```
php yii migrate
```

Instalación desde un archivo comprimido

1. Descargue el archivo comprimido de Yii-download

2. Descomprima el archivo descargado en una carpeta accesible desde la Web.
3. Modifique el archivo config / web.php ingresando una clave secreta para el elemento de configuración cookieValidationKey

   Puedes agregar cualquier tipo de clave que quieras:

   ```
   'cookieValidationKey' => '',

   For example : xyctuyvibonp

   'cookieValidationKey' => 'xyctuyvibonp',
   ```

```
 //insert a secret key in the following (if it is empty) - this is required by cookie
validation
'cookieValidationKey' => 'enter your secret key here',
```

Instale Yii2 avanzado en Ubuntu

Primero necesitamos instalar compositor. Pasos para instalar compositor Instalar compositor.

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

Ahora cambia el directorio:

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

Ver compositor trabajando

```
composer
```

Ahora Composer instalado.

Hay dos maneras de instalar Yii2 por adelantado.

1.Instalación desde un archivo comprimido

Obtener archivo zip desde el siguiente enlace.

Descomprímalo en el directorio de destino, por ejemplo, `/var/www/html` .

https://github.com/yiisoft/yii2/releases/download/2.0.8/yii-advanced-app-2.0.8.tgz

Mover dentro de la carpeta "avanzada". Mover manualmente o escribir debajo del comando.

```
cd advanced
```

Ejecutar debajo del comando.

```
php init
```

2.Instalación a través del compositor

La instalación a través de compositor requiere token de autenticación github. Para token necesitas registrarte en GitHub.

Después de registrarte puedes generar tu token:

Pasos para generar un token

1. En la esquina superior derecha de cualquier página, haga clic en su foto de perfil, luego haga clic en Configuración.
2. En la barra lateral de configuración del usuario, haga clic en Tokens de acceso personal.
3. Haga clic en Generar nuevo token.
4. Dale a tu ficha un nombre descriptivo.
5. Seleccione los ámbitos que desea otorgar a este token.
6. Haga clic en Generar token.
7. Copia el token a tu portapapeles. Por razones de seguridad, después de salir de esta página, nadie podrá ver el token nuevamente.

Referencia: https://help.github.com/articles/creating-an-access-token-for-command-line-use/

Después de generar token copiarlo

Cambio de directorio

```
cd /var/www/html/
```

Ejecutar debajo del comando

```
composer config -g github-oauth.github.com <AuthToken>
```

ejemplo:

```
composer config -g github-oauth.github.com f1eefb8f188c22dd6467f1883cb2615c194d1ce1
```

Instalar yii2

```
composer create-project --prefer-dist yiisoft/yii2-app-advanced advanced
```

Mover dentro de la carpeta "avanzada". Mover manualmente o escribir debajo del comando.

```
cd advanced
```

Ejecutar debajo del comando.

```
php init
```

¡Está hecho!

Ahora puedes comprobarlo.

http: // localhost / advanced / frontend / web

y

http: // localhost / advanced / backend / web

Lea Empezando con yii2 en línea: https://riptutorial.com/es/yii2/topic/788/empezando-con-yii2

Capítulo 2: API de descanso

Examples

Comience con el api de descanso

Tenemos una tabla que incluye de países, por lo que creamos un modelo que se llama modelo de compatilista.

```
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "countrylist".
 *
 * @property integer $id
 * @property string $iso
 * @property string $name
 * @property string $nicename
 * @property string $iso3
 * @property integer $numcode
 * @property integer $phonecode
 */
class Countrylist extends \yii\db\ActiveRecord
{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'countrylist';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['iso', 'name', 'nicename', 'phonecode'], 'required'],
            [['numcode', 'phonecode'], 'integer'],
            [['iso'], 'string', 'max' => 2],
            [['name', 'nicename'], 'string', 'max' => 80],
            [['iso3'], 'string', 'max' => 3]
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
```

```
            'id' => 'ID',
            'iso' => 'Iso',
            'name' => 'Name',
            'nicename' => 'Nicename',
            'iso3' => 'Iso3',
            'numcode' => 'Numcode',
            'phonecode' => 'Phonecode',
        ];
    }
}
```

y creo un servicio web de descanso para eso, creamos un controlador para restapi y configuramos la variable modelClass para nuestro modelo.

```
 <?php
    namespace app\controllers;
    use yii\rest\ActiveController;
    use Yii;
    class CountrylistController extends ActiveController
    {
      public $modelClass='app\models\Countrylist';
    }
?>
```

Para usar Restapi necesitamos URLs bonitas y
Añadimos esta regla para url bonita.

```
'urlManager' => [
   'class' => 'yii\web\UrlManager',
   'enablePrettyUrl' => true,
   'showScriptName' => false,
   'rules' => [
       ['class'=>'yii\rest\UrlRule','controller'=>'countrylist']
       ],
    ],
```

después de eso accedemos a podemos probar nuestra API de descanso como ejemplo

http: // localhost / countrylist nos da una lista de condados.

Cómo anular acciones predeterminadas de resto api Yii2

Como ejemplo, desea deshabilitar la paginación en su acción de índice predeterminada y obtener todos los resultados en el índice. Como puedes hacer eso? Es sencillo. Debería anular la acción de índice en su controlador de esta manera:

```
public function actions() {
    $actions = parent::actions();
    unset($actions['index']);
    return $actions;
}

public function actionIndex() {
    $activeData = new ActiveDataProvider([
```

```
        'query' => \common\models\Yourmodel::find(),
        'pagination' => false
    ]);
    return $activeData;
}
```

Anular el tipo de contenido para una acción específica

Caso de uso: solo una acción que debe devolver un contenido plano (texto) como está:

```
public function actionAsXML()
{
    $this->layout = false;
    Yii::$app->response->format = Response::FORMAT_XML;

    return ['aaa' => [1, 2, 3, 4]];;
}
```

Los formatos de respuesta predefinidos son:

- FORMAT_HTML
- FORMAT_XML
- FORMAT_JSON
- FORMAT_JSONP
- FORMAT_RAW

No hay un tipo de mime para `text/plain` fuera del cuadro, use este en su lugar:

```
public function actionPlainText()
{
    $this->layout = false;
    Yii::$app->response->format = Response::FORMAT_RAW;
    Yii::$app->response->headers->add('Content-Type', 'text/plain');

    return $this->render('plain-text'); // outputs template as plain text
}
```

Lea API de descanso en línea: https://riptutorial.com/es/yii2/topic/6102/api-de-descanso

Capítulo 3: Cargas de archivos

Examples

Cómo hacerlo

Cargando archivos

La carga de archivos en Yii se realiza generalmente con la ayuda de [[yii \ web \ UploadedFile]], que encapsula cada archivo cargado como un objeto `UploadedFile` . Combinado con [[yii \ widgets \ ActiveForm]] y modelos, puede implementar fácilmente un mecanismo seguro de carga de archivos.

Creando Modelos

Al igual que trabajar con entradas de texto sin formato, para cargar un solo archivo, crearía una clase de modelo y usaría un atributo del modelo para mantener la instancia del archivo cargado. También debe declarar una regla de validación para validar la carga del archivo. Por ejemplo,

```
namespace app\models;

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

class UploadForm extends Model
{
    /**
     * @var UploadedFile
     */
    public $imageFile;

    public function rules()
    {
        return [
            [['imageFile'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg'],
        ];
    }

    public function upload()
    {
        if ($this->validate()) {
            $this->imageFile->saveAs('uploads/' . $this->imageFile->baseName . '.' . $this-
>imageFile->extension);
            return true;
        } else {
            return false;
        }
    }
}
```

En el código anterior, el atributo `imageFile` se usa para mantener la instancia del archivo cargado. Se asocia con una regla de validación de `file` que utiliza [[yii \ validators \ FileValidator]] para garantizar que se cargue un archivo con el nombre de extensión `png` o `jpg` . El método `upload()` realizará la validación y guardará el archivo cargado en el servidor.

El validador de `file` permite verificar las extensiones de archivo, el tamaño, el tipo MIME, etc. Consulte la sección Validadores básicos para obtener más detalles.

> Sugerencia: si está cargando una imagen, puede considerar usar el validador de `image` . El validador de `image` se implementa a través de [[yii \ validators \ ImageValidator]] que verifica si un atributo ha recibido una imagen válida que se puede guardar o procesar con la Extensión de Imagine .

Archivo de entrada de render

A continuación, cree una entrada de archivo en una vista:

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

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

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

    <button>Submit</button>

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

Es importante recordar que agrega la opción `enctype` al formulario para que el archivo se pueda cargar correctamente. La llamada `fileInput()` mostrará una etiqueta `<input type="file">` que permitirá a los usuarios seleccionar un archivo para cargar.

> Consejo: desde la versión 2.0.8, [[yii \ web \ widgets \ ActiveField :: fileInput | fileInput]] agrega la opción `enctype` al formulario automáticamente cuando se usa el campo de entrada del archivo.

Cableado

Ahora, en una acción del controlador, escriba el código para conectar el modelo y la vista para implementar la carga de archivos:

```
namespace app\controllers;

use Yii;
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) {
            $model->imageFile = UploadedFile::getInstance($model, 'imageFile');
            if ($model->upload()) {
                // file is uploaded successfully
                return;
            }
        }

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

En el código anterior, cuando se envía el formulario, se llama al método [[yii \ web \ UploadedFile :: getInstance ()]] para representar el archivo cargado como una instancia de `UploadedFile` . Luego confiamos en la validación del modelo para asegurarnos de que el archivo cargado sea válido y guarde el archivo en el servidor.

Subiendo múltiples archivos

También puede cargar varios archivos a la vez, con algunos ajustes al código enumerados en las subsecciones anteriores.

Primero debe ajustar la clase del modelo agregando la opción `maxFiles` en la regla de validación de `file` para limitar el número máximo de archivos que se pueden cargar. Establecer `maxFiles` en `0` significa que no hay límite en la cantidad de archivos que se pueden cargar simultáneamente. El número máximo de archivos que se pueden cargar simultáneamente también está limitado con la directiva de PHP `max_file_uploads` , cuyo valor predeterminado es 20. El método `upload()` también debe actualizarse para guardar los archivos cargados uno por uno.

```
namespace app\models;

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

class UploadForm extends Model
{
    /**
     * @var UploadedFile[]
     */
    public $imageFiles;

    public function rules()
    {
        return [
            [['imageFiles'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg',
'maxFiles' => 4],
        ];
    }

    public function upload()
    {
```

```
        if ($this->validate()) {
            foreach ($this->imageFiles as $file) {
                $file->saveAs('uploads/' . $file->baseName . '.' . $file->extension);
            }
            return true;
        } else {
            return false;
        }
    }
}
```

En el archivo de vista, debe agregar la opción `multiple` a la llamada `fileInput()` para que el campo de carga de archivos pueda recibir varios archivos:

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

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

    <?= $form->field($model, 'imageFiles[]')->fileInput(['multiple' => true, 'accept' =>
'image/*']) ?>

    <button>Submit</button>

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

Y finalmente, en la acción del controlador, debe llamar a `UploadedFile::getInstances()` lugar de `UploadedFile::getInstance()` para asignar una matriz de instancias de `UploadedFile` a `UploadForm::imageFiles` .

```
namespace app\controllers;

use Yii;
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) {
            $model->imageFiles = UploadedFile::getInstances($model, 'imageFiles');
            if ($model->upload()) {
                // file is uploaded successfully
                return;
            }
        }

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

Lea Cargas de archivos en línea: https://riptutorial.com/es/yii2/topic/2221/cargas-de-archivos

Capítulo 4: Componentes

Examples

Creación y uso de componentes de aplicaciones.

Pasos para crear un componente:

- Cree una carpeta llamada `components` en su carpeta raíz del proyecto
- Cree su componente dentro de la carpeta de componentes, por ejemplo: `MyComponent.php`

```
namespace app\components;

use Yii;
use yii\base\Component;
use yii\base\InvalidConfigException;


class MyComponent extends Component
{
    public function demo()
    {
        return "welcome";
    }
}
```

- Registre su componente dentro del archivo `config/web.php`

```
components' => [
    'mycomponent' => [
        'class' => 'app\components\MyComponent',
    ],
]
```

Ahora puedes usar tu método componente:

```
namespace app\controllers;

use Yii;

class DemoController extends \yii\web\Controller
{
    public function actionTest()
    {
        echo Yii::$app->mycomponent->demo();
    }
}
```

Lista desplegable usando la función de componente

Crear función en MyComponent.php

```
    namespace app\components;

    use Yii;
    use yii\base\Component;
    use yii\base\InvalidConfigException;
    use yii\helpers\Url;
    use yii\helpers\ArrayHelper;

    use app\models\User;

        class MyComponent extends Component
        {

          // Function return list of  id & user Names,used for dropdownlist
          public function getUserID()
          {
            $code = User::find()->select('id,name')
            ->where(['is_deleted'=>'n'])
            ->all();

            $result = ArrayHelper::map($code, 'id', 'name');
            if($result)
                return $result;
            else
                return ["null"=>"No User"];
          }
        }
```

-> Registrar componente en web.php

```
components' => [
        'mycomponent' => [
             'class' => 'app\components\MyComponent',
            ],
         ]
```

-> usalo en tu vista

```
<?= $form->field($model, 'user_id')->dropDownList(Yii::$app->mycomponent->getUserID())?>
```

Lea Componentes en línea: https://riptutorial.com/es/yii2/topic/2217/componentes

Capítulo 5: Enrutamiento y URLs

Observaciones

Todas las URL deben crearse a través de helper `yii\helpers\Url` . Le ayudará mucho si decide cambiar las reglas de url en urlManager.

Examples

Creando URLs

Helper yii \ helpers \ Url proporciona un conjunto de métodos estáticos para administrar URL. Este ayudante puede usarse en el código de vistas / controladores.

URL a una ruta:

```
echo Url::to(['post/index']);
```

URL a una ruta con parámetros:

```
echo Url::to(['post/view', 'id' => 100]);
```

URL anclada:

```
echo Url::to(['post/view', 'id' => 100, '#' => 'content']);
```

URL absoluta:

```
echo Url::to(['post/index'], true);
```

URL absoluta usando el esquema https:

```
echo Url::to(['post/index'], 'https');
```

Nota: La ruta que se pasa al método `Url::to()` es sensible al contexto. Puede usar el módulo actual y el controlador actual. Por ejemplo, suponga que el módulo actual es `admin` y el controlador actual es `post` :

ruta relativa solo con ID de acción (no contiene barras):

```
echo Url::to(['index']);    // -->> '/index.php?r=admin%2Fpost%2Findex'
```

Ruta relativa (no tiene barra diagonal):

```
echo Url::to(['post/index']);    // -->> '/index.php?r=admin%2Fpost%2Findex'
```

Ruta absoluta (comienza con barra):

```
echo Url::to(['/post/index']);    // -->> '/index.php?r=post%2Findex'
```

URL solicitada actual:

```
echo Url::to();
echo Url::to(['']);
```

Para crear una URL basada en la **ruta actual** y los **parámetros GET** use Url :: current () .

Supongamos `$_GET = ['id' => 10, 'page' => 7]` , la ruta actual es `post/view` .

URL actual:

```
echo Url::current();    // -->> '/index.php?r=post%2Fview&id=10&page=7'
```

URL actual sin parámetro de `page` :

```
echo Url::current(['page' => null]);  // -->> '/index.php?r=post%2Fview&id=10'
```

URL actual con parámetro de `page` cambiado:

```
echo Url::current(['page' => 12]);    // -->> '/index.php?r=post%2Fview&id=10&page=12'
```

Lea Enrutamiento y URLs en línea: https://riptutorial.com/es/yii2/topic/5510/enrutamiento-y-urls

Capítulo 6: Galletas

Observaciones

Las cookies son parte de la solicitud HTTP, por lo que es una buena idea hacer ambas cosas en el controlador, cuya responsabilidad es tratar con la solicitud y la respuesta.

Examples

Configurando una cookie

Para configurar una cookie, es decir, para crearla y programar el envío al navegador, debe crear una nueva instancia de clase `\yii\web\Cookie` y agregarla a la colección de cookies de respuesta:

```
$cookie = new Cookie([
    'name' => 'cookie_monster',
    'value' => 'Me want cookie!',
    'expire' => time() + 86400 * 365,
]);
\Yii::$app->getResponse()->getCookies()->add($cookie);
```

En lo anterior estamos pasando parámetros al constructor de la clase de cookie. Estos son básicamente los mismos que se utilizan con la función nativa de setcookie de PHP:

- `name` - nombre de la cookie.
- `value` - valor de la cookie. Asegúrate de que sea una cuerda. Los navegadores normalmente no están contentos con los datos binarios en las cookies.
- `domain` - dominio para el que está configurando la cookie.
- `expire` : la marca de tiempo de Unix que indica la hora en que se debe eliminar automáticamente la cookie.
- `path` : la ruta en el servidor en el que estará disponible la cookie.
- `secure` : si es `true` , la cookie se configurará solo si se usa HTTPS.
- `httpOnly` : si es `true` , la cookie no estará disponible a través de JavaScript.

Leyendo una galleta

Para leer una cookie usa el siguiente código:

```
$value = \Yii::$app->getRequest()->getCookies()->getValue('my_cookie');
```

Nota: este código permite leer las cookies que se han configurado mediante el componente de cookies (porque las firma de forma predeterminada). Por lo tanto, si agrega / actualiza una cookie usando el código JS, no puede leerlo usando este método (por defecto).

Galletas para subdominios

Por razones de seguridad, de forma predeterminada, las cookies son accesibles solo en el mismo dominio desde el que se configuraron. Por ejemplo, si ha configurado una cookie en el dominio `example.com` , no puede obtenerla en el dominio `www.example.com` . Entonces, si planea usar subdominios (es decir, admin.example.com, profile.example.com), necesita establecer el `domain` explícitamente:

```
$cookie = new Cookie([
    'name' => 'cookie_monster',
    'value' => 'Me want cookie everywhere!',
    'expire' => time() + 86400 * 365,
    'domain' => '.example.com' // <<<=== HERE
]);
\Yii::$app->getResponse()->getCookies()->add($cookie);
```

Ahora las cookies se pueden leer en todos los subdominios de `example.com` .

Autenticación entre subdominios y cookies de identidad

En el caso de las cookies de autologin o "recordarme", se aplican las mismas peculiaridades que en el caso de las cookies de subdominio. Pero esta vez necesita configurar el componente de usuario, configurando la matriz `identityCookie` a la configuración de cookie deseada.

Abra su archivo de configuración de la aplicación y agregue los parámetros de `identityCookie` a la configuración del componente de usuario:

```
$config = [
    // ...
    'components' => [
        // ...
        'user' => [
            'class' => 'yii\web\User',
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
            'loginUrl' => '/user/login',
            'identityCookie' => [ // <---- here!
                'name' => '_identity',
                'httpOnly' => true,
                'domain' => '.example.com',
            ],
        ],
        'request' => [
            'cookieValidationKey' => 'your_validation_key'
        ],
        'session' => [
            'cookieParams' => [
                'domain' => '.example.com',
                'httpOnly' => true,
            ],
        ],

    ],
];
```

Tenga en cuenta que `cookieValidationKey` debe ser el mismo para todos los subdominios.

Tenga en cuenta que debe configurar la propiedad `session::cookieParams` para que samedomain sea su `user::identityCookie` para garantizar que el `login logout` y el `logout` funcionen en todos los subdominios. Este comportamiento se explica mejor en la siguiente sección.

Parámetros de cookie de sesión

Los parámetros de las cookies de sesión son importantes tanto si necesita mantener la sesión mientras pasa de un subdominio a otro o cuando, por el contrario, aloja la aplicación back-end en `/admin` URL y desea manejar la sesión por separado.

```
$config = [
    // ...
    'components' => [
        // ...
        'session' => [
            'name' => 'admin_session',
            'cookieParams' => [
                'httpOnly' => true,
                'path' => '/admin',
            ],
        ],
    ],
];
```

Lea Galletas en línea: https://riptutorial.com/es/yii2/topic/2945/galletas

Capítulo 7: Gestión de activos

Sintaxis

- los activos dependientes se cargarán antes de estos activos en un orden dado
    - público $ depende = ['yii \ web \ YiiAsset', 'yii \ bootstrap \ BootstrapAsset', 'yii \ bootstrap \ BootstrapPluginAsset', 'cinghie \ fontawesome \ FontAwesomeAsset',];

Observaciones

este ejemplo se basa en la plantilla avanzada https://github.com/yiisoft/yii2-app-advanced

El activo de Cinghie en este ejemplo es el paquete de activos para adminLTE https://github.com/cinghie/yii2-admin-lte

Examples

Esto es parte del archivo de diseño

```
<?php
/* this example is based on the advanced template
 * This file is located in
 * backend/views/layouts/main.php
 */

use yii\helpers\Html;

// here the asset is registered
use cinghie\adminlte\AdminLTEAsset;
AdminLTEAsset::register($this);


?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>

 <body class="hold-transition skin-blue sidebar-mini">

<?php $this->beginBody() ?>

<?= $content ?>

<?php $this->endBody() ?>
```

```
</body>
</html>
<?php $this->endPage() ?>
```

Este es el archivo de activos

```
<?php

/**
* This file is the Asset Bundle File located in
*  vendor/cinghie/yii2-admin-lte/AdminLTEAsset.php
* @copyright Copyright &copy; Gogodigital Srls
* @company Gogodigital Srls - Wide ICT Solutions
* @website http://www.gogodigital.it
* @github https://github.com/cinghie/yii2-admin-lte
* @license GNU GENERAL PUBLIC LICENSE VERSION 3
* @package yii2-AdminLTE
* @version 1.3.10
*/

namespace cinghie\adminlte;

use yii\web\AssetBundle;

/**
 * Class yii2-AdminLTEAsset
 * @package cinghie\adminlte
 */
class AdminLTEAsset extends AssetBundle
{

    /**
     * @inherit
     */
    public $sourcePath = '@bower/';

    /**
     * @inherit
     */
    public $css = [
        'ionicons/css/ionicons.css',
        'admin-lte/dist/css/AdminLTE.css',
        'admin-lte/dist/css/skins/_all-skins.css'
    ];

    /**
     * @inherit
     */
    public $js = [
        'admin-lte/dist/js/app.js'
    ];

    /**
     * @inherit
     */
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap\BootstrapAsset',
        'yii\bootstrap\BootstrapPluginAsset',
```

```
    'cinghie\fontawesome\FontAwesomeAsset',
    ];

}
```

El HTML generado con activos cargados automáticamente.

```
<!DOCTYPE html>
<html lang="en-EN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-param" content="_csrf">
    <meta name="csrf-token"
content="M01tTVZLdlBlBQEqGyYcYHc5PwI1CRknfB1bBiAaPTNBfyk0Ehg8EQ==">
    <title>Profil</title>
    <link href="/assets/f3e48cde/css/bootstrap.css?v=1473788138" rel="stylesheet">
<link href="/assets/24e44190/css/font-awesome.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/ionicons/css/ionicons.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/admin-lte/dist/css/AdminLTE.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/admin-lte/dist/css/skins/_all-skins.css?v=1473866258"
rel="stylesheet"></head>
<body class="hold-transition skin-blue sidebar-mini">

....

</script><script src="/assets/69b4ffbe/jquery.js?v=1473788138"></script>
<script src="/assets/6aa8a809/yii.js?v=1473788138"></script>
<script src="/assets/f3e48cde/js/bootstrap.js?v=1473788138"></script>
<script src="/assets/fa4335a5/admin-lte/dist/js/app.js?v=1473866258"></script>

</body>
</html>
```

Lea Gestión de activos en línea: https://riptutorial.com/es/yii2/topic/6900/gestion-de-activos

Capítulo 8: Instalación manual de la extensión

Examples

Instalar la extensión sin compositor

> Nota: se recomienda encarecidamente utilizar Composer. La siguiente instrucción es básicamente lo que el compositor hace por ti.

- Descargar archivo de extensión de archivo de la versión necesaria desde Github
- Abrir `composer.json`
- Busque la sección de carga automática del `PSR-4` y recuérdelo, por ejemplo, para `kmit/select2`
- Extraiga los archivos a la carpeta correspondiente en la carpeta del proveedor, como `vendor/kmit/select2`
- Agregue el siguiente código al `vendor/composer/autoload_psr4.php`

```
'kmit\\select2\\' => array($vendorDir . '/kmit/select2'),
```

- Agregue el siguiente código al `vendor/yiisoft/extensions.php` :

```
'kmit/select2'(name of extension from composer.json file of extension) =>
    array (
        'name' => 'kmit/select2',
        'version' => '1.0.0.0',
        'alias' => array (
            '@vendor/kmit/select2'(path of extension folder alias) => $vendorDir .
'/kmit/select2' (path of extension folder),
        ),
    ),
```

Video Tutroial

Lea Instalación manual de la extensión en línea:
https://riptutorial.com/es/yii2/topic/2224/instalacion-manual-de-la-extension

Capítulo 9: Migraciones de base de datos

Examples

Creando Migraciones

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

El argumento del nombre requerido proporciona una breve descripción de la nueva migración. Por ejemplo, si la migración se trata de crear una nueva tabla llamada noticias, puede usar el nombre create_news_table y ejecutar el siguiente comando

```
yii migrate/create create_news_table
```

Ejemplo de archivo de migración

```
<?php

use yii\db\Migration;

class m150101_185401_create_news_table extends Migration
{
public function up()
{

}

public function down()
{
    echo "m101129_185401_create_news_table cannot be reverted.\n";

    return false;
}

/*
// Use safeUp/safeDown to run migration code within a transaction
public function safeUp()
{
}

public function safeDown()
{
}
*/
}
```

Mesa plegable

```
public function up()
{
    $this->dropTable('post');
```

```
}
```

Crear campos de tabla de inmediato

```
yii migrate/create create_post_table --fields="title:string,body:text"
```

Genera:

```
/**
 * Handles the creation for table `post`.
 */
class m150811_220037_create_post_table extends Migration
{
/**
 * @inheritdoc
 */
public function up()
{
    $this->createTable('post', [
        'id' => $this->primaryKey(),
        'title' => $this->string(),
        'body' => $this->text(),
    ]);
}

/**
 * @inheritdoc
 */
public function down()
{
    $this->dropTable('post');
}
}
```

Crear mesa

```
public function up()
{
    $this->createTable('post', [
        'id' => $this->primaryKey()
    ]);
}
```

Drop / Rename / Alter Column

```
public function up()
{
    $this->dropColumn('post', 'position');

    $this->renameColumn('post', 'owner_id', 'user_id');

    $this->alterColumn('post', 'updated', $this->timestamp()->notNull()->defaultValue('0000-
00-00 00:00:00'));
}
```

Añadir columna

```
public function up()
{
    $this->addColumn('post', 'position', $this->integer());
}
```

Revertir las migraciones

```
yii migrate/down     # revert the most recently applied migration
yii migrate/down 3   # revert the most 3 recently applied migrations
```

Migraciones transaccionales

```
public function safeUp()
{
    $this->createTable('news', [
        'id' => $this->primaryKey(),
        'title' => $this->string()->notNull(),
        'content' => $this->text(),
    ]);

    $this->insert('news', [
        'title' => 'test 1',
        'content' => 'content 1',
    ]);
}

public function safeDown()
{
    $this->delete('news', ['id' => 1]);
    $this->dropTable('news');
}
```

Una forma aún más fácil de implementar migraciones transaccionales es colocar el código de migración en los `safeUp()` y `safeDown()` . Estos dos métodos difieren de `up()` y `down()` en que están encerrados implícitamente en una transacción. Como resultado, si falla alguna operación en estos métodos, todas las operaciones anteriores se revertirán automáticamente.

Migración de múltiples bases de datos

De forma predeterminada, las migraciones se aplican a la misma base de datos especificada por el componente de la aplicación db. Si desea que se apliquen a una base de datos diferente, puede especificar la opción de línea de comandos db como se muestra a continuación:

```
yii migrate --db=db2
```

Rehacer migraciones

```
yii migrate/redo        # redo the last applied migration
```

```
yii migrate/redo 3      # redo the last 3 applied migrations
```

Listado de Migraciones

```
yii migrate/history     # showing the last 10 applied migrations
yii migrate/history 5   # showing the last 5 applied migrations
yii migrate/history all # showing all applied migrations

yii migrate/new         # showing the first 10 new migrations
yii migrate/new 5       # showing the first 5 new migrations
yii migrate/new all     # showing all new migrations
```

Modificar el historial de migración

```
yii migrate/mark 150101_185401                      # using timestamp to specify the migration
yii migrate/mark "2015-01-01 18:54:01"              # using a string that can be parsed by
strtotime()
yii migrate/mark m150101_185401_create_news_table   # using full name
yii migrate/mark 1392853618                         # using UNIX timestamp
```

Aplicando Migraciones

```
yii migrate
```

Este comando mostrará una lista de todas las migraciones que no se han aplicado hasta el momento. Si confirma que desea aplicar estas migraciones, ejecutará el método up () o safeUp () en cada nueva clase de migración, una tras otra, en el orden de sus valores de marca de hora. Si falla alguna de las migraciones, el comando se cerrará sin aplicar el resto de las migraciones.

```
yii migrate 3
yii migrate/to 150101_185401                      # using timestamp to specify the migration
yii migrate/to "2015-01-01 18:54:01"              # using a string that can be parsed by
strtotime()
yii migrate/to m150101_185401_create_news_table   # using full name
yii migrate/to 1392853618                         # using UNIX timestamp
```

Lea Migraciones de base de datos en línea: https://riptutorial.com/es/yii2/topic/1929/migraciones-de-base-de-datos

Capítulo 10: Pjax

Examples

Paso 1 Añadir Estructura

En views \ site \ form-submit.php

```
<?php Pjax::begin(['id'=>'id-pjax']); ?>
<?= Html::beginForm(['site/form-submission'], 'post', ['data-pjax' => '', 'class' => 'form-
inline']); ?>
    <?= Html::input('text', 'string', Yii::$app->request->post('string'), ['class' => 'form-
control']) ?>
    <?= Html::submitButton('Hash String', ['class' => 'btn btn-lg btn-primary', 'name' =>
'hash-button']) ?>
<?= Html::endForm() ?>
<h3><?= $stringHash ?></h3>
<?php Pjax::end(); ?>
```

Paso 2 Código del lado del servidor

```
public function actionFormSubmission()
{
    $security = new Security();
    $string = Yii::$app->request->post('string');
    $stringHash = '';
    if (!is_null($string)) {
        $stringHash = $security->generatePasswordHash($string);
    }
    return $this->render('form-submission', [
        'stringHash' => $stringHash,
    ]);
}
```

como usar pjax

Agregue esta línea al comienzo de su vista.

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

Agregue las siguientes dos líneas alrededor del contenido que necesita actualización parcial.

```
<?php Pjax::begin(['id'=>'id-pjax']); ?>
Content that needs to be updated
<?php Pjax::end(); ?>
```

recargar pjax

```
$.pjax.reload({container: '#id-pjax'});
```

usa el argumento timeout en pjax

```
<?php Pjax::begin(['id'=>'id-pjax', 'timeout' => false]); ?>
```

puede especificar un valor entero para el argumento de tiempo de espera, que sería el número de milisegundos a esperar (su valor predeterminado es 1000). Si el tiempo de ejecución en el servidor es mayor que este valor de tiempo de espera, se activará una carga de página completa.

Por defecto, pjax enviará el formulario utilizando el método GET. Puede cambiar el método de envío de formulario a POST como en el siguiente ejemplo

```
<?php Pjax::begin(['id'=>'id-pjax', 'timeout' => false, 'clientOptions' => ['method' =>
'POST']]); ?>
```

Pjax uso avanzado

Yii Framework 2.0 se entrega con soporte incorporado para Pjax , una biblioteca de JavaScript que reduce los tiempos de carga de la página. Esto se logra al actualizar solo la parte de la página que ha cambiado a través de Ajax, lo que puede traducirse en ahorros sustanciales si tiene muchos otros activos en sus páginas. Algunos de nuestros proyectos utilizan esta funcionalidad y quisimos compartir algunas lecciones aprendidas.

Problema : la página 1 es una página estática simple que contiene pocos elementos. La página 2 incluye un ActiveForm así como otros widgets. Los recursos de JavaScript de ActiveForm deben cargarse para que se ejecute el JavaScript en línea, pero como la página 1 no incluía esos recursos, la página 2 se encontró con un error de JavaScript al intentar ejecutar la línea de formulario activo: 'Uncught TypeError: undefined no es un función'.

Solución : incluya los activos de ActiveForm en un paquete de activos compartidos que se cargará en todas las páginas, asegurándose de que cualquier página de entrada permita la disponibilidad de los scripts correctos.

```
class AppAsset extends AssetBundle
{
    ...
    public $depends = [
        'yii\widgets\ActiveFormAsset',
        'yii\validators\ValidationAsset',
    ];
    ...
}
```

Problema : en el mismo ejemplo anterior, la página 1 incluye algunos widgets (barra de navegación, etc.). La página 2 incluye los mismos widgets y algunos más (ActiveForm, etc.). Al cargar la página a través de Pjax, se ejecutaba algún JavaScript en línea personalizado, pero el script en línea colocado por el widget ActiveForm no parecía funcionar, ya que el código de

validación no funcionaba. En la depuración, encontramos que la función de inicio ActiveForm se estaba ejecutando, pero la variable 'this' no parecía corresponder al ActiveForm. En realidad correspondía a la div NavBar. Al investigar los ID de div, vimos que ActiveForm esperaba tener el ID de # w1, pero a la barra de navegación ya se le asignó ese ID en la página 1 ya que ese fue el primer widget que se encontró en esa página.

Solución : No confíe en Yii para generar automáticamente las ID de los widgets por usted. En su lugar, siempre pase una ID al crear el widget para mantener el control de esas ID.

Problema : la solicitud Pjax se estaba cancelando exactamente 1.000 ms después de que se inició la solicitud.

Solución : Aumente la configuración del tiempo de espera de Pjax. El valor predeterminado es de 1 segundo, que debería ser aceptable para los sitios de producción. Sin embargo, en desarrollo, mientras se usa xdebug, nuestros tiempos de carga de la página superan regularmente este límite.

Problema : la aplicación web implementa el patrón Post-Redirect-Get (PRG) . Pjax recarga toda la página en lugar de solo la redirección.

Solución : Este es el comportamiento pretendido de Pjax. La redirección no cumple su función cuando se usa Pjax, por lo que puede determinar si una solicitud es Pjax y, de ser así, renderizar el contenido en lugar de redirigirlo. Un ejemplo puede parecer:

```
$endURL = "main/endpoint";
if (Yii::$app->request->isPjax) {
    return $this->run($endURL);
} else {
    return $this->redirect([$endURL]);
}
```

¿Cuál ha sido tu experiencia con Pjax y Yii? ¡Comenta a continuación si has encontrado algún problema o tienes mejores soluciones que las nuestras!

Lea Pjax en línea: https://riptutorial.com/es/yii2/topic/4554/pjax

Capítulo 11: Plantilla de proyecto avanzado

Examples

Despliegue en entorno de alojamiento compartido.

Implementar una plantilla de proyecto avanzada para alojamiento compartido es un poco más complicado que uno básico porque tiene dos webroots, que los servidores web de alojamiento compartido no son compatibles. Necesitaremos ajustar la estructura del directorio para que la URL de frontend sea http: //site.local y la URL de backend sea http: //site.local/admin .

Mueva los scripts de entrada a un solo webroot

En primer lugar necesitamos un directorio webroot. Cree un nuevo directorio y asígnele un nombre que coincida con su nombre de webroot de alojamiento, por ejemplo, www o public_html o similar. Luego cree la siguiente estructura donde www es el directorio webroot de alojamiento que acaba de crear:

```
www
    admin
backend
common
console
environments
frontend
...
```

> **www** será nuestro directorio frontend, así que mueva los contenidos de **frontend / web** a él. Mueva los contenidos de **backend / web** a **www / admin** . En cada caso, deberá ajustar las rutas en **index.php** y **index-test.php** .

Ajustar sesiones y cookies.

Originalmente, el backend y el frontend están diseñados para ejecutarse en diferentes dominios. Cuando lo movamos todo al mismo dominio, el frontend y el backend compartirán las mismas cookies, creando un choque. Para solucionarlo, ajuste la **configuración de la** aplicación **backend backend / config / main.php de la** siguiente manera:

```
'components' => [
    'request' => [
        'csrfParam' => '_csrf-backend',
        'csrfCookie' => [
            'httpOnly' => true,
            'path' => '/admin',
        ],
    ],
    'user' => [
        'identityClass' => 'common\models\User',
```

```
        'enableAutoLogin' => true,
        'identityCookie' => [
            'name' => '_identity-backend',
            'path' => '/admin',
            'httpOnly' => true,
        ],
    ],
    'session' => [
        // this is the name of the session cookie used for login on the backend
        'name' => 'advanced-backend',
        'cookieParams' => [
            'path' => '/admin',
        ],
    ],
],
```

Espero que esto ayude a los usuarios de alojamiento compartido a implementar aplicaciones avanzadas.

> Créditos: https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/topic-shared-hosting.md

Compartir archivos cargados entre el frontend y el backend usando enlaces simbólicos

Así que ha subido sus archivos a una carpeta say `/backend/web/uploads/` y desea que estas cargas también sean visibles en la interfaz. La opción más fácil es crear un enlace simbólico en el frontend que enlaza con el backend:

```
ln -s /path/to/backend/web/uploads/ /path/to/frontend/web/uploads
```

En tus vistas puedes usar enlaces relativos a los archivos ahora:

```
<img src='/uploads/<?= $model->image?>' alt='My Image goes here'>
<a href='/uploads/<?= $model->filename?>' target='_blank'>Download File</a>
```

Asegúrese de que su servidor web permita que se sigan los enlaces simbólicos.

Lea Plantilla de proyecto avanzado en línea: https://riptutorial.com/es/yii2/topic/944/plantilla-de-proyecto-avanzado

Capítulo 12: Pruebas

Examples

Configurar entorno de prueba

Instala Codeception:

```
composer global status
composer global require "codeception/codeception=~2.0.0" "codeception/specify=*"
"codeception/verify=*"
```

Instalar faker:

```
cd /var/www/yii                                    // Path to your application
composer require --dev yiisoft/yii2-faker:*
```

Cree una base de datos, que se usará solo para las pruebas. Puede duplicar la base de datos existente o aplicar migraciones:

```
cd tests
codeception/bin/yii migrate
```

Ajuste la configuración de los `components['db']` en `tests/codeception/config/config-local.php` .

Agregue el directorio `/var/www/yii/vendor/bin` a su ruta.

Revise todos los archivos de configuración y `.yml` .

Iniciar el servidor web, por ejemplo:

```
php -S localhost:8080
```

Ejecutar las pruebas:

```
codecept run
```

Más información:

- http://www.yiiframework.com/doc-2.0/guide-test-environment-setup.html
- http://codeception.com/install
- https://github.com/yiisoft/yii2-app-basic/tree/master/tests
- https://github.com/yiisoft/yii2-app-advanced/tree/master/tests

Nota: estas instrucciones son válidas para la versión 2.0.9 de Yii2. En la versión 2.0.10, de acuerdo con Sam Dark, la parte de prueba se volverá a factorizar (y las instrucciones deben actualizarse). La versión 2.0.10 debería lanzarse el 11 de septiembre de 2016:

https://github.com/yiisoft/yii2/milestones

Cómo burlarse de ActiveRecord

Si desea simular un AR que no intenta conectarse a la base de datos, puede hacerlo de la siguiente manera (si usa PHPUnit):

```
$post = $this->getMockBuilder('\app\model\Post')
    ->setMethods(['save', 'attributes'])
    ->getMock();
$post->method('save')->willReturn(true);
$post->method('attributes')->willReturn([
    'id',
    'status',
    'title',
    'description',
    'text'
]);
```

El problema es que debemos anular el método de los atributos (), ya que ActiveRecord de forma predeterminada obtiene una lista de atributos del esquema de base de datos que intentamos evitar.

Lea Pruebas en línea: https://riptutorial.com/es/yii2/topic/2226/pruebas

Capítulo 13: Registro activo

Observaciones

AR es perfecto cuando necesita eliminar, actualizar o crear uno o más registros secuencialmente. Su compatibilidad con los atributos sucios (al guardar solo lo que realmente se cambió) da como resultado declaraciones de ACTUALIZACIÓN optimizadas que elevan la carga de la base de datos de manera significativa y reducen las posibilidades de varios conflictos relacionados con la edición del mismo registro por varias personas al mismo tiempo.

Si no tiene una lógica realmente compleja en su aplicación y, por lo tanto, no requiere entidades abstractas, AR es la mejor opción para eliminar, actualizar y crear.

AR también está bien para consultas simples que resultan en menos de 100 registros por página. No es tan eficaz como trabajar con matrices producidas por el generador de consultas o asArray () pero es más placentero trabajar con ellas.

AR no es recomendable para consultas complejas. Por lo general, se trata de agregar o transformar datos, por lo que lo que se devuelve no se ajusta al modelo AR de todos modos. Es preferible utilizar el generador de consultas en este caso.

Lo mismo ocurre con la importación y exportación. Es mejor utilizar el generador de consultas debido a las grandes cantidades de datos y posiblemente a las consultas complejas

Examples

Encontrar todos los registros

```
Post::find()->all();
// SELECT * FROM post
```

o la taquigrafía

(Devuelve una instancia de modelo de registro activo mediante una clave principal o una matriz de valores de columna).

```
Post::findAll(condition);
```

devuelve una matriz de instancias de ActiveRecord.

Encuentra todos con donde Condición

```
$model = User::find()
        ->where(['id' => $id])
        ->andWhere('status = :status', [':status' => $status])
        ->all();
```

Encuentra a todos con el pedido por

```
$model = User::find()
         ->orderBy(['id'=>SORT_DESC])
        ->all();
Or

$model = User::find()
         ->orderBy(['id'=>SORT_ASC])
        ->all();
```

Dónde cláusula

Los operadores

```
$postsGreaterThan = Post::find()->where(['>', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at > '2016-01-25'

$postsLessThan = Post::find()->where(['<', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at < '2016-01-25'

$postsNotEqual = Post::find()->where(['<>', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at <> '2016-01-25'
```

EN

```
$postsInArray = Post::find()->where(['id' => [1,2,3]])->all();
// SELECT * FROM post WHERE id IN (1,2,3)
```

ENTRE

```
$postsInBetween = Post::find()
->where(['between', 'date', "2015-06-21", "2015-06-27" ])
->all();
```

NULO

```
$postsWithNullTitle = Post::find()->where(['title' => null]);
// SELECT * FROM post WHERE title IS NULL
```

Y

```
$postsAND = Post::find()->where(['title' => null, 'body' => null]);
// SELECT * FROM post WHERE title IS NULL AND body IS NULL
```

O

```
$postsAND = Post::find()->where(['OR', 'title IS NULL', 'body IS NULL']);
// SELECT * FROM post WHERE title IS NULL OR body IS NULL
```

NO

```
 $postsNotEqual = Post::find()->where(['NOT', ['created_at'=>'2016-01-25']])->all();
// SELECT * FROM post WHERE created_at IS NOT '2016-01-25'
```

CLASES ANULADAS

```
$postsNestedWhere = Post::find()->andWhere([
    'or',
    ['title' => null],
    ['body' => null]
])->orWhere([
    'and',
    ['not', ['title' => null]],
    ['body' => null]
]);
// SELECT * FROM post WHERE (title IS NULL OR body IS NULL) OR (title IS NOT NULL AND body IS
NULL)
```

Operador como con filtro, donde métodos de registro activo

Por ejemplo, en el filtro de búsqueda, desea filtrar la publicación al desgarrar el título de la publicación o la descripción publicada por el usuario que ha iniciado sesión actualmente.

```
$title = 'test';
$description = 'test';
```

i) yFilterWhere ()

```
$postLIKE = Post::find()->where(['user_id' => Yii::$app->user->getId()])-
>andfilterWhere(['or', ['title' => $title, 'description' => $description]])->all();
//SELECT * FROM post WHERE user_id = 2 AND ((`title` LIKE '%test%') OR (`description` LIKE
'%test%'))
```

ii) orFilterWhere ()

```
$postLIKE = Post::find()->where(['user_id' => Yii::$app->user->getId()])->orFilterWhere(['or',
['title' => $title, 'description' => $description]])->all();
//SELECT * FROM post WHERE user_id = 2 OR ((`title` LIKE '%test%') OR (`description` LIKE
'%test%'))
```

iii) filtroDonde ()

```
$postLIKE = Post::find()->filterWhere(['AND', ['title' => $title, 'description' =>
$description]])->andWhere(['user_id' => Yii::$app->user->getId()])->all();
//SELECT * FROM post WHERE ((`title` LIKE '%test%') AND (`description` LIKE '%test%')) AND
user_id = 2
```

Nota: mientras usamos filterWhere () tenemos que llamar a all ywhere () o or Where () después de filterWhere (), de lo contrario, todas las condiciones se eliminarán, excepto filterWhere ().

Crear una clase ActiveRecord con valor de campos basados en eventos

```
    <?php
    namespace models;

    use yii\db\ActiveRecord;
    use yii\behaviors\TimestampBehavior;

    class Post extends ActiveRecord
    {
        public static function tableName()
        {
            return 'post';
        }

        public function rules() {
            return [
                [['created_at', 'updated_at'], 'safe'],
            ];
        }
        public function behaviors() {
            parent::behaviors();

            return [
              'timestamp' => [
                'class' => TimestampBehavior::className(),
                'attributes' => [
                  ActiveRecord::EVENT_BEFORE_INSERT => ['created_at', 'updated_at'],
                  ActiveRecord::EVENT_BEFORE_UPDATE => ['updated_at']
                ],
                'value' => date('Y-m-d H:i:s'),
              ]
            ];
          }
    }
```

O esto puede ser usado

```
public function beforeSave($insert)
{
    if($this->isNewRecord){
        //When create
    }else{
         //When update
    }

    return parent::beforeSave($insert);
}

  public function afterSave($insert, $changedAttributes )
{
    if($insert){
        //When create
    }else{
         //When update
    }
    return parent::afterSave($insert, $changedAttributes);
}
```

Encontrar un registro

```
$customer = Customer::findOne(10);
```

o

```
$customer = Customer::find()->where(['id' => 10])->one();
```

o

```
$customer = Customer::find()->select('name,age')->where(['id' => 10])->one();
```

o

```
$customer = Customer::findOne(['age' => 30, 'status' => 1]);
```

o

```
$customer = Customer::find()->where(['age' => 30, 'status' => 1])->one();
```

Encontrar una consulta

Encuentra un solo registro basado en id.

```
$model = User::findOne($id);
```

Seleccione una sola columna basada en id.

```
$model = User::findOne($id)->name;
```

Recuperar el registro único de la base de datos en función de la condición.

```
$model = User::find()->one();  // give first record

$model = User::find()->where(['id' => 2])->one(); // give single record based on id
```

Seleccione el registro de columnas individuales de la base de datos según la condición.

```
$model = User::find()->select('name,email_id')->where(['id' => 1])->one();
```

O

```
$model = User::find()->select(['id','name','email_id'])->where(['id' => 1])->one();
```

Orden por

```
$model = User::find()->select(['id','name','email_id'])->orderBy(['id' => SORT_DESC])->one();

OR
```

```
$model = User::find()->select(['id','name','email_id'])->orderBy(['id' => SORT_ASC])->one();
```

Registros activos con sub consultas

Ejemplo: Clientes que pueden crear una publicación. Cada cliente puede crear múltiples publicaciones. Tan pronto como el cliente cree la publicación, la publicación estará bajo la revisión de los administradores. Ahora tenemos que buscar la lista de clientes que tienen todas las publicaciones activas mediante una subconsulta.

Nota: Si un cliente tiene 5 publicaciones, entre 5 publicaciones si tiene al menos una inactiva, debemos excluir a este cliente de la lista de clientes.

```
$subQuery = Post::find()->select(['customer_id'])->where(['status' => 2]); //fetch the
customers whos posts are inactive - subquery
$query = Customer::find()->where(['NOT IN', 'id', $subQuery])->all(); //Exclude the customers
whos posts are inactive by using subquery
```

Lea Registro activo en línea: https://riptutorial.com/es/yii2/topic/1516/registro-activo

Capítulo 14: Sesión

Examples

Sesión en yii2

importar clase de sesión

```
use  yii\web\Session;
```

Crear una sesion

```
$session = Yii::$app->session;
$session->open(); // open a session
$session->close();  // close a session
```

Almacenar el valor en la variable de sesión.

```
$session = Yii::$app->session;

$session->set('name', 'stack');
OR
$session['name'] = 'stack';
OR
$_SESSION['name'] = 'stack';
```

Obtener el valor de la variable de sesión.

```
$name = $session->get('name');
OR
$name = $session['name'];
```

Eliminar la variable de sesión

```
$session->remove('name');
OR
unset($session['name']);
OR
unset($_SESSION['name']);

$session->destroy(); // destroy all session
```

Eliminar todas las variables de sesión

```
$session->removeAll();
```

Compruebe la variable de sesión

```
$session->has('name')
OR
isset($session['name'])
//both function return boolean value [true or false]
```

Flash de sesión

Establecer flash de sesión

```
$session = Yii::$app->session;
$session->setFlash('error', 'Error in login');
```

Obtener flash de sesión

```
echo $session->getFlash('error');
```

Ver flash de sesión

```
$result = $session->hasFlash('error');
```

Eliminar la sesión de flash

```
$session->removeFlash('error');
```

Eliminar todas las variables flash de sesión

```
$session->removeAllFlashes();
```

Utilizar directamente la variable de sesión.

Establecer y obtener la variable de sesión

```
\Yii::$app->session->set('name','stack');
\Yii::$app->session->get('name');
```

Flash de sesión

```
\Yii::$app->getSession()->setFlash('flash_msg','Message');
\Yii::$app->getSession()->getFlash('flash_msg');
```

Creando y editando variables de sesión que son matrices.

Guardar la variable de sesión como una variable.

```
$session = Yii::$app->session;

$sess = $session['keys'];
```

Luego crea o actualiza el valor de la matriz que deseas

```
$sess['first'] = 'abc';
```

Y finalmente guardar en la variable de sesión.

```
$session['keys'] = $sess
```

Recordar URL para volver a visitar más tarde

Caso de uso: recuerde la URL actual a la que volver después de agregar un nuevo registro en un controlador diferente (relacionado), por ejemplo, cree un nuevo contacto para agregar a la factura que se está editando.

InvoiceController / actionUpdate:

```
Url::remember(Url::current(), 'returnInvoice');
```

ContactController / actionCreate:

```
if ($model->save()) {
    $return = Url::previous('returnInvoice');
    if ($return) {
        return $this->redirect($return);
    }
    // ...
}
```

Puede restablecer la URL recordada una vez que haya terminado:

InvoiceController / actionUpdate:

```
if ($model->save()) {
    Url::remember(null, 'returnInvoice');
    // ...
}
```

El nombre de la clave - `returnInvoice` en este ejemplo - es opcional.

Lea Sesión en línea: https://riptutorial.com/es/yii2/topic/3584/sesion

Capítulo 15: Solicitud de Ajax

Examples

Enviando el formulario Ajax

Ver archivo:

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

<?php
$form = ActiveForm::begin([
    'action' => ['comments/ajax-comment'],
    'options' => [
        'class' => 'comment-form'
    ]
]);
?>
    <?= $form->field($model, 'comment'); ?>

    <?= Html::submitButton("Submit", ['class' => "btn"]); ?>

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

Javascript:

```
 jQuery(document).ready(function($) {
       $(".comment-form").submit(function(event) {
            event.preventDefault(); // stopping submitting
            var data = $(this).serializeArray();
            var url = $(this).attr('action');
            $.ajax({
                url: url,
                type: 'post',
                dataType: 'json',
                data: data
            })
            .done(function(response) {
                if (response.data.success == true) {
                    alert("Wow you commented");
                }
            })
            .fail(function() {
                console.log("error");
            });

        });
    });
```

Acción del controlador:

```
public function actionAjaxComment()
{
    $model = new Comments();
    if (Yii::$app->request->isAjax) {
        Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;

        if ($model->load(Yii::$app->requset->post()) && $model->save()) {
            return [
                'data' => [
                    'success' => true,
                    'model' => $model,
                    'message' => 'Model has been saved.',
                ],
                'code' => 0,
            ];
        } else {
            return [
                'data' => [
                    'success' => false,
                    'model' => null,
                    'message' => 'An error occured.',
                ],
                'code' => 1, // Some semantic codes that you know them for yourself
            ];
        }
    }
}
```

Vista Ajax Render

`Controller::renderAjax()` método `Controller::renderAjax()` se puede utilizar para responder a una solicitud Ajax. Este método es similar a renderPartial () excepto que se inyectará en el resultado de renderizado con scripts JS / CSS y archivos registrados con la vista

Supongamos que tenemos un formulario de inicio de sesión en un archivo de vista:

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

\yii\bootstrap\BootstrapAsset::register($this);

<div class="site-login">

    <?php $form = ActiveForm::begin(); ?>

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

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

        <?= Html::submitButton('Login',['class' => 'btn btn-primary btn-block']) ?>

    <?php ActiveForm::end(); ?>
</div>
```

Cuando usamos `renderPartial()` en una acción de controlador:

```
public function actionLogin()
{
    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        return $this->goBack();
    }
    return $this->renderPartial('login', [
        'model' => $model,
    ]);
}
```

Salida de acción:

```
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
</div>
```

Cuando usamos `renderAjax()` en una acción de controlador:

```
    ...
    return $this->renderAjax('login', [
        'model' => $model,
    ]);
    ...
```

Salida de acción (JS, CSS inyectado):

```
<link href="/assets/f1759119/css/bootstrap.css" rel="stylesheet">
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
</div>
<script src="/assets/13aa7b5d/jquery.js"></script>
<script src="/assets/302a2946/yii.js"></script>
```

```
<script src="/assets/302a2946/yii.validation.js"></script>
<script src="/assets/302a2946/yii.activeForm.js"></script>
```

Si queremos excluir algunos activos de la vista (para evitar publicaciones):

```
...
Yii::$app->assetManager->bundles = [
    'yii\bootstrap\BootstrapAsset' => false,
];
return $this->renderAjax('login', [
    'model' => $model,
]);
...
```

Salida de acción (no bootstrap.css):

```
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
</div>
<script src="/assets/13aa7b5d/jquery.js"></script>
<script src="/assets/302a2946/yii.js"></script>
<script src="/assets/302a2946/yii.validation.js"></script>
<script src="/assets/302a2946/yii.activeForm.js"></script>
```

Lea Solicitud de Ajax en línea: https://riptutorial.com/es/yii2/topic/2944/solicitud-de-ajax

Capítulo 16: Trabajar con bases de datos

Examples

Usando el constructor de consultas Yii2

Yii2 proporciona formas eficientes de recuperar datos de la base de datos. ***Considere*** un ejemplo de una tabla de empleados simple con campos ***emp_id, emp_name y emp_salary*** . Para recuperar los nombres de los empleados y sus salarios, utilizamos la consulta.

```
select emp_name,emp_salary from employee
```

Para generar la consulta anterior en Yii2, hay muchos métodos. ***Uno*** de los métodos es usar un objeto ***yii \ db \ Quer*** *y* .

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();

$rows=$query->select(['emp_name','emp_salary']) //specify required columns in an array
             ->from('employee') //specify table name
             ->all(); //returns an array of rows with each row being an associative array of
name-value pairs.
```

Podemos hacer uso de un bucle foreach para recorrer cada par nombre-valor en la matriz ***$ rows*** .

```
foreach ($rows as $row) {
    echo "Employee Name: ".$row['emp_name'].",Employee Salary: ".$row['emp_salary']."<br>";
}
```

Esto dará salida

> Nombre del empleado: Kiran, Salario del empleado: 25000
>
> Nombre del empleado: Midhun, Salario del empleado: 50000
>
> Nombre del empleado: Jishnu, Salario del empleado: 20000
>
> Nombre del empleado: Ajith, Salario del empleado: 25000
>
> Nombre del empleado: Akshay, Salario del empleado: 750000

Más ejemplos

Supongamos que necesitamos encontrar el nombre de los empleados cuyo salario es igual a 25000. Podemos escribir la consulta en SQL como

```
select emp_name from employee where salary=25000
```

En Yii2, el código para generar la consulta anterior.

```
$query=new \yii\db\Query();

$rows=$query->select(['emp_name'])
            ->from('employee')
            ->where(['emp_salary'=>25000]) //specify the condition as an associative array
where key is column name
            ->all();
```

Si necesitamos encontrar nombres de empleados cuyo salario sea superior a 25000, podemos escribir el código en Yii2 como

```
  $rows=$query->select(['emp_name'])
        ->from('employee')
        ->where(['>','emp_salary', 25000])
//Here first element in the above array specify relational operator used, second element
specify the table name and third the value itself.
        ->all();
```

Más verificación de condición usando where ()

Se pueden escribir múltiples condiciones usando el método **where ()** como se indica a continuación.

```
// Creates a new \yii\db\Query() object
$query = new \yii\db\Query();
$rows = $query->select(['emp_name','emp_salary'])
        ->from('employee')
        ->where(['emp_name' => 'Kiran', 'emp_salary' => 25000]) // Specify multiple conditions
        ->one(); // Returns the first row of the result
```

El código anterior buscará a un empleado que tenga el nombre de **kiran** y el salario **25000** . Si varios empleados cumplen con la condición anterior, el llamado ***one ()*** se asegura de que solo se obtenga el primer resultado. Para obtener todos los resultados debes usar ***all ()*** .

Tenga en cuenta que si usa ***all ()*** el resultado siempre será una matriz; Incluso si hay solo uno o cero resultados. Esta matriz contiene todos los resultados como matrices o está vacía cuando no coinciden los registros. La llamada ***one ()*** devolverá la matriz resultante directamente o falsa si la consulta no devuelve nada.

El código equivalente en sql se da a continuación.

```
select emp_name, emp_salary from employee where emp_name = 'Kiran' and emp_salary = 25000
limit 1;
```

A continuación se proporciona una forma alternativa de escribir la consulta anterior en Yii2.

```
$rows = $query->select(['emp_name', 'emp_salary'])
    ->from('employee')
    ->where(['emp_name' => 'Kiran'])
```

```
    ->andWhere(['emp_salary' => 25000])
    ->one();
```

Se puede especificar un conjunto adicional de condiciones usando **andWhere** . Esto será útil si necesitamos agregar una verificación de condición adicional a la consulta más adelante.

Otra forma de especificar múltiples condiciones es mediante el uso del **formato** de **operador del** método **where ().** La consulta anterior también se puede escribir como se indica a continuación.

```
$rows = $query->select(['emp_name','emp_salary'])
    ->from('employee')
    ->where(['and', 'emp_name="kiran"', 'emp_salary=25000'])
    ->one();
```

Aquí especificamos el operador ' **y** ' como el primer elemento de la matriz. De manera similar, también podemos usar ' **o** ', ' **entre** ', ' **no entre** ', ' **en** ', ' **no en** ', ' **como** ', ' **o como** ', ' **no como** ', ' **o no como** ', ' **existe** ' , ' **no existe** ', ' **>** ', ' **<=** ' etc. como operadores.

Ejemplos de uso de 'in' y 'like'

Supongamos que necesitamos encontrar empleados que tengan salarios **20000, 25000 y 50000** . En sql normal escribiríamos la consulta como

```
select * from employee where salary in (20000,25000,50000)
```

En Yii2 podemos escribir esto como se indica a continuación.

```
$rows = $query->from('employee')
        ->where(['emp_salary' => [20000,25000,50000]])
        ->all();
```

Otra forma de especificar la misma condición es

```
$rows = $query->from('employee')
    ->where(['in', 'emp_salary', [20000,25000,50000]]) // Making use of operator format of
where() method
    ->all();
```

De manera similar, " **no en** " se puede especificar en lugar de " **en** " si queremos que todos los empleados no tengan salarios 20000, 25000 y 50000.

Ahora veamos algunos ejemplos de uso de "me **gusta** " dentro de la condición (). Supongamos que necesitamos encontrar a todos los empleados que tienen la cadena ' **gopal** ' en su nombre. Los nombres pueden ser venugopal, rajagopal, gopalakrishnan, etc. La consulta de SQL se proporciona a continuación.

```
select * from employee where emp_name like '%gopal%'
```

En Yii2 escribiremos esto como

```
 $rows = $query->from('employee')
        ->where(['like', 'emp_name', 'gopal']) // Making use of operator format of where()
method
        ->all();
```

Si necesitamos encontrar a todos los empleados que tienen la cadena ' **gopal** ' y ' **nair** ' en su nombre. Podemos escribir como

```
   $rows = $query->from('employee')
        ->where(['like', 'emp_name', ['gopal','nair']]) // Making use of operator format of
where() method
        ->all();
```

Esto evaluaría como

seleccione * del empleado donde emp_name como '% gopal%' y '% nair%'

De manera similar, podemos usar " **no me gusta** " para indicar a todos los empleados que no tienen la cadena " **gopal** " y " **nair** " en sus nombres.

Usando orderBy ()

El método orderBy () especifica el fragmento ORDER BY de una consulta SQL. Por ejemplo, considere nuestra tabla de empleados con campos **emp_id, emp_first_name, emp_last_name y emp_salary.** Supongamos que debemos ordenar el resultado por orden creciente de salarios de los empleados. Podemos hacerlo sql como se indica a continuación.

```
Select * from employee order by emp_salary
```

En yii2, podemos construir la consulta como se indica a continuación.

```
 //creates a new \yii\db\Query() object
    $query=new \yii\db\Query();

    $rows= $query->from('employee')->orderBy([
    'emp_salary' => SORT_ASC //specify sort order ASC for ascending DESC for descending
    ])->all();
```

Si necesitamos ordenar a los empleados con su primer nombre en orden ascendente y luego sus salarios en orden descendente, podemos escribirlo en sql simple de la siguiente manera.

```
Select * from employee order by emp_first_name ASC, emp_salary DESC
```

El sql equivalente se puede construir usando yii2 de la siguiente manera

```
//creates a new \yii\db\Query() object
    $query=new \yii\db\Query();

    $rows= $query->from('employee')->orderBy([
    'emp_first_name' => SORT_ASC
    'emp_salary' => SORT_DESC
```

```
    ])->all();
```

También puede especificar ORDER BY utilizando una cadena, al igual que lo hace al escribir declaraciones de SQL sin formato. Por ejemplo, la consulta anterior también se puede generar como se indica a continuación.

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();
$rows=$query->from('employee')->orderBy('emp_first_name ASC, emp_salary DESC')->all();
```

Puede llamar a addOrderBy () para agregar columnas adicionales al fragmento ORDER BY. Por ejemplo

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();
$rows=$query->from('employee')->orderBy('emp_first_name ASC')
    ->addOrderBy('emp_salary DESC')->all();
```

Lea Trabajar con bases de datos en línea: https://riptutorial.com/es/yii2/topic/4167/trabajar-con-bases-de-datos

Capítulo 17: Validación

Examples

Validar el valor único de la base de datos en Yii2

Pocas personas tienen problemas con el mensaje de error que no se muestra si el valor existente se ingresa en el cuadro de texto.

Entonces, por ejemplo, no estoy *permitiendo que el* usuario ingrese el *correo electrónico existente* .

signup.php

(Página en la que desea registrar un nuevo usuario sin la ID de correo electrónico existente)

1. Eliminar `use yii\bootstrap\ActiveForm;` (si está presente)
2. Añadir `use yii\widgets\ActiveForm;`
3. Agregue `'enableAjaxValidation' => true` (en ese campo donde desea detener al usuario para ingresar la ID de correo electrónico existente).

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

<?= $form->field($modelUser, 'email',['enableAjaxValidation' => true])
    ->textInput(['class'=>'form-control','placeholder'=>'Email']); ?>
```

Controlador

Agregue estas líneas en la parte superior `use yii\web\Response;use yii\widgets\ActiveForm;`

```
<?php
use yii\web\Response;
use yii\widgets\ActiveForm;

  .
  .// Your code
  .

  public function actionSignup() {

    $modelUser = new User();

    //Add This For Ajax Email Exist Validation
    if(Yii::$app->request->isAjax && $modelUser->load(Yii::$app->request->post())){
      Yii::$app->response->format = Response::FORMAT_JSON;
      return ActiveForm::validate($modelUser);
    }
    else if ($model->load(Yii::$app->request->post())) {
```

```
    }

  }
?>
```

Modelo

```
[['email'],'unique','message'=>'Email already exist. Please try another one.'],
```

Validación del valor único de la base de datos: validación única

Algunas personas tienen problemas con respecto a los mensajes de error que no se muestran si se ingresa un valor existente. Por ejemplo, no estoy permitiendo que un usuario se registre con un correo electrónico existente.

Ver

```
<?php
.....................

    <?= $form->field($modelUser, 'email')->textInput(['class'=>'form-
control','placeholder'=>'Email']) ?>
.....................
```

Controlador

```
<?php
use yii\web\Response; // important lines
use yii\widgets\ActiveForm; // important lines

.
.// Your code
.

public function actionSignup()
{

    $modelUser = new User();

    //Add This For Ajax Validation
    if(Yii::$app->request->isAjax && $modelUser->load(Yii::$app->request->post())){
        Yii::$app->response->format = Response::FORMAT_JSON;
        return ActiveForm::validate($modelUser);
    }
    if ($modelUser->load(Yii::$app->request->post()) && $modelUser->save()) {
        return $this->redirect(['someplace nice']);
    }
    return $this->render('update', [
        'modelUser' => $modelUser,
    ]);
}
```

Modelo

```
public function rules()
{
    return [
        ..............
        ['email', 'unique', 'message'=>'Email already exist. Please try another one.'],
        ...........
    ]
}
```

Deshabilitar el mensaje de error de validación en el enfoque / Key Up

Por defecto, aparece un mensaje de error debajo del `textbox` de `textbox` en `<div class="help-block"></div>` en *keyUp* o *después de presionar el botón de enviar* si no se cumplen las restricciones de validación.

A veces queremos que solo se envíe un mensaje, es decir, que no haya validación en el evento `onKeyup` .

Revisemos el `yii2/widgets/ActiveForm.php` :

```
<?php

namespace yii\widgets;

use Yii;
use yii\base\InvalidCallException;
use yii\base\Widget;
use yii\base\Model;
use yii\helpers\ArrayHelper;
use yii\helpers\Url;
use yii\helpers\Html;
use yii\helpers\Json;

class ActiveForm extends Widget
{
  public $action = '';
  public $method = 'post';
  public $options = [];
  .
  .
  .
  public $validateOnSubmit = true;
  public $validateOnChange = true;
  public $validateOnBlur = true;
  public $validateOnType = false;

  .
  .
  .

}
```

Allí vemos que `$validateOnBlur` se establece en `true` de forma predeterminada. Cambiar los archivos del marco es algo muy malo, por lo que debemos anularlo al utilizar el formulario:

```
<?php $form = ActiveForm::begin([ 'id' => 'register-form','validateOnBlur' => false]); ?>
```

Escenario en Validación

Usando el escenario puede realizar validación en diferentes situaciones

Definir el escenario en la clase modelo.

```
    class User extends \yii\db\ActiveRecord
    {
      public static function tableName()
      {
          return 'user_master';
      }

    // define validation in rule() function
  public function rules()
  {
    return [
      [['email_id'],'email'],
      [['first_name',],'required','on'=>['create','update']],  // create scenario
      [['email_id',],'required','on'=> ['admin','create','update','forgotpassword']],
      [['mobile',],'required','on'=>['admin','create','update']],
    ];
  }
}
```

Usar escenario en el controlador

```
public function actionCreate()
{
  $model = new User();
  $model->scenario="create";  // use create scenario, create scenario validaion applied in
this model


}
public function actionUpdate()
{
    $model = new User();
    $model->scenario="update";  // use update scenario, update scenario validaion applied in
this model
}
```

Validar matriz

Desde la versión 2.0.4 de Yii2, se utiliza EachValidator para validar cada elemento de una matriz.

```
[
    // ... other rules
    ['userIDs', 'each', 'rule' => ['integer']],
]
```

La parte `['integer']` puede ser cualquier otro objeto validador que ofrece Yii2 y puede contener los argumentos específicos para el validador. Me gusta: `['integer', 'min' => 1337]` . Si los ID de usuario no contienen una matriz, la validación de la regla fallará.

Si solo desea ver si un atributo contiene una matriz sin validar el contenido, puede escribir su propio validador.

```
[
    ['myAttr', function($attribute, $params) {
        if (!is_array($this->$attribute)) {
            $this->addError($attribute, "$attribute isn't an array!");
        }
    }]
]
```

Lea Validación en línea: https://riptutorial.com/es/yii2/topic/839/validacion

Capítulo 18: Validaciones personalizadas

Introducción

Yii2 tiene algunos validadores incorporados que se pueden usar al resolver validaciones generales o relacionadas con la programación. Cuando necesita crear una nueva validación de lógica de negocios, necesita crear sus propios validadores.

Examples

Tipos de validaciones

Entendamos inicialmente los tipos básicos de validadores personalizados:

1. Validador en línea
2. Validador independiente

Validador en línea : es el tipo de validador que creamos dentro de la clase que es básicamente un método que definimos al igual que otros métodos pero con parámetros adicionales que se pasan por Yii2.

```
....
public function ValidateMyBusiness($attr, $params){
    // adding an error here means our validation is failed.
    if ($this->{$attr} > 1100) {
        $this->addError($attr, "Some error occured");
    }
}
...
// calling above validator is simple as below:
public function rules(){
  return [
     ['money', 'validateMyBusiness', 'params' => ['targetAccount' => $this->account]];
  ]
}

# params array will be passed to our inline parameter as a second argument.
```

Lea Validaciones personalizadas en línea: https://riptutorial.com/es/yii2/topic/9187/validaciones-personalizadas

Capítulo 19: Yii2 ActiveForm

Examples

Campos de formulario en Yii2

Muestra el ejemplo básico de la página de visualización en Yii2 para nuevos alumnos

Estas son clases básicas que debe agregar para crear un formulario usando yii2 ActiveForm

```
<?php

Use yii\helpers\Html;
Use yii\widgets\ActiveForm;
```

La línea de abajo comenzará la etiqueta de formulario para nuestro formulario de abajo que muestra un ejemplo que muestra cómo especificar id para el formulario y cómo aplicar cualquier clase para el formulario.

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

Aquí $ modelo Especifique qué campo de la tabla de base de datos queremos vincular con la forma en que se modela el objeto almacenado aquí en esta variable que se ha pasado desde el controlador relevante.

```
   <?= $form->field($model, 'username') ?>
   <?= $form->field($model, 'password')->passwordInput() ?>
```

'nombre de usuario' y 'contraseña' es el nombre del campo de la tabla con el que nuestro valor estará vinculado.

Aquí, en el siguiente código, estamos poniendo el botón Enviar para enviar el formulario y aplicamos 'Iniciar sesión' como texto de botón y clases básicas de CSS.

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

Aquí en el siguiente código estamos cerrando formulario

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

Crear campo de contraseña:

```
<?= $form->field($model, 'password')->passwordInput() ?>
```

Crear campo de texto:

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

Crear campo de formulario oculto:

```
echo $form->field($model, 'hidden1')->hiddenInput()->label(false);
```

Crear desplegable:

```
<?php echo $form->field($model, 'name')
->dropdownList(
Stud::find()->select(['name'])
->indexBy('name')->column(),
['prompt'=>'Select no']); ?>
```

Lista desplegable con identificación y nombre

```
<?= $form->field($model, 'name')->dropDownList(
            ArrayHelper::map(Stud::find()->all(), 'no', 'name'),['prompt' => 'Select Car
Name']
 ) ?>
```

Crear FileUploader:

```
echo $form->field($model, 'imagepath')->fileInput();
```

Agregar un marcador de posición y una etiqueta personalizada

```
 <?= $form->field($model, 'username')->textInput()->hint('Please enter your  name')-
>label('Name') ?>
```

Validaciones de ActiveForm

Puede habilitar / deshabilitar ajax y validaciones de clientes en forma activa.

```
$form = ActiveForm::begin([
    'id' => 'signup-form',
    'enableClientValidation' => true,
    'enableAjaxValidation' => true,
    'validationUrl' => Url::to('signup'),
]);
```

1. `enableClientValidation` está habilitado de forma predeterminada en ActiveForm. Si no necesita la validación del cliente en forma, podemos deshabilitarla asignándola como falsa.
2. `enableAjaxValidation` está deshabilitado de forma predeterminada en ActiveForm. Si desea habilitarlo, tenemos que agregarlo manualmente en ActiveForm como arriba.
3. `validationUrl`

: si desea mantener toda la codificación de validación en una acción de controlador separada para este formulario, podemos configurar el formulario activo utilizando `validationUrl` . Si no configuramos esto, tomará el valor de acción del formulario.

Los dos argumentos anteriores afectarán para toda la forma. Si desea verificar la validación de ajax solo para un campo en particular en el formulario, puede agregar `enableAjaxValidation` para ese campo en particular. Solo funcionará para ese campo no de forma completa.

Por ejemplo, en el formulario de registro, desea verificar que el nombre de usuario ya existe. La validación a la hora del usuario ingrese el formulario. puede utilizar este argumento `enableAjaxValidation` para ese campo.

```
echo $form->field($model, 'username', ['enableAjaxValidation' => true]);
```

Lea Yii2 ActiveForm en línea: https://riptutorial.com/es/yii2/topic/6807/yii2-activeform

Capítulo 20: Yii2 calendario de consultas para el campo de texto

Examples

Agregue calendario de jquery para un campo de texto con el máximo como fecha actual

Si queremos mostrar un calendario jquery para el usuario final que puede elegir la fecha máxima como la fecha actual en el calendario. El siguiente código será útil para este escenario.

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

<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....
<?= $form->field($model, 'date_of_birth')->widget(DatePicker::classname(), ['dateFormat' =>
'php:M d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'maxDate' => '+0d']]) ?>
.....
<?php ActiveForm::end(); ?>
```

Añadir calendario jquery para un campo de texto con fecha mínima

Para algunos formularios que desea visualizar, los días de los días futuros / pasados y otros días deben estar deshabilitados, entonces este escenario ayudará.

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

<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....

<?php
$day = '+5d'; //if you want to display +5 days from current date means for future days.
#(or)
$day = '-5d'; //if you want to display -5 days from current date means older days.
?>
<?= $form->field($model, 'date_of_birth')->widget(DatePicker::classname(), ['dateFormat' =>
'php:M d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'maxDate' => $day]]) ?>
.....
<?php ActiveForm::end(); ?>
```

Añadir calendario jquery con desde fecha y hasta la fecha

Si desea tener el calendario desde la fecha y hasta la fecha y también hasta la fecha, los días del calendario siempre serán mayores que desde el campo de la fecha, entonces el escenario siguiente ayudará.

```
<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....
<?= $form->field($model, 'from_date')->widget(DatePicker::classname(), ['dateFormat' => 'php:M
d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'onSelect' => new
yii\web\JsExpression('function(selected) { var dt = new Date(selected);
dt.setDate(dt.getDate() + 1); $("#filter-date-to").datepicker("option", "minDate", dt); }')]])
?>

<?= $form->field($model, 'to_date')->widget(DatePicker::classname(), ['dateFormat' => 'php:M
d, Y', 'options' => ['readonly' => true, 'id' => 'filter-date-to'], 'clientOptions' => [
'changeMonth' => true, 'changeYear' => true, 'yearRange' => '1980:'.date('Y')]]) ?>
.....
<?php ActiveForm::end(); ?>
```

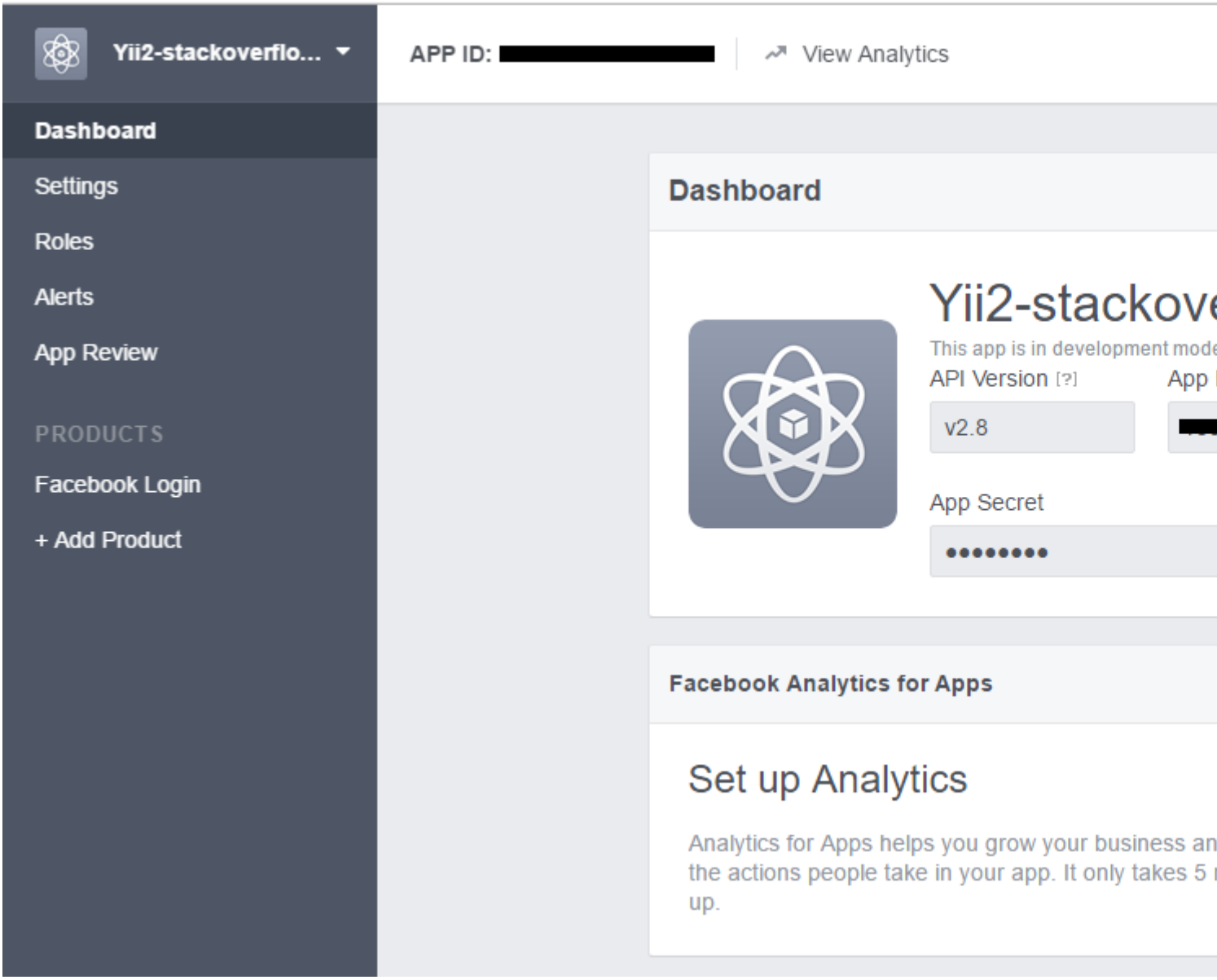
Lea Yii2 calendario de consultas para el campo de texto en línea:
https://riptutorial.com/es/yii2/topic/6366/yii2-calendario-de-consultas-para-el-campo-de-texto

Capítulo 21: Yii2 OAuth2 - Ej: consumidor facebook OAuth2

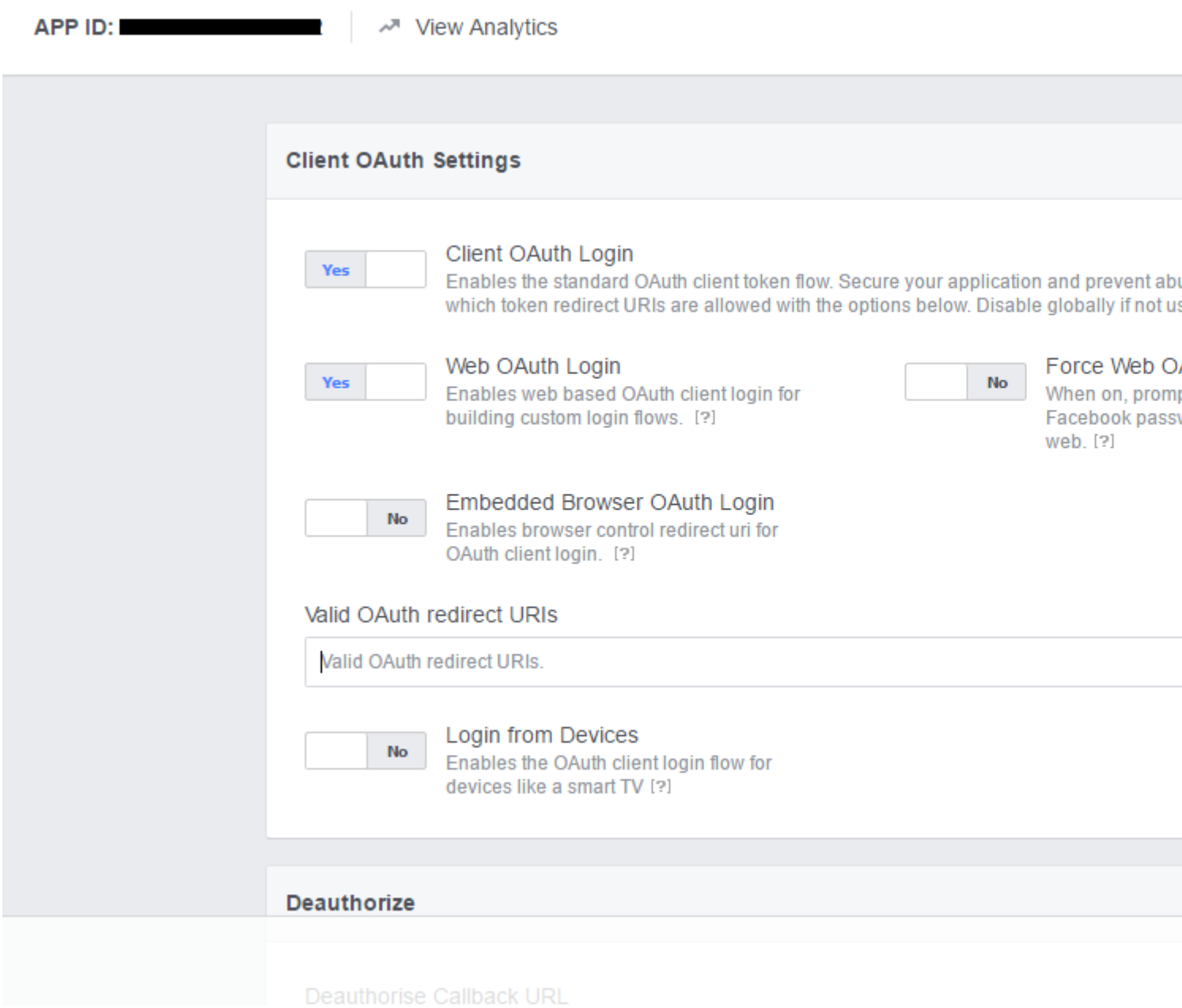
Examples

Crear una aplicación en el desarrollador de facebook

Vaya a la [https://developers.facebook.com/◆(https://developers.facebook.com/) y cree su aplicación.



Haga clic en `Add product` y elija `Facebook Login`



Instalar yii2-authclient

Antes de instalar esta extensión, debes instalar la aplicación yii2. En este ejemplo, uso la plantilla yii2-basic. Guía para la instalación aquí .

correr

```
composer require --prefer-dist yiisoft/yii2-authclient
```

o agregar

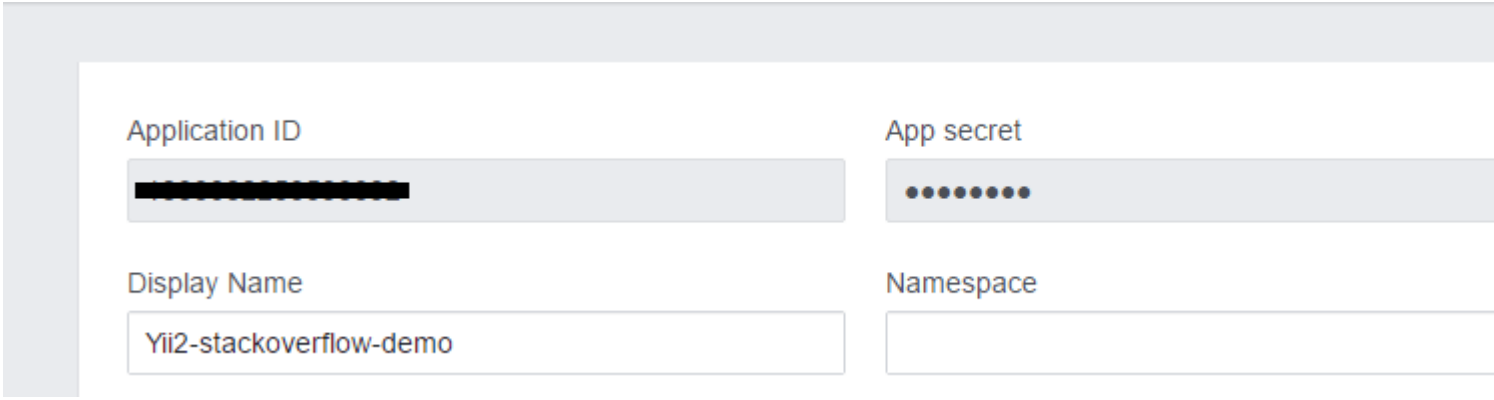
```
"yiisoft/yii2-authclient": "~2.1.0"
```

a la sección de `require` de su `composer.json` .

Agregue config `authClientCollection` a sus `components` configuración:

```
return [
    'components' => [
        'authClientCollection' => [
            'class' => 'yii\authclient\Collection',
            'clients' => [
                'facebook' => [
                    'class' => 'yii\authclient\clients\Facebook',
                    'clientId' => 'facebook_client_id',
                    'clientSecret' => 'facebook_client_secret',
                ],
            ],
        ]
    ],
    // ...
];
```

`facebook_client_id` es id de aplicación y `facebook_client_secret` es secreto de aplicación.



Añadir acción de autenticación y configurar devolución de llamada

1. Agregar botón `Login as facebook account` en su vista de inicio de sesión:

Edite `site/login.php` en la carpeta de vistas, agregue esta línea al contenido del inicio de sesión de la página:

```
<?= yii\authclient\widgets\AuthChoice::widget([
     'baseAuthUrl' => ['site/auth'],
     'popupMode' => false,
]) ?>
```

Anteriormente, establecemos que la acción de `auth` en `SiteController` manejará el flujo de OAuth2.

Ahora lo creamos.

```
class SiteController extends Controller
{
    public function actions()
    {
        return [
            'auth' => [
                'class' => 'yii\authclient\AuthAction',
```

```
                'successCallback' => [$this, 'onAuthSuccess'],
            ],
        ];
    }

    public function onAuthSuccess($client)
    {
        // do many stuff here, save user info to your app database
    }
}
```

Usamos `yii\authclient\AuthAction` para crear url y redirigir a la página de inicio de sesión de Facebook.

La función `onAuthSuccess` utilizada para obtener información del usuario, inicie sesión en su aplicación.

Añadir redirect_url a la configuración de la aplicación de Facebook

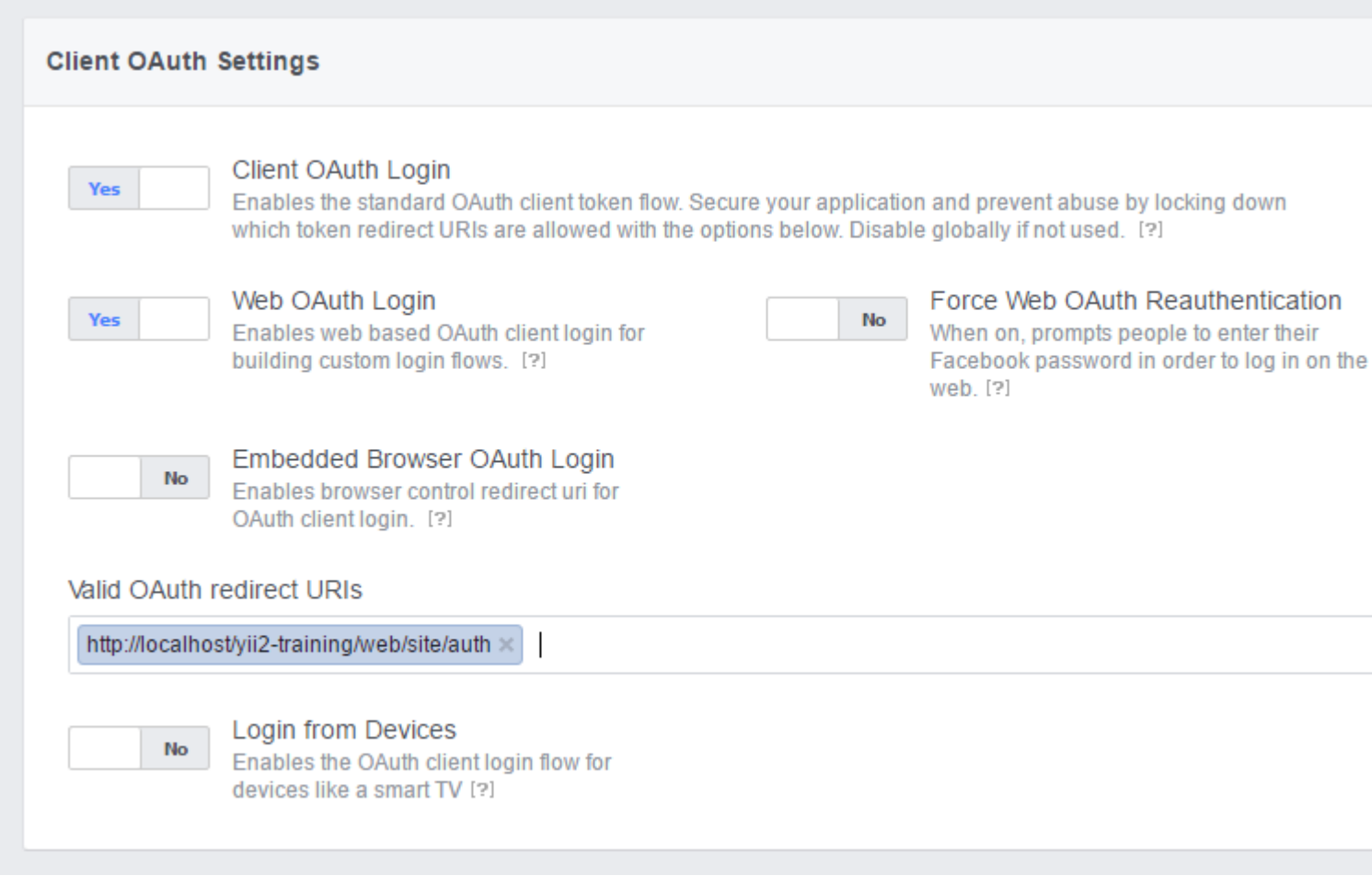
Si habilitas prettyUrl en tu aplicación yii2, tu redirect_uri será:

```
http://<base_url>/web/site/auth
```

Y deshabilitar URL bonita:

```
http://<base_url>/web/index.php?r=site%2Fauth
```

Ejemplo:



Ejemplo para la función onAuthSuccess

```
/**
 * @param $client ClientInterface
 */
public function onAuthSuccess($client)
{
    //Get user info
    /** @var array $attributes */
    $attributes = $client->getUserAttributes();
    $email = ArrayHelper::getValue($attributes, 'email'); //email info
    $id = ArrayHelper::getValue($attributes, 'id'); // id facebook user
    $name = ArrayHelper::getValue($attributes, 'name'); // name facebook account

    //Login user
    //For demo, I will login with admin/admin default account
    $admin = User::findByUsername('admin');
    Yii::$app->user->login($admin);
}
```

Lea Yii2 OAuth2 - Ej: consumidor facebook OAuth2 en línea:
https://riptutorial.com/es/yii2/topic/7428/yii2-oauth2---ej--consumidor-facebook-oauth2

Creditos

S. No	Capítulos	Contributors
1	Empezando con yii2	Bibek Lekhak, Community, Farcaller, jagsler, Mohan Rex, Muhammad Shahzad, Pasha Rumkin, Sam Dark, urmaul, Yasar Arafath, Yasin Patel, Yatin Mistry
2	API de descanso	jagsler, jlapoutre, yafater, Yasin Patel
3	Cargas de archivos	Sam Dark
4	Componentes	Sam Dark, Yasin Patel
5	Enrutamiento y URLs	IStranger, Ярослав Гойса
6	Galletas	IStranger, Sam Dark
7	Gestión de activos	Thomas Rohde
8	Instalación manual de la extensión	Insane Skull, mnoronha, Sam Dark, vishuB
9	Migraciones de base de datos	Ali MasudianPour, jlapoutre, Sam Dark
10	Pjax	gmc, Manikandan S, Muaaz Rafi, Shaig Khaligli, yafater, Yasin Patel
11	Plantilla de proyecto avanzado	mnoronha, Mohan Rex, Salem Ouerdani, Sam Dark, topher
12	Pruebas	Antonín Slejška, Bizley, Sam Dark, XzAeRo
13	Registro activo	Insane Skull, Manikandan S, Michael St Clair, Mike Artemiev, particleflux, saada, Sam Dark, Yasar Arafath, Yasin Patel
14	Sesión	Brett, Goke Obasa, jlapoutre, MikelG, Yasin Patel
15	Solicitud de Ajax	Anton Rybalko, Ilyas karim, meysam
16	Trabajar con bases de datos	jagsler, Kiran Muralee
17	Validación	jagsler, Mihai P., Nana Partykar, Sam Dark, Yasin Patel

18	Validaciones personalizadas	Ejaz Karim
19	Yii2 ActiveForm	Hina Vaja, Manikandan S, particleflux
20	Yii2 calendario de consultas para el campo de texto	Manikandan S
21	Yii2 OAuth2 - Ej: consumidor facebook OAuth2	ThanhPV