



EBook Gratuito

APPENDIMENTO

yii2

Free unaffiliated eBook created from
Stack Overflow contributors.

#yii2

Sommario

Di.....	1
Capitolo 1: Iniziare con yii2.....	2
Osservazioni.....	2
Versioni.....	2
Examples.....	2
Installazione o configurazione.....	2
Installazione tramite Composer.....	3
Installazione di Composer.....	3
Installazione di Yii.....	3
Installazione da un file di archivio.....	3
Installa Yii2 avanzato in ubuntu.....	4
Capitolo 2: analisi.....	7
Examples.....	7
Configurare l'ambiente di test.....	7
Come deridere ActiveRecord.....	8
Capitolo 3: API riposante.....	9
Examples.....	9
Inizia con il riposo api.....	9
Come sovrascrivere le azioni predefinite di rest api Yii2.....	10
Sovrascrivi Content-Type per un'azione specifica.....	11
Capitolo 4: Biscotti.....	12
Osservazioni.....	12
Examples.....	12
Impostazione di un cookie.....	12
Leggere un cookie.....	12
Cookie per sottodomini.....	13
Autenticazione cross-subdominio e cookie di identità.....	13
Parametri del cookie di sessione.....	14
Capitolo 5: componenti.....	15
Examples.....	15

Creazione e utilizzo dei componenti dell'applicazione	15
Elenco a discesa utilizzando la funzione componente	15
Capitolo 6: Convalide personalizzate	17
introduzione	17
Examples	17
Tipi di convalida	17
Capitolo 7: Gestione delle risorse	18
Sintassi	18
Osservazioni	18
Examples	18
Questo fa parte del file di layout	18
Questo è il file di asset	19
L'HTML generato con le risorse caricate automaticamente	20
Capitolo 8: Installazione manuale dell'estensione	21
Examples	21
Installa estensione senza Composer	21
Capitolo 9: Lavorare con i database	22
Examples	22
Utilizzando il generatore di query Yii2	22
Più controllo delle condizioni usando where ()	23
Usare orderBy ()	25
Capitolo 10: Migrazioni del database	27
Examples	27
Creazione di migrazioni	27
Esempio di file di migrazione	27
Drop Table	27
Crea campi tabella immediatamente	28
Crea tabella	28
Rilascia / Rinomina / Modifica colonna	28
Aggiungi colonna	29
Ripristino delle migrazioni	29
Migrazioni transazionali	29

Migrazione di più database.....	29
Ripristino delle migrazioni.....	29
Elenco delle migrazioni.....	30
Modifica della cronologia delle migrazioni.....	30
Applicazione delle migrazioni.....	30
Capitolo 11: Modello di progetto avanzato.....	31
Examples.....	31
Distribuzione in ambiente di hosting condiviso.....	31
Sposta gli script di accesso in un singolo webroot.....	31
Regola le sessioni e i cookie.....	31
Condivisione di file caricati tra frontend e back-end tramite symlink.....	32
Capitolo 12: Pjax.....	33
Examples.....	33
Passaggio 1 Aggiungi struttura.....	33
Passaggio 2 Codice lato server.....	33
come usare pjax.....	33
ricarica pjax.....	33
utilizzare l'argomento di timeout in pjax.....	34
Utilizzo avanzato Pjax.....	34
Capitolo 13: Record attivo.....	36
Osservazioni.....	36
Examples.....	36
Trova tutti i record.....	36
Dove la clausola.....	37
Crea una classe ActiveRecord con valore dei campi basati sugli eventi.....	38
Trova un record.....	39
Trova una query.....	40
Record attivi con sottocategorie.....	41
Capitolo 14: Richiesta Ajax.....	42
Examples.....	42
Invio del modulo Ajax.....	42
Renderizza la vista Ajax.....	43

Capitolo 15: Routing e URL	46
Osservazioni	46
Examples	46
Creazione di URL	46
Capitolo 16: Sessione	48
Examples	48
Sessione in yii2	48
import Session Class	48
Crea una sessione	48
Memorizza il valore nella variabile di sessione	48
Ottieni il valore dalla variabile di sessione	48
Rimuovi la variabile di sessione	48
Rimuovi tutte le variabili di sessione	48
Controlla la variabile Session	49
Flash di sessione	49
Utilizza direttamente la variabile di sessione	49
Creazione e modifica delle variabili di sessione che sono matrici	49
Ricorda l'URL da rivedere più tardi	50
Capitolo 17: Upload di file	51
Examples	51
Come farlo	51
Caricamento di file	51
Creazione di modelli	51
Input del file di rendering	52
Cablaggio	52
Caricamento di più file	53
Capitolo 18: Validazione	55
Examples	55
Convalida un valore univoco dal database in Yii2	55
Convalida del valore univoco dal database: convalida univoca	56
Disabilita messaggio di errore di convalida su Focus / Key Up	57

Scenario in convalida.....	58
Convalida matrice.....	58
Capitolo 19: Yii2 ActiveForm.....	60
Examples.....	60
Campi modulo in Yii2.....	60
Validazioni ActiveForm.....	61
Capitolo 20: Yii2 JQuery Calendar per il campo di testo.....	63
Examples.....	63
Aggiungi il calendario jquery per un campo di testo con il massimo della data corrente.....	63
Aggiungi il calendario jquery per un campo di testo con data minima.....	63
Aggiungi calendario jquery con dalla data e alla data.....	63
Capitolo 21: Yii2 OAuth2 - Es: consumer facebook OAuth2.....	65
Examples.....	65
Crea un'app sullo sviluppatore di Facebook.....	65
Installa yii2-authclient.....	66
Aggiungi azione auth e imposta callback.....	67
Aggiungi redirect_url alle impostazioni dell'app Facebook.....	68
Esempio per la funzione onAuthSuccess.....	69
Titoli di coda.....	70

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capitolo 1: Iniziare con yii2

Osservazioni

Yii è un framework di programmazione Web generico, il che significa che può essere utilizzato per sviluppare tutti i tipi di applicazioni Web utilizzando PHP. Grazie alla sua architettura basata su componenti e al sofisticato supporto di memorizzazione nella cache, è particolarmente adatto per lo sviluppo di applicazioni su larga scala quali portali, forum, sistemi di gestione dei contenuti (CMS), progetti di e-commerce, servizi Web RESTful e così via.

Versioni

Versione	Data di rilascio
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

Installazione o configurazione

Yii2 può essere installato in due modi. Loro sono

1. Installazione tramite Composer
2. Installazione da un file di archivio

Installazione tramite Composer

Installazione di Composer

Se non hai ancora installato Composer, puoi farlo seguendo le istruzioni su getcomposer.org . Su Linux e Mac OS X, eseguirai i seguenti comandi:

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

Per Windows è sufficiente scaricare e installare [composer-setup.exe](#) Potrebbe essere necessario configurare il token di accesso dell'API github al di sopra del limite di frequenza dell'API Github.

Installazione di Yii

Con Composer installato, è possibile installare Yii eseguendo i seguenti comandi in una cartella accessibile tramite Web:

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

quindi eseguire il seguente comando per installare Yii2 con template di base.

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

Per installare Yii2 con modello avanzato eseguito

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

Successivamente creare un nuovo database e regolare la configurazione dei componenti ['db'] in common / config / main-local.php di conseguenza. quindi eseguire il seguente comando per

```
php yii migrate
```

Installazione da un file di archivio

1. Scarica il file di archivio da [Yii-download](#)
2. Scompattare il file scaricato in una cartella accessibile dal Web.

3. Modifica il file config / web.php inserendo una chiave segreta per l'elemento di configurazione cookieValidationKey

Puoi aggiungere qualsiasi tipo di chiave che desideri:

```
'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',
```

Installa Yii2 avanzato in ubuntu

Per prima cosa dobbiamo installare il compositore. Passi per installare il compositore Installa compositore.

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

Ora cambia directory:

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

Controlla il compositore che lavora

```
composer
```

Ora è installato Composer.

Ci sono due modi per installare Yii2 advance.

1.Installazione da un file di archivio

Ottieni il file zip dal link sottostante.

Decomprimilo nella directory di destinazione, ad esempio /var/www/html .

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

Spostati all'interno della cartella "avanzata". Muovere manualmente o digitare sotto il comando.

```
cd advanced
```

Esegui sotto il comando.

```
php init
```

2. Installazione tramite Composer

L'installazione tramite il compositore richiede un token di autenticazione github. Per il token devi registrarti su GitHub.

Dopo l'iscrizione puoi generare il tuo token:

Passi per generare un token

1. Nell'angolo in alto a destra di qualsiasi pagina, fai clic sulla foto del profilo, quindi fai clic su Impostazioni.
2. Nella barra laterale delle impostazioni utente, fai clic su Token di accesso personali.
3. Fai clic su Genera nuovo token.
4. Dai al tuo token un nome descrittivo.
5. Seleziona gli ambiti che desideri concedere a questo token.
6. Fai clic su Genera token.
7. Copia il token negli Appunti. Per motivi di sicurezza, dopo aver superato questa pagina, nessuno potrà più visualizzare il token.

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

Dopo aver generato un token, copialo

Cambia directory

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

Esegui sotto il comando

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

esempio:

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

Installa yii2

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

Spostati all'interno della cartella "avanzata". Muovere manualmente o digitare sotto il comando.

```
cd advanced
```

Esegui sotto il comando.

```
php init
```

E 'fatto!

Ora puoi controllarlo.

<http://localhost/avanzato/frontend/web>

e

<http://localhost/avanzato/backend/web>

Leggi Iniziare con yii2 online: <https://riptutorial.com/it/yii2/topic/788/iniziare-con-yii2>

Capitolo 2: analisi

Examples

Configurare l'ambiente di test

Installa Codeception:

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

Installa Faker:

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

Creare un database, che verrà utilizzato solo per i test. È possibile duplicare il database esistente o applicare le migrazioni:

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

Regola la configurazione dei `components['db']` nei `tests/codeception/config/config-local.php`.

Aggiungi la directory `/var/www/yii/vendor/bin` al tuo percorso.

Esamina tutti i file di configurazione e `.yml`.

Avvia il server web, ad esempio:

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

Esegui i test:

```
codecept run
```

Maggiori informazioni:

- <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: queste istruzioni sono valide per la versione 2.0.9 di Yii2. Nella versione 2.0.10 sarà secondo Sam Dark la parte di test rielaborata (e le istruzioni devono essere aggiornate). La versione 2.0.10 dovrebbe essere rilasciata l'11 settembre 2016:

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

Come deridere ActiveRecord

Se vuoi prendere in giro AR che non prova a connettersi al database puoi farlo nel modo seguente (se usi 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'
]);
```

Il problema è che dobbiamo sovrascrivere il metodo attributes () poiché ActiveRecord di default sta ottenendo l'elenco degli attributi dallo schema del database che stiamo cercando di evitare.

Leggi analisi online: <https://riptutorial.com/it/yii2/topic/2226/analisi>

Capitolo 3: API riposante

Examples

Inizia con il riposo api

Abbiamo una tabella che include paesi, quindi creiamo un modello chiamato modello di countrylist

```
<?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 $nickname
 * @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', 'nickname', 'phonecode'], 'required'],
            [['numcode', 'phonecode'], 'integer'],
            [['iso'], 'string', 'max' => 2],
            [['name', 'nickname'], 'string', 'max' => 80],
            [['iso3'], 'string', 'max' => 3]
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
        ];
    }
}
```

```

        'iso' => 'Iso',
        'name' => 'Name',
        'nickname' => 'Nickname',
        'iso3' => 'Iso3',
        'numcode' => 'Numcode',
        'phonecode' => 'Phonecode',
    ];
}
}

```

e creo il webservice di riposo per questo, creiamo un controller per restapi e impostiamo la variabile `modelClass` per il nostro modello.

```

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

```

per l'utilizzo di restapi abbiamo bisogno di url piuttosto e aggiungiamo questa regola per un bel URL

```

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

```

dopo di che abbiamo accesso a testare la nostra pausa API come esempio

<http://localhost/countrylist> ci fornisce l'elenco delle contee.

Come sovrascrivere le azioni predefinite di rest api Yii2

Ad esempio, si desidera disabilitare l'impaginazione nell'azione indice predefinita e ottenere tutti i risultati in indice. Come puoi farlo? È semplice. Devi eseguire l'override dell'azione indice nel tuo controller in questo modo:

```

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;
}
```

Sovrascrivi Content-Type per un'azione specifica

Caso d'uso: una sola azione che dovrebbe restituire un semplice contenuto (di testo) così com'è:

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

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

I formati di risposta predefiniti sono:

- FORMAT_HTML
- FORMAT_XML
- FORMAT_JSON
- FORMAT_JSONP
- FORMAT_RAW

Non esiste un tipo mime per `text/plain` out of the box, usa invece:

```
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
}
```

Leggi API riposante online: <https://riptutorial.com/it/yii2/topic/6102/api-riposante>

Capitolo 4: Biscotti

Osservazioni

I cookie fanno parte della richiesta HTTP, quindi è una buona idea fare entrambe le cose nel controller, la cui responsabilità è esattamente quella che riguarda la richiesta e la risposta.

Examples

Impostazione di un cookie

Per impostare un cookie, ad esempio per crearlo e pianificare l'invio al browser, è necessario creare una nuova istanza di classe `\yii\web\Cookie` e aggiungerla alla raccolta dei cookie di risposta:

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

In quanto sopra passiamo i parametri al costruttore della classe cookie. Questi sono praticamente gli stessi usati con la funzione [setcookie PHP nativa](#) :

- `name` - nome del cookie.
- `value` - valore del cookie. Assicurati che sia una stringa. I browser non sono generalmente contenti dei dati binari nei cookie.
- `domain` - dominio per il quale stai impostando il cookie.
- `expire` - unix timestamp che indica l'ora in cui il cookie deve essere automaticamente cancellato.
- `path` : il percorso sul server in cui sarà disponibile il cookie.
- `secure` - se `true` , il cookie verrà impostato solo se si utilizza HTTPS.
- `httpOnly` : se è `true` , il cookie non sarà disponibile tramite JavaScript.

Leggere un cookie

Per leggere un cookie utilizzare il seguente codice:

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

Nota: questo codice consente di leggere i cookie che sono stati impostati utilizzando il componente cookie (poiché per impostazione predefinita firma tutti i cookie). Pertanto se aggiungi / aggiorni cookie usando il codice JS, non puoi leggerlo usando questo metodo (di default).

Cookie per sottodomini

Per motivi di sicurezza, i cookie di default sono accessibili solo sullo stesso dominio da cui sono stati impostati. Ad esempio, se hai impostato un cookie su domain `example.com`, non puoi ottenerlo sul dominio `www.example.com`. Pertanto, se hai intenzione di utilizzare sottodomini (ad esempio `admin.example.com`, `profile.example.com`), devi impostare il `domain` esplicito:

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

Ora i cookie possono essere letti da tutti i sottodomini di `example.com`.

Autenticazione cross-sottodominio e cookie di identità

In caso di autologin o cookie "ricordami", si applicano le stesse stranezze come nel caso dei cookie di sottodominio. Ma questa volta è necessario configurare il componente utente, impostando l'array `identityCookie` sulla configurazione dei cookie desiderata.

Apri il file di configurazione dell'applicazione e aggiungi i parametri `identityCookie` alla configurazione del componente utente:

```
$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,
        ],
    ],
],
];
```

Nota che `cookieValidationKey` dovrebbe essere lo stesso per tutti i sottodomini.

Nota che devi configurare la proprietà `session::cookieParams` per avere il `samedomain` come `user::identityCookie` per garantire il `login` e il `logout` per tutti i sottodomini. Questo comportamento è spiegato meglio nella prossima sezione.

Parametri del cookie di sessione

I parametri dei cookie di sessione sono importanti sia se si ha la necessità di mantenere la sessione mentre si passa da un sottodominio a un altro o quando, al contrario, si ospita l'app di back-end sotto `/admin` URL e si desidera gestire separatamente la sessione.

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

Leggi Biscotti online: <https://riptutorial.com/it/yii2/topic/2945/biscotti>

Capitolo 5: componenti

Examples

Creazione e utilizzo dei componenti dell'applicazione

Passaggi per creare un componente:

- Creare una cartella denominata `components` nella cartella principale del progetto
- Crea il tuo componente all'interno della cartella dei componenti es: `MyComponent.php`

```
namespace app\components;

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

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

- Registra il tuo componente all'interno del file `config/web.php`

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

Ora puoi usare il tuo metodo componente:

```
namespace app\controllers;

use Yii;

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

Elenco a discesa utilizzando la funzione componente

Crea una funzione in `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"];
    }
}

```

-> Registra componente in web.php

```

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

```

-> usalo nella tua vista

```

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

```

Leggi componenti online: <https://riptutorial.com/it/yii2/topic/2217/componenti>

Capitolo 6: Convalide personalizzate

introduzione

Yii2 ha alcuni validatori incorporati che possono essere utilizzati durante la risoluzione di convalide di programmazione generale o generale. Quando è necessario creare una nuova convalida della logica aziendale, è necessario creare i propri validatori.

Examples

Tipi di convalida

Iniziamo a capire i tipi base di validatori personalizzati:

1. In linea Validator
2. Validatore standalone

Inline Validator : è il tipo di validatore che creiamo all'interno della classe, che è fondamentalmente un metodo che definiamo proprio come altri metodi, ma con parametri aggiuntivi che vengono passati da Yii2.

```
....
public function ValidateMyBusiness($attr, $params){
    // adding an error here means our validation is failed.
    if ($this->{$attr} > 1100) {
        $this->addError($attr, "Some error occurred");
    }
}
...
// 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.
```

Leggi Convalide personalizzate online: <https://riptutorial.com/it/yii2/topic/9187/convalide-personalizzate>

Capitolo 7: Gestione delle risorse

Sintassi

- le attività dipese verranno caricate prima di queste risorse in un determinato ordine
 - `public $ depends = ['yii \ web \ YiiAsset', 'yii \ bootstrap \ BootstrapAsset', 'yii \ bootstrap \ BootstrapPluginAsset', 'cinghie \ fontawesome \ FontAwesomeAsset'];`

Osservazioni

questo esempio si basa sul modello avanzato <https://github.com/yiisoft/yii2-app-advanced>

L'asset cinghie in questo esempio è il pacchetto di asset per adminLTE

<https://github.com/cinghie/yii2-admin-lte>

Examples

Questo fa parte del file di layout

```
<?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() ?>
```

Questo è il file di asset

```
<?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 = [
        'icons/css/ionsicons.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',  
];  
  
}
```

L'HTML generato con le risorse caricate automaticamente

```
<!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>
```

Leggi Gestione delle risorse online: <https://riptutorial.com/it/yii2/topic/6900/gestione-delle-risorse>

Capitolo 8: Installazione manuale dell'estensione

Examples

Installa estensione senza Composer

Nota: si consiglia vivamente di utilizzare Composer. Le istruzioni che seguono sono in pratica ciò che il compositore fa per te.

- Scarica il file di estensione dell'archivio della versione richiesta da Github
- Apri `composer.json`
- Trova la sezione di autoload PSR-4 e ricordala per es `kmit/select2`
- Estrai i file nella cartella corrispondente nella cartella del venditore, come `vendor/kmit/select2`
- Aggiungi il seguente codice al `vendor/composer/autoload_psr4.php`

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

- Aggiungi il seguente codice a `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 Tutorial

Leggi Installazione manuale dell'estensione online:

<https://riptutorial.com/it/yii2/topic/2224/installazione-manuale-dell-estensione>

Capitolo 9: Lavorare con i database

Examples

Utilizzando il generatore di query Yii2

Yii2 fornisce metodi efficienti per recuperare i dati dal database. **Considerare** un esempio di una semplice tabella dei dipendenti con campi **emp_id, emp_name e emp_salary** . Per recuperare i nomi dei dipendenti e i loro stipendi, utilizziamo la query.

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

Per generare la query sopra in Yii2, ci sono molti metodi. Uno dei metodi è usare un oggetto **yii\db\Query** .

```
//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.
```

Possiamo fare uso di un ciclo foreach per eseguire il ciclo di ogni coppia nome-valore nell'array **\$rows** .

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

Questo uscirà

Nome del dipendente: Kiran, Stipendio dei dipendenti: 25000

Nome impiegato: Midhun, Stipendio dipendente: 50000

Nome del dipendente: Jishnu, Stipendio dei dipendenti: 20000

Nome del dipendente: Ajith, Stipendio dei dipendenti: 25000

Nome del dipendente: Akshay, Stipendio dei dipendenti: 750000

Altri esempi

Supponiamo di dover trovare il nome dei dipendenti il cui stipendio è uguale a 25000. Possiamo scrivere la query in sql come

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

In Yii2, il codice per generare la query sopra

```
$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();
```

Se abbiamo bisogno di trovare nomi di dipendenti il cui stipendio sia maggiore di 25000, possiamo scrivere il codice in Yii2 come

```
$rows=$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();
```

Più controllo delle condizioni usando where ()

È possibile scrivere più condizioni utilizzando il metodo **where ()** come indicato di seguito.

```
// 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
```

Il codice sopra richiamerà un impiegato che ha il nome **kiran** e lo stipendio **25000** . Se più dipendenti soddisfano la condizione di cui sopra, la chiamata **uno ()** si assicura che venga recuperato solo il primo risultato. Per recuperare tutti i risultati dovresti usare **all ()** .

Nota che se usi **tutto ()** il risultato sarà sempre un array; Anche se ci sono solo uno o zero risultati. Questo array contiene tutti i risultati come array o è vuoto quando nessun record corrisponde. La chiamata **one ()** restituirà l'array risultante direttamente o false se la query non restituisce nulla.

Di seguito è riportato il codice equivalente in sql.

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

Di seguito è riportato un modo alternativo di scrivere la query sopra riportata in Yii2.

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

È possibile specificare ulteriori set di condizioni utilizzando e **dove** . Questo sarà utile se avremo bisogno di aggiungere ulteriori controlli di condizione alla query più tardi.

Un altro modo per specificare più condizioni consiste nell'utilizzare il **formato operatore del metodo where ()**. La query precedente può anche essere scritta come indicato di seguito.

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

Qui specifichiamo l'operatore ' e ' come il primo elemento dell'array. Allo stesso modo possiamo anche usare " o ", " tra ", " non tra ", " in ", " non in ", " come ", " o come ", " non come ", " o non come ", " esiste ", ' non esiste ', '>', '<=' ecc come operatori.

Esempi di utilizzo di "in" e "mi piace"

Supponiamo di dover trovare gli impiegati con stipendi **20000, 25000 e 50000** . In sql normale scriveremo la query come

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

In Yii2 possiamo scrivere come indicato di seguito.

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

Un altro modo per specificare la stessa condizione è

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

Allo stesso modo ' **non in** ' può essere specificato invece di ' **in** ' se vogliamo che tutti i dipendenti non abbiano stipendi 20000, 25000 e 50000.

Ora vediamo alcuni esempi di utilizzo di "mi **piace** " all'interno della condizione where ().

Supponiamo di dover trovare tutti i dipendenti che hanno la stringa " **gopal** " nel loro nome. I nomi possono essere venugopal, rajagopal, gopalakrishnan ecc. La query sql è riportata di seguito.

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

In Yii2 scriveremo questo come

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

Se abbiamo bisogno di trovare tutti i dipendenti che hanno la stringa " **gopal** " e " **nair** " nel loro nome. Possiamo scrivere come

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

Questo valuterà come

seleziona * dal dipendente dove emp_name come '% gopal%' e '% nair%'

Allo stesso modo possiamo usare ' **non come** ' per indicare che tutti i dipendenti non hanno la stringa ' **gopal** ' e ' **nair** ' nei loro nomi.

Usare orderBy ()

Il metodo orderBy () specifica il frammento ORDER BY di una query SQL. Ad esempio, considera la nostra tabella dei dipendenti con campi **emp_id**, **emp_first_name**, **emp_last_name** e **emp_salary**. Supponiamo di dover ordinare il risultato aumentando l'ordine degli stipendi dei dipendenti. Possiamo farlo in sql come indicato di seguito.

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

In yii2, possiamo costruire la query come indicato di seguito

```
//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();
```

Se abbiamo bisogno di ordinare i dipendenti con il loro nome in ordine ascendente e poi i loro salari in ordine decrescente, possiamo scriverlo in plain sql come segue.

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

L'equivalente sql può essere compilato usando yii2 come segue

```
//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();
```

È inoltre possibile specificare ORDER BY utilizzando una stringa, proprio come quando si scrivono istruzioni SQL non elaborate. Ad esempio, la query precedente può anche essere

generata come indicato di seguito.

```
//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();
```

Puoi chiamare `addOrderBy ()` per aggiungere ulteriori colonne al frammento ORDER BY. Per esempio

```
//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();
```

Leggi **Lavorare con i database online**: <https://riptutorial.com/it/yii2/topic/4167/lavorare-con-i-database>

Capitolo 10: Migrazioni del database

Examples

Creazione di migrazioni

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

L'argomento del nome richiesto fornisce una breve descrizione della nuova migrazione. Ad esempio, se la migrazione riguarda la creazione di una nuova tabella denominata news, è possibile utilizzare il nome `create_news_table` ed eseguire il seguente comando

```
yii migrate/create create_news_table
```

Esempio di file di migrazione

```
<?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()
    {

    }
    */
}
```

Drop Table

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

```
}
```

Crea campi tabella immediatamente

```
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');
    }
}
```

Crea tabella

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

Rilascia / Rinomina / Modifica colonna

```
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'));
}
```

Aggiungi colonna

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

Ripristino delle migrazioni

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

Migrazioni transazionali

```
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');
}
```

Un modo ancora più semplice di implementare le migrazioni transazionali è inserire il codice di migrazione nei `safeUp()` e `safeDown()`. Questi due metodi differiscono da `up()` e `down()` in quanto sono racchiusi implicitamente in una transazione. Di conseguenza, se qualsiasi operazione in questi metodi fallisce, tutte le operazioni precedenti verranno automaticamente ripristinate.

Migrazione di più database

Per impostazione predefinita, le migrazioni vengono applicate allo stesso database specificato dal componente dell'applicazione `db`. Se si desidera che vengano applicati a un altro database, è possibile specificare l'opzione della riga di comando `db` come mostrato di seguito:

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

Ripristino delle migrazioni

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

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

Elenco delle migrazioni

```
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
```

Modifica della cronologia delle migrazioni

```
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
```

Applicazione delle migrazioni

```
yii migrate
```

Questo comando elencherà tutte le migrazioni che non sono state applicate finora. Se confermi di voler applicare queste migrazioni, eseguirà il metodo `up()` o `safeUp()` in ogni nuova classe di migrazione, una dopo l'altra, nell'ordine dei loro valori di timestamp. Se una qualsiasi delle migrazioni fallisce, il comando si chiude senza applicare il resto delle migrazioni.

```
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
```

Leggi Migrazioni del database online: <https://riptutorial.com/it/yii2/topic/1929/migrazioni-del-database>

Capitolo 11: Modello di progetto avanzato

Examples

Distribuzione in ambiente di hosting condiviso

La distribuzione di un modello di progetto avanzato sull'hosting condiviso è un po' più complicata rispetto a quella di base perché ha due web-root, che i server Web di hosting condiviso non supportano. Dovremo modificare la struttura delle directory in modo che l'URL di frontend sia <http://site.local> e l'URL di backend sia <http://site.local/admin>.

Sposta gli script di accesso in un singolo webroot

Prima di tutto abbiamo bisogno di una directory webroot. Creare una nuova directory e denominarla in modo che corrisponda al nome del webroot di hosting, ad esempio `www` o `public_html` o simile. Quindi crea la seguente struttura dove `www` è la directory webroot di hosting che hai appena creato:

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

www sarà la nostra directory di frontend in modo da spostare il contenuto di **frontend / web** in esso. Sposta il contenuto del **back-end / web** in **www / admin**. In ogni caso sarà necessario regolare i percorsi in **index.php** e **index-test.php**.

Regola le sessioni e i cookie

Originariamente il backend e il frontend sono destinati a funzionare in diversi domini. Quando lo spostiamo tutti nello stesso dominio, il frontend e il backend condivideranno gli stessi cookie, creando uno scontro. Per sistemarlo, regola il backend dell'applicazione **config backend / config / main.php** come segue:

```
'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',
  ],
],
],
],
```

Spero che questo aiuti gli utenti di hosting condiviso a implementare un'applicazione avanzata.

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

Condivisione di file caricati tra frontend e back-end tramite symlink

Quindi hai caricato i tuoi file in una cartella dire `/backend/web/uploads/` e vuoi che questi caricamenti siano visibili anche sul frontend. L'opzione più semplice è creare un collegamento simbolico nel frontend che collega al back-end:

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

Nelle tue visualizzazioni puoi ora utilizzare i collegamenti relativi ai file:

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

Assicurati che il tuo webserver permetta ai follower di essere seguiti.

Leggi Modello di progetto avanzato online: <https://riptutorial.com/it/yii2/topic/944/modello-di-progetto-avanzato>

Capitolo 12: Pjax

Examples

Passaggio 1 Aggiungi struttura

Nelle viste \ site \ form-submission.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(); ?>
```

Passaggio 2 Codice lato server

```
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,
    ]);
}
```

come usare pjax

Aggiungi questa riga all'inizio della tua vista.

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

Aggiungi le seguenti due righe attorno al contenuto che richiede un aggiornamento parziale.

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

ricarica pjax

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

utilizzare l'argomento di timeout in pjax

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

è possibile specificare un valore intero per l'argomento di timeout, che sarebbe il numero di millisecondi da attendere (il suo valore predefinito è 1000). Se il tempo di esecuzione nel server è maggiore di questo valore di timeout, verrà attivato un caricamento di pagina completo.

Per impostazione predefinita, pjax invierà il modulo utilizzando il metodo GET. È possibile modificare il metodo di invio del modulo a POST come nell'esempio seguente

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

Utilizzo avanzato Pjax

Yii Framework 2.0 viene fornito con il supporto [integrato](#) per [Pjax](#), una libreria JavaScript che riduce i tempi di caricamento delle pagine. Ciò si ottiene semplicemente aggiornando la parte della pagina che è stata modificata tramite Ajax, che può tradursi in notevoli risparmi se si hanno molte altre risorse sulle proprie pagine. Alcuni dei nostri progetti utilizzano questa funzionalità e volevamo condividere alcune lezioni apprese.

Problema : la pagina 1 è una semplice pagina statica che contiene pochi elementi. La pagina 2 include un ActiveForm e altri widget. Le risorse JavaScript ActiveForm devono essere caricate per poter eseguire JavaScript inline, ma poiché la pagina 1 non ha incluso tali risorse, si è verificato un errore JavaScript durante il tentativo di esecuzione della linea activeform: 'Uncaught TypeError: undefined non è un funzione'.

Soluzione : includi le risorse ActiveForm in un pacchetto di risorse condivise che verrà caricato su tutte le pagine, assicurando che qualsiasi pagina di entrata consenta la disponibilità degli script corretti.

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

Problema : nello stesso esempio sopra, la Pagina 1 include alcuni widget (NavBar, ecc.). La pagina 2 include gli stessi widget e alcuni altri (ActiveForm, ecc.). Durante il caricamento della pagina tramite Pjax, era in esecuzione qualche inline JavaScript personalizzato, ma lo script inline inserito dal widget ActiveForm non sembrava funzionare, in quanto il codice di convalida non

funzionava. Nel debug, abbiamo rilevato che la funzione `init` di `ActiveForm` era in esecuzione, ma la variabile `"this"` non sembrava corrispondere a `ActiveForm`. In realtà corrispondeva al `div NavBar`. Analizzando gli ID `div`, abbiamo visto che `ActiveForm` si aspettava di avere l'ID di `# w1`, ma il `NavBar` aveva già assegnato quell'ID al `Page 1` poiché era il primo widget incontrato su quella pagina.

Soluzione : non fare affidamento su `Yii` per generare automaticamente gli ID del widget. Invece, passare sempre un ID durante la creazione del widget per mantenere il controllo di tali ID.

Problema : la richiesta `Pjax` veniva annullata esattamente 1.000 ms dopo l'avvio della richiesta.

Soluzione : aumentare l'impostazione di `timeout Pjax`. Il valore predefinito è 1 secondo, che dovrebbe essere accettabile per i siti di produzione. Tuttavia, in fase di sviluppo, durante l'utilizzo di `xdebug`, i tempi di caricamento della pagina superano regolarmente questo limite.

Problema : l'applicazione `Web` implementa il pattern [Post-Redirect-Get \(PRG\)](#) . `Pjax` ricarica tutta la pagina invece del solo reindirizzamento.

Soluzione : questo è il comportamento previsto di `Pjax`. Il reindirizzamento non raggiunge il suo scopo quando si utilizza `Pjax`, quindi è possibile determinare se una richiesta è `Pjax` e, in tal caso, eseguire il rendering del contenuto anziché reindirizzarlo. Un esempio può essere simile a:

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

Qual è stata la tua esperienza con `Pjax` e `Yii`? Commenta di seguito se hai trovato qualche trucco o hai soluzioni migliori della nostra!

Leggi `Pjax` online: <https://riptutorial.com/it/yii2/topic/4554/pjax>

Capitolo 13: Record attivo

Osservazioni

AR è perfetto quando devi eliminare, aggiornare o creare uno o più record in sequenza. Il suo supporto degli attributi dirty (salvando solo ciò che è stato realmente cambiato) risulta in istruzioni UPDATE ottimizzate che sollevano il carico dal database in modo significativo e riducono le possibilità di conflitti diversi connessi alla modifica dello stesso record da più persone contemporaneamente.

Se non si dispone di una logica veramente complessa nella propria applicazione e quindi non richiede entità di astrazione, AR è la soluzione migliore per eliminare, aggiornare e creare.

AR è anche OK per query semplici con meno di 100 record per pagina. Non è così performante come lavorare con gli array prodotti da query builder o `asArray()` ma è più piacevole lavorare con.

AR non è raccomandato per query complesse. Solitamente si tratta di aggregare o trasformare dati, quindi ciò che viene restituito non si adatta comunque al modello AR. In questo caso è preferibile utilizzare il generatore di query.

Lo stesso vale per l'importazione e l'esportazione. Meglio usare il generatore di query a causa di grandi quantità di dati e di query possibilmente complesse.

Examples

Trova tutti i record

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

o la stenografia

(Restituisce un'istanza del modello di record attivo da una chiave primaria o una matrice di valori di colonna.)

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

restituisce una matrice di istanze di ActiveRecord.

Trova tutto con condizioni

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

Trova tutto con orderBy

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

Or

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

Dove la clausola

OPERATORI

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

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

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

NEL

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

FRA

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

NULLO

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

E

```
$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
```

NON

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

CLIENTI NASCOSTI

```
$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)
```

LIKE OPERATOR con filterWhere metodi activerecord

Ad esempio, nel filtro di ricerca si desidera filtrare il post bruciando il titolo del post o la descrizione postata dall'utente attualmente connesso.

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

i) andFilterWhere ()

```
$postLIKE = Post::find()->where(['user_id' => Yii::$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) oFilterWhere ()

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

iii) filterWhere ()

```
$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: durante l'utilizzo di filterWhere () dobbiamo chiamare all andwhere () o oWhere () dopo filterWhere () altrimenti tutte le condizioni verranno rimosse tranne filterWhere ()

Crea una classe ActiveRecord con valore dei campi basati sugli eventi

```

<?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 questo può essere usato

```

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);
}

```

Trova un record

```
$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();
```

Trova una query

Trova un singolo record basato su id.

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

Seleziona una singola colonna in base all'ID.

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

Recupera il singolo record dal database in base alle condizioni.

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

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

Seleziona record di singole colonne dal database in base alle condizioni.

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

O

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

Ordinato da

```
$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();
```

Record attivi con sottocategorie

Esempio: clienti che possono creare un post. Ogni cliente può creare più post. Non appena il cliente crea il post, il post sarà sottoposto a revisione da parte degli amministratori. Ora dobbiamo recuperare l'elenco dei clienti che hanno tutti i post attivi utilizzando la sotto-query.

Nota: se un cliente ha 5 post, tra 5 post se ha almeno uno inattivo, dobbiamo escludere questo cliente dall'elenco dei clienti.

```
$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
```

Leggi Record attivo online: <https://riptutorial.com/it/yii2/topic/1516/record-attivo>

Capitolo 14: Richiesta Ajax

Examples

Invio del modulo Ajax

Vedi il file:

```
<?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");
        });
    });
});
```

Azione controller:


```

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->request->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 occurred.',
                ],
                'code' => 1, // Some semantic codes that you know them for yourself
            ];
        }
    }
}

```

Renderizza la vista Ajax

`Controller::renderAjax()` metodo `Controller::renderAjax()` può essere utilizzato per rispondere a una richiesta Ajax. Questo metodo è simile a `renderPartial()`, tranne per il fatto che verrà iniettato nel risultato di rendering con script e file JS / CSS che sono registrati con la vista

Supponiamo di avere un modulo di accesso in un file di visualizzazione:

```

<?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>

```

Quando usiamo `renderPartial()` in un'azione controller:

```

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

```

Uscita azione:

```

<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>

```

Quando usiamo `renderAjax()` in un'azione del controller:

```

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

```

Output di azione (JS, CSS iniettato):

```

<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>
```

Se vogliamo escludere alcune risorse dalla vista (per evitare duplicati):

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

Uscita azione (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>
```

Leggi Richiesta Ajax online: <https://riptutorial.com/it/yii2/topic/2944/richiesta-ajax>

Capitolo 15: Routing e URL

Osservazioni

Tutti gli URL dovrebbero essere creati tramite helper `yii\helpers\Url` che ti aiuta molto se decidi di cambiare le regole url in `urlManager`.

Examples

Creazione di URL

Helper `yii\helpers\Url` fornisce una serie di metodi statici per la gestione degli URL. Questo helper può essere utilizzato nel codice visualizzazioni / controller.

URL di una rotta:

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

URL di una rotta con parametri:

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

URL ancorato:

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

URL assoluto:

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

URL assoluto utilizzando lo schema https:

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

Nota: il percorso passato al metodo `Url::to()` è sensibile al contesto. Può utilizzare il modulo corrente e il controller corrente. Ad esempio, supponiamo che il modulo corrente sia `admin` e il controller corrente sia `post` :

percorso relativo solo con ID azione (non contiene alcuna barra):

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

percorso relativo (non ha una barra principale):

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

percorso assoluto (inizia con una barra):

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

URL corrente richiesto:

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

Per creare l'URL in base al **percorso corrente** e i **parametri GET**, utilizzare `Url::current()`.

Supponi `$_GET = ['id' => 10, 'page' => 7]`, la rotta corrente è `post/view`.

URL corrente:

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

URL corrente senza parametro di `page`:

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

URL corrente con parametro di `page` modificato:

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

Leggi Routing e URL online: <https://riptutorial.com/it/yii2/topic/5510/routing-e-url>

Capitolo 16: Sessione

Examples

Sessione in yii2

import Session Class

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

Crea una sessione

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

Memorizza il valore nella variabile di sessione.

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

Ottieni il valore dalla variabile di sessione.

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

Rimuovi la variabile di sessione

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

Rimuovi tutte le variabili di sessione

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

Controlla la variabile Session

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

Flash di sessione

Imposta il flash di sessione

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

Ottieni flash di sessione

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

Controlla il flash della sessione

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

Rimuovi il flash di sessione

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

Rimuovi tutte le variabili flash di sessione

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

Utilizza direttamente la variabile di sessione

Imposta e ottieni la variabile di sessione

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

Flash di sessione

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

Creazione e modifica delle variabili di sessione che sono matrici

Salva la variabile di sessione come variabile.

```
$session = Yii::$app->session;  
  
$sess = $session['keys'];
```

Quindi creare o aggiornare il valore dell'array desiderato

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

E infine salvare nella variabile di sessione

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

Ricorda l'URL da rivedere più tardi

Caso d'uso: ricorda l'URL corrente in cui tornare dopo aver aggiunto un nuovo record in un controller (correlato) diverso, ad esempio crea un nuovo contatto da aggiungere a una fattura modificata.

InvoiceController / actionUpdate:

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

ContactController / actionCreate:

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

Puoi reimpostare l'URL memorizzato una volta terminato:

InvoiceController / actionUpdate:

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

Il nome della chiave - `returnInvoice` in questo esempio - è facoltativo.

Leggi Sessione online: <https://riptutorial.com/it/yii2/topic/3584/sessione>

Capitolo 17: Upload di file

Examples

Come farlo

Caricamento di file

Il caricamento dei file in Yii avviene di solito con l'aiuto di `[[yii \ web \ UploadedFile]]` che incapsula ogni file caricato come oggetto `UploadedFile`. In combinazione con `[[yii \ widgets \ ActiveForm]]` e modelli, è possibile implementare facilmente un meccanismo di caricamento sicuro dei file.

Creazione di modelli

Come quando si utilizzano input di testo normale, per caricare un singolo file si crea una classe del modello e si utilizza un attributo del modello per mantenere l'istanza del file caricato. Devi anche dichiarare una regola di convalida per convalidare il caricamento del file. Per esempio,

```
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;
        }
    }
}
```

Nel codice sopra, l'attributo `imageFile` viene utilizzato per mantenere l'istanza del file caricato. È

associato a una regola di convalida dei `file` che utilizza `[[yii \ validators \ FileValidator]]` per garantire che un file con estensione nome `png` o `jpg` sia caricato. Il metodo `upload()` eseguirà la convalida e salverà il file caricato sul server.

Il validatore di `file` ti consente di controllare le estensioni dei file, le dimensioni, il tipo MIME, ecc. Per ulteriori dettagli, fai riferimento alla sezione Validatori di base.

Suggerimento: se stai caricando un'immagine, puoi prendere in considerazione l'utilizzo del validatore di `image`. Il validatore di `image` è implementato tramite `[[yii \ validators \ ImageValidator]]` che verifica se un attributo ha ricevuto un'immagine valida che può essere quindi salvata o elaborata utilizzando l' [estensione Image](#).

Input del file di rendering

Successivamente, crea un input di file in 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() ?>
```

È importante ricordare che si aggiunge l'opzione `enctype` al modulo in modo che il file possa essere caricato correttamente. La chiamata `fileInput()` eseguirà il rendering di un `<input type="file">` che consentirà agli utenti di selezionare un file da caricare.

Suggerimento: dalla versione 2.0.8, `[[yii \ web \ widgets \ ActiveForm :: fileInput | fileInput]]` aggiunge `enctype` opzione `enctype` al modulo quando viene utilizzato il campo di input del file.

Cablaggio

Ora in un'azione controller, scrivi il codice per collegare il modello e la vista per implementare il caricamento dei file:

```
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]);
}
}

```

Nel codice precedente, quando viene inviato il modulo, viene chiamato il metodo `[[yii \ web \ UploadedFile :: getInstance ()]]` per rappresentare il file caricato come istanza di `UploadedFile` . Ci affidiamo quindi alla convalida del modello per assicurarci che il file caricato sia valido e salvare il file sul server.

Caricamento di più file

Puoi anche caricare più file contemporaneamente, con alcune modifiche al codice elencato nelle sottosezioni precedenti.

Innanzitutto è necessario regolare la classe del modello aggiungendo l'opzione `maxFiles` nella regola di convalida dei `file` per limitare il numero massimo di file consentiti per il caricamento. L'impostazione di `maxFiles` su `0` significa che non esiste un limite al numero di file che possono essere caricati simultaneamente. Il numero massimo di file che possono essere caricati simultaneamente è anche limitato dalla direttiva PHP `max_file_uploads` , che ha valore predefinito `20`. Il metodo `upload()` dovrebbe anche essere aggiornato per salvare i file caricati uno per 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;
    }
}
}
}

```

Nel file di visualizzazione, è necessario aggiungere l'opzione `multiple` alla chiamata `fileInput()` modo che il campo di caricamento del file possa ricevere più file:

```

<?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() ?>

```

Infine, nell'azione del controller, è necessario chiamare `UploadedFile::getInstances()` anziché `UploadedFile::getInstance()` per assegnare un array di istanze di `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]);
    }
}

```

Leggi Upload di file online: <https://riptutorial.com/it/yii2/topic/2221/upload-di-file>

Capitolo 18: Validazione

Examples

Convalida un valore univoco dal database in Yii2

Poche persone hanno problemi con il messaggio di errore che non viene visualizzato se il valore esistente viene inserito nella casella di testo.

Quindi, ad esempio, non sto *permettendo* all'utente di inserire l' *e-mail esistente* .

signup.php

(Pagina in cui si desidera registrare un nuovo utente senza un ID e-mail esistente)

1. Rimuovere `use yii\bootstrap\ActiveForm;` (se presente)
2. Aggiungi `use yii\widgets\ActiveForm;`
3. Aggiungi `'enableAjaxValidation' => true` (In quel campo dove vuoi fermare l'utente per inserire l'e-mail esistente.)

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

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

controllore

Aggiungi queste linee in cima `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())) {

    }
}
```

```
}  
?>
```

Modello

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

Convalida del valore univoco dal database: convalida univoca

Alcune persone hanno problemi con i messaggi di errore che non vengono visualizzati se viene inserito un valore esistente. Ad esempio, non autorizzo l'iscrizione di un utente con un'e-mail esistente.

vista

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

controllore

```
<?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,  
    ]);  
}
```

Modello

```
public function rules()
```

```

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

```

Disabilita messaggio di errore di convalida su Focus / Key Up

Per impostazione predefinita, il messaggio di errore viene visualizzato sotto la `textbox` di `textbox` in `<div class="help-block"></div>` su *keyUp* o *dopo aver premuto il pulsante di invio* se non vengono soddisfatti i vincoli di convalida.

A volte vogliamo un messaggio solo `onKeyUp` ovvero nessuna convalida a evento `onKeyUp`.

Controlliamo il file `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;

    .
    .
    .
}

```

Vediamo che `$validateOnBlur` è impostato su `true` per impostazione predefinita. Cambiare i file framework è una cosa molto brutta da fare, quindi è necessario sovrascriverli quando si utilizza il modulo:

```

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

```

Scenario in convalida

Utilizzando lo scenario è possibile eseguire la convalida in situazioni diverse

Definire lo scenario nella classe del modello

```
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']],
        ];
    }
}
```

Usa scenario nel controller

```
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
}
```

Convalida matrice

Dalla versione 2.0.4 di Yii2 è presente EachValidator per convalidare ogni elemento in una matrice.

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

La parte ['integer'] può essere ogni altro oggetto validatore che Yii2 offre e può contenere gli argomenti specifici per il validatore. Tipo: ['integer', 'min' => 1337] . Se gli ID utente non

contengono un array, la convalida della regola avrà esito negativo.

Se vuoi solo vedere se un attributo contiene un array senza convalidare il contenuto, puoi scrivere il tuo validatore.

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

Leggi Validazione online: <https://riptutorial.com/it/yii2/topic/839/validazione>

Capitolo 19: Yii2 ActiveForm

Examples

Campi modulo in Yii2

Visualizzazione dell'esempio di base della pagina di visualizzazione in Yii2 per i nuovi studenti

Queste sono le classi base che devi aggiungere per creare un modulo usando yii2 ActiveForm

```
<?php

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

La riga sottostante avvierà il tag modulo per il modulo sottostante che mostra l'esempio mostra come specificare l'id per il modulo e come applicare qualsiasi classe per il modulo.

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

Qui \$ modello Specificare quale campo della tabella del database si desidera associare con il modulo dell'oggetto modello memorizzato qui in questa variabile che è stata passata dal relativo controller.

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

'username' e 'password' è il nome del campo della tabella con cui verrà associato il nostro valore.

Qui sotto il codice stiamo inserendo il pulsante di invio per l'invio del modulo e applicando 'Login' come pulsante Testo e classi di base CSS ad esso.

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

Qui sotto il codice stiamo chiudendo il modulo

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

Crea campo password:

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

Crea campo di testo:

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

Crea campo modulo nascosto:

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

Crea menu a discesa:

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

Elenco a discesa con ID e nome

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

Crea FileUploader:

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

Aggiunta di un segnaposto e un'etichetta personalizzata

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

Validazioni ActiveForm

È possibile abilitare / disabilitare le convalide di ajax e client in forma attiva.

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

1. `enableClientValidation` è abilitato per impostazione predefinita in ActiveForm. Se non è necessaria la convalida del client nel modulo, è possibile disabilitare assegnando come falso.
2. `enableAjaxValidation` è disabilitato per impostazione predefinita in ActiveForm. Se si desidera abilitarlo, dobbiamo aggiungere manualmente in ActiveForm come sopra.
3. `validationUrl` : se si desidera mantenere tutta la codifica di convalida nell'azione del

controller separata per questo modulo, è possibile configurare ActiveForm utilizzando `validationUrl`. Se non lo abbiamo impostato, prenderà il valore dell'azione del modulo.

I suddetti due argomenti avranno effetto per l'intera forma. Se si desidera verificare la convalida di ajax solo per un campo particolare nel modulo, è possibile aggiungere `enableAjaxValidation` per quel particolare campo. Funzionerà solo per quel campo non in forma intera.

Ad esempio nel modulo di registrazione si desidera verificare che il nome utente esista già validazione al momento dell'inserimento nel modulo. puoi usare questo argomento `enableAjaxValidation` per quel campo.

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

Leggi Yii2 ActiveForm online: <https://riptutorial.com/it/yii2/topic/6807/yii2-activeform>

Capitolo 20: Yii2 JQuery Calendar per il campo di testo

Examples

Aggiungi il calendario jquery per un campo di testo con il massimo della data corrente

Se vogliamo visualizzare un calendario jquery per l'utente finale che può scegliere la data massima come data corrente nel calendario. Il codice seguente sarà utile per questo scenario.

```
<?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(); ?>
```

Aggiungi il calendario jquery per un campo di testo con data minima

Per alcuni moduli che si desidera visualizzare i giorni da giorni futuri / passati e altri giorni devono essere disattivati, questo scenario sarà di aiuto.

```
<?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(); ?>
```

Aggiungi calendario jquery con dalla data e alla data

Se si desidera avere il calendario per la data e la data e anche per i giorni di calendario della data sarà sempre maggiore del campo della data, quindi lo scenario sottostante aiuterà.

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

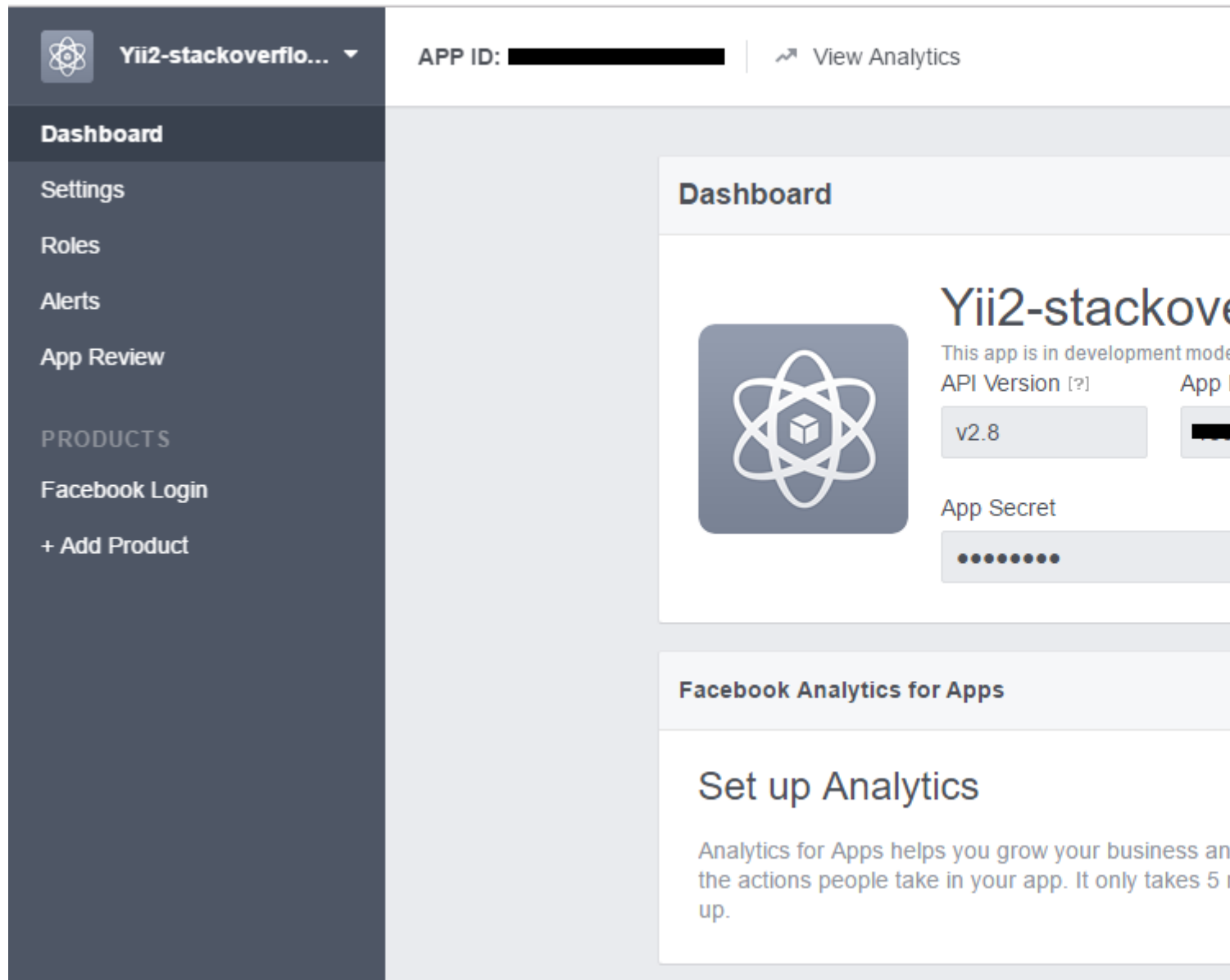
Leggi Yii2 JQuery Calendar per il campo di testo online: <https://riptutorial.com/it/yii2/topic/6366/yii2-jquery-calendar-per-il-campo-di-testo>

Capitolo 21: Yii2 OAuth2 - Es: consumer facebook OAuth2

Examples

Crea un'app sullo sviluppatore di Facebook

Vai a <https://developers.facebook.com/> e crea la tua app.



Fai clic su `Add product` e seleziona `Facebook Login`

Client OAuth Settings

 Yes

Client OAuth Login

Enables the standard OAuth client token flow. Secure your application and prevent abuse by specifying which token redirect URIs are allowed with the options below. Disable globally if not used.

 Yes

Web OAuth Login

Enables web based OAuth client login for building custom login flows. [?]

 No

Force Web OAuth Login

When on, prompts users to use web based login. [?]

 No

Embedded Browser OAuth Login

Enables browser control redirect uri for OAuth client login. [?]

Valid OAuth redirect URIs

 No

Login from Devices

Enables the OAuth client login flow for devices like a smart TV [?]

Deauthorize

Deauthorise Callback URL

Installa yii2-authclient

Prima di installare questa estensione, è necessario installare yii2-app. In questo esempio, utilizzo il modello yii2-basic. Guida per l'installazione [qui](#).

Correre

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

o aggiungi

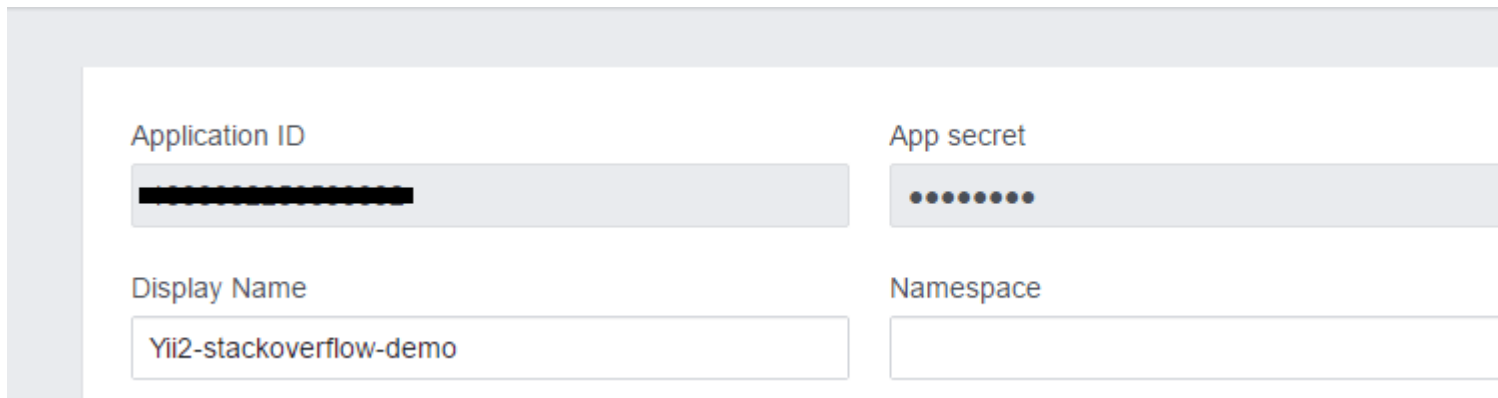
```
"yiisoft/yii2-authclient": "~2.1.0"
```

alla sezione `require` del tuo `composer.json`.

Aggiungi config `authClientCollection` ai tuoi `components` configurazione:

```
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` è application id e `facebook_client_secret` è app secret.



The screenshot shows a form with four fields:

- Application ID:** A text input field containing a redacted value (black bars).
- App secret:** A text input field containing a redacted value (black dots).
- Display Name:** A text input field containing the value "Yii2-stackoverflow-demo".
- Namespace:** An empty text input field.

Aggiungi azione auth e imposta callback

1. Aggiungi pulsante `Login as facebook account` alla tua vista di accesso:

Modifica `site/login.php` nella cartella delle visualizzazioni, aggiungi la linea di tesi al contenuto dell'accesso alla pagina:

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

Sopra, impostiamo che l'azione di `auth` in `SiteController` gestirà il flusso di OAuth2.

Ora lo creiamo.

```
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
}
}
```

Utilizziamo `yii\authclient\AuthAction` per creare URL e reindirizzare alla pagina di accesso di Facebook.

Funzione `onAuthSuccess` utilizzata per ottenere informazioni utente, accedere alla tua app.

Aggiungi `redirect_url` alle impostazioni dell'app Facebook

Se abiliti `prettyUrl` nella tua app yii2, il tuo `redirect_uri` sarà:

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

E disabilita l'URL carino:

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

Esempio:

Client OAuth Settings

<input checked="" type="checkbox"/> Yes	Client OAuth Login Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]	
<input checked="" type="checkbox"/> Yes	Web OAuth Login Enables web based OAuth client login for building custom login flows. [?]	<input type="checkbox"/> No Force Web OAuth Reauthentication When on, prompts people to enter their Facebook password in order to log in on the web. [?]
<input type="checkbox"/> No	Embedded Browser OAuth Login Enables browser control redirect uri for OAuth client login. [?]	

Valid OAuth redirect URIs

<input type="checkbox"/> No	Login from Devices Enables the OAuth client login flow for devices like a smart TV [?]	
-----------------------------	--	--

Esempio per la funzione 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);
}
```

Leggi Yii2 OAuth2 - Es: consumer facebook OAuth2 online:

<https://riptutorial.com/it/yii2/topic/7428/yii2-oauth2---es--consumer-facebook-oauth2>

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con yii2	Bibek Lekhak , Community , Farcaller , jagsler , Mohan Rex , Muhammad Shahzad , Pasha Rumkin , Sam Dark , urmaul , Yasar Arafath , Yasin Patel , Yatin Mistry
2	analisi	Antonín Slejška , Bizley , Sam Dark , XzAeRo
3	API riposante	jagsler , jlapoutre , yafater , Yasin Patel
4	Biscotti	IStranger , Sam Dark
5	componenti	Sam Dark , Yasin Patel
6	Convalide personalizzate	Ejaz Karim
7	Gestione delle risorse	Thomas Rohde
8	Installazione manuale dell'estensione	Insane Skull , mnoronha , Sam Dark , vishuB
9	Lavorare con i database	jagsler , Kiran Muralee
10	Migrazioni del database	Ali MasudianPour , jlapoutre , Sam Dark
11	Modello di progetto avanzato	mnoronha , Mohan Rex , Salem Ouerdani , Sam Dark , topher
12	Pjax	gmc , Manikandan S , Muaaz Rafi , Shaig Khaligli , yafater , Yasin Patel
13	Record attivo	Insane Skull , Manikandan S , Michael St Clair , Mike Artemiev , particleflux , saada , Sam Dark , Yasar Arafath , Yasin Patel
14	Richiesta Ajax	Anton Rybalko , Ilyas karim , meysam
15	Routing e URL	IStranger , Ярослав Гойса
16	Sessione	Brett , Goke Obasa , jlapoutre , MikelG , Yasin Patel

17	Upload di file	Sam Dark
18	Validazione	jagsler , Mihai P. , Nana Partykar , Sam Dark , Yasin Patel
19	Yii2 ActiveForm	Hina Vaja , Manikandan S , particleflux
20	Yii2 JQuery Calendar per il campo di testo	Manikandan S
21	Yii2 OAuth2 - Es: consumer facebook OAuth2	ThanhPV