Kostenloses eBook

LERNEN yii2

Free unaffiliated eBook created from
Stack Overflow contributors.

#yii2

Inhaltsverzeichnis

Über

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

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

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

Kapitel 1: Erste Schritte mit yii2

Bemerkungen

Yii ist ein generisches Web-Programmier-Framework, das heißt, es kann für die Entwicklung aller Arten von Webanwendungen mit PHP verwendet werden. Aufgrund seiner komponentenbasierten Architektur und der ausgefeilten Caching-Unterstützung eignet es sich besonders für die Entwicklung großer Anwendungen wie Portale, Foren, Content Management Systeme (CMS), E-Commerce-Projekte, RESTful-Webdienste usw.

Versionen

Ausführung	Veröffentlichungsdatum
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

Installation oder Setup

Yii2 kann auf zwei Arten installiert werden. Sie sind

1. Installation über Composer
2. Installation aus einer Archivdatei

Installation über Composer

Composer installieren

Wenn Sie Composer noch nicht installiert haben, können Sie dies tun, indem Sie den Anweisungen auf getcomposer.org folgen . Unter Linux und Mac OS X führen Sie die folgenden Befehle aus:

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

Für Windows einfach Composer-setup.exe herunterladen und installieren. Möglicherweise müssen Sie das Github-API-Zugriffstoken so konfigurieren, dass die Github-API-Ratenbeschränkung überschrieben wird.

Installieren von Yii

Wenn Composer installiert ist, können Sie Yii installieren, indem Sie die folgenden Befehle in einem über Web zugänglichen Ordner ausführen:

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

Führen Sie dann den folgenden Befehl aus, um Yii2 mit der Basisvorlage zu installieren.

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

Zur Installation von Yii2 mit erweitertem Vorlagenlauf

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

Danach erstellen Sie eine neue Datenbank und passen Sie die Konfiguration der Komponenten ['db'] in common / config / main-local.php entsprechend an. Führen Sie dann den folgenden Befehl aus

```
php yii migrate
```

Installation aus einer Archivdatei

1. Laden Sie die Archivdatei von Yii-download herunter
2. Entpacken Sie die heruntergeladene Datei in einen über das Web zugänglichen Ordner.
3. Ändern Sie die Datei config / web.php, indem Sie einen geheimen Schlüssel für das Konfigurationselement cookieValidationKey eingeben

   Sie können jeden gewünschten Schlüsseltyp hinzufügen:

   ```
   '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',
```

Installieren Sie Yii2 advanced in Ubuntu

Zuerst müssen wir den Composer installieren. Schritte zum Installieren von Composer Installieren Sie Composer.

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

Jetzt das Verzeichnis wechseln:

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

Überprüfen Sie, ob der Komponist funktioniert

```
composer
```

Jetzt ist Composer installiert.

Es gibt zwei Möglichkeiten, Yii2 advance zu installieren.

1.Installation aus einer Archivdatei

Zip-Datei von unten erhalten.

Entpacken Sie es in das Zielverzeichnis, z. B. `/var/www/html` .

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

Bewegen Sie sich in den "Erweiterten" Ordner. Bewegen Sie sich manuell oder geben Sie den folgenden Befehl ein.

```
cd advanced
```

Führen Sie den folgenden Befehl aus.

```
php init
```

2.Installation über Composer

Für die Installation über Composer ist ein Github-Authentifizierungstoken erforderlich. Für Token müssen Sie sich bei GitHub anmelden.

Nach der Anmeldung können Sie Ihr Token generieren:

Schritte zum Generieren eines Tokens

1. Klicken Sie in der oberen rechten Ecke einer beliebigen Seite auf Ihr Profilfoto und anschließend auf Einstellungen.
2. Klicken Sie in der Seitenleiste für Benutzereinstellungen auf Persönliche Zugriffstoken.
3. Klicken Sie auf Neues Token generieren.
4. Geben Sie Ihrem Token einen beschreibenden Namen.
5. Wählen Sie die Bereiche aus, die Sie diesem Token gewähren möchten.
6. Klicken Sie auf Token generieren.
7. Kopieren Sie das Token in Ihre Zwischenablage. Aus Sicherheitsgründen kann das Token nach der Navigation von dieser Seite nicht mehr angezeigt werden.

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

Nach dem Generieren des Tokens kopieren Sie es

Ändere die Richtung

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

Führen Sie den folgenden Befehl aus

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

Beispiel:

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

Installieren Sie yii2

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

Bewegen Sie sich in den "Erweiterten" Ordner. Bewegen Sie sich manuell oder geben Sie den folgenden Befehl ein.

```
cd advanced
```

Führen Sie den folgenden Befehl aus.

```
php init
```

Es ist fertig!

Jetzt kannst du es überprüfen.

http: // localhost / advanced / frontend / web

und

http: // localhost / advanced / backend / web

Erste Schritte mit yii2 online lesen: https://riptutorial.com/de/yii2/topic/788/erste-schritte-mit-yii2

Kapitel 2: Ajax-Anfrage

Examples

Ajax-Formular absenden

Datei ansehen:

```
<?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");
            });

        });
    });
```

Controller-Aktion:

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

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

Ajax-Ansicht rendern

`Controller::renderAjax()` -Methode kann verwendet werden, um auf eine Ajax-Anforderung zu antworten. Diese Methode ähnelt renderPartial (), wird jedoch mit JS / CSS-Scripts und -Dateien, die in der Ansicht registriert sind, in das Renderingergebnis eingefügt

Angenommen, wir haben ein Anmeldeformular in einer Ansichtsdatei:

```
<?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>
```

Wenn wir `renderPartial()` in einer Controller-Aktion verwenden:

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

Aktionsausgabe:

```
<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>
```

Wenn wir `renderAjax()` in einer Controller-Aktion verwenden:

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

Aktionsausgabe (JS, CSS injiziert):

```
<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>
```

Wenn wir einige Assets aus der Sicht ausschließen möchten (um Dubletten zu verhindern):

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

Aktionsausgabe (keine 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>
```

Ajax-Anfrage online lesen: https://riptutorial.com/de/yii2/topic/2944/ajax-anfrage

Kapitel 3: Aktiver Rekord

Bemerkungen

AR ist perfekt, wenn Sie einen oder mehrere Datensätze nacheinander löschen, aktualisieren oder erstellen müssen. Durch die Unterstützung von Dirty-Attributen (nur das, was wirklich geändert wurde) werden optimierte UPDATE-Anweisungen erzielt, die die Datenbank erheblich entlasten und die Wahrscheinlichkeit für verschiedene Konflikte verringern, die mit der gleichzeitigen Bearbeitung desselben Datensatzes durch mehrere Personen zusammenhängen.

Wenn Sie in Ihrer Anwendung keine wirklich komplexe Logik haben und daher keine abstrakten Elemente benötigen, ist AR die beste Lösung für das Löschen, Aktualisieren und Erstellen.

AR ist auch für einfache Abfragen geeignet, die zu weniger als 100 Datensätzen pro Seite führen. Es ist nicht so leistungsfähig wie das Arbeiten mit Arrays, die von Query Builder oder asArray () erstellt wurden, aber es macht mehr Spaß, damit zu arbeiten.

AR wird für komplexe Abfragen nicht empfohlen. Hierbei geht es in der Regel darum, Daten zu aggregieren oder zu transformieren, sodass das zurückgegebene Ergebnis sowieso nicht in das AR-Modell passt. In diesem Fall ist es vorzuziehen, den Query Builder zu verwenden.

Gleiches gilt für Import und Export. Verwenden Sie den Query Builder wegen der großen Datenmengen und möglicherweise komplexer Abfragen.

Examples

Alle Datensätze finden

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

oder die Abkürzung

(Gibt eine aktive Datensatzmodellinstanz über einen Primärschlüssel oder ein Array von Spaltenwerten zurück.)

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

gibt ein Array von ActiveRecord-Instanzen zurück.

Alle mit wo Zustand finden

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

Alle mit orderBy finden

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

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

Wo Klausel

BETREIBER

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

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

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

IM

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

ZWISCHEN

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

NULL

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

UND

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

ODER

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

NICHT

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

NESTED CLAUSES

```
$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 with filterWhere Aktivierungsmethoden

Im Suchfilter möchten Sie beispielsweise den Beitrag nach gebranntem Beitragstitel oder einer Beschreibung filtern, die vom aktuell angemeldeten Benutzer gepostet wird.

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

i) undFilterWhere ()

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

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

iii) 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
```

Hinweis: Bei der Verwendung von filterWhere () müssen wir all undwhere () oder orWhere () nach dem filterWhere () aufrufen, andernfalls werden alle Bedingungen entfernt, an denen die Bedingungen außer filterWhere () entfernt werden.

Erstellen Sie eine ActiveRecord-Klasse mit ereignisbasiertem Feldwert

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

Oder das kann verwendet werden

```
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);
}
```

Einen Datensatz finden

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

oder

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

oder

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

oder

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

oder

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

Suchen Sie eine Abfrage

Einzelnen Datensatz anhand der ID suchen

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

Wählen Sie eine einzelne Spalte basierend auf der ID aus.

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

Rufen Sie den einzelnen Datensatz je nach Bedingung aus der Datenbank ab.

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

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

Wählen Sie einen einzelnen Spaltensatz aus der Datenbank basierend auf den Bedingungen aus.

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

ODER

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

Sortieren nach

```
$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();
```

Aktive Datensätze mit Unterabfragen

Beispiel: Kunden, die einen Beitrag erstellen können. Jeder Kunde kann mehrere Beiträge erstellen. Sobald der Kunde die Post erstellt, wird die Post von den Administratoren geprüft. Jetzt müssen wir die Kundenliste abrufen, die alle aktiven Beiträge enthält, indem wir die Unterabfrage verwenden.

Hinweis: Wenn ein Kunde 5 Stellen hat, von denen 5 mindestens eine inaktiv hat, müssen wir diesen Kunden von der Kundenliste ausschließen.

```
$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
```

Aktiver Rekord online lesen: https://riptutorial.com/de/yii2/topic/1516/aktiver-rekord

Kapitel 4: Anlagenmanagement

Syntax

- Die abhängigen Vermögenswerte werden vor diesem Vermögen in der angegebenen Reihenfolge geladen
    - public $ hängt = ['yii \ web \ YiiAsset', 'yii \ bootstrap \ BootstrapAsset', 'yii \ bootstrap \ BootstrapPluginAsset', 'cinghie \ fontawesome \ FontAwesomeAsset']];

Bemerkungen

Dieses Beispiel basiert auf der erweiterten Vorlage https://github.com/yiisoft/yii2-app-advanced

Das Cinghie-Asset in diesem Beispiel ist das Asset-Paket für adminLTE
https://github.com/cinghie/yii2-admin-lte

Examples

Dies ist Teil der Layoutdatei

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

Dies ist die Asset-Datei

```
<?php

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

namespace cinghie\adminlte;

use yii\web\AssetBundle;

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

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

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

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

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

```
        'yii\bootstrap\BootstrapAsset',
        'yii\bootstrap\BootstrapPluginAsset',
    'cinghie\fontawesome\FontAwesomeAsset',
    ];

}
```

Das erzeugte HTML mit automatisch geladenen Assets

```
<!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>
```

Anlagenmanagement online lesen: https://riptutorial.com/de/yii2/topic/6900/anlagenmanagement

Kapitel 5: Benutzerdefinierte Validierungen

Einführung

Yii2 verfügt über einige eingebaute Validatoren, die beim Lösen von programmgesteuerten oder allgemeinen Validierungen verwendet werden können. Wenn Sie eine neue Validierung der Geschäftslogik erstellen müssen, müssen Sie Ihre eigenen Validatoren erstellen.

Examples

Arten von Validierungen

Verstehen wir zunächst grundlegende Typen von benutzerdefinierten Validatoren:

1. Inline-Validator
2. Standalone-Validator

Inline-Validator : **Dies** ist der Typ des Validators, den wir innerhalb der Klasse erstellen. Dies ist eine Methode, die wir genau wie andere Methoden definieren, jedoch mit zusätzlichen Parametern, die von Yii2 übergeben werden.

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

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

Benutzerdefinierte Validierungen online lesen:
https://riptutorial.com/de/yii2/topic/9187/benutzerdefinierte-validierungen

Kapitel 6: Datei-Uploads

Examples

Wie es geht

Dateien hochladen

Das Hochladen von Dateien in Yii erfolgt normalerweise mit Hilfe von [[yii \ web \ UploadedFile]], die jede hochgeladene Datei als `UploadedFile` Objekt kapselt. In Kombination mit [[yii \ widgets \ ActiveForm]] und Modellen können Sie problemlos einen sicheren Mechanismus zum Hochladen von Dateien implementieren.

Modelle erstellen

Wie beim Arbeiten mit Klartexteingaben würden Sie zum Hochladen einer einzelnen Datei eine Modellklasse erstellen und ein Attribut des Modells verwenden, um die hochgeladene Dateiinstanz beizubehalten. Sie sollten auch eine Validierungsregel deklarieren, um den Upload der Datei zu überprüfen. Zum Beispiel,

```
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;
        }
    }
}
```

Im obigen Code wird das `imageFile` Attribut verwendet, um die hochgeladene `imageFile` . Es ist mit einer zugehörigen `file` , die verwendet [[yii \ Validatoren \ FileValidator]] , um eine Datei mit der Erweiterung Namen , um sicherzustellen , `png` oder `jpg` hochgeladen. Die `upload()` -Methode führt die Validierung durch und speichert die hochgeladene Datei auf dem Server.

Der `file` - Validator können Sie Dateierweiterungen, Größe, MIME - Typ, usw. Bitte beachten Sie die Core - Validatoren Abschnitt , um weitere Informationen zu überprüfen.

> Tipp: Wenn Sie ein Bild hochladen, können Sie prüfen, die unter Verwendung von `image` statt Validator. Der `image` Validator wird über [[yii \ validators \ ImageValidator]] implementiert, der überprüft, ob ein Attribut ein gültiges Image erhalten hat, das entweder gespeichert oder mit der Imagine-Erweiterung verarbeitet werden kann.

Rendern von Dateieingaben

Als Nächstes erstellen Sie eine Dateieingabe in einer Ansicht:

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

`enctype` Sie, dass Sie die Option `enctype` zum Formular hinzufügen, damit die Datei ordnungsgemäß hochgeladen werden kann. Der Aufruf `fileInput()` gibt ein Tag `<input type="file">` dem Benutzer eine hochzuladende Datei auswählen können.

> Tipp: seit Version 2.0.8 [[yii \ web \ Widgets \ ActiveField :: FileInput- | FileInput-]] fügt `enctype` Option in das Formular automatisch , wenn Dateieingabefeld verwendet wird.

Verdrahtung

Schreiben Sie nun in einer Controller-Aktion den Code, um das Modell zu verkabeln und die Ansicht, um das Hochladen von Dateien zu implementieren:

```
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]);
    }
}
```

Wenn im obigen Code das Formular gesendet wird, wird die Methode [[yii \ web \ UploadedFile :: getInstance ()]] aufgerufen, um die hochgeladene Datei als `UploadedFile` Instanz darzustellen. Wir verlassen uns dann auf die Modellvalidierung, um sicherzustellen, dass die hochgeladene Datei gültig ist, und die Datei auf dem Server speichern.

Mehrere Dateien hochladen

Sie können auch mehrere Dateien gleichzeitig hochladen, wobei der Code in den vorherigen Unterabschnitten angepasst wurde.

Zuerst sollten Sie die Modellklasse anpassen , indem Sie das Hinzufügen `maxFiles` Option in der `file` Gültigkeitsregel die maximale Anzahl von Dateien zu begrenzen , erlaubt zu. Wenn Sie `maxFiles` auf `0` , ist die Anzahl der Dateien, die gleichzeitig hochgeladen werden können, unbegrenzt. Die maximale Anzahl von Dateien, die gleichzeitig hochgeladen werden dürfen, ist auch mit der PHP-Direktive `max_file_uploads` begrenzt. Die `max_file_uploads` ist 20. Die `upload()` - Methode sollte auch aktualisiert werden, um die hochgeladenen Dateien einzeln zu speichern.

```
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;
        }
    }
}
```

In der Ansichtsdatei sollten Sie dem Aufruf von `fileInput()` die Option `multiple` hinzufügen, damit das Feld zum Hochladen von Dateien mehrere Dateien empfangen kann:

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

Und schließlich sollten Sie in der Controller-Aktion `UploadedFile::getInstances()` anstelle von `UploadedFile::getInstance()` `UploadForm::imageFiles` um `UploadForm::imageFiles` ein Array von `UploadedFile` Instanzen `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]);
    }
}
```

Datei-Uploads online lesen: https://riptutorial.com/de/yii2/topic/2221/datei-uploads

Kapitel 7: Datenbankmigrationen

Examples

Migrationen erstellen

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

Das erforderliche Namensargument enthält eine kurze Beschreibung der neuen Migration. Wenn es sich bei der Migration beispielsweise um das Erstellen einer neuen Tabelle mit dem Namen news handelt, können Sie den Namen create_news_table verwenden und den folgenden Befehl ausführen

```
yii migrate/create create_news_table
```

Beispiel für eine Migrationsdatei

```
<?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()
{
}
*/
}
```

Tabelle ablegen

```
public function up()
```

```
{
    $this->dropTable('post');
}
```

Erstellen Sie sofort Tabellenfelder

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

Erzeugt:

```
/**
 * 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');
}
}
```

Tabelle erstellen

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

Spalte löschen / umbenennen / ändern

```
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'));
}
```

Spalte hinzufügen

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

Migrationen rückgängig machen

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

Transaktionsmigrationen

```
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');
}
```

Eine noch einfachere Möglichkeit, Transaktionsmigrationen zu implementieren, besteht darin, Migrationscode in die `safeUp()` und `safeDown()` . Diese beiden Methoden unterscheiden sich von `up()` und `down()` dass sie implizit in einer Transaktion eingeschlossen sind. Wenn ein Vorgang in diesen Methoden fehlschlägt, werden alle vorherigen Vorgänge automatisch zurückgesetzt.

Migrieren mehrerer Datenbanken

Standardmäßig werden Migrationen auf dieselbe Datenbank angewendet, die in der Datenbankanwendungskomponente angegeben ist. Wenn Sie möchten, dass sie auf eine andere Datenbank angewendet werden, können Sie die Befehlszeilenoption db wie folgt angeben:

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

Migrationen wiederholen

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

Auflisten von Migrationen

```
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
```

Migrationsverlauf ändern

```
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
```

Migrationen anwenden

```
yii migrate
```

Dieser Befehl listet alle Migrationen auf, die noch nicht angewendet wurden. Wenn Sie bestätigen, dass Sie diese Migrationen anwenden möchten, wird die up () - oder safeUp () - Methode in jeder neuen Migrationsklasse nacheinander in der Reihenfolge ihrer Zeitstempelwerte ausgeführt. Wenn eine der Migrationen fehlschlägt, wird der Befehl beendet, ohne die restlichen Migrationen anzuwenden.

```
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
```

Datenbankmigrationen online lesen:
https://riptutorial.com/de/yii2/topic/1929/datenbankmigrationen

Kapitel 8: Erweiterte Projektvorlage

Examples

Bereitstellung in einer Shared-Hosting-Umgebung

Die Bereitstellung einer erweiterten Projektvorlage für Shared Hosting ist etwas komplizierter als eine grundlegende, da sie über zwei Webroots verfügt, die von Shared Hosting-Webservern nicht unterstützt werden. Wir müssen die Verzeichnisstruktur anpassen, damit die Frontend-URL http: //site.local und die Backend-URL http: //site.local/admin lautet .

Verschieben Sie Eintrittsskripte in eine einzelne Webroot

Zunächst benötigen wir ein Webroot-Verzeichnis. Erstellen Sie ein neues Verzeichnis und benennen Sie es entsprechend Ihrem Hosting-Webroot-Namen, z. B. www oder public_html oder Ähnlichem. Erstellen Sie dann die folgende Struktur, wobei www das soeben erstellte Hosting-Webroot-Verzeichnis ist:

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

> **www** ist unser Frontend-Verzeichnis, also verschieben Sie den Inhalt von **Frontend / Web** in dieses Verzeichnis. Verschieben Sie den Inhalt des **Backends / web** in **www / admin** . In jedem Fall müssen Sie die Pfade in **index.php** und **index-test.php anpassen** .

Passen Sie Sitzungen und Cookies an

Ursprünglich sollten das Backend und das Frontend auf verschiedenen Domains laufen. Wenn wir alle auf dieselbe Domain umstellen, teilen das Frontend und das Backend dieselben Cookies auf, was zu einem Konflikt führt. Um dies zu beheben, passen Sie die Backend-Anwendung **config backend / config / main.php** wie folgt an:

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

Hoffen Sie, dass dies den Shared Hosting-Benutzern hilft, erweiterte Anwendungen bereitzustellen.

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

Hochladen von Dateien zwischen Frontend und Backend mithilfe von Symlinks

Sie haben also Ihre Dateien in einen Ordner hochgeladen, z. B. `/backend/web/uploads/` und Sie möchten, dass diese Uploads auch im Frontend sichtbar sind. Die einfachste Möglichkeit ist, im Frontend einen Symlink zu erstellen, der mit dem Backend verknüpft ist:

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

In Ihren Ansichten können Sie jetzt relative Links zu den Dateien verwenden:

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

Stellen Sie sicher, dass auf Ihrem Webserver Symlinks verfolgt werden können.

Erweiterte Projektvorlage online lesen: https://riptutorial.com/de/yii2/topic/944/erweiterte-projektvorlage

Kapitel 9: Erweiterung manuell installieren

Examples

Installieren Sie die Erweiterung ohne Composer

> Hinweis: Es wird dringend empfohlen, Composer zu verwenden. Die folgende Anweisung ist im Wesentlichen das, was Composer für Sie tut.

- Laden Sie die Archiv-Erweiterungsdatei der benötigten Version von Github herunter
- Öffnen Sie `composer.json`
- Suchen Sie nach dem `PSR-4` Autoload-Abschnitt, und merken Sie sich das für `kmit/select2`
- Extrahieren Sie die Dateien in den entsprechenden Ordner im Herstellerordner, z. B. `vendor/kmit/select2`
- Fügen Sie folgenden Code zu `vendor/composer/autoload_psr4.php`

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

- Fügen Sie folgenden Code zu `vendor/yiisoft/extensions.php` :

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

Video Tutroial

Erweiterung manuell installieren online lesen: https://riptutorial.com/de/yii2/topic/2224/erweiterung-manuell-installieren

Kapitel 10: Kekse

Bemerkungen

Cookies sind Teil der HTTP-Anfrage, daher ist es eine gute Idee, sowohl Controller als auch Controller zu erledigen, deren Verantwortlichkeit sich genau auf Anfrage und Antwort bezieht.

Examples

Cookie setzen

Um ein Cookie zu setzen, z. B. um es zu erstellen und für den Versand an den Browser zu `\yii\web\Cookie` müssen Sie eine neue Instanz der Klasse `\yii\web\Cookie` erstellen und der `\yii\web\Cookie` hinzufügen:

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

Im obigen Abschnitt übergeben wir Parameter an den Cookie-Klassenkonstruktor. Diese sind im Wesentlichen die gleichen wie bei der systemeigenen PHP- Setcookie- Funktion:

- `name` - Name des Cookies.
- `value` - Wert des Cookies. Stellen Sie sicher, dass es eine Zeichenfolge ist. Browser sind normalerweise nicht glücklich über binäre Daten in Cookies.
- `domain` - Domäne, für die Sie den Cookie setzen.
- `expire` - Unix-Zeitstempel, der den Zeitpunkt angibt, zu dem der Cookie automatisch gelöscht werden soll.
- `path` - Der Pfad auf dem Server, in dem das Cookie verfügbar sein wird.
- `secure` - Wenn `true` , wird das Cookie nur gesetzt, wenn HTTPS verwendet wird.
- `httpOnly` - Wenn `true` , ist ein Cookie nicht über JavaScript verfügbar.

Einen Cookie lesen

Um ein Cookie zu lesen, verwenden Sie den folgenden Code:

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

Hinweis: Dieser Code ermöglicht das Lesen von Cookies, die mithilfe der Cookie-Komponente festgelegt wurden (da standardmäßig alle Cookies unterzeichnet werden). Wenn Sie Cookies mit JS-Code hinzufügen / aktualisieren, können Sie sie daher nicht standardmäßig lesen.

Cookies für Subdomains

Aus Sicherheitsgründen sind Cookies standardmäßig nur auf derselben Domäne verfügbar, von der aus sie gesetzt wurden. Wenn Sie beispielsweise ein Cookie auf der Domain `example.com` , können Sie es nicht auf der Domain `www.example.com` . Wenn Sie also Subdomains (z. B. admin.example.com, profile.example.com) verwenden möchten, müssen Sie die `domain` explizit festlegen:

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

Jetzt kann ein Cookie von allen Subdomains von `example.com` gelesen werden.

Subdomainübergreifende Authentifizierungs- und Identitäts-Cookies

Im Falle eines Autologin oder "Remember Me" -Cookies gelten die gleichen Macken wie bei Subdomain-Cookies. Diesmal müssen Sie jedoch die Benutzerkomponente konfigurieren und das `identityCookie` Array auf die gewünschte Cookie-Konfiguration setzen.

Öffnen Sie Ihre Anwendungskonfigurationsdatei und fügen Sie der Benutzerkomponenten-Konfiguration die Parameter " `identityCookie` hinzu:

```
$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,
            ],
        ],

    ],
];
```

Beachten Sie, dass `cookieValidationKey` für alle `cookieValidationKey` gleich sein sollte.

Beachten Sie, dass Sie die Eigenschaft `session::cookieParams` konfigurieren `session::cookieParams` , dass samedomain als `user::identityCookie` um sicherzustellen, dass die `login` und `logout` für alle Unterdomänen funktioniert. Dieses Verhalten wird im nächsten Abschnitt besser erläutert.

Parameter für Sitzungscookies

Parameter für Sitzungscookies sind wichtig, wenn Sie die Sitzung aufrechterhalten müssen, während Sie von einer Subdomain zur anderen wechseln, oder wenn Sie die Backend-App dagegen unter `/admin` URL hosten und die Sitzung separat behandeln möchten.

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

Kekse online lesen: https://riptutorial.com/de/yii2/topic/2945/kekse

Kapitel 11: Komponenten

Examples

Anwendungskomponenten erstellen und verwenden

Schritte zum Erstellen einer Komponente:

- Erstellen Sie einen Ordner mit dem Namen `components` in Ihrem Projektstammordner
- Erstellen Sie Ihre Komponente im Komponentenordner, z. B. `MyComponent.php`

```
namespace app\components;

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


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

- Registrieren Sie Ihre Komponente in der Datei `config/web.php`

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

Jetzt können Sie Ihre Komponentenmethode verwenden:

```
namespace app\controllers;

use Yii;

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

Dropdown-Liste mit Komponentenfunktion

Funktion in MyComponent.php erstellen

```
    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"];
          }
        }
```

-> Komponente in web.php registrieren

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

-> benutze es in deiner Sicht

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

Komponenten online lesen: https://riptutorial.com/de/yii2/topic/2217/komponenten

Kapitel 12: Mit Datenbanken arbeiten

Examples

Verwenden des Yii2 Query Builder

Yii2 bietet effiziente Methoden zum Abrufen von Daten aus der Datenbank. ***Beachten Sie*** ein Beispiel für eine einfache ***Employer-*** Tabelle mit den Feldern ***emp_id, emp_name und emp_salary*** . Um die Mitarbeiternamen und deren Gehälter abzurufen, verwenden wir die Abfrage.

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

Um die obige Abfrage in Yii2 zu generieren, gibt es viele Methoden. Eine der Methoden ist die Verwendung eines ***yii \ db \ Quer*** *y-* Objekts.

```
//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.
```

Wir können eine foreach-Schleife verwenden, um jedes Name-Wert-Paar im ***$ rows-*** Array zu durchlaufen.

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

Dies wird ausgegeben

> Name des Mitarbeiters: Kiran, Gehalt des Mitarbeiters: 25000
>
> Name des Mitarbeiters: Midhun, Mitarbeitergehalt: 50000
>
> Name des Mitarbeiters: Jishnu, Gehalt des Angestellten: 20000
>
> Name des Mitarbeiters: Ajith, Gehalt des Mitarbeiters: 25000
>
> Name des Mitarbeiters: Akshay, Gehalt des Angestellten: 750000

Mehr Beispiele

Angenommen, wir müssen den Namen der Mitarbeiter ermitteln, deren Gehalt 25000 beträgt. Wir können die Abfrage in SQL schreiben

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

In Yii2 der Code zum Generieren der obigen Abfrage

```
$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();
```

Wenn wir Mitarbeiternamen suchen müssen, deren Gehalt mehr als 25000 beträgt, können wir den Code in Yii2 als schreiben

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

Weitere Zustandsüberprüfung mit where ()

Mehrere Bedingungen können mit der **where ()** -Methode wie unten angegeben geschrieben werden.

```
// 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
```

Der obige Code holt einen Mitarbeiter mit dem Namen **Kiran** und einem Gehalt von **25000 ab** . Wenn mehrere Mitarbeiter die obige Bedingung erfüllen, stellt der Anruf ***eins ()*** sicher, dass nur das erste Ergebnis abgerufen wird. Um alle Ergebnisse abzurufen, sollten Sie ***all () verwenden*** .

Wenn Sie ***all () verwenden, ist*** das Ergebnis immer ein Array. Auch wenn es nur eins oder null gibt. Dieses Array enthält alle Ergebnisse als Arrays oder ist leer, wenn keine Datensätze übereinstimmen. Der Aufruf ***one ()*** gibt das resultierende Array direkt zurück oder false, wenn die Abfrage nichts zurückgibt.

Der äquivalente Code in SQL ist unten angegeben.

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

Eine alternative Schreibweise für die obige Abfrage in Yii2 ist unten angegeben.

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

```
    ->from('employee')
    ->where(['emp_name' => 'Kiran'])
    ->andWhere(['emp_salary' => 25000])
    ->one();
```

Zusätzliche Bedingungen können mit **undWhere** angegeben werden. Dies ist hilfreich, wenn wir der Abfrage später eine zusätzliche Bedingungsprüfung hinzufügen müssen.

Eine weitere Möglichkeit, mehrere Bedingungen anzugeben, besteht darin, das **Operator-Format der where ()** -Methode zu verwenden. Die obige Abfrage kann auch wie folgt geschrieben werden.

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

Hier geben wir den Operator ' **and** ' als erstes Element im Array an. In ähnlicher Weise können wir auch ' **oder** ', ' **zwischen** ', ' **nicht zwischen** ', ' **in** ', ' **nicht in** ', ' **wie** ', ' **oder ähnlich** ', ' **nicht wie** ', ' **oder nicht wie** ', ' **existieren** ' verwenden. , ' **nicht vorhanden** ', ' **>** ', ' **<=** ' usw. als Operatoren.

Beispiele für 'in' und 'like'

Angenommen, wir müssen die Mitarbeiter finden, deren Gehälter **20000, 25000 und 50000 betragen** . In normalen SQL würden wir die Abfrage als schreiben

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

In Yii2 können wir dies wie folgt schreiben.

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

Eine andere Möglichkeit, dieselbe Bedingung anzugeben, ist

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

Ebenso kann ' **nicht in** ' anstelle von ' **in** ' angegeben werden, wenn alle Mitarbeiter ohne Gehälter 20000, 25000 und 50000 erhalten möchten.

Lassen Sie uns nun einige Beispiele für die Verwendung von ' **like** ' innerhalb von where () anzeigen. Nehmen wir an, wir müssen alle Mitarbeiter finden, die in ihrem Namen die Zeichenfolge " **gopal** " haben. Die Namen können venugopal, rajagopal, gopalakrishnan usw. sein. Die SQL-Abfrage ist unten angegeben.

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

In Yii2 schreiben wir das als

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

Wenn wir alle Mitarbeiter finden müssen, die in ihrem Namen die Zeichenfolge ' **gopal** ' und ' **nair** ' enthalten. Wir können als schreiben

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

Dies würde als bewerten

> Wählen Sie * aus dem Mitarbeiter aus, in dem der Emp_name wie '% gopal%' und '% nair%' enthalten ist.

In ähnlicher Weise können wir " **nicht mögen** " verwenden, um anzugeben, dass alle Mitarbeiter nicht die Zeichenfolge " **gopal** " und " **nair** " im Namen haben.

OrderBy () verwenden

Die orderBy () -Methode gibt das ORDER BY-Fragment einer SQL-Abfrage an. Betrachten Sie zum Beispiel unsere Employee-Tabelle mit den Feldern **emp_id, emp_first_name, emp_last_name und emp_salary** sql wie unten angegeben.

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

In yii2 können wir die Abfrage wie unten beschrieben erstellen

```
//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();
```

Wenn wir die Mitarbeiter mit ihrem Vornamen in aufsteigender Reihenfolge und dann mit ihren Gehältern in absteigender Reihenfolge bestellen müssen, können wir sie wie folgt in plain sql schreiben.

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

Das entsprechende sql kann mit yii2 wie folgt erstellt werden

```
//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();
```

Sie können ORDER BY auch mithilfe einer Zeichenfolge angeben, genau wie beim Schreiben von SQL-Rohanweisungen. Die obige Abfrage kann beispielsweise auch wie unten angegeben generiert werden.

```
//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();
```

Sie können addOrderBy () aufrufen, um dem ORDER BY-Fragment zusätzliche Spalten hinzuzufügen. Zum Beispiel

```
//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();
```

Mit Datenbanken arbeiten online lesen: https://riptutorial.com/de/yii2/topic/4167/mit-datenbanken-arbeiten

Kapitel 13: Pjax

Examples

Schritt 1 Struktur hinzufügen

In den Ansichten \ 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(); ?>
```

Schritt 2 Server Side Code

```
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,
    ]);
}
```

Wie benutze ich pjax?

Fügen Sie diese Zeile am Anfang Ihrer Ansicht hinzu.

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

Fügen Sie die folgenden zwei Zeilen um den Inhalt hinzu, für den eine teilweise Aktualisierung erforderlich ist.

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

pjax neu laden

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

Verwenden Sie das Timeout-Argument in pjax

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

Sie können einen ganzzahligen Wert für das Timeout-Argument angeben. Dies ist die Anzahl der zu wartenden Millisekunden (der Standardwert ist 1000). Wenn die Ausführungszeit auf dem Server diesen Timeout-Wert überschreitet, wird ein vollständiger Seitenladevorgang ausgelöst.

Standardmäßig sendet pjax das Formular mit der GET-Methode. Sie können die Formularübermittlungsmethode wie im folgenden Beispiel in POST ändern

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

Pjax erweiterte Verwendung

Yii Framework 2.0 wird mit integrierter Unterstützung für Pjax geliefert , einer JavaScript-Bibliothek, die die Ladezeiten von Seiten reduziert. Dies wird erreicht, indem nur der Teil der Seite aktualisiert wird, der durch Ajax geändert wurde. Dies kann zu erheblichen Einsparungen führen, wenn sich auf Ihren Seiten viele andere Assets befinden. Einige unserer Projekte nutzen diese Funktionalität und wir wollten einige Lektionen teilen.

Problem : Seite 1 ist eine einfache statische Seite, die wenige Elemente enthält. Seite 2 enthält eine ActiveForm sowie andere Widgets. Die ActiveForm-JavaScript-Ressourcen müssen geladen werden, damit das Inline-JavaScript ausgeführt werden kann. Da diese Assets jedoch nicht enthalten waren, trat beim Versuch, die Activeform-Zeile auszuführen, ein JavaScript-Fehler auf: 'Uncught TypeError: undefined ist kein Funktion'.

Lösung : Fügen Sie ActiveForm-Elemente in ein Paket mit gemeinsam genutzten Bestandteilen ein, das über alle Seiten geladen wird, und stellen Sie so sicher, dass auf jeder Einstiegsseite die richtigen Skripts verfügbar sind.

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

Problem : Im selben Beispiel oben enthält Seite 1 einige Widgets (NavBar usw.). Seite 2 enthält

dieselben Widgets sowie einige weitere (ActiveForm usw.). Beim Laden der Seite über Pjax wurden einige benutzerdefinierte Inline-JavaScript ausgeführt, das Inline-Skript des ActiveForm-Widgets schien jedoch nicht zu funktionieren, da der Validierungscode nicht funktionierte. Beim Debugging stellten wir fest, dass die ActiveForm-Init-Funktion ausgeführt wurde, aber die 'this'-Variable schien nicht der ActiveForm zu entsprechen. Es entsprach tatsächlich der NavBar-Div. Bei der Untersuchung der Div-IDs sahen wir, dass ActiveForm die ID von # w1 erwartete, aber der NavBar war diese ID bereits auf der Seite 1 zugewiesen, da dies das erste Widget auf dieser Seite war.

Lösung : Verlassen Sie sich nicht auf Yii, um die Widget-IDs automatisch für Sie zu generieren. Übergeben Sie beim Erstellen des Widgets immer eine ID, um die Kontrolle über diese IDs zu behalten.

Problem : Die Pjax-Anforderung wurde genau 1.000 ms nach dem Initiieren der Anforderung abgebrochen.

Lösung : Erhöhen Sie die Pjax-Zeitlimiteinstellung. Die Standardeinstellung ist 1 Sekunde, was für Produktionsstandorte akzeptabel sein sollte. Bei der Entwicklung von xdebug liegen die Ladezeiten unserer Seiten jedoch regelmäßig über dieser Grenze.

Problem : Webanwendung implementiert das PRG- Muster (Post-Redirect-Get) . Pjax lädt die gesamte Seite anstelle der Umleitung neu.

Lösung : Dies ist das beabsichtigte Verhalten von Pjax. Die Weiterleitung erfüllt bei Verwendung von Pjax nicht ihren Zweck. Sie können also bestimmen, ob eine Anforderung Pjax ist. Wenn dies der Fall ist, rendern Sie den Inhalt, anstatt ihn umzuleiten. Ein Beispiel kann wie folgt aussehen:

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

Welche Erfahrungen haben Sie mit Pjax und Yii gemacht? Kommentieren Sie unten, wenn Sie Gotchas gefunden haben oder bessere Lösungen als unsere haben!

Pjax online lesen: https://riptutorial.com/de/yii2/topic/4554/pjax

Kapitel 14: Restful API

Examples

Beginnen Sie mit Rest API

Wir haben eine Tabelle mit Ländern, also erstellen wir ein Modell, das als länderspezifisches Modell bezeichnet wird

```
<?php

namespace app\models;

use Yii;

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

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

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

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

und ich erstelle einen rest Webservice, wir erstellen einen Controller für restapi und setzen die modelClass-Variable für unser Modell.

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

für restapi benötigen wir hübsche URLs und
Wir fügen diese Regel für eine hübsche URL hinzu

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

Danach können wir unsere Rest-API als Beispiel testen

http: // localhost / countrylist gibt uns eine Liste der Grafschaften.

Wie werden die Standardaktionen von rest api Yii2 überschrieben?

Als Beispiel möchten Sie die Paginierung in Ihrer Standardindexaktion deaktivieren und alle Ergebnisse im Index abrufen. Wie kannst du das machen? Es ist einfach. Sie sollten die Indexaktion in Ihrem Controller folgendermaßen überschreiben:

```
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;
}
```

Überschreiben Sie den Inhaltstyp für bestimmte Aktionen

Anwendungsfall: Nur eine Aktion, die einen einfachen (Text-) Inhalt unverändert zurückgeben soll

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

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

Vordefinierte Antwortformate sind:

- FORMAT_HTML
- FORMAT_XML
- FORMAT_JSON
- FORMAT_JSONP
- FORMAT_RAW

Es gibt keinen MIME-Typ für `text/plain` aus der Box.

```
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
}
```

Restful API online lesen: https://riptutorial.com/de/yii2/topic/6102/restful-api

Kapitel 15: Routing und URLs

Bemerkungen

Alle URLs sollten über das `yii\helpers\Url` Es hilft Ihnen sehr, wenn Sie sich dazu entscheiden, die URL-Regeln in URLManager zu ändern.

Examples

URLs erstellen

Hilfe yii \ helpers \ Url bietet eine Reihe statischer Methoden zum Verwalten von URLs. Dieser Helfer kann in Ansichten / Controller-Code verwendet werden.

URL zu einer Route:

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

URL zu einer Route mit Parametern:

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

verankerte URL:

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

absolute URL:

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

absolute URL mit dem https-Schema:

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

Hinweis: Die an die `Url::to()` Methode übergebene Route ist kontextsensitiv. Es kann Strommodule und Stromregler verwenden. Angenommen, das aktuelle Modul ist `admin` und der aktuelle Controller ist `post` :

Relative Route nur mit Aktions-ID (enthält überhaupt keine Schrägstriche):

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

relative Route (hat keinen führenden Schrägstrich):

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

absolute Route (beginnt mit Schrägstrich):

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

aktuell angeforderte URL:

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

Um eine URL basierend auf der **aktuellen Route** und den **GET-Parametern** zu erstellen, verwenden Sie Url :: current () .

Angenommen, `$_GET = ['id' => 10, 'page' => 7]` , ist die aktuelle Route `post/view` .

aktuelle URL:

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

aktuelle URL ohne `page` Parameter:

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

aktuelle URL mit geänderten `page` Parameter:

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

Routing und URLs online lesen: https://riptutorial.com/de/yii2/topic/5510/routing-und-urls

Kapitel 16: Session

Examples

Sitzung in yii2

Sitzungsklasse importieren

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

Sitzung erstellen

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

Speichern Sie den Wert in der Sitzungsvariablen.

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

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

Rufen Sie den Wert aus der Sitzungsvariablen ab.

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

Entfernen Sie die Sitzungsvariable

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

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

Entfernen Sie alle Sitzungsvariablen

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

Überprüfen Sie die Sitzungsvariable

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

Session Flash

Sitzungsblitz einstellen

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

Session-Flash abrufen

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

Überprüfen Sie den Sitzungsblitz

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

Sitzungsblitz entfernen

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

Entfernen Sie alle Session-Flash-Variablen

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

Sitzungsvariable direkt verwenden

Sitzungsvariable setzen und abrufen

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

Session-Flash

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

Erstellen und Bearbeiten von Sitzungsvariablen, die Arrays sind

Speichern Sie die Sitzungsvariable als Variable.

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

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

Erstellen oder aktualisieren Sie dann den gewünschten Array-Wert

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

Speichern Sie schließlich in die Sitzungsvariable

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

Erinnern Sie sich an die URL, um sie später erneut aufzurufen

Anwendungsfall: Erinnern Sie sich an die aktuelle URL, zu der Sie zurückkehren möchten, nachdem Sie einen neuen Datensatz in einem anderen (zugehörigen) Controller hinzugefügt haben. Erstellen Sie beispielsweise einen neuen Kontakt, den Sie zu einer bearbeiteten Rechnung hinzufügen möchten.

InvoiceController / actionUpdate:

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

ContactController / actionCreate:

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

Sie können die gespeicherte URL zurücksetzen, wenn Sie fertig sind:

InvoiceController / actionUpdate:

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

Der Schlüsselname - in diesem Beispiel `returnInvoice` - ist optional.

Session online lesen: https://riptutorial.com/de/yii2/topic/3584/session

Kapitel 17: Testen

Examples

Testumgebung einrichten

Codeception installieren:

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

Faker installieren:

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

Erstellen Sie eine Datenbank, die nur für die Tests verwendet wird. Sie können die vorhandene Datenbank duplizieren oder Migrationen anwenden:

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

Passen Sie die Konfiguration der `components['db']` in `tests/codeception/config/config-local.php` .

Fügen Sie das Verzeichnis `/var/www/yii/vendor/bin` zu Ihrem Pfad hinzu.

Überprüfen Sie alle Konfigurations- und `.yml` Dateien.

Starten Sie den Webserver, zB:

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

Führen Sie die Tests durch:

```
codecept run
```

Mehr Informationen:

- 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

Hinweis: Diese Anweisungen gelten für die Yii2-Version 2.0.9. In der Version 2.0.10 wird laut Sam Dark der Testteil umgestaltet (und die Anweisungen müssen aktualisiert werden). Die Version 2.0.10 sollte am 11. September 2016 veröffentlicht werden:

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

Wie wird ActiveRecord gespielt?

Wenn Sie AR übermitteln möchten, das nicht versucht, eine Verbindung zur Datenbank herzustellen, können Sie dies auf folgende Weise tun (bei Verwendung von 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'
]);
```

Der Haken ist, dass wir die attributes () -Methode überschreiben müssen, da ActiveRecord standardmäßig die Attributliste aus dem Datenbankschema abruft, das wir zu vermeiden versuchen.

Testen online lesen: https://riptutorial.com/de/yii2/topic/2226/testen

Kapitel 18: Validierung

Examples

Überprüfen Sie den eindeutigen Wert aus der Datenbank in Yii2

Nur wenige Personen haben Probleme mit der Fehlermeldung, wenn der vorhandene Wert in das Textfeld eingegeben wird.

So zum Beispiel bin ich *nicht so dass* Benutzer *vorhandene E - Mail* einzugeben.

signup.php

(Seite, auf der Sie einen neuen Benutzer ohne vorhandene E-Mail-ID registrieren wollten)

1. Entfernen Sie `use yii\bootstrap\ActiveForm;` (Falls vorhanden)
2. Fügen Sie `use yii\widgets\ActiveForm;`
3. Fügen Sie `'enableAjaxValidation' => true` (In diesem Feld, wo Sie den Benutzer für die Eingabe der vorhandenen E-Mail-ID anhalten möchten.)

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

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

Regler

Fügen Sie diese Zeilen oben hinzu. `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())) {

    }
```

```
  }
?>
```

Modell

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

Validieren des eindeutigen Werts aus der Datenbank: Eindeutige Validierung

Bei einigen Personen treten Probleme mit Fehlermeldungen auf, wenn ein vorhandener Wert eingegeben wird. Zum Beispiel erlaube ich keine Benutzeranmeldung mit einer vorhandenen E-Mail.

Aussicht

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

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

Regler

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

Modell

```
public function rules()
```

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

Deaktivieren Sie die Überprüfungsfehlermeldung bei Fokus / Key Up

Standardmäßig erscheint eine Fehlermeldung unter dem `textbox` in `<div class="help-block"></div>` bei *keyUp* oder *nach dem Drücken des " Submit* `<div class="help-block"></div>` *-Buttons* , wenn keine Validierungsbedingungen erfüllt sind.

Manchmal möchten wir eine Nachricht nur beim `onKeyup` , dh keine Validierung beim Ereignis `onKeyup` .

`yii2/widgets/ActiveForm.php` wir `yii2/widgets/ActiveForm.php` Datei `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;

  .
  .
  .

}
```

Dort sehen wir, dass `$validateOnBlur` standardmäßig auf `true` gesetzt ist. Das Ändern von Framework-Dateien ist eine sehr schlechte Sache, daher müssen wir sie bei Verwendung des Formulars überschreiben:

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

Szenario in der Validierung

Mit Hilfe des Szenarios können Sie die Validierung in verschiedenen Situationen durchführen

Szenario in Modellklasse definieren

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

Szenario im Controller verwenden

```
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
}
```

Array validieren

Seit Yii2 Version 2.0.4 gibt es den EachValidator, um jedes Element in einem Array zu überprüfen.

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

Der Teil `['integer']` kann jedes andere von Yii2 angebotene Prüferobjekt sein und kann die

spezifischen Argumente für den Prüfer enthalten. Wie: `['integer', 'min' => 1337]` . Wenn die Benutzer-IDs kein Array enthält, schlägt die Überprüfung der Regel fehl.

Wenn Sie nur sehen möchten, ob ein Attribut ein Array enthält, ohne den Inhalt zu überprüfen, können Sie einen eigenen Prüfer schreiben.

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

Validierung online lesen: https://riptutorial.com/de/yii2/topic/839/validierung

Kapitel 19: Yii2 ActiveForm

Examples

Formularfelder in Yii2

Ein einfaches Beispiel für die Ansichtsseite in Yii2 für neue Lernende anzeigen

Dies sind grundlegende Klassen, die Sie hinzufügen müssen, um mit yii2 ActiveForm ein Formular zu erstellen

```
<?php

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

In der unteren Zeile wird das Formular-Tag für unser Formular unten angezeigt. Es zeigt ein Beispiel, wie die ID für das Formular angegeben wird und wie Klassen für das Formular angewendet werden.

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

Here $ model Geben Sie an, welches Datenbanktabellenfeld mit diesem Modellobjekt verknüpft werden soll, das hier in dieser Variablen gespeichert ist, die vom entsprechenden Controller übergeben wurde.

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

'Benutzername' und 'Passwort' ist der Name des Tabellenfeldes, an das unser Wert gebunden wird.

Im folgenden Code wird der Submit-Button für die Formularübermittlung eingefügt und "Login" als Button-Text und grundlegende CSS-Klassen angewendet.

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

Hier unten im Code schließen wir das Formular aus

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

Passwortfeld erstellen:

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

Textfeld erstellen:

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

Verstecktes Formularfeld erstellen:

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

Dropdown erstellen:

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

Dropdown-Liste mit ID und Name

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

FileUploader erstellen:

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

Platzhalter und benutzerdefinierte Beschriftung hinzufügen

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

ActiveForm-Überprüfungen

Sie können AJAX- und Client-Validierungen in aktiver Form aktivieren / deaktivieren.

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

1. `enableClientValidation` ist standardmäßig in ActiveForm aktiviert. Wenn Sie keine Client-Validierung in Form benötigen, können Sie sie als falsch deaktivieren.
2. `enableAjaxValidation` ist in ActiveForm standardmäßig deaktiviert. Wenn Sie es aktivieren möchten, müssen Sie es wie oben manuell in ActiveForm hinzufügen.
3. `validationUrl`

- Wenn Sie die gesamte Validierungscodierung in einer separaten Controller-Aktion für dieses Formular beibehalten möchten, können Sie die aktive Form mithilfe von `validationUrl` konfigurieren. Wenn wir dies nicht festgelegt haben, wird der Aktionswert des Formulars übernommen.

Die beiden obigen Argumente wirken sich auf die gesamte Form aus. Wenn Sie die Ajax-Validierung nur für ein bestimmtes Feld im Formular prüfen möchten, können `enableAjaxValidation` für dieses bestimmte Feld `enableAjaxValidation` hinzufügen. Es funktioniert nur für dieses Feld und nicht für das gesamte Formular.

Zum Beispiel möchten Sie im Anmeldeformular überprüfen, ob der Benutzername bereits gültig ist, sobald der Benutzer das Formular eingegeben hat. Sie können dieses Argument `enableAjaxValidation` für dieses Feld verwenden.

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

Yii2 ActiveForm online lesen: https://riptutorial.com/de/yii2/topic/6807/yii2-activeform

Kapitel 20: Yii2 Jquery-Kalender für Textfeld

Examples

Fügen Sie für ein Textfeld einen Jquery-Kalender mit dem aktuellen Datum als max hinzu

Wenn wir einen Jquery-Kalender für Endbenutzer anzeigen möchten, der das maximale Datum als aktuelles Datum im Kalender auswählen kann. Der folgende Code ist für dieses Szenario hilfreich.

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

Hinzufügen eines Jquery-Kalenders für ein Textfeld mit Mindestdatum

Bei einigen Formularen möchten Sie die Tage der zukünftigen / vergangenen Tage anzeigen und andere Tage müssen deaktiviert werden, dann hilft dieses Szenario.

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

Fügen Sie einen Jquery-Kalender mit Datum und Datum hinzu

Wenn Sie möchten, dass Kalender für Datum und Datum und auch bis Datum Kalendertage immer größer sind als vom Datumsfeld, dann hilft das folgende Szenario.

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

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

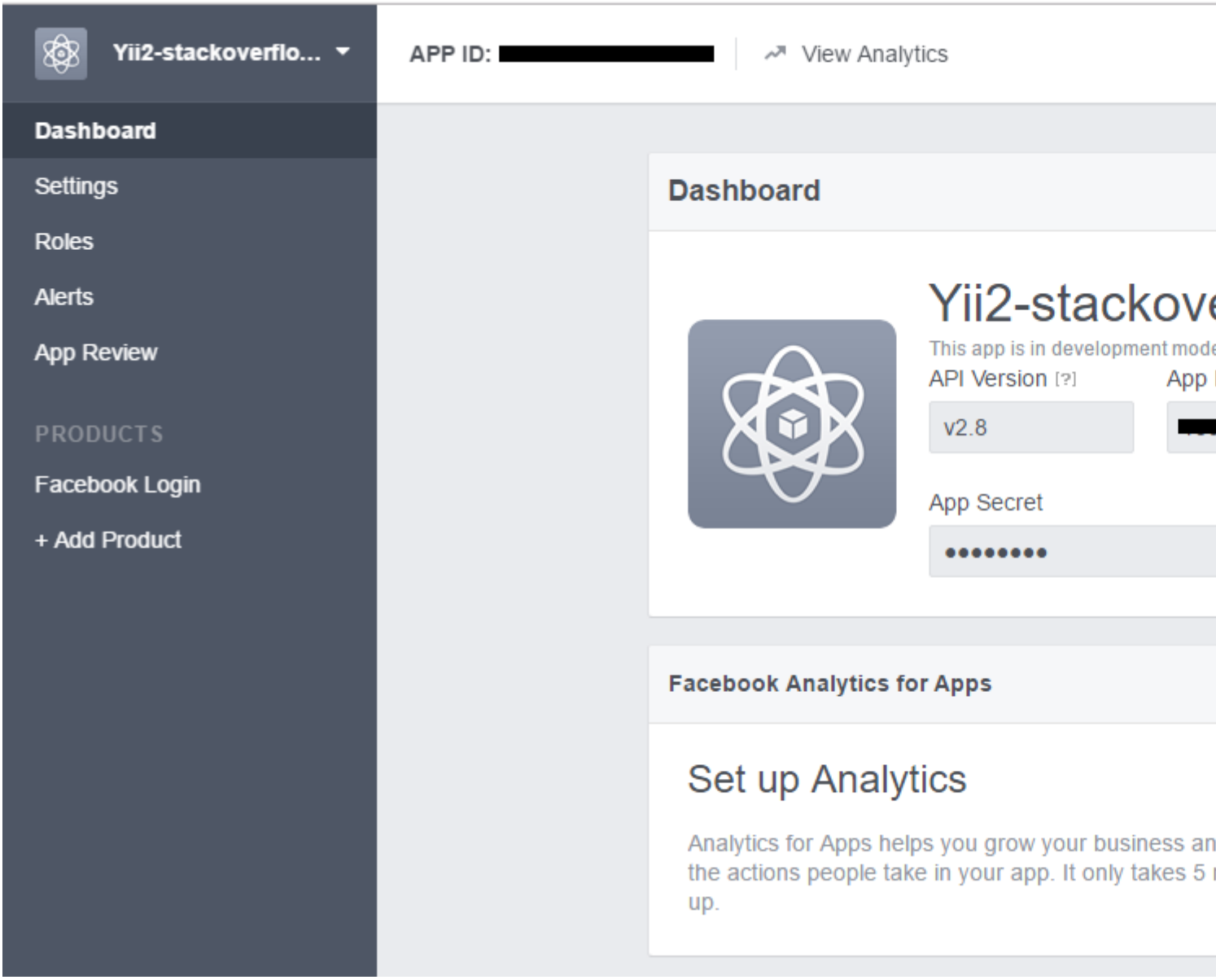
Yii2 Jquery-Kalender für Textfeld online lesen: https://riptutorial.com/de/yii2/topic/6366/yii2-jquery-kalender-fur-textfeld

Kapitel 21: Yii2 OAuth2 - Ex: Verbraucher facebook OAuth2

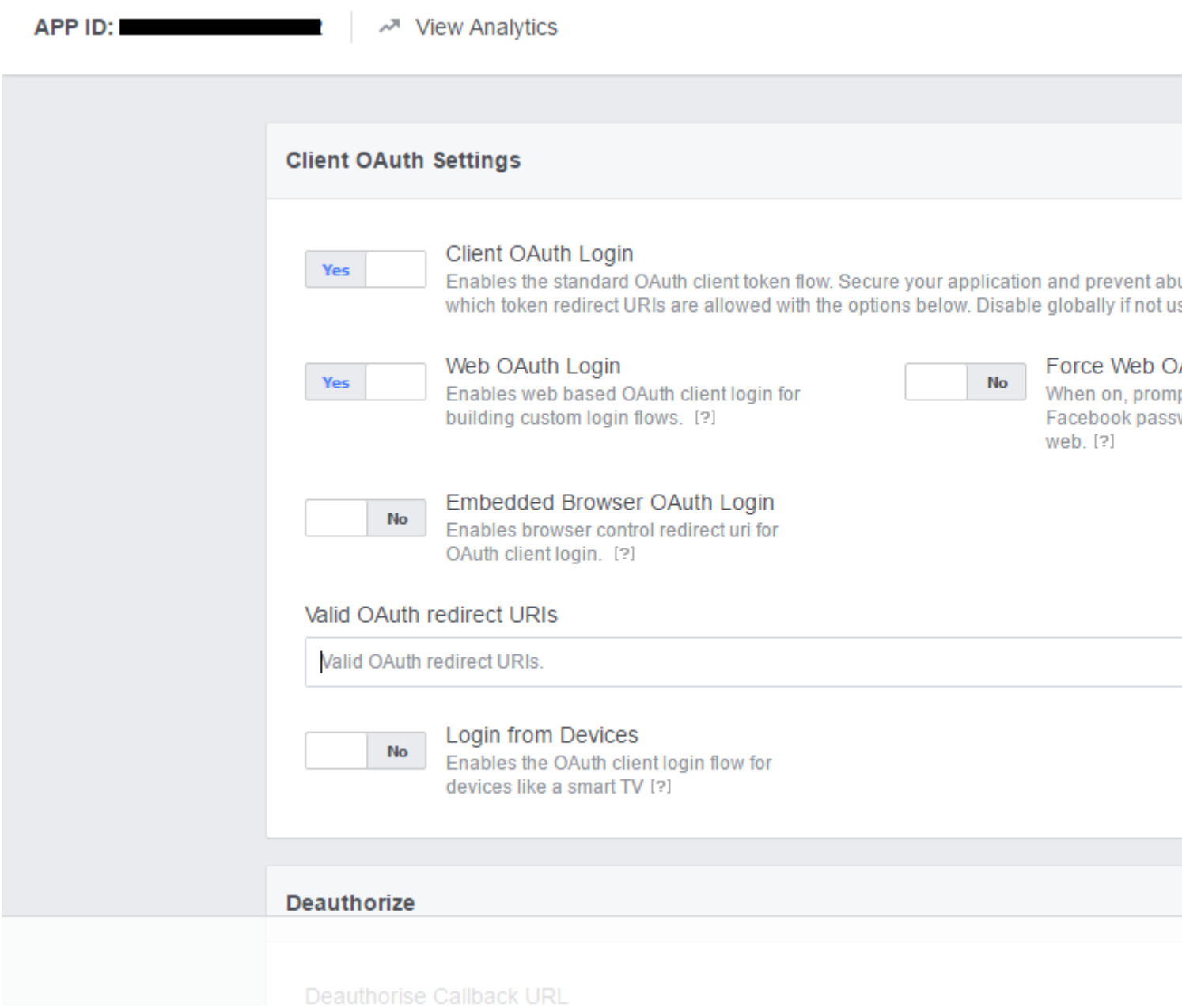
Examples

Erstellen Sie eine App für Facebook-Entwickler

Gehen Sie zu [https://developers.facebook.com/(((ttps://developers.facebook.com/) und erstellen Sie Ihre App.



Klicken `Add product` und wählen Sie `Facebook Login`



Installieren Sie den yii2-authclient

Bevor Sie diese Erweiterung installieren, müssen Sie yii2-app installieren. In diesem Beispiel verwende ich die Vorlage yii2-basic. Anleitung zur Installation hier .

Lauf

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

oder hinzufügen

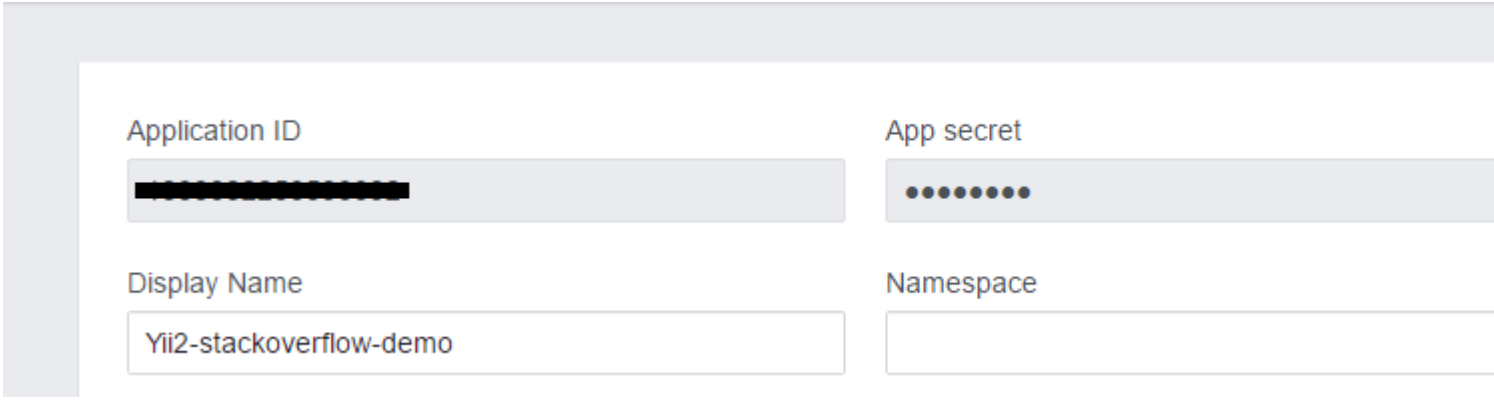
```
"yiisoft/yii2-authclient": "~2.1.0"
```

zum `require` Abschnitt Ihres `composer.json` .

In Config `authClientCollection` zu Ihren Konfigurations `components` :

```
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` ist Anwendungs - ID und `facebook_client_secret` ist app Geheimnis.



Fügen Sie eine Auth-Aktion hinzu und richten Sie einen Rückruf ein

1. Schaltfläche `Login as facebook account` für Ihre Login-Ansicht an:

Bearbeiten Sie `site/login.php` im Ordner "views" und fügen Sie diese Zeile zum Inhalt der Seitenanmeldung hinzu:

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

Oben setzen wir, dass die `auth` Aktion in `SiteController` den OAuth2-Fluss behandelt.

Jetzt schaffen wir es.

```
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
    }
}
```

Wir verwenden `yii\authclient\AuthAction` um eine URL zu erstellen und zur Facebook-Login-Seite umzuleiten.

Funktion `onAuthSuccess` die zum `onAuthSuccess` von Benutzerinformationen verwendet wird, `onAuthSuccess` bei Ihrer App an.

Füge redirect_url zur Facebook-App-Einstellung hinzu

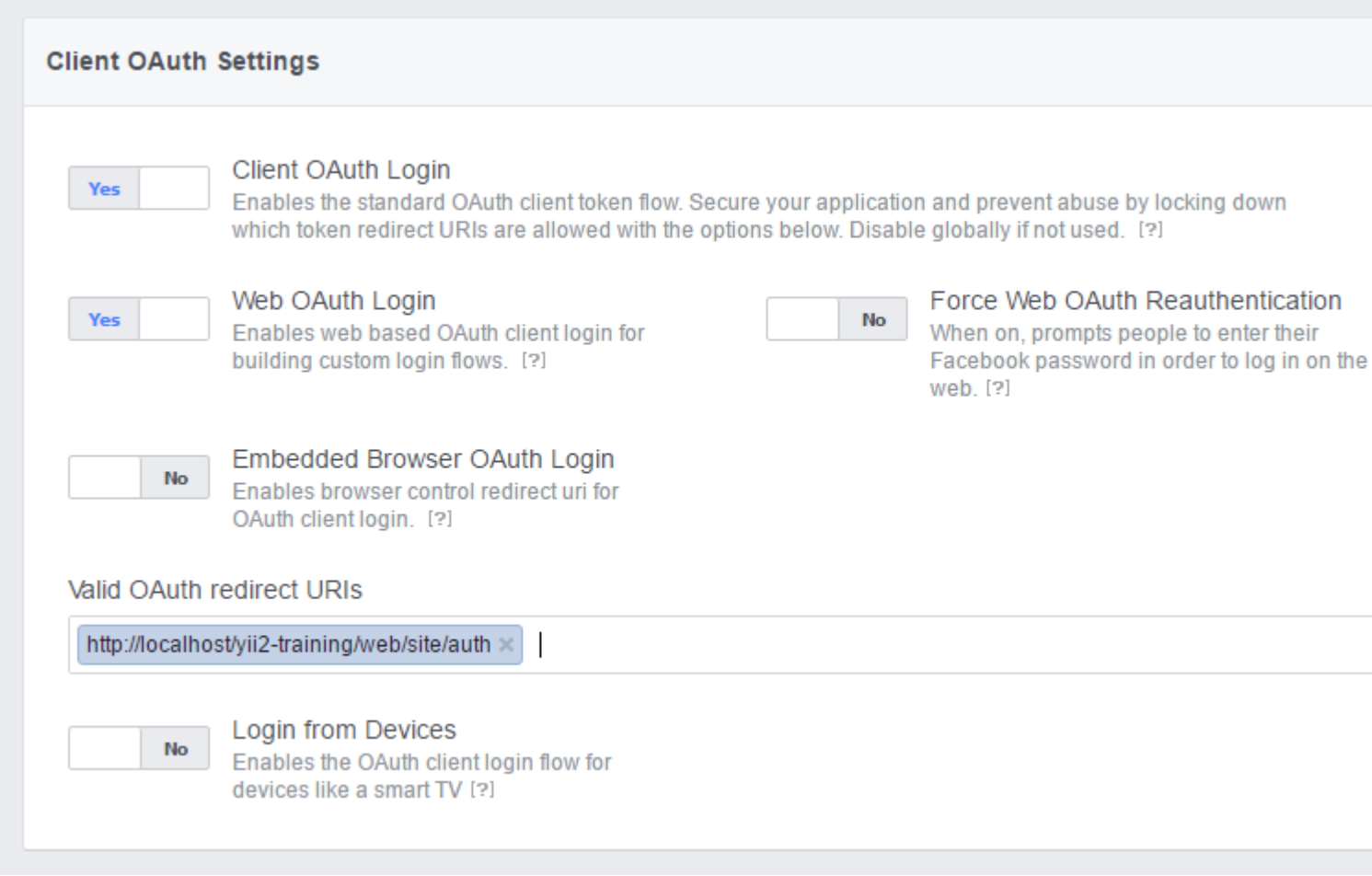
Wenn Sie prettyUrl in Ihrer yii2-App aktivieren, lautet Ihre Weiterleitung:

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

Und deaktiviere die hübsche URL:

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

Beispiel:



Beispiel für die Funktion 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);
}
```

Yii2 OAuth2 - Ex: Verbraucher facebook OAuth2 online lesen:
https://riptutorial.com/de/yii2/topic/7428/yii2-oauth2---ex--verbraucher-facebook-oauth2

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit yii2	Bibek Lekhak, Community, Farcaller, jagsler, Mohan Rex, Muhammad Shahzad, Pasha Rumkin, Sam Dark, urmaul, Yasar Arafath, Yasin Patel, Yatin Mistry
2	Ajax-Anfrage	Anton Rybalko, Ilyas karim, meysam
3	Aktiver Rekord	Insane Skull, Manikandan S, Michael St Clair, Mike Artemiev, particleflux, saada, Sam Dark, Yasar Arafath, Yasin Patel
4	Anlagenmanagement	Thomas Rohde
5	Benutzerdefinierte Validierungen	Ejaz Karim
6	Datei-Uploads	Sam Dark
7	Datenbankmigrationen	Ali MasudianPour, jlapoutre, Sam Dark
8	Erweiterte Projektvorlage	mnoronha, Mohan Rex, Salem Ouerdani, Sam Dark, topher
9	Erweiterung manuell installieren	Insane Skull, mnoronha, Sam Dark, vishuB
10	Kekse	IStranger, Sam Dark
11	Komponenten	Sam Dark, Yasin Patel
12	Mit Datenbanken arbeiten	jagsler, Kiran Muralee
13	Pjax	gmc, Manikandan S, Muaaz Rafi, Shaig Khaligli, yafater, Yasin Patel
14	Restful API	jagsler, jlapoutre, yafater, Yasin Patel
15	Routing und URLs	IStranger, Ярослав Гойса
16	Session	Brett, Goke Obasa, jlapoutre, MikelG, Yasin Patel
17	Testen	Antonín Slejška, Bizley, Sam Dark, XzAeRo
18	Validierung	jagsler, Mihai P., Nana Partykar, Sam Dark, Yasin Patel

19	Yii2 ActiveForm	Hina Vaja, Manikandan S, particleflux
20	Yii2 Jquery-Kalender für Textfeld	Manikandan S
21	Yii2 OAuth2 - Ex: Verbraucher facebook OAuth2	ThanhPV