Yii2, PHP, Web

Projekt: Einstieg PHP

Inhalt

Native PHP basics

Zu Beginn möchte ich hier ganz kurz die absoluten Basics von PHP nochmals notieren.

Ein PHP Skript beginnt mit *<?php* und endet mit *?>*. PHP bedient sich grösstenteils an der gewöhnlichen C-Syntax und ist daher schnell zu erlernen.

Keywords

Die Keywords sind nicht Case Sensitive. Das wichtigste Keyword ist «Echo», es wird benutzt um eine Ausgabe zu machen. Die meisten anderen Schlüsselwörter sind die üblichen verdächtigen: Break, return, if/else, for, foreach, try, ...
Auch «and» und «or» können sehr hilfreich sein.

Variablen

Variablen werden mit «$» deklariert. Hier taucht ein grosser Unterschied zu meinen Gewohnheiten auf: Es muss kein Datentyp angegeben werden.
Die Namen der Variablen sind Case-Sensitive.
Konstanten können mit dem Schlüsselwort «Const» erstellt werden.

Codebeispiel

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Statischer/Dynamischer Inhalt

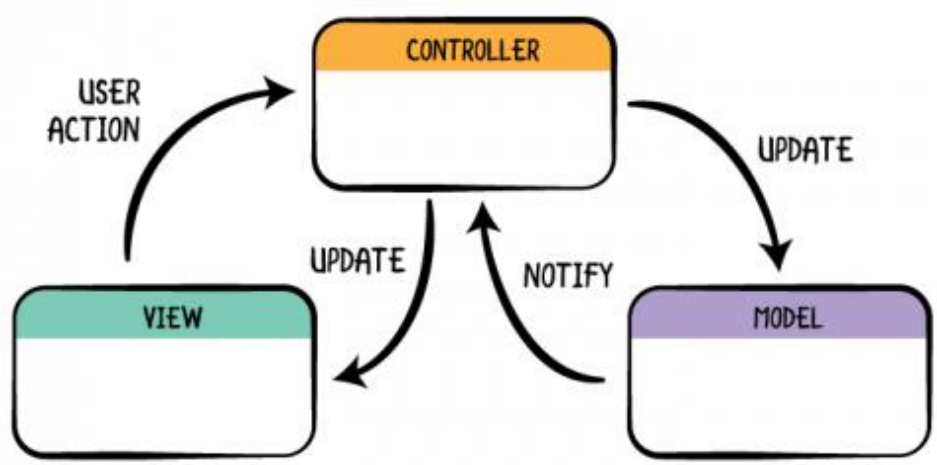
Statische Webseiten zeigen immer denselben, vom Ersteller definierten Inhalt. Es gibt natürlich Medien wie Videos und Bilder, es sind allerdings immer dieselben. Um etwas anderes darzustellen, muss der Source Code verändert werden. Dynamische Inhalte sind fähig, verschiedene Inhalte für verschiedene User zu kreieren. Zum Beispiel werden ja auf Youtube nicht immer dieselben Videos angezeigt. Der Inhalt der Startseite/Abobox o. Ä. verändert sich stetig, wenn z.B. ein neues Video erscheint. Solche dynamische Inhalte können mit PHP oder JavaScript erzielt werden. Oft werden auch Frameworks wie Ruby verwendet.

MVC

MVC ist ein Muster zur Auftrennung von Webapplikationen in Programmsteuerung, Datenmodell (Datenbanken) und Präsentation. Im Yii Framework sieht das folgendermassen aus.

Programmsteuerung	Datenmodell	Präsentation
Controller	Model	View
Sämtliche Funktionen der Applikation	Enthält die Daten, mit denen gearbeitet wird	Alles, was für den Benutzer sichtbar ist.

Zum Datenmodell käme im Falle noch eine Datenbank hinzu.



Quelle: https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach

User Action

Jegliche Funktionen werden in Controllern ausgeführt/programmiert. Es gibt mehrere Controller in einer Applikation. Ein Controller ist auch dafür verantwortlich, neue Views zu laden. Dieses Beispiel wird SiteController genannt.

Codebeispiel (SiteController)

```php
<?php

namespace app\controllers;

use …

class SiteController extends Controller
{
    /**
     * Displays homepage.
     *
     * @return string
     */
    public function actionIndex()
    {
        return $this->render('index');
    }
```

Actions

Ein Controller besteht aus Actions. Actions sind die kleinste Einheit, die man ansteuern und ausführen kann. Man erstellt sie, in dem man eine **Actionmethode** erstellt.

Requests

Wenn der User einen Input gibt, findet ein **Request** statt. Dieser wird von Yii folgendermassen bearbeitet:

1. Request an das Entry Script (web/index.php)
2. Instanz der Applikation wird erstellt
3. Route wird aufgelöst.
4. Controller Instanz wird erstellt um den Request zu behandeln.
5. Action Instanz wird erstellt und die Filter (Regeln) für die Action werden angewandt.
6. Wenn ein Filter einen Error ausgibt, wird die Action abgebrochen.
7. Wenn kein Fehler ensteht, wird die Action ausgeführt.
8. Das Model wird geladen, wenn vorhanden mit einer Datenbank.
9. Die View wird gerendert und mit den Daten vom Model versorgt. Es wird HTML geschrieben.
10. Das Resultat wird an den Browser des Users gesendet.

Erkenntnis

PHP generiert HTML Code, der dann vom Browser angezeigt werden kann. Jedes Mal, wenn der User einen Request auslöst, wird das HTML neu gerendert.

Arbeiten mit Daten

Models verwendet man, um Daten zu speichern und mit ihnen zu arbeiten. Eine Model Klasse repräsentiert die Daten in Form von Attributen, die wie Properties zu verwenden sind. Auch Arrays sind verwendbar.

Codebeispiel

```php
<?php

namespace app\models;

use Yii;
use yii\base\Model;

/**
 * ContactForm is the model behind the contact form.
 */
class ContactForm extends Model
{
    public $name;
    public $email;
    public $subject;
    public $body;
    public $verifyCode;
```

Validatoren

Validatoren sind Teil der Sicherheitsmassnahmen in Yii. Sie überprüfen einen User Input nach selbst bestimmten Regeln. Diese müssen definiert werden. Das fehlen der Validatoren ist eine häufige Fehlerquelle, weshalb etwas nicht funktioniert. Die Regeln werden mittels der Rules-methode konfiguriert.

Codebeispiel

```
public function rules()
{
   return [
      // username, email and password are all required in "register" scenario
      [['username', 'email', 'password'], 'required', 'on' =>
self::SCENARIO_REGISTER],

      // username and password are required in "login" scenario
      [['username', 'password'], 'required', 'on' => self::SCENARIO_LOGIN],
   ];
}
```

Views

Wie oben erklärt befindet sich alles für den User sichtbare in der View. Hier kann man die Webseite designen, Buttons, Textboxen usw. hinzufügen. (Input in der Textbox würde in das Model übertragen und im Controller weiter verarbeitet werden.)

Formulare

Um Input vom User entgegen zu nehmen verwendet man Forms. Man benutzt dazu das Widget «ActiveForm». (Widgets kann man sich wie Bausteine vorstellen, die von Yii zur verfügung gestellt werden.)

Codebeispiel

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

$form = ActiveForm::begin([
   'id' => 'login-form',
   'options' => ['class' => 'form-horizontal'],
]) ?>
<?= $form->field($model, 'username') ?>
<?= $form->field($model, 'password')->passwordInput() ?>

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

HTML Helper

Möchte ich in meinem Yii Projekt in einem PHP Skript einfach ganz normale HTML tags generieren, kann ich das mittels HTML Helper tun. Es handelt sich hierbei um eine Helferklasse. Im folgenden Beispiel wird mittels Tag-Methode ein Paragraph mit dem Text 'Hallo!' ausgegeben:

```php
<?php
Html::tag('p', 'Hallo!');
?>
```