Programmieren mit CakePHP

Dies hier wird keine Beschreibung, wie man gut strukturiert mit CakePHP eine Webanwendung programmiert. Hier geht es einzig darum, wie schnell ein Fehler behoben werden oder eine Funktionalität modifiziert werden kann.

CakePHP

… ist ein Framework zur Entwicklung datenbankgestützer Webanwendungen mit PHP. Bei TEMA sind das Jobsystem und die Onlineregistrierungen (Kingcon) mit CakePHP realisiert. Die Kingcons nutzen CakePHP 1.2.XXX, das Jobsystem und die NdU-Backends 1.3.YYY. Die Unterschiede sind minimal, aber es kann schon mal passieren, dass es eine Funktion nur in einem der Cakes gibt.

Wenn es Fragen zur Syntax oder zu Best-Practice gibt, hilft meist googeln. Die erste Quelle ist http://book.cakephp.org/1.2/ (gibt's auch für 1.3)

Model-Controller-View

Dies sind die zentralen Begriffe einer jeden Cake-Anwendung.

Das Model definiert die Schnittstelle zwischen MySQL-Datenbank-Tabellen und der Cake-Anwendung.

Der Controller liefert die zentrale Logik, er nutzt Methoden des Models (z.B. read(), find(), save(), validate()) und rendert einen View.

Der View bringt das Ergebnis dann an den Bildschirm. (Oder als Excel oder PDF zum Download.)

CakePHP ist objektorientiert

Aber das muss niemanden schrecken. Um CakePHP nutzen zu können, reicht es, die Bedeutung der Variabe **`$this`** zu verstehen und die Dereferenzierung mit dem Pfeil, z.B. **`$this->Model->read()`**. Erst wenn man komplexe Aufgaben lösen will und eigene Helpers oder Components dafür entwickelt, muss man die Objekthierarchie verstehen.

Model-Controller-View, die 2.

Wenn jemand zu euch kommt, weil etwas in Kingcon, im NdU-Backend oder im Jobsystem nicht funktioniert, fragt als erstes nach der URL!

Die URL einer Cake-Anwendung ist wie folgt aufgebaut:

http://Webservername/evtl.-ein-Unterverzeichnis/Controller/View/Parameter

http://jobsys.tema-dev.de/auftrags/edit/1111
- Webserver: jobsys.tema-dev.de
- Controller: auftrags
- View: edit
- der Rest sind Parameter: 1111

http://anmeldung.nachtderunternehmen.de/aachen/firmas/add_firm
- Webserver: anmeldung.nachtderunternehmen.de
- Unterverzeichnis: aachen
- Controller: firmas
- View: add_firm

http://www.gma2012.de/register/en/people/ShowSpecial/Ticket.gruppe/Tageskarte
- Webserver: www.gma2012.de
- Unterverzeichnis: register (weil CakePHP genau wie TYPO3 alle Pfade verarbeiten würde)
- Sprachsteuerkürzel: en

Controller: people
View: ShowSpecial
1. Parameter: Ticket.gruppe
2. Parameter: Tageskarte

Model-Controller-View, die 3.

Nachdem aus der URL Controller und View bestimmt wurden, gilt es, die entsprechenden Stellen auf dem Webserver zu finden.

Der Zugang auf den Webserver erfolgt ganz normal per FTP (oder WebDAV)). Ist es eine Subdomain (z.B. Jobsystem oder NdU-Backend), muss man dorthin wechseln; gibt es ein Unterverzeichnis, muss man auch in dieses wechseln. Gibt es kein Unterverzeichnis (z.B. Jobsystem), so muss man in das Verzeichnis ‚app' wechseln.

Auf dieser Ebene muss es die Order ‚model', ‚controller' und ‚views' sowie einige andere (z.B. ‚config', ‚locale', ‚webroot')geben.

Im Ordner ‚config' gibt es die Datei ‚database.php', die definiert, mit welcher Datenbank Kontakt aufgenommen wird. (In diesem Ordner gibt es auch die Datei ‚routes.php', die die Zuordnung von Sprachkürzeln in der URL zum Lang-Parameter regelt. Des weiteren die Datei ‚core.php', in der man den `Configure::write('debug', 0);` mal kurzfristig auf 2 setzen kann fürs Debugging.)

Jetzt kommen wir zu den CakePHP-Konventionen. (Die leider bei einigen Kingcons nicht eingehalten wurden – böse Falle!) – Tabellen in der Datenbank werden in der Mehrzahl benannt. Also z.B. ‚people', ‚tickets', ‚firmas'. Das Model, das auf diese Tabelle zugreift wird im Singular benannt und groß geschrieben; dabei berücksichtigt Cake die englische Grammatik. Also z.B. als Model: ‚Person', ‚Ticket', ‚Firma'.

Die Controller, die mit diesem Model arbeiten werden wieder im Plural benannt, in CamelCase-Schreibweise. Im Namen der Datei und im Namen des PHP-Objects ist das ‚Controller' mit enthalten. Wenn ein Controller ein anderes Model nutzen will, als sein Name suggeriert (oder z.B. mehrere Models), kann man mit der Variable ‚$uses' diese festlegen.

Im Controller werden Variablen definiert und Funktionen. Im Sinne der OO-Programmierung sind es Methoden. In CakePHP spricht man von ‚Action'. In jeder Action wird ganz gewöhnliche PHP-Logik programmiert und am Ende ein View aufgerufen. In der Regel heißt der View so, wie die Action, aber mit der Methode `$this->render('anderer_view')` kann man das ändern.

Die Views liegen im Ordner ‚views' und sind dort nach Controllern sortiert. Im Ordner ‚views' gibt es zwei spezielle Unterordner: ‚layouts' und ‚elements'. Es ist möglich, im View mit `$this->element()` oder `$this->renderElement()` ein „Miniview" aus dem Elementordner einzubinden. (Es lebe die Modularität!). Zu jeder View-Anzeige gehört auch ein Layout; ist im Controller nichts angegeben, so wird das Layout ‚default' benutzt. Die Kingcons nutzen den Layout-Mechanismus, um den http-Header zu setzen und z.B. XLS oder PDF zu generieren.

Model

Wenn ein bestehendes Kingcon plötzlich Macken hat, liegt es eigentlich nie am Model.

Im Model kann man sehr detailliert die Zuordnung zu einer Datenbanktabelle beschreiben, falls man die Cake-Konventionen verletzen will.

Im Model kann man beforeSave und afterFind-Callbacks definieren.

Im Model werden Validierungsregeln festgelegt – ein sehr mächtiges Werkzeug, weil es z.B. viele fertige Regeln gibt; z.B. was eine Emailadresse ist – natürlich ist es eine serverseitige Validierung.

Im Model werden Verknüpfungen zu anderen Models definiert, so dass ich z.B. beim Find-Aufruf einer Person auch alle Tickets, Papers, usw. zu dieser Person bekommen kann.

Geradezu genial ist die Methode ‚saveAll', die Datensätze in mehrere Tabellen schreibt und beim Neuanlegen sogar den Fremdschlüssel in die abhängigen Tabellen einträgt.

Action

Das ist die Logik der CakePHP-Anwendung. Ehrlicherweise muss man sagen, dass die URL nicht aus Controller und View, sondern aus Controller und Action besteht. Die Parameter, die in der URL /-getrennt nach der Action folgen, werden der Action als Positionsparameter mitgegeben.

Also z.B. URL: **`http://www.gma2012.de/register/en/people/invoice/7/3/true/de`**

Und die Funktionsdeklaration der Action ‚invoice' lautet:

```
function invoice($id=null, $re_id=null, $send_to_browser="true", $language=null)
```

Neben Positionsparametern gibt es in Cake auch ‚named parameters', die aber in Kingcon kaum genutzt werden. (Außer der Parameter ‚lang' für die Sprache. Im Jobsystem wird es intensiver genutzt.)

In den Actions erfolgt ganz normale PHP-Programmiererei.

Im Controller sind Methoden des Models verfügbar; also z.B.: **`$this->Person->find(…)`** oder **`$this->Person->save(…)`**. Ist die Model-Verknüpfung definiert, geht auch **`$this->Person->Ticket->save(…)`**, um z.B. nur Ticketdaten zu ändern.

Es gibt eine Reihe von Components, die man im Controller bekannt machen und dann nutzen kann; - häufig anzutreffen ist die Email-Komponente.

Neben der ‚normalen' Variante, die Action über eine URL aufzurufen, erlaubt es Cake auch per **`$this->requestAction()`** eine Action auszuführen. Es mag sinnvolle Anwendungen dafür geben, im Fall von Kingcon scheint es mir ein Zeichen schlechter Strukturiertheit zu sein. (Erlaubt ist, was geht!)

Was ebenfalls geht, aber CakePHP ad absurdum führt: – in den Actions oder gar im View mit ‚mysql_query' Daten aus einer bestimmten Tabelle zu holen oder zu schreiben. Pfui!!

View

Die Views (und Elements und Layouts) in CakePHP haben die Endung .ctp oder .thtml. Im Prinzip sind es gewöhnliche PHP-Dateien. Es gibt einige sehr praktische Helper zum Bauen der Views: den HTML-Helper für Links, Images u.ä. und den Form-Helper für alle Arten von Formularfeldern. In der Syntax unterscheiden sich diese Helper bei CakePHP1.2 und 1.3 gelegentlich. Das kann schon mal ein Grund für eine weiße Seite sein, dass man einen Helper-Aufruf im View geändert hat und es diesen Parameter in dieser Cake-Version so nicht gibt. – Hier gibt's die Infos: http://api12.cakephp.org/classes

Alle CakePHP-Dateien sollten als UTF8-without-BOM gespeichert werden. Solange die Datei keine Umlaute enthält, ist es nicht wesentlich. Es gibt einige ältere Kingcons, da müssen die Views, die PDF-Dateien erzeugen (Invoice, Badge, Umschlag), als ANSI gespeichert werden, weil sonst die PDF-Bibliothek streikt.

Sprachumschaltung

CakePHP nutzt die gettxt-Sprachverwaltung. D.h. Textübersetzungen findet man im Ordner ‚locale' … bis zur Datei default.po. Cake nutzt dafür die Doppelunterstrich-Funktion. Wird der Text in der Übersetzungsdatei gefunden, wird die Übersetzung genutzt, sonst der Text. Wird die Funktion ohne zweiten Parameter=true genutzt, führt sie auch gleich noch das ‚echo' aus. (Bzw. andersherum: will ich nur ein String übersetzt haben, um ihn dann weiter zu verarbeiten, muss ich den zweiten Parameter auf true oder 1 setzen.)

Die Sprachumschaltung erfordert einiges an Initialisierung, die in den Kingcons im app_controller erfolgt (das ist die Datei bzw. das Object, von der jeder konkrete Controller abgeleitet wird); bitte fragt mich nicht nach Einzelheiten. Ich bin froh, dass es funktioniert.

auth-Component

CakePHP bietet mit der auth-Komponente ein sehr mächtiges Werkzeug zur Benutzeranmeldung. Die Zugangsdaten liegen in einer Datenbanktabelle; die Zugangskontrolle erfolgt über Sessionvariablen. Die Komponente wird in fast allen Kingcons genutzt. Wenn es daran etwas zu ändern gibt, dann ist in der üblichen Weise zu verfahren: Modifizieren in Analogie und Testen. Entweder klappt es oder man muss doch im Online-Manual nachlesen.

Array `$this->data`

Mit diesem in CakePHP immer vorhandenen Array kann man Daten zwischen dem View und dem Controller austauschen.

Wurden die Formularfelder im View mit dem Form-Helper „gebaut“, so erhält man alle Formulardaten im Controller im $this-data-Array (Wenn man sich den Quelltext des fertig gerenderten Views anschaut, versteht man auch warum.) Kein Hantieren mit $_POST-Variablen nötig!

Werden die Daten im Controller im $this->data-Array gespeichert, so sorgt der Form-Helper im View für die richtige Zuordnung zu den Formularfeldern, Also im Controller in der Action z.B.

```
$this->data=$this->Person->read();
```

und schon werden im View nicht nur die Textfelder, sondern auch Radio-, Check- und Selectboxes mit den Werten angezeigt.

Daten aus dem Controller an das View werden mit der Controllermethode set() übergeben. Z.B. `$this->set('varname', $Zeugs_im_Controller);` macht im View eine Variable ‚$varname’ bekannt und füllt sie mit Daten. Im View greift man wie auf eine lokale Variable darauf zu.

Der Form-Helper hilft auch beim Ausfüllen der Selectboxes – eine Hilfestellung, die bei Kingcon bisher nicht, dafür aber intensiv beim Jobsystem genutzt wird. (Prinzip: Im Formular gibt es ein Feld, das als Selectbox definiert wurde namens ‚abteilung_id’; vom Controller an das View wird ein Array ‚abteilungs’ mit den zwei Feldern ‚id’ und ‚name’ übergeben. Dann baut der Form-Helper daraus die Optionen für die Selectbox und aus $this->data wählt es den aktiven Wert. Fertig! Convention over Configuration – das Credo von CakePHP!)