



CakePHP 1.2

ContainableBehavior

1 Einleitung

CakePHP verbindet über das Modell verschiedene Tabellen miteinander. Dadurch wird es dem Anwender sehr leicht gemacht, verknüpfte Tabellenabfragen und Pflegeviews zu erstellen. Doch jede Medaille hat zwei Seiten: durch die automatische Verknüpfung entstehen teilweise sehr komplexe Tabellenabfragen. Die Laufzeit erhöht sich enorm da nicht nur durch die Abfrage auf die Tabellen – sondern auch durch die Verarbeitung des Ergebnisses. Denn dadurch, dass die Tabellen mit einem Left-Join verknüpft werden, entstehen zahlreiche NULL-Einträge – die für das Ergebnis nicht interessant sind. Sie werden zwar reduziert durch das Kommando GROUP BY – doch die Datenbank wird dadurch unnötig belastet.

Unter CakePHP 1.1 konnte man durch unbind diese Verbindung im Controller lösen. Es gab auch verschiedene Ansätze dieses zu erleichtern (es sind noch weitere Kommandos notwendig). Unter CakePHP 1.2 gibt es jetzt eine neuen Zusatz: ContainableBehavior. Dieses Modell „Verhalten“ ermöglicht es die Operationen im find-Befehl einzuschränken und zu filtern.

Hinweis: ContainableBehavior funktioniert erst ab RC2! Die Beispiele wurden mit CakePHP 1.2.0.7692 RC3 erstellt.

1.1 Normales Verhalten

Wenn eine Abfrage find('all') ausgeführt wird, werden alle im Modell eingebundenen Tabellen (hasMany, hasOne) und alle Felder der Tabellen im SELECT berücksichtigt.

Nehmen wir an, wir haben eine Tabelle immobilien auf die wir eine Abfrage machen, um alle Einträge zu selektieren. Im Modell haben wir folgende Tabellen (Modelle) mit eingebunden:

- Anhaenge (1:n, enthält die Bilder)
- Freitexte (1:n, enthält zusätzliche Texte)
- immobilienart (1:1 Beziehung der Immobilienart > Texttabelle)

Wenn wir nun eine Abfrage find('all') ausführen, erhalten wir ein mehrdimensionales Array als Ergebnis das in etwa wie folgt aussieht – für eine Zeile (Hinweis: Um die Darstellung nicht zu ausschweifend zu gestalten, wurden zahlreiche Felder entfernt)!

```
Array
(
    [0] => Array
        (
            [Immobilien] => Array
                (
                    [objektnummer] => {0F57FA1E-76CC-4806-B317-A6327BE4E848}
                    [immobilienart] => 1
                    [maklerobjnummer] => 6708
                    [modified] =>
                    [plz] => 67281
                    [iso_land] => DEU
                    [mietekauf] => 2
                    [stichwort] => Dachgeschosswohnung mit großer Terrasse
                    [gesamtpreis] => 0.00
                    [warmmiete] => 450.00
                    [aussen_courtage] => 2MM zuzüglich MwSt.
                    [inkl_mwst] => true
```



```

        [iso_waehrung] => EUR
    )

    [Immobilienart] => Array
    (
        [id] => 1
        [infotext] => Wohnungen
    )

    [Anhaenge] => Array
    (
        [objektnummer] =>
        [nummer] =>
        [gruppe] =>
        [anhangtitel] =>
        [format] =>
        [daten] =>
        [pfad] =>
    )

    [Freitexte] => Array
    (
        [objektnummer] => {0F57FA1E-76CC-4806-B317-A6327BE4E848}
        [textname] => dreizeiler
        [freitext] => Dachgeschosswohnung mit großer Dachterrasse!
    )
)

```

In meinem Beispiel habe ich die Abfrage auf eine bestimmte Immobilienart eingeschränkt (Wohnungen). Wenn ich auf den Befehl GROUP BY verzichte, erhalte ich in meinem Beispiel 36 Zeilen – anstatt sechs Zeilen d.h. die Selektion unter Berücksichtigung der vier verbundenen Tabellen ergibt das sechsfache an Zeilen, das durch das GROUP BY durch die Datenbank erst einmal verdichtet werden muss.

ContainableBehavior hilft nun, das Ergebnis auf das zu reduzieren, was wir tatsächlich benötigen. In der Abfrage in der ich die Objekte als Liste darstellen möchte, benötige ich nur die Ergebnisse der Tabellen immobilien und anhaenge.

Um das neue Verhalten zu verwenden, können Sie es in die \$actsAs Eigenschaft im Modell einfügen:

```

class Immobilien extends AppModel {
    var $actsAs = array('Containable');
}

```

Genauso kann das Verhalten „on the fly“ im Controller eingebunden werden:

```

$this->Immobilien->Behaviors->attach('Containable');

```

In meiner Index-Funktion habe ich einfach ein find('all') definiert – ohne Pagination.

Um nun die Abfrage so einzuschränken, dass nur die Tabelle *anhaenge* berücksichtigt wird, genügt folgendes Kommando:



```
$this->Immobilien->contain('Anhaenge');
```

Oder, um alle anderen Tabellen bzw. Modelle auszuschließen:

```
$this->Immobilien->contain();
```

So sieht dann die komplette Funktion aus:

```
function index() {

    $this->Immobilien->contain();

    $conditions = array();
    $eintraege = $this->Immobilien->find('all', array(
        'conditions' => $conditions,
        'group' => array('Immobilien.objektnummer')
    ));
    $this->set('Immobilien', $eintraege);
}
```

Das Ergebnis ist wie erwartet, es wird nur noch die Tabelle *immobilien* berücksichtigt.

Doch das ist nicht alles. Wir können auch noch einschränken, welche Felder gelesen werden sollen:

```
$this->Immobilien->contain('Anhaenge.objektnummer', 'Anhaenge.pfad');
```

Hinweis: Versuchen Sie nicht die zu selektierenden Felder des aufgerufenen Modells auf diesen Weg zu definieren – es funktioniert nicht:

Warning (512): Model "Immobilien" is not associated with model "Immobilien" [CORE\cake\libs\model\behaviors\containable.php, line 342]

Sie können auch direkt im ein find('all') die einzubindenden Tabellen und Felder mitgeben:

```
function index() {

    $conditions = array();
    $eintraege = $this->Immobilien->find('all', array(
        'conditions' => $conditions,
        'containt' => array(
            'Anhaenge' => array('Anhaenge.objektnummer', 'Anhaenge.pfad'),
            'Freitexte' => array('Freitexte.textname')
        ),
        'group' => array('Immobilien.objektnummer')
    ));
    $this->set('Immobilien', $eintraege);
}
```

1.2 CustomBehavior und Pagination

Die oben dargestellten Varianten funktionieren nur in Verbindung mit der Funktion find. Wenn mit Pagination gearbeitet wird, funktioniert das CustomBehavior etwas anders.

Entweder werden die einzubindenden Tabellen und Felder wie folgt angegeben:



```
$this->paginate['Immobilien']['contain'] = array (
    'Anhaenge' => array (
        'fields' => array ('objektnummer','pfad','anhangtitel')
    )
);
```

Oder direkt in der paginate-Funktion:

```
$this->paginate['Immobilien'] = array(
    'limit' => 5,
    'conditions' => $conditions,
    'recursive' => 0,
    'contain' => array(
        'Anhaenge' => array(
            'fields' = array(
                'Anhaenge.objektnummer',
                'Anhaenge.pfad',
                'Anhaenge.gruppe',
                'Anhaenge.anhangtitel')
            )
        ),
    'group' => 'Immobilien.objektnummer');
```