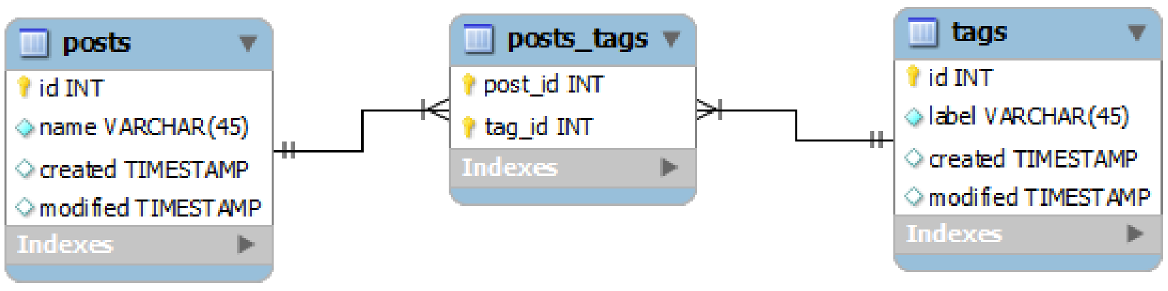
PabloLM "...todo se repite muchas veces"

Cakephp 2.x Saving and validating a HABTM relation example

Even if the documentation on this subject is quite complete, I will write an example of how to do it including a control witch validates the number of elements saved.

First I'll give you only the code and after that (for those who are interested) I'll give some notes about it. Or you could download/pull the github repository to test it. I've also added some unit test that I'll use for new versions of cakephp. Please do clone this code and feel free to fork, repair, etc.

The database model



Read the naming conventions

The Model

Post.php

```
<?php
App::uses('AppModel', 'Model');
/**
 * Post Model
 *
 * @property Tag $Tag
 */
class Post extends AppModel {
```

```
/**
 * Display field
 *
 * @var string
 */
	public $displayField = 'name';

/**
 * Validation rules
 *
 * @var array
 */
	public $validate = array(
		'name' => array(
			'notempty' => array(
				'rule' => array('notempty'),
			),
		),
		'Tag' => array(
			'multiple' => array(
				'rule' => array('multiple', array('min' => 1)),
				'message' => 'You need to select at least one tag',
				'required' => true,
			),
		),
	);

/**
 * hasAndBelongsToMany associations
 *
 * @var array
 */
	public $hasAndBelongsToMany = array(
		'Tag' => array(
			'className' => 'Tag',
			'joinTable' => 'posts_tags',
			'foreignKey' => 'post_id',
			'associationForeignKey' => 'tag_id',
			'unique' => 'keepExisting',
		)
	);

/**
 * Transforms the data array to save the HABTM relation
 */
	public function beforeSave($options = array()){
		foreach (array_keys($this->hasAndBelongsToMany) as $model){
			if(isset($this->data[$this->name][$model])){
				$this->data[$model][$model] = $this->data[$this->name][$mo
				unset($this->data[$this->name][$model]);
			}
		}
		return true;
	}

}
```

app/Model/Post.php view raw

Tag.php

```
<?php
```

```
App::uses('AppModel', 'Model');
/**
 * Tag Model
 *
 */
class Tag extends AppModel {

/**
 * Display field
 *
 * @var string
 */
        public $displayField = 'label';

}
```

app/Model/Tag.php view raw

PostController.php

```
        public function add() {
                if ($this->request->is('post')) {
                        $this->Post->create();
                        $this->Post->validator()->remove('Tag');
                        if ($this->Post->save($this->request->data)) {
                                $this->Session->setFlash(__('The post has been saved'));
                                return $this->redirect(array('action' => 'index'));
                        } else {
                                $this->Session->setFlash(__('The post could not be saved.
                        }
                }
                $tags = $this->Post->Tag->find('list');
                $this->set(compact('tags'));
        }
```

app/Controller/PostsController.php view raw

Post/add.ctp

```
<div class="posts form">
<?php echo $this->Form->create('Post'); ?>
        <fieldset>
                <legend><?php echo __('Add Post'); ?></legend>
        <?php
                echo $this->Form->input('name');
                echo $this->Form->input('Post.Tag',array('label'=>'Tags', 'type'=>'select
        ?>
        </fieldset>
<?php echo $this->Form->end(__('Submit')); ?>
</div>
<div class="actions">
        <h3><?php echo __('Actions'); ?></h3>
        <ul>
                <li><?php echo $this->Html->link(__('List Posts'), array('action' => 'inde
        </ul>
</div>
```

app/View/Posts/add.ctp view raw

Where’s the magic?

If you look closely, you’ll find that almost everything is in the Model. The add() method on the Controller is just a simple (baked) method used to save the Post model and its associated Tags.

Basically to save a HABTM relation you need the data to be structured like this:

```
Array
        (
        [Post] => Array
                (
                        [name] => my test post
                )
        [Tag] => Array
                (
                [Tag] => Array
                        (
                                [0] => 1
                                [1] => 3
                        )
                )
        )
```

habtm_structure hosted with ❤ by **GitHub** **view raw**

(Why does the Tag array needs to be formatted like that??? I really have no idea.)

If you want to validate the number of tags that can be added to a post, you could use the “multiple validator”. But in order to be able to use the multiple validator, the data must be structured like this:

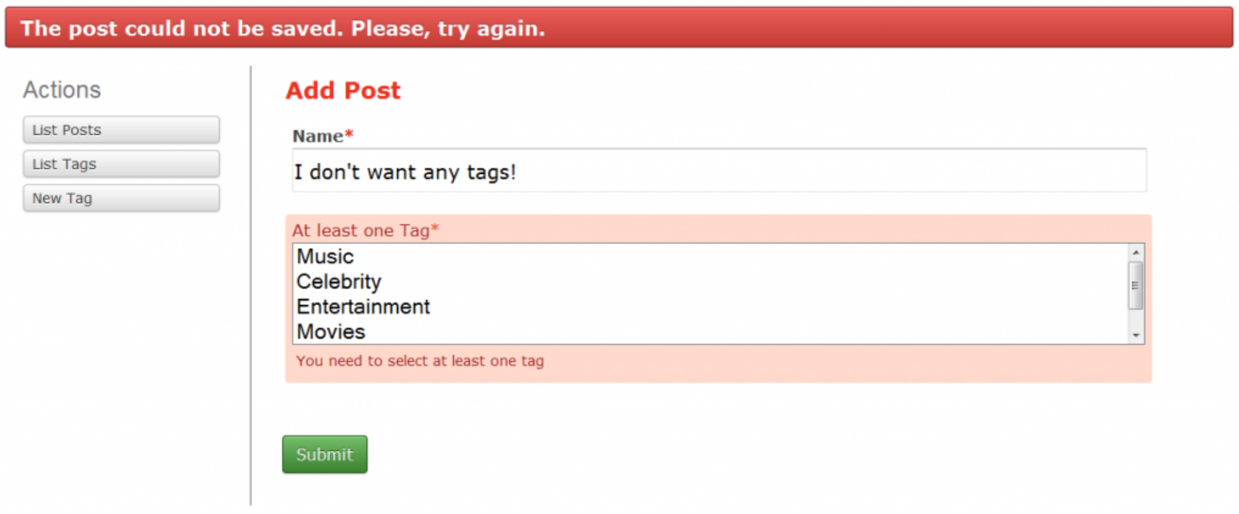
```
Array
        (
        [Post] => Array
                (
                        [name] => my other test post
                        [Tag] => Array
                                (
                                [0] => 2
                                [1] => 4
                        )
                )
        )
```

validator_structure hosted with ❤ by **GitHub** **view raw**

(Again: Why does the Tag array needs to be formatted like that??? I really have no idea.)

As you can see, the HABTM relationship needs the data in a different way that the validator, clearly it is a contradiction between the validation and the data saving. The validator needs the Tag array inside the Post, and the save method needs the Tags at the same level of the Post. Surely you could transform the data inside the controller, but where's the fun in that? Let's better use the before save callback, because maybe you have multiple HABTM relations and you want to preserve a skinny controller and a fat model. In this case I used the before save on the Post Model, but you could also define it in the AppModel so it will work for all models.

And that's it!. Now if you try to save the Post without choosing any tag, you should see something like this:


The post could not be saved. Please, try again.

Actions

List Posts

List Tags

New Tag

Add Post

Name*

I don't want any tags!

At least one Tag*

Music

Celebrity

Entertainment

Movies

You need to select at least one tag

Submit

This might not be the cleanest way, but I couldn't find a better way to do it (without using js validation of course). And the documentation doesn't say anything about this, so....

Hope this helps!

Hey! if you want to give it a try. I have created a repository so you could pull, test, correct, improve this code. Just go to the repository on github.

Posted on January 24, 2014 by pablo in Tutorial Tagged cakephp 2.x, habtm, save, validation

7 thoughts on “Cakephp 2.x Saving and validating a HABTM relation example”

Said Bakr says:
March 19, 2014 at 3:35 pm

Could you able to discuss in new article the dealing with haveMany through. i.e in your example database we just able to deal with an extra data field in posts_tags table such as `order` which is an integer used to make an order value for the tags listed per each post. For example, suppose that blogs contains just three tags, politics, sports, science. So we have a political post that may talk about science but we want to be sure that tags will be in the following order: politics, science, so we will use the order field.

We need to know how to make add or edit view that is going to be able to list all tags and next to each tag the value of the order could be inserted.

Reply

pablo says:
June 11, 2014 at 3:45 pm

I’ll try to do write about it later 😛 but for now i’m into python and JS… but it seems that HABTM relations with extra fields could give me more visits so I’ll try to write about it 😛

Reply

Laura says:
June 11, 2014 at 3:37 pm

Hi,
what if there is a dropdown with one selection, instead of multiple selection?
What will be the code in the view? I ‘ve found a lot of examples but none of them are correct..

Thank you

Laura

Reply

pablo says:
June 11, 2014 at 3:50 pm

Hi Laura,

If you have a dropdown (only one possible value selected) then probably the relation is not a HABTM but a simple HasMany relation.

If you use a HasMany relationship, you could simply `bake` a view and it will add the dropdown and all

Hope this helps

Reply

Laura says:
June 11, 2014 at 4:08 pm

Thank you for your quick reply, I appreciate it.
I have habtn relationship become a user could belong to one or many groups, and viceverse a group could be made by one or many users.
So the relation is not one to many, right?
In my case, one user has just one group associated (If a use one multiple selection with just one group selected, or a single selection with one group selected, isn't it the same thing?

Thanks again for the help.

Reply

pablo says:

June 11, 2014 at 4:20 pm

Ahh ok, so yes, in that case it is a HABTM relation. If I understood correctly, you have a HABTM but in your form you want to limit to only one single Group. So yes you can use a dropdown to do it, you'll probablly need to modify a little bit the `$this->request->data` before calling the `User->save()` method on your controller.

In the end, you'll nedd to call $this->User->save($data) where $data is something like:

```
Array

        (

        [User] => Array

                (

                        [name] => just an example

                        ....

                )

        [Group] => Array

                (

                [Group] => Array

                        (

                                [0] => 123  // your group id sent

                        )

                )

        )
```

Reply

Laura says:
June 11, 2014 at 3:52 pm

I have the same model (instead of Post I have Users and instead of Tags i have

Groups, and a dropdown list (witha single selection).
I insert User info correctly in the table of users but when I select inside the dropdown list the group name (related to the user) nothing is saved in the join table groups_users.
What am i doing wrong?
Please can you help me?
Thank you

LAura

Reply