CRUD plugin

Introduction

The Crud plugin allow high reusability of the default Create, Retrieve, Update and Delete (CRUD) actions in our controllers

Usually the forms is very simple, and always look the same - this plugin will add the actions to your controller so you don't have to reimplement them over and over

Installation

Make sure Cake loads the plugin by adding the following to your **app/Config/bootstrap.php** - make sure to include bootstrap inclucion

Installation code for Crud plugin

```
CakePlugin::load('Crud', array('bootstrap' => true));
```

In your app controller make sure to load the Crud **component**

Initiallization in the controller

```
<?php
/**
 * Application wide controller
 *
 * @abstract
 * @copyright Nodes ApS 2010-2012 <tech@nodes.dk>
 * @package App.Controller
 */
abstract class AppController extends Controller {
    /**
    * List of global controller components
    *
    * @cakephp
    * @var array
    */
    public $components = array(
        // Enable CRUD actions
        'Crud.Crud' => array(
            'actions' => array('index', 'add', 'edit', 'view', 'delete')
        )
    );
?>
```

Configuration

Router prefixes

We only use routes without prefix, "admin_" is also supported by default

In the code example above, we pass in an **actions** array with all the controller actions we wish the Crud component to handle - you can easily omit some of the actions

Example #1

```
public $components = array(
    // Enable CRUD actions
    'Crud.Crud' => array(
        'actions' => array('index', 'add', 'edit', 'view') // All actions but delete() will be
implemented
    )
);
```

In the above example, if **/delete** is called on the controller, cake will raise it's normal error as if nothing has happend

You can enable and disable Crud actions on the fly

Enable / Disable Crud actions

```
<?php
/**
 * Application wide controller
 *
 * @abstract
 * @copyright Nodes ApS 2010-2012 <tech@nodes.dk>
 * @package App.Controller
 */
abstract class AppController extends Controller {
    /**
    * List of global controller components
    *
    * @cakephp
    * @var array
    */
    public $components = array(
        // Enable CRUD actions
        'Crud.Crud' => array(
            'actions' => array('index', 'add', 'edit', 'view', 'delete')
        )
    );

    public function beforeFilter() {
        // Will ignore delete action
        $this->Crud->disableAction('delete');

        // Will process delete action again
        $this->Crud->enableAction('delete')
    }
}
```

You can also change the default view used for a Crud action

Change view file for a Crud action

```
public function beforeFilter() {
    // Do some funky shit here
    // ....
    // ....
    // ....
    // ....

    // Change the view for add crud action to be "public_add.ctp"
    $this->Crud->mapActionView('add',  'public_add');

    // Change the view for edit crud action to be "public_edit.ctp"
    $this->Crud->mapActionView('edit', 'public_edit');

    // Convenient shortcut to change both at once
    $this->Crud->mapActionView(array('add' => 'public_add', 'edit' => 'public_edit'));
}
```

Convention

The Crud component always operates on the **modelClass** of your controller, that is the first model in your **$uses** array

By default Crud component asumes your add and edit actions is identical, and will attempt to render the "form.ctp" file

There is no view for **delete** action, it will always just redirect

The callback system

Callbacks

When you bind a callback object to the Collection all callbacks below will be called on the model
If you only wish a specific callback to be called in one action, please remember to check for this with the **shouldProcess** method in the callback object

index()

- init
- beforePaginate

add():

- init
- beforeSave
- afterSave
- beforeRender

edit($id = null):

- init
- beforeSave
- afterSave
- beforeFind
- recordNotFound
- beforeRender

view($id = null):

- init

- beforeFind
- recordNotFound
- beforeRender

delete($id = null):

- init
- beforeFind
- recordNotFound
- beforeDelete
- afterDelete

Callback parameters

- init(Controller $Controller, $action) - **run parent::init($Controller, $action) first**

- beforeSave()
- afterSave($success, $id = null)
- beforeFind($query)
- afterFind($query)
- beforeRender()
- beforeDelete($id)
- afterDelete($id, $success = false)
- recordNotFound($id)
- beforePaginate()

Always remember to call parent... never break the chain!!

Creating your first Form Decorator

Create the Decorator in **Lib/Form/<ModelName>sFormDecorator.php**

Lib/Form/<ModelName>sFormDecorator.php

```
<?php
App::uses('BaseFormDecorator', 'Crud.Form');
class <ModelName>sFormDecorator extends BaseFormDecorator {
 public function beforeRender() {
  // Check about this is admin, and about this function should be process for this action
  if ($this->Controller->isAdmin() && $this->shouldProcess('only', array('admin_add'))) {
   // We only wanna do something, if this is admin request, and only for "admin_add"
  }


  return parent::beforeRender();
 }

 public function afterSave($success, $id) {
  if ($this->Controller->isAdmin()) {
   // In this test, we want afterSave to do one thing, for admin_add and another for admin_edit
   // If admin_add redirect to parent
   if ($this->shouldProcess('only', array('admin_add'))) {
    if ($success) {
     $controller->redirect(array('action' => 'index'));
    }
   }
   // If admin_edit redirect to self
   elseif ($this->shouldProcess('only', array('admin_edit'))) {
    if ($success) {
     $controller->redirect(array('action' => 'edit', $id));
    }
   }
  }


  return parent::afterSave($success, $id);
 }
}
```

Examples

Controller

```
<?php
/**
 * Admin add will call following callbacks
 *
 * init
 * beforeSave
 * afterSave
 * beforeRender
 *
 * @return void
 */
public function admin_add() {
 $this->Crud->addCallback('admin_add', '<Pluigname>.<ModelName>'); // Decorator to call

 return $this->Crud->addAction();
}
```