Using helpers to overload a back-office template

Using helpers to overload a back-office template

Back-office template architecture

The back-office templates are stored in the `admin/themes/default/template` folder, but several locations can have an impact:

- `admin/themes/default/template/helpers/type_of_the_helper/`: templates that are directly called by helpers if there is no template overloading.
- `admin/themes/default/template/controllers/name_of_the_controller/`: templates that are directly overloaded by a controller.
- `override/controllers/admin/templates/name_of_the_controller/`: overloading folder for templates that already exist in `admin/themes/default/template/controllers/name_of_the_controller/`

Examples

Prerequisites

Create this test table in your MySQL database:

```
CREATE TABLE `ps_test` (
  `id_test` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` VARCHAR( 255 ) NOT NULL
);
```

Create this `Test` class:

```
<?php

class TestCore extends ObjectModel
{
     /** @var string Name */
    public $name;

    /**
     * @see ObjectModel::$definition
     */
    public static $definition = array(
        'table' => 'test',
        'primary' => 'id_test',
        'fields' => array(
            'name' => array(
                'type' => self::TYPE_STRING,
                'validate' => 'isGenericName',
                'required' => true,
                'size' => 64
            )
        ),
    );
}
```

Example 1: Using templates for helpers within a controller

We are going to create a controller by using the default helpers' templates.

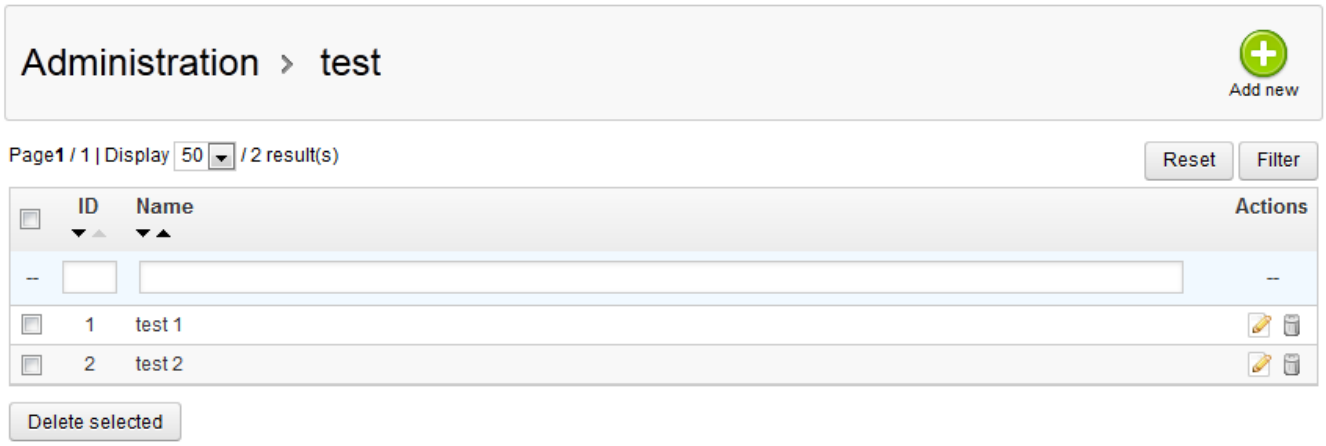
First, we create a Test controller:

```
<?php
```

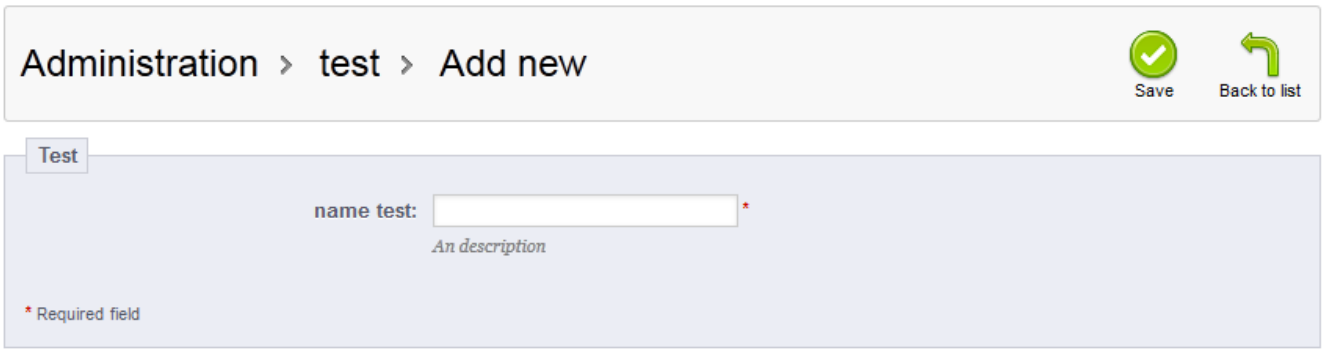
```
class AdminTestControllerCore extends AdminController
{
    public function __construct()
    {
         $this->table = 'test';
         $this->className = 'Test';

         $this->lang = false;

        // Building the list of records stored within the "test" table
        $this->fields_list = array(
            'id_test' => array(
                'title' => $this->l('ID'),
                'align' => 'center',
                'width' => 25
            ),
            'name' => array(
                'title' => $this->l('Name'),
                'width' => 'auto'
            )
        );

        // This adds a multiple deletion button
        $this->bulk_actions = array(
            'delete' => array(
                'text' => $this->l('Delete selected'),
                'confirm' => $this->l('Delete selected items?')
            )
        );

        parent::__construct();
    }

    // This method generates the list of results
    public function renderList()
    {
        // Adds an Edit button for each result
        $this->addRowAction('edit');

        // Adds a Delete button for each result
        $this->addRowAction('delete');

        return parent::renderList();
    }

    // This method generates the Add/Edit form
    public function renderForm()
    {
        // Building the Add/Edit form
        $this->fields_form = array(
            'legend' => array(
                'title' => $this->l('Test')
            ),
            'input' => array(
                array(
                    'type' => 'text',
                    'label' => $this->l('name test:'),
                    'name' => 'name',
                    'size' => 33,
                    'required' => true,
                    'desc' => $this->l('A description'),
                )
            ),
            'submit' => array(
                'title' => $this->l('    Save   '),
                'class' => 'button'
            )
        );

        return parent::renderForm();
    }
```

```
}
```

Here is the result of the list generation (`renderList()` method):



Here is the result of the form generation (`renderForm()` method):



Example 2: Overloading a helper template

You might want something different than what the default helper template provide. Thankfully, you can overload the helps template themselves!

First, you need to create a folder in the `admin/themes/default/template/controllers/` folder. Let's call it `/test`.
This folder name is the name of the controller without the "Admin", "Controller" and "Core" sections. For example:

- `AdminTestControllerCore` -> `admin/themes/default/template/controllers/`**`test/`**
- `AdminTestPrestashopControllerCore` -> `admin/themes/default/template/controllers/`**`test-prestashop/`**

You can change the name of the folder if you put this code in the constructor:

```
$this->tpl_folder = 'test_controller';
```

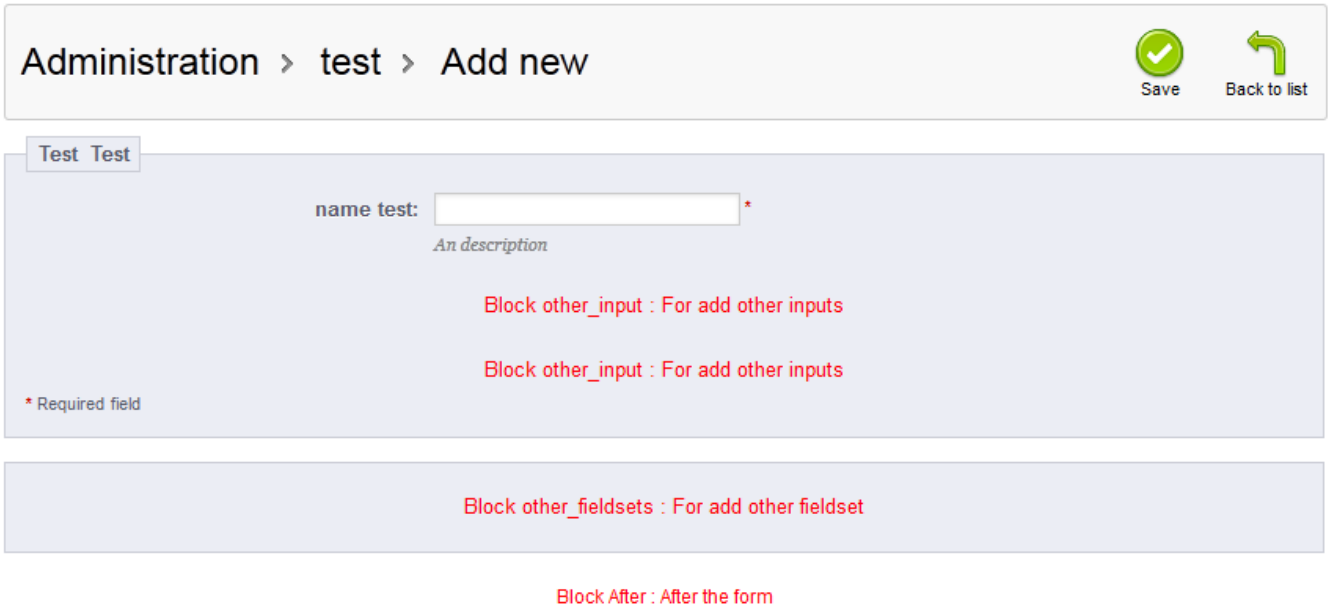
...or you could define a new architecture with subfolders by putting this code in the constructor:

```
$this->tpl_folder = 'test_controller/content';
```

Here is how you can overload a form's `.tpl` file (in `admin/themes/default/template/controllers/test/helpers/form/form.tpl`):

```
{extends file="helpers/form/form.tpl"}

{block name="other_input"}
    {if isset($input.name) && $input.type == 'text'}
        <p style="color:red;text-align:center;">other_input block: to add other input</p>
    {/if}
{/block}

{block name="other_fieldsets"}
    <br />
    <fieldset>
        <p style="color:red;text-align:center;">other_fieldsets block: to add other fieldsets</p>
    </fieldset>
{/block}

{block name="after"}
    <br /><p style="color:red;text-align:center;">After block: to add after the form</p>
{/block}

{block name=script}
    {* Add your scripts here *}
    $(document).ready(function() {

    });
{/block}
```

Here is the result:



Example 3: Overloading an existing template

You might want to overload a template using a controller. In this case, you just have to put the overloaded file in the following folder: `override/controllers/admin/template/`

For instance, let's say you want to add a message in the General Preferences page. You must have add the following file in this location: `override/controllers/admin/templates/preferences/helpers/options/options.tpl`

```
{extends file="helpers/options/options.tpl"}

{block name="after"}
    <p style="color:red;text-align:center;">Block after : add a text after the form</p>
{/block}
```

Here is the result: