# Lab zend form

1. Tổng quát:

Những vấn đề mà mình xin trình bày trong bài lap này:

1. Zend\_Form, Zend\_Form\_SubForm, Zend\_Form\_Element, Zend\_Decorator
2. Tạo DataGridView với ZendForm, Phân trang

Những vấn đề mình không nói tới trong bài lap này: JavaScript(Ajax, jquery,…); phân trang với Zend\_Paginator; vân vân.

Tiêu chí của bài lap: dựa trên cơ sở áp dụng thực tế của mình, đưa ra lý thuyết kèm theo output thực tế để kiểm chứng. Các bạn có thể sử dụng demo mình đính kèm để có được sự nhìn nhận khách quan.

1. Zend\_Form, Zend\_Form\_Element, Zend\_Decorator
2. Zend\_Form

- Zend\_Form được Zend xây dựng lại từ Form của HTML theo hướng đối tượng dựa trên nền tảng PHP

- Các bạn xây dựng zendform bằng cách tạo trực tiếp đối tượng zend\_form cũng được, hoặc kế thừa lại class Zend\_Form. Mình thường dùng cách 2

- Zend\_Form có DefaultDecorator, có thể add các Element(tương được với 1 cặp thẻ ở trong HTML), các SubForm (lớp con của Zend\_Form, là lớp trừu tượng tượng trưng cho 1 nhóm thẻ HTML)

- Nên override lại defaultDecorator của Zend\_Form.

Với khởi tạo như sau ($this là form)

$this->setMethod('get')

->setAction('index');

$this->setDecorators(**array**(

'FormElements',

**array**('HtmlTag',**array**('tag'=>'table')),

'Form'

));

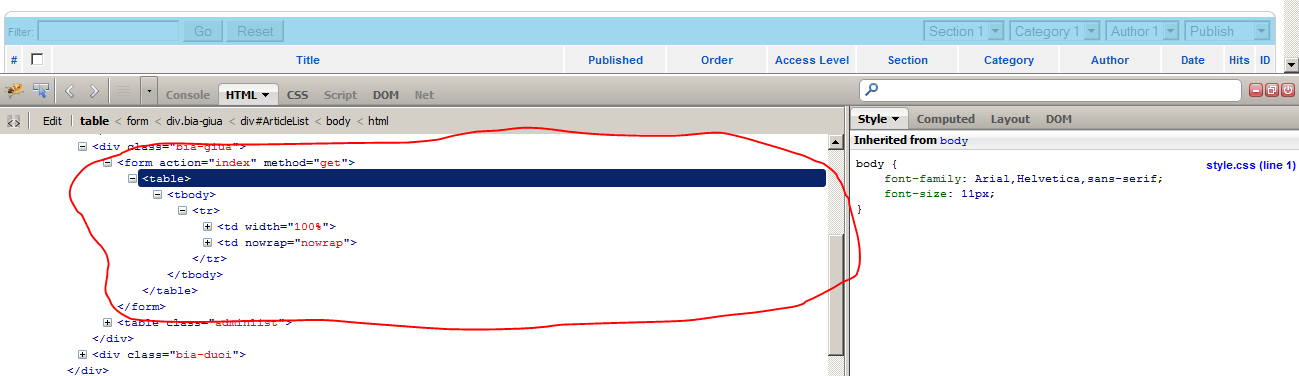
$this->setDecorators(**array**(

'FormElements',

**array**('HtmlTag',**array**('tag'=>'table')),

'Form'));

Các bạn sẽ có output nhìn như sau



Phần action, bạn đặt trong đó là tên action tương ứng, phần method thì post hoặc get tùy bạn chọn.

1. Zend\_Form\_SubForm

Là lớp con kế thừa từ Zend\_Form. Có khả các khả năng của Zend\_Form, thường được sử dụng để add 1 nhóm Element và đặt chúng vào chung trong 1 Decorator của SubForm rồi mới add vô Form.

Các bạn xem ví dụ sau

/\*\*

\*

\* SubForm thu nhat chua filter text

\* **@var** unknown\_type

\*/

$subform1st = **new** Zend\_Form\_SubForm();

$subform1st->setDecorators(**array**(

'FormElements',**array**(

'HtmlTag',**array**(

'tag'=>'td','width'=>'100%'))));

$filterBox = **new** Zend\_Form\_Element\_Text('filter');

$filterBox->setDecorators(**array**(

'ViewHelper','Label'))->setLabel('Filter: ');

$goButton = **new** Zend\_Form\_Element\_Button('Go');

$goButton->setDecorators(**array**('ViewHelper'));

$resetButton = **new** Zend\_Form\_Element\_Button('Reset');

$resetButton->setDecorators(**array**('ViewHelper'));

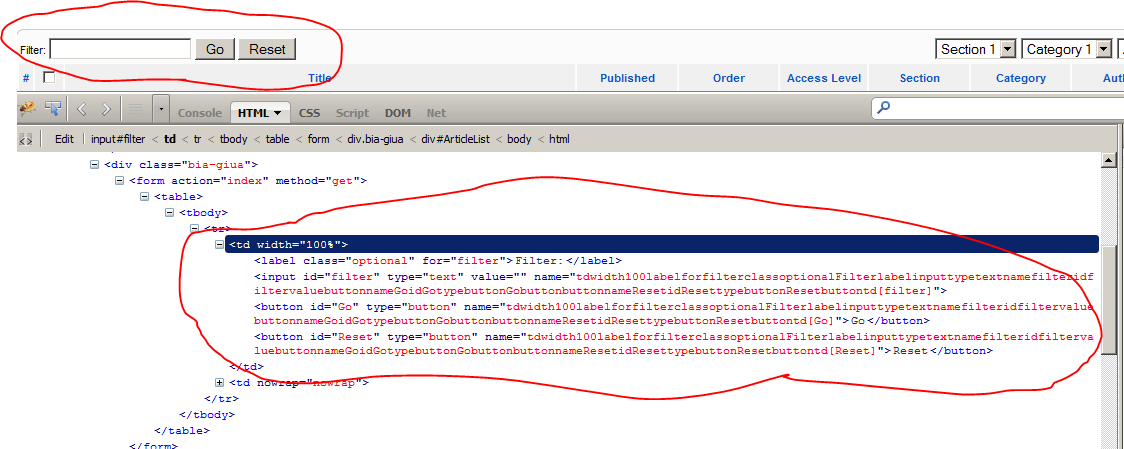
$subform1st->addElements(**array**(

$filterBox,$goButton,$resetButton));

$subform1st->setElementsBelongTo($subform1st);

$this->addSubForm($subform1st, 'filtertext');

Phần output tương ứng với Subform đó như sau



Giải thích:

Đầu tiên các bạn khởi tạo Subform

$subform1st = **new** Zend\_Form\_SubForm();

Cho subform này có thẻ bao ngoài là <td width=100%> bằng cách setDecorators cho $subform1st như sau:

$subform1st->setDecorators(**array**(

'FormElements',**array**(

'HtmlTag',**array**(

'tag'=>'td','width'=>'100%'))));

Tạo các element con để add vô $subform1st và setDecorators sao cho chúng chỉ render cặp thẻ của element mà thôi, các cặp thẻ khác đều clear hết

$filterBox = **new** Zend\_Form\_Element\_Text('filter');

$filterBox->setDecorators(**array**(

'ViewHelper','Label'))->setLabel('Filter: ');

$goButton = **new** Zend\_Form\_Element\_Button('Go');

$goButton->setDecorators(**array**('ViewHelper'));

$resetButton = **new** Zend\_Form\_Element\_Button('Reset');

$resetButton->setDecorators(**array**('ViewHelper'));

Thông thường ta chỉ cần dùng ‘ViewHelper’ trong decorator, nhưng nếu bạn có setLabel cho element và muốn hiển thị nó ra thì bỏ thêm decorator là ‘Label’

Add các element vào $subform1st, hàm setElementsBelongTo($subform1st) là bắt buộc. Xong xuôi, ta dùng hàm addSubForm để thêm Subform vào Form

$subform1st->addElements(**array**(

$filterBox,$goButton,$resetButton));

$subform1st->setElementsBelongTo($subform1st);

$this->addSubForm($subform1st, 'filtertext');

1. Zend\_Form\_Element

Zend tạo sẵn cho chúng ta nhiều element: button, image, select,… Không phải element nào mình cũng có điều kiện nghiên cứu nên mình chỉ hướng dẫn lại cách sử dụng những element mà mình đã sử dụng qua. Và một element đặc biệt cho phép ta bỏ text/html vào đó, có thể hiển thị bất cứ text/html nào.

Các Element đều kế thừa từ lớp Zend\_Form\_Element, sử dụng các phương thức như: setAtrrib, setLabel, setDecorator/setDecorators, addValidator,… và nhiều phương thức khác. Từ tên gọi label bạn dễ dàng suy ra tác dụng của những hàm này. Cụ thể

* Zend\_Form\_Element\_Text

Bạn xem code ví dụ

$orderbox2 = **new** Zend\_Form\_Element\_Text('orderbox',

**array**('value'=>$rowdata['order']));

$orderbox2->setDecorators(**array**(

'ViewHelper',**array**(

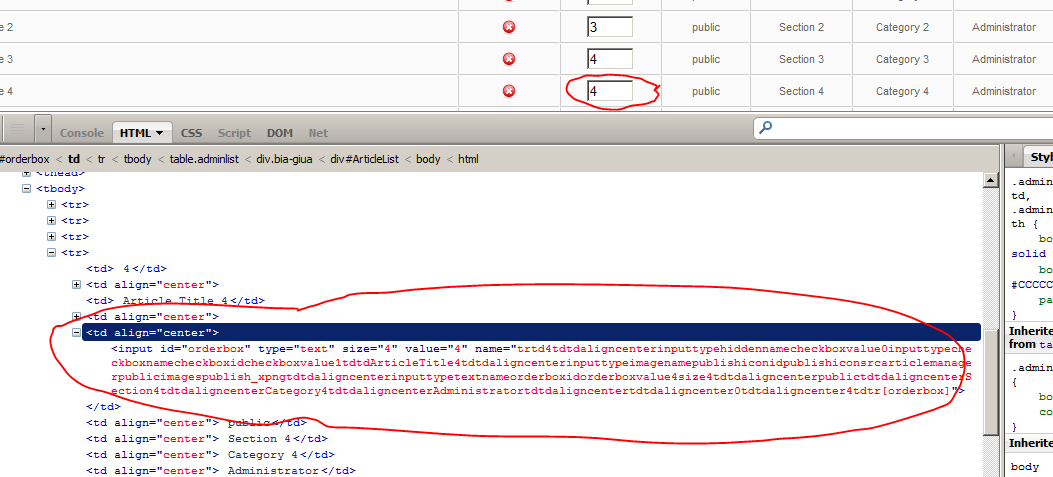
'HtmlTag',**array**(

'tag'=>'td','align'=>'center'))))

->setAttribs(**array**('size'=>'4'));

Ở code trên, $orderbox2 là một Text Element(textbox). Phần khởi tạo tạo một textbox với id là “oderbox”, giá trị khởi tạo là $rowdata[‘order’], là giá trị của 1 phần tử mảng với key là ‘order’

Các hàm sử dụng là setDecorators, setAtrribs. Với khởi tạo như thế này, mình sẽ có output như sau



* Zend\_Form\_Element\_Button

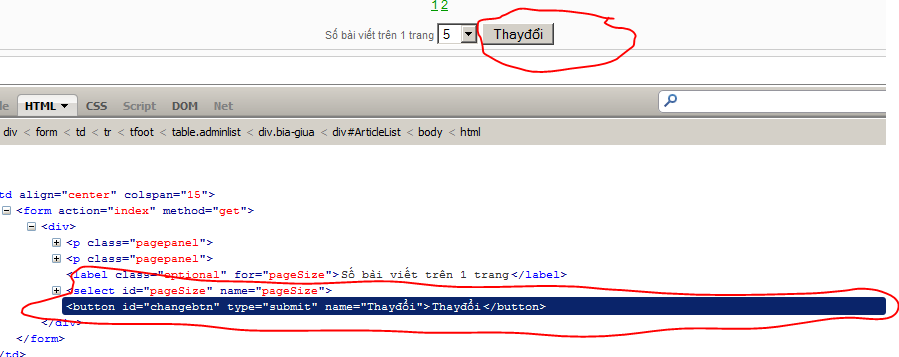
Bạn xem ví dụ sau

$changebtn = **new** Zend\_Form\_Element\_Button('Thay đổi');

$changebtn

->setAttribs(**array**('id'=>'changebtn','type'=>'submit')) ->setDecorators(**array**('ViewHelper'));

Với 1 button được khởi tạo như trên, ta sẽ có 1 button hiển thị chũ “thay đổi”, id là “changebtn”, và có khả năng submit form. Phần setAtrrib type=submit là bắt buộc nếu bạn muốn button thực hiện submit. Output của nó như sau



* Zend\_Form\_Element\_Image

Các bạn xem code ví dụ sau

$publishicon = **new** Zend\_Form\_Element\_Image('publishicon');

$publishicon->setDecorators(**array**('ViewHelper',**array**('HtmlTag',**array**('tag'=>'td','align'=>'center'))));

**if**($rowdata['is\_published']==1)

$publishicon->setImage(

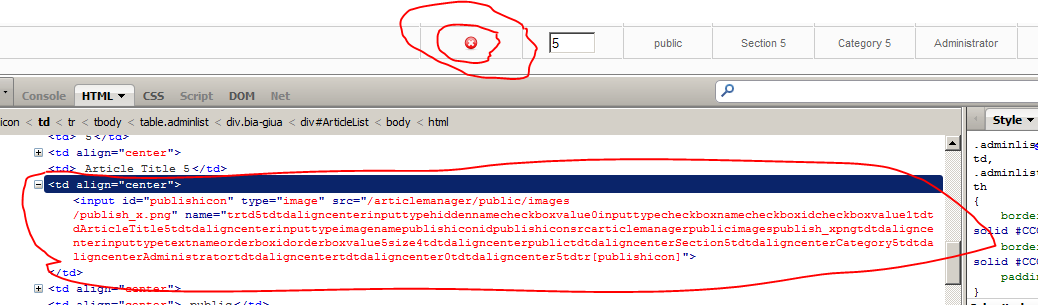
'/articlemanager/public/images/publish\_g.png');

**else** $publishicon->setImage(

'/articlemanager/public/images/publish\_x.png');

Decorator thì các bạn set như các element khác, phần khác biệt là phương thức setImage($imageUrl) sẽ set hình ảnh (tương đương với thuộc tính src=”” của thẻ <img>. Lưu ý phải là đường dẫn Url, ko phải là đường dẫn vật lý (kiểu như C:\AppServe\www\image.png là không được, đường dẫn tương tự như mình ở trên, articlemanager là thư mục chứa source).

Output của nó như sau



* Zend\_Form\_Element\_Select

Các bạn xem code ví dụ

$displayperpage = **new** Zend\_Form\_Element\_Select('pageSize');

$option = **array**(

'5'=>5,

'10'=>10,

'15'=>15,

'20'=>20,

'25'=>25,

'30'=>30);

$displayperpage

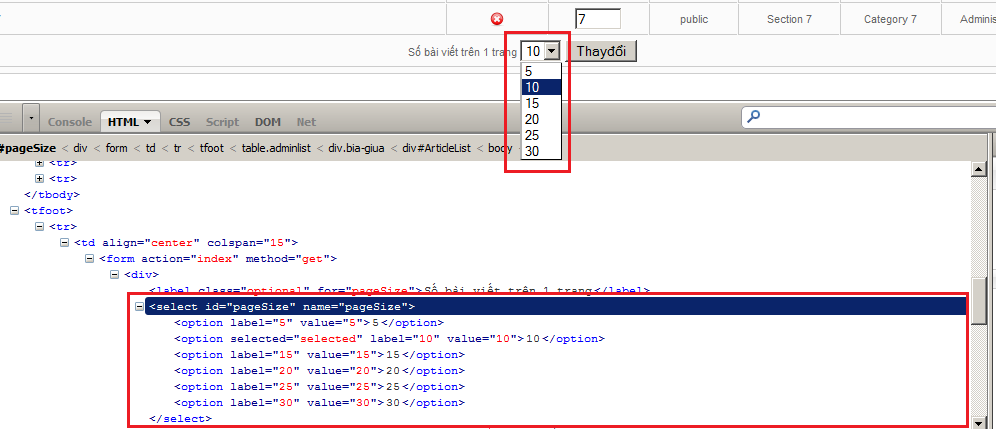
->setMultiOptions($option)

->setLabel('Số bài viết trên 1 trang ')

->setDecorators(**array**('ViewHelper','Label')) ->setValue(10);

Các bạn khởi tạo như trên, 1 thẻ select với id=”pageSize”. Để bỏ các option cho select, bạn cần tạo một chuỗi, ở trên là chuỗi $option. Ứng với key của phần tử sẽ là thuộc tính value của thẻ <option> và giá trị của phần tử thì ứng với chữ sẽ hiển thị ra ngoài. Sau đó các bạn dùng hàm setMultiOptions($option) để bỏ option vào. Để set default selected value thì các b ạn dùng hàm setValue(10) với 10 là value cần set (tức key trong mảng option)

Output của nó trông như sau



* Application\_Form\_Element\_Note

Application\_Form\_Element\_Note là một class do mình tạo, kế thừa từ Zend\_Form\_Element\_Xhtml

<?php **class** Application\_Form\_Element\_Note **extends** Zend\_Form\_Element\_Xhtml

{

**public** $helper = 'formNote';

}

Với element tự tạo này, mình sử dụng phướng thức setValue($chuoicaninra) thì nó sẽ render ra trọn vẹn phần value được set vào đó, bao gồm cả thẻ html nếu có. Nó giúp ta có thể tạo các element chỉ chứa text hoặc các phần định dạng kèm theo, thẻ <p>, <span>, <a>, …

Một ví dụ như sau

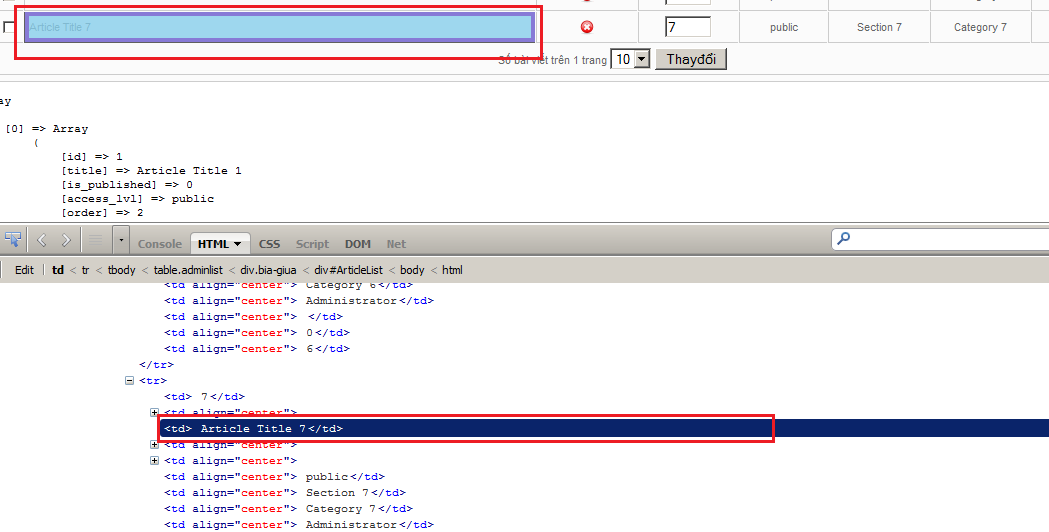
$title = **new** Application\_Form\_Element\_Note(

'title',**array**('value'=>$rowdata['title']));

$title->setDecorators(**array**(

'ViewHelper',**array**('HtmlTag',**array**('tag'=>'td'))));

Với Note được tạo như trên, mình sẽ có output như sau:



1. Tạo DataGridView với ZendForm

Giải thuật tạo datagridview động(có nghĩa là nhận dữ liệu đầu vào và xuất ra tương ứng) dựa trên những hiểu biết của mình như sau:

- Sử dụng table

- Ứng với mỗi ô(cặp thẻ <td></td>) là 1 element, hiển thị một ô dữ liệu

- Các element thuộc cùng dòng thì add chung vào 1 SubForm, setDecorator cho SubForm đó là cặp thẻ <tr>

- Cuối cùng add các cặp thẻ trên vào form, setDecorator cho form là cặp thẻ <table> (có thể không cần, như thế thì ta phải echo từng SubForm dựa vào tên bạn sử dụng khi dùng hàm addSubForm)

Chi tiết, các bạn xem demo đính kèm.

1. Phân trang:

Giải thuật phân trang đòi hỏi mình phải sử dụng một số biến sau ở controller:

$pageSize; *// Số row hiển thị trên gridview*

$pageNum; *// trang hiện tại*

$offset; *// offset của thanh phân trang. Ví dụ khi offset = 2 thì chỉ hiển thị số trang từ 1->4, nếu bạn vào trang 2 thì hiển thị từ 2->5 (nếu form có >=5 trang)*

$startrow = ($pageNum - 1)\*$pageSize; // chỉ số của record bắt đầu lấy trong database

$totalRows = 0; *// tổng số trang, được tính bằng cách đếm tổng số row có trong csdl và chia cho $pageSize, làm tròn*

Cơ bản nó có những biến như vậy. Giải thuật của mình như sau:

1/Khi mới đầu load trang,hệ thống dựa trên giá trị khởi tạo của các biến trên, get data từ csdl để tạo form tương ứng.

2/ Khi user thực hiện việc chuyển trang, trên đường link sẽ có biến pageNum= để controller get về, load lại form tương ứng với trang hiện tại mới

3/ Nếu user thay đổi pageSize, biến pageSize tương ứng được gửi về thông qua phương thức get, controllet get lại pageSize và tạo, render ra form mới tương ứng.

Chi tiết mời các bạn xem demo

Đánh giá: giải thuật trên của mình chỉ dừng ở mức dùng được, chưa phải là giải thuật tốt. Tuy nhiên do giới hạn về kiến thức và thời gian nên mình chưa có điều kiện hoàn thiện. Sẽ cố gắng hoàn thiện trong thời gian tới, mong sự đóng góp ý kiến của các bạn.