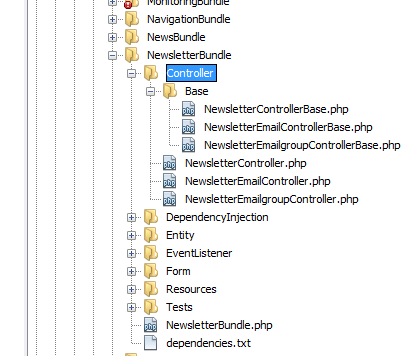
السلام عليكم ورحمة الله

فيما يلي شرح توثيقي للـ CRUD

يمكن أخذ NewsletterEmailgroup CRUD في NewsletterBundle كمثال توضيحي.



**Controllers**

يتكون الـ Controller من الـ Controller الرئيسي NewsletterEmailgroupController.php وبفتح الـ Source Code الخاص به نجد أنه موروث من NewsletterEmailgroupControllerBase.php الموجود في المجلد Base

وفي الـ Controller الرئيسي NewsletterEmailgroupController.php

وهو Class فارغ يمكن أن نكتب فيه فقط الـ Actions التي نريد عمل Override لها لوجود شئ مختلف بها عن الـ Actions الموروثة من NewsletterEmailgroupControllerBase.php

يعمل الكود التالي على تفعيل الـ Auto generated routing لكل الـ Actions الموجودة في الـ Controller

1. **use** Rawafed\SecurityBundle\Annotations as Security;
3. */\*\**
4. *\* @Security\SecuredArea(group="content")*
5. *\*/*

بالإنتقال إلى الـ Parent Class

NewsletterEmailgroupControllerBase.php

نجد أنه موروث من الـ Abstract Class التالي: Rawafed\AdminBundle\Classes\AdminCrudController

الذي به كل الـ Business logic المستخدم عادة في إنشاء CRUD, وفيما يلي شرح تفصيلي للـ Class

NewsletterEmailgroupControllerBase.php

**<?php**  
  
**namespace** Rawafed\NewsletterBundle\Controller\Base;  
  
**use** Rawafed\AdminBundle\Classes\AdminCrudController;  
**use** Symfony\Component\HttpFoundation\Request;  
  
**use** Rawafed\NewsletterBundle\Entity\NewsletterEmailgroup;  
**use** Rawafed\NewsletterBundle\Form\NewsletterEmailgroupType;  
  
**class** NewsletterEmailgroupControllerBase **extends** AdminCrudController  
{  
    **public** **function** \_\_construct() {

حفظ المجلد الموجود فيه ملفات التويج وحفظ كل Routs التي سوف تستخدم في الـ Crud في هذه المصفوفة لاستخدامها بعد ذلك بأسماء مختصرة في باقي الـ Actions وفي الViews أيضاً.

        $this->setMainViewDirectory('NewsletterBundle:NewsletterEmailgroup');  
        $this->setPaths([array](http://www.php.net/array)(  
            'home'      => 'newsletter\_newsletter\_home',  
            'index'     => 'newsletter\_newsletteremailgroup\_index',  
            'new'       => 'newsletter\_newsletteremailgroup\_new',  
            'create'    => 'newsletter\_newsletteremailgroup\_create',  
            'edit'      => 'newsletter\_newsletteremailgroup\_edit',  
            'update'    => 'newsletter\_newsletteremailgroup\_update',  
            'show'      => 'newsletter\_newsletteremailgroup\_show',  
            'delete'    => 'newsletter\_newsletteremailgroup\_delete',  
            'batch'     => 'newsletter\_newsletteremailgroup\_batch',  
        ));  
    }

تكوين الQuery الذي سوف يعرض الـ Grid باستخدام QueryBuilder حتى يقوم الفلتر بإضافة أجزاء الـ Where بشكل أوتوماتيكي بعد ذلك قبل أن ينفذ الـ Query

    **protected** **function** getListQueryBuilder() {  
        $repository = $this->getDoctrine()->getRepository('NewsletterBundle:NewsletterEmailgroup');  
        $queryBuilder = $repository->createQueryBuilder('g')  
        ->select('g,e, g.id, g.name, COUNT(e.id) AS numEmails')  
        ->leftJoin('g.email','e')  
        ->groupBy('g.id');  
        return $queryBuilder;  
    }

هذه الـ Method اختيارية يمكن كتابتها أو عدم كتابتها وفي حالة كتابتها فهي تسرع بشكل كبير الـ Queries التي والتي بها Joins وترجع كمية كبيرة من ال rows وذلك لإنها تعطيه الناتج النهائي لعدد الـ Rows الراجعة أولاً بدون استخدام Join ثم بعد ذلك يقوم الـ Paginator بعمل حساباته بناء على هذا الرقم, وإن لم يتم كتابة هذه الـ Method سيقوم الPaginator بعمل حساباته ولكن بطريقة أبطأ.

    **protected** **function** getListCountQueryBuilder() {  
        $repository = $this->getDoctrine()->getRepository('NewsletterBundle:NewsletterEmailgroup');  
        $queryBuilder = $repository->createQueryBuilder('g')->select('COUNT(g.id)');  
        return $queryBuilder;  
    }

تقوم هذه الmethod بإرجاع Array بها وصف للفلتر ويقوم البرنامج ببناءه آلياً بناء على هذا الوصف

Array كل عنصر بها عبارة عن Array فرعية مكوناتها كالتالي:

Title: العنوان

name: Form element name (unique)

field: الحقل الذي سوف يعقد المقارنة عليه, لابد أن يكون مختاراً في الكويري

type: نوع الحقل الذي سوف يظهر في الفورم

ولابد أن يكون أحد القيم التالية:

Text, Date, Boolean, Entity, List, Choice

في حالة Entity, List, Choice لابد أن تكون الخاصية list بها Array بكل الخيارات المتاحة

Comparison: نوع المقارنة

LIKE, =, <, >, <=, >=, !=

      
    **protected** **function** getListFilterDefinition() {  
        $t = $this->get('translator');  
        return [array](http://www.php.net/array)(  
            [array](http://www.php.net/array)('title' => $t->trans('admin.titles.id'), 'name'=>'id', 'field'=>'g.id', 'type'=>'Text', 'comparison'=>'='),  
            [array](http://www.php.net/array)('title' => $t->trans('admin.titles.name'), 'name'=>'name', 'field'=>'g.name', 'type'=>'Text','comparison'=>'LIKE'),  
        );  
    }

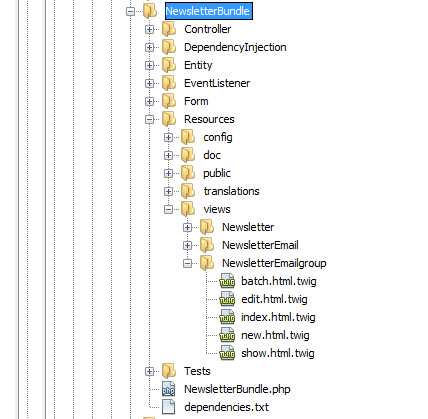
عرض الـ Grid

السبب أني عملت implementation الـ Actions في هذا الClass ولم أعمل implementations للـ Actions في الـ Abstract Class هو أنه ربما يكون CRUD لا أريد فيه Action معين مثل الـ Show مثلاً فلا يكون معمول له implementation رغما عني.. وإنما في حالة الإحتياج للـ Business logic الخاص به أستدعيه من خلال الـ method المناسبة له getShowActionResponse مثلاً أو غيرها وأمرر لها فقط الباراميترات المناسبة.

      
    **public** **function** indexAction(Request $request) {  
        return $this->getIndexActionResponse($request);  
    }  
      
    **public** **function** newAction() {  
        $entity = **new** NewsletterEmailgroup();  
        return $this->getNewActionResponse(  
            $entity,  
            **new** NewsletterEmailgroupType()  
        );  
    }  
      
    **public** **function** createAction(Request $request) {  
        $entity  = **new** NewsletterEmailgroup();  
        return $this->getCreateActionResponse(  
            $request,  
            $entity,  
            **new** NewsletterEmailgroupType()  
        );  
    }  
      
    **public** **function** editAction($id) {  
        $em = $this->getDoctrine()->getManager();  
        $entity = $em->getRepository('NewsletterBundle:NewsletterEmailgroup')->find($id);  
        return $this->getEditActionResponse(  
            $entity,  
            **new** NewsletterEmailgroupType()  
        );  
    }  
  
    **public** **function** updateAction(Request $request, $id) {  
        $em = $this->getDoctrine()->getManager();  
        $entity = $em->getRepository('NewsletterBundle:NewsletterEmailgroup')->find($id);  
        return $this->getUpdateActionResponse(  
            $request,  
            $entity,  
            **new** NewsletterEmailgroupType()  
        );  
    }  
      
    **public** **function** showAction($id) {  
        $em = $this->getDoctrine()->getManager();  
        $entity = $em->getRepository('NewsletterBundle:NewsletterEmailgroup')->find($id);  
        return $this->getShowActionResponse($entity);  
    }  
      
    **public** **function** deleteAction(Request $request, $id = **null**) {  
        return $this->getDeleteActionResponse($request, $id);  
    }  
  
    **public** **function** batchAction(Request $request) {  
        return $this->getBatchActionResponse($request);  
    }

    **protected** **function** removeEntityById($id) {  
        $em = $this->getDoctrine()->getManager();  
        $entity = $em->getRepository('NewsletterBundle:NewsletterEmailgroup')->find($id);  
        return $this->removeEntity($entity);  
    }  
}

**الـ Twig Views**



موجودون في المجلد الذي ذكرته في الـ Constructor (NewsletterBundle:NewsletterEmailgroup)

**class** NewsletterEmailgroupControllerBase **extends** AdminCrudController  
{  
    **public** **function** \_\_construct() {  
        $this->setMainViewDirectory('NewsletterBundle:NewsletterEmailgroup');  
        $this->setPaths([array](http://www.php.net/array)(  
            'home'      => 'newsletter\_newsletter\_home',  
            'index'     => 'newsletter\_newsletteremailgroup\_index',  
            'new'       => 'newsletter\_newsletteremailgroup\_new',  
            'create'    => 'newsletter\_newsletteremailgroup\_create',  
            'edit'      => 'newsletter\_newsletteremailgroup\_edit',  
            'update'    => 'newsletter\_newsletteremailgroup\_update',  
            'show'      => 'newsletter\_newsletteremailgroup\_show',  
            'delete'    => 'newsletter\_newsletteremailgroup\_delete',  
            'batch'     => 'newsletter\_newsletteremailgroup\_batch',  
        ));  
    }

في كل الـ Views نجد القسم التالي الخاص ببناء الـ breadcrumb

أعطه الـ Title والـ Path الخاص بكل رابط, وفي الرابط الأخير لا أعطه الـ Path فلا يعتبره clickable

{% block breadcrumb %}  
    {{ backendHelpers.breadcrumb({   
        ("newsletter.titles.newsletter\_home"|trans) : (path(paths.home)),  
        ("newsletter.titles.email\_groups"|trans) : (path(paths.index)),  
        ("newsletter.titles.email\_group\_creation"|trans) : ''  
    }) }}  
{% endblock %}

وهو يستخدم macro يكون الـ html الخاص ببناء الـ breadcrumb وكل هذه الـ macros موجودة في الملف

Rawafed/AdminBundle/Resources/views/Admin/backendHelpers.html.twig

واستدعاء هذا الملف معرف في الـ layout لذلك فهو يتم التعرف على الماكروز الموجودة فيه تلقائياً في كل الـ back-end views

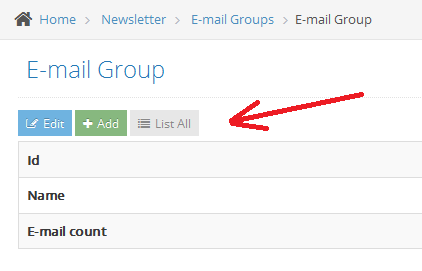
والغرض من هذا هو عندما أريد التعديل على كود الـ html الخاص ببناء الـ breadcrumbs على سبيل المثال أو غيرها أعدل في مكان واحد وليس في عشرات الأماكن.

{{ backendHelpers.pageTitle( "newsletter.titles.email\_group\_creation"|trans ) }}

هذا الكود خاص بتكوين عنوان الصفحة

<div class="actionButtons">  
 {{ backendHelpers.addActionButton(path(paths.edit,{ 'id': entity.id }),paths.edit, ('admin.titles.edit'|trans), 'btn-info','icon-edit' )}}  
 {{ backendHelpers.addActionButton(path(paths.new), paths.new, ('admin.titles.add'|trans), 'btn-success', 'icon-plus' )}}  
 {{ backendHelpers.addActionButton(path(paths.index), paths.index, ('admin.titles.list'|trans), 'btn-light', 'icon-list')}}  
</div>

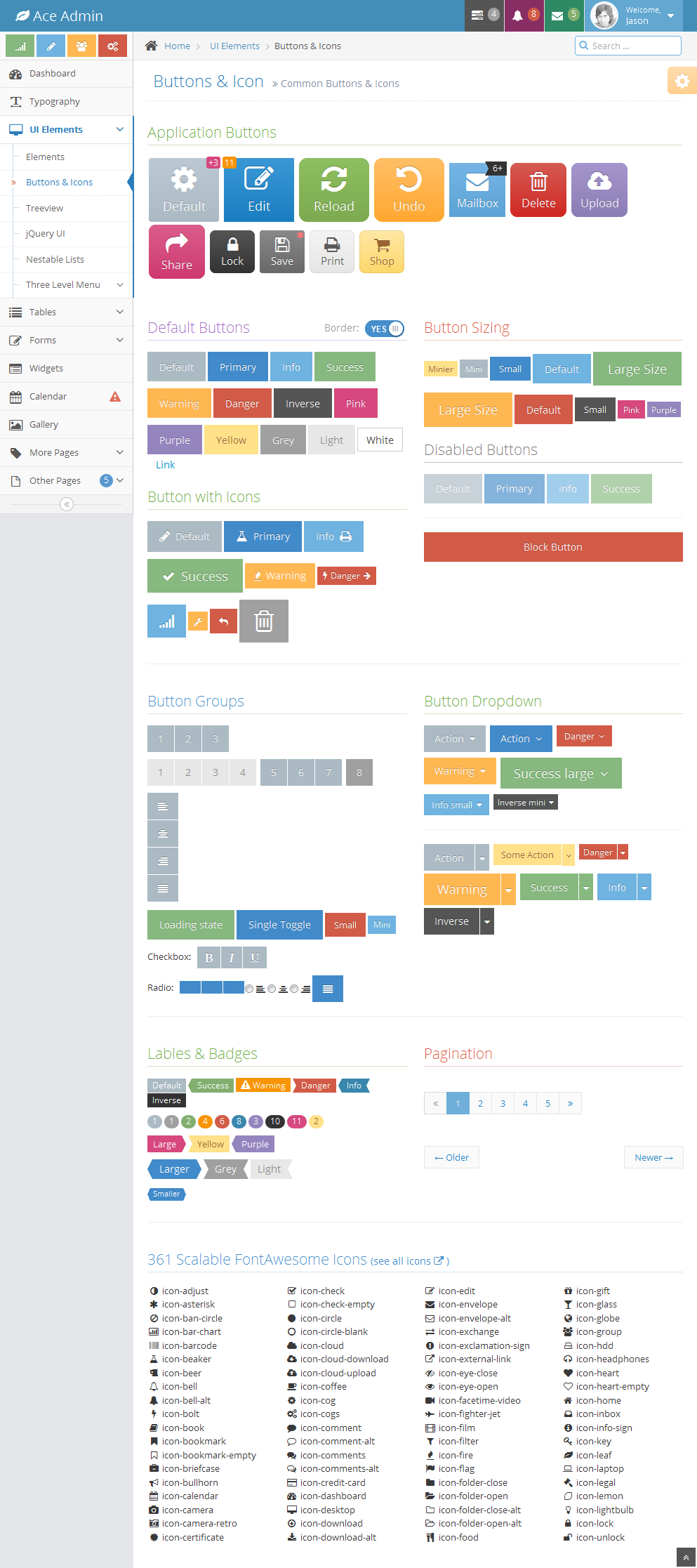
هذا الكود مثال على تكوين الـ Action Buttons



واستدعاء كل ماكرو منهم يكون زر واحد يأخذ الـ parameters اللازمة مثل الpath ليكون الـ url والrout ليحدد إن كان المستخدم له صلاحية لضغط هذا الزر أم لا وإم لم يكن له صلاحية يظهر الزر غير مفعل Disabled

والـ Class الخاص بالزر والClass الخاص بالأيقونة, والمرجع لنا في معرفة هذه الـ classes هو الـ Theme

والسبب أنني تركت الـ Div متاح هو إتاحة الفرصة إن أردنا عمل button مخصص, Drop down button مثلا أو menu أو أي شئ custom



في صفحة الـ Grid وهي index.html.twig لابد أن تكون الأكواد التالية موجودة

يستدعي مكتبات جافا سكريبت الخاصة بمكتبة bazinga وهي تعرف عبارات الترجمة الخاصة بالمشروع للجافا سكريبت حتى يمكن استخدام الترجمة في ملفات جافا سكريبت خارجية

{% block head %}  
    {{ parent() }}  
    {{ backendHelpers.bazingajsTranslationJavaScripts() }}  
{% endblock %}

الكود التالي يقوم بعمل include لبعض ملفات الجافا سكريبت اللازمة لعمل الـ Grid

{% block bottomScripts %}  
    {{ parent() }}  
    {{ backendHelpers.gridJavaScripts() }}  
{% endblock %}

هذا الجزء يقوم ببناء الفلتر بشكل آلي

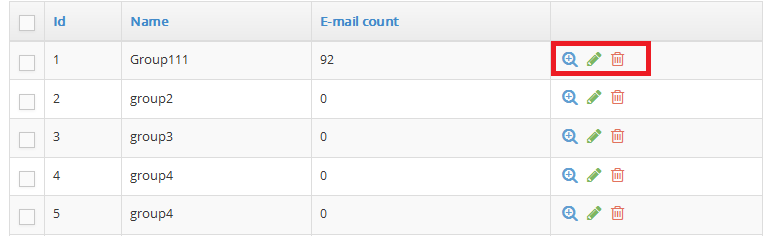
{% block gridFilter %}  
    {{ backendHelpers.gridFilter(filterDefinition, filterValues, path(paths.index,{'mode':'filter'}), path(paths.index)) }}  
{% endblock %}

في الـ html الخاص بالـ Grid نفسها لابد من مراعاة أن الـ id الخاص بها هو gridForm لأن هذه القيمة ذات معنى لبعض أكواد الجافا سكريبت

<**form** id="gridForm" action="{{ path(paths.batch) }}" method="post">

{{ backendHelpers.gridActionButtons([  
{'title': ('admin.titles.show'|trans), 'iconColorClass':'blue', 'iconClass':'icon-zoom-in', 'rout':paths.show, 'path': path(paths.show, { 'id': entity.id })},  
{'title': ('admin.titles.edit'|trans), 'iconColorClass':'green', 'iconClass':'icon-pencil', 'rout':paths.edit, 'path': path(paths.show, { 'id': entity.id })},  
{'title': ('admin.titles.delete'|trans), 'iconColorClass':'red', 'iconClass':'icon-trash', 'rout':paths.delete, 'path': path(paths.delete, { 'id': entity.id, 'redirect': app.request.uri|[base64\_encode](http://www.php.net/base64_encode) }), 'class':'gridDeleteRow', 'gridDeleteLink':**true**}  
]) }}

هذا الكود الخاص بتكوين الـ Action buttons الخاصة بكل Row في الـ Grid

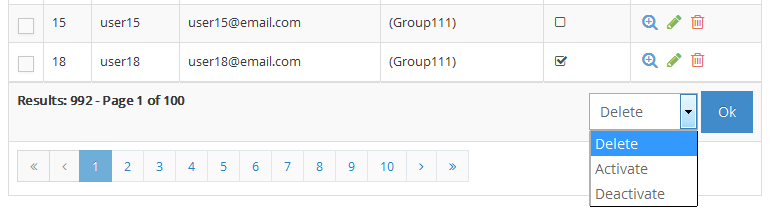


وهو أيضاً بناء على الباراميترات المدخلة له يأخذ في الإعتبار كلاس الزر والأيقونة وما إذا كان المستخدم له صلاحية الضغط عليه أم لا ويظهر الزر باهت وغير مفعل في حالة عدم وجود صلاحية.

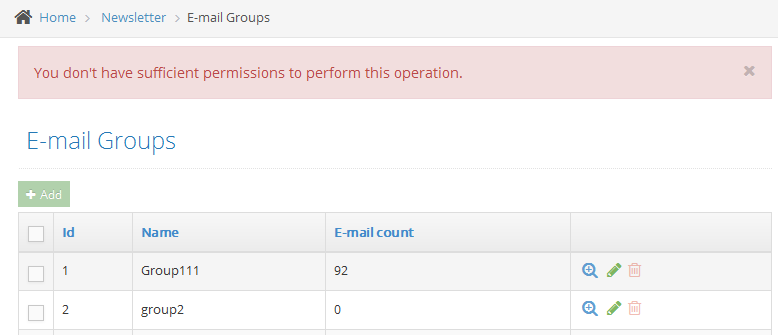
{{ backendHelpers.gridBatch(pagination, [  
   {'path': path(paths.delete), 'rout':paths.delete, 'title':('admin.titles.delete'|trans)},  
   {'path': path(paths.delete, { 'activate': 1 }), 'rout':paths.activate, 'title':('admin.titles.activate'|trans)},  
   {'path': path(paths.delete, { 'activate': 0 }), 'rout':paths.activate, 'title':('admin.titles.deactivate'|trans)}  
]) }}

الكود الخاص ببناء الـ batch operation (الكود السابق خاص ببناء drop down تعمل delete, activate & deactivate)

والبارامترات المطلوبة هي الـ path (أو الـ URL) والـ rout لمعرفة إن كان هناك صلاحية أم لا, وعنوان كل عمليةالذي سيظهر للمستخدم.



وعند قيام المستخدم بعملية batch ليس له صلاحية القيام به ستظهر له هذه الرسالة



كانت هذه أهم الملاحظات بخصوص بناء CRUD

خالص تحياتي

علي حسن