Exercise Solutions

Section 7 Lesson 4. Adminhtml

Exercise 1: Create a Grid for the Customer Distributor Entity

Use your IDE to do this exercise.

Step 1. Create a grid action and reference it from the *adminhtml.xml* menu entry

a) Edit the file *app/code/local/Training/Distributor/etc/system.xml*, ensuring that a `Training_Distributor` controllers directory is added to the adminhtml router configuration.

```
<?xml version="1.0"?>
<config>
      <!-- snip... -->
      <admin>
            <routers>
                  <adminhtml>
                        <args>
                              <!--
                                    add
Training/Distributor/controllers/Adminhtml/ to list of
                                    directories under the Mage_Adminhtml
frontname
                              -->
                              <modules>
                                    <Training_Distributor
after="Mage_Adminhtml">Training_Distributor_Adminhtml</Training_Distributor>
                              </modules>
                        </args>
                  </adminhtml>
            </routers>
      </admin>
</config>
```


b) Create the file *Training/Distributor/controllers/Adminhtml/DistributorController.php* and define the class.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      public function indexAction()
      {
            $this->getResponse()->setBody("It works!");
      }
}
```

c) Edit the file *app/code/local/Training/Distributor/etc/adminhtml.xml*, ensuring that the menu node created in Lesson 2 has an `<action>` parameter as follows.

```
<?xml version="1.0"?>
<config>
      <!-- Add adminhtml menu entry under the Catalog section -->
      <menu>
            <catalog>
                  <children>
                        <training_distributor translate="title"
module="training_distributor">
                              <title>Manage Distributors</title>
                              <action>adminhtml/distributor</action>
                              <!-- aka adminhtml/distributor/index -->
                              <sort_order>250</sort_order>
                        </training_distributor>
                  </children>
            </catalog>
      </menu>
</config>
```

d) Click the link in the menu and ensure that you see the "It works!" message.

e) After verifying that the menu item and controller are working as they should, add the ACL method `_isAllowed()` for the controller and set the `indexAction()` method to redirect to a `listAction()` method.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      protected function _isAllowed()
      {
            return Mage::getSingleton('admin/session')
                  ->isAllowed('catalog/training_distributor');
      }

      public function indexAction()
      {
            $this->_redirect('*/*/list');
      }
}
```

f) Next, define the `listAction()` method that will be used to render the grid.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      //snip...

      public function listAction()
      {
            $this->_getSession()->setFormData(null); //housekeeping
            $this->_title($this->__('Catalog'))->_title($this-
>__('Distributors'));

            $this->loadLayout();
            $this->_setActiveMenu('catalog/training_distributor');
            $this->renderLayout();
      }
}
```

g) Finally, verify that everything is working by clicking the "Manage Distributors" link in the menu, triggering a browser redirect to the list action, and displaying an Admin page with empty content area.

Step 2. Register and create a layout update file and add an update that adds the grid container to the content block

a) Edit the `Training_Distributor` *config.xml* file once again, adding the configuration for a layout update file to the adminhtml area.

```
<?xml version="1.0"?>
<config>
      <!-- snip... -->
      <adminhtml>
            <layout>
                  <updates>
                        <training_distributor>
                              <file>training/distributor.xml</file>
                        </training_distributor>
                  </updates>
            </layout>
      </adminhtml>
</config>
```

b) Next, create the layout update file *app/design/adminhtml/default/default/layout/training/distributor.xml* and add the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout>
      <adminhtml_distributor_list>
            <reference name="content">
<block type="training_distributor/adminhtml_distributor" name="distributor"/>
            </reference>
      </adminhtml_distributor_list>
</layout>
```

Step 3. Create the grid container block

- Create the file *Training/Distributor/Block/Adminhtml/Distributor.php* and define the class and `_construct()` method as follows.


```
<?php

class Training_Distributor_Block_Adminhtml_Distributor
      extends Mage_Adminhtml_Block_Widget_Grid_Container
{
      protected function _construct()
      {
            $this->_blockGroup = 'training_distributor';
            $this->_controller = 'adminhtml_distributor';
            $this->_headerText = $this->__('List Distributors');
            $this->_addButtonLabel = $this->__('Add Distributor');

            parent::_construct();
      }
}
```

Step 4. Create the grid block, using AJAX updates

a) Create the file *Training/Distributor/Block/Adminhtml/Distributor/Grid.php* and define the class and `_construct()` method as follows.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Grid
      extends Mage_Adminhtml_Block_Widget_Grid
{
      protected function _construct()
      {
            parent::_construct();

            $this->setId('training_distributor_list');
            $this->setDefaultSort('id');

            /*
             * Override method getGridUrl() in this class to provide URL
for AJAX
             */
            $this->setUseAjax(true);
      }
}
```


b) Based on the choice to use AJAX , define the `getGridUrl()` method, which will return the URL for the grid-only display.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Grid
      extends Mage_Adminhtml_Block_Widget_Grid
{
      //snip...
      public function getGridUrl()
      {
            return $this->getUrl('*/*/grid', array('_current' => true));
      }
}
```

c) Next, define the `_prepareCollection()` method, which will be called to prepare and set the collection object from which the grid rows will be derived.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Grid
      extends Mage_Adminhtml_Block_Widget_Grid
{
      //snip...

      protected function _prepareCollection()
      {
            $collection =
Mage::getResourceModel('training_distributor/distributor_collection');
            $this->setCollection($collection);

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

d) With the collection object defined, map the model attributes to grid columns in the `_prepareColumns()` method.


```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Grid
      extends Mage_Adminhtml_Block_Widget_Grid
{
      //snip...

      protected function _prepareColumns()
      {
            $this->addColumn('id', array(
                  'header' => $this->__('ID'),
                  'sortable' => true,
                  'width' => '60px',
                  'index' => 'entity_id'
            ));

            $this->addColumn('email', array(
                  'header' => $this->__('Distributor Email'),
                  'index' => 'email',
            ));

            $this->addColumn('name', array(
                  'header' => $this->__('Name'),
                  'index' => 'name',
                  'column_css_class' => 'name'
            ));

            $this->addColumn('action', array(
                  'header' => $this->__('Action'),
                  'width' => '100px',
                  'type' => 'action',
                  'getter' => 'getId',
                  'actions' => array(
                        array(
                              'caption' => $this->__('Edit'),
                              'url' => array('base' => '*/*/edit'),
                              'field' => 'id',
                        ),
                        array(
                              'caption' => $this->__('Delete'),
                              'url' => array('base' => '*/*/delete'),
                              'field' => 'id',
                        ),
                  ),
                  'filter' => false,
                  'sortable' => false,
            ));

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


e) To make the rows "clickable," override the `getRowUrl()` method.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Grid
      extends Mage_Adminhtml_Block_Widget_Grid
{
      //snip...

      public function getRowUrl($row)
      {
            return $this->getUrl('*/*/edit', array('id' => $row->getId()));
      }
}
```

f) Verify that everything is correct by viewing/refreshing the "Manage Distributors" page.

Step 5. Create the AJAX update action and add appropriate layout XML, returning only the grid block

a) Edit *app/design/adminhtml/default/default/layout/training/distributor.xml*, adding the grid action's full action name layout update handle and specifying the grid block as an output block.

```
<?xml version="1.0"?>
<layout>
      <!-- snip... -->
      <adminhtml_distributor_grid>
            <block type="training_distributor/adminhtml_distributor_grid"
name="ajax_grid" output="toHtml"/>
      </adminhtml_distributor_grid>
</layout>
```


b) Define a `gridAction()` method in the `Training_Distributor_Adminhtml_IndexController` class.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      public function gridAction()
      {
            $this->loadLayout(false);
            $this->renderLayout();
      }
}
```

Exercise 2: Create a Form for the Customer Distributor Entity

Use your IDE to do this exercise.

Step 1. Create an edit action and a new action and forward to the edit action from there

a) Define an `editAction()` method in the `Training_Distributor_Adminhtml_DistributorController` class.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      public function editAction()
      {
      }
}
```

This method will render a form for both the creation and updating of `Training_Distributor` entities.

b) In the `editAction()` method, creates a distributor data model instance and add it to the registry for use in the view.

```
<?php

class Training_Distributor_Adminhtml_DistributorController
      extends Mage_Adminhtml_Controller_Action
{
      public function editAction()
      {
            $model = Mage::getModel('training_distributor/distributor');
            Mage::register('current_distributor', $model);
            $id = $this->getRequest()->getParam('id');
      }
}
```

Note: The `id` parameter is mapped to a local variable for convenience and cleanliness.

c) Next, add the logic to distinguish between editing versus creating (including error handling for an invalid entity id being passed) and render using layout XML.

```
<?php

class Training_Distributor_Adminhtml_DitributorController
      extends Mage_Adminhtml_Controller_Action
{
      public function editAction()
      {
            $model =
Mage::getModel('training_distributor/distributor');
            Mage::register('current_distributor', $model);
            $id = $this->getRequest()->getParam('id');

            try {
                  if ($id) {
                        if (!$model->load($id)->getId()) {
                              Mage::throwException(
                                    $this->__('No record with ID "%s"
found.', $id)
                              );
                        }
                  }
                  if ($model->getId()) {
                        $pageTitle =
                              $this->__('Edit %s (%s)', $model-
>getName(), $model->getId());
                  }
                  else {
                        $pageTitle = $this->__('New Distributor');
                  }
                  $this->_title($this->__('Catalog'))
                        ->_title($this->__('Distributors'))-
>_title($pageTitle);

                  $this->loadLayout();
                  $this-
>_setActiveMenu('catalog/training_distributor');
                  $this->renderLayout();
            }
            catch (Exception $e)
            {
                  Mage::logException($e);
                  $this->_getSession()->addError($e->getMessage());
                  $this->_redirect('*/*/list');
            }
      }
}
```


d) Create a method `newAction()` forwarding the user to the edit action internally.

```
public function newAction()
{
      // Redirect the user via a magento internal redirect
      $this->_forward('edit');
}
```

Step 2. Add a layout update for the edit action that adds the form container to the content block

- Edit the `Training_Distributor` adminhtml layout update file, adding the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout>
      <!-- snip... -->
      <adminhtml_distributor_edit>
            <reference name="content">
                  <block
type="training_distributor/adminhtml_distributor_edit"
name="training_distributor_form" />
            </reference>
      </adminhtml_distributor_edit>
</layout>
```

Step 3. Create the form container block

a) Create the file *Training/Distributor/Block/Adminhtml/Distributor/Edit.php.*

b) Define the block container class and `_construct()` method.


```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit
      extends Mage_Adminhtml_Block_Widget_Form_Container
{
      protected function _construct()
      {
            parent::_construct();
            $this->_objectId = 'id';
            $this->_blockGroup = 'training_distributor';
            $this->_controller = 'adminhtml_distributor';
            $this->_mode = 'edit';
   }
}
```

c) Next, update the default button labels and add and configure a "Save and Continue" button in the `_prepareLayout()` method.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit
      extends Mage_Adminhtml_Block_Widget_Form_Container
{
      //snip ...
      protected function _prepareLayout()
      {
            $this->_updateButton('save', 'label', $this->__('Save
Distributor'));
            $this->_updateButton('delete', 'label', $this->__('Delete
Distributor'));

            $this->_addButton('save_and_continue', array(
                  'label' => Mage::helper('adminhtml')->__('Save And
Continue Edit'),
                  'onclick' => 'saveAndContinueEdit()',
                  'class' => 'save',
            ), -100);

            $this->_formScripts[] = "
                  function saveAndContinueEdit(){

      editForm.submit($('edit_form').action+'back/edit/');
                  }
            ";
            return parent::_prepareLayout();
      }
}
```


d) Finally, add header text by defining the `getHeaderText()` method, including logic to handle update versus create.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit
      extends Mage_Adminhtml_Block_Widget_Form_Container
{
      public function getHeaderText()
      {
            $model = Mage::registry('current_distributor');
            if ($model && $model->getId()) {
                  return $this->__('Edit Distributor "%s" (%s)',
                              $this->escapeHtml($model->getName()),
                              $this->escapeHtml($model->getId())
                  );
            } else {
                  return $this->__('New Distributor');
            }
      }
}
```

Step 4. Create the form block.

a) Create the file Training/Distributor/Block/Adminhtml/Distributor/Edit/Form.php

b) Define the block container class and begin to configure the form object in the `_prepareForm()` method.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit_Form
      extends Mage_Adminhtml_Block_Widget_Form
{
      protected function _prepareForm()
      {
            $form = new Varien_Data_Form(array(
            'id'     => 'edit_form',
                        'action' => $this->getUrl('*/*/save',
                                    array('id' => $this->getRequest()-
>getParam('id'))),
                'method' => 'post'
          ));

              $fieldset = $form->addFieldset('general_form', array(
                    'legend' => $this->__('General Setup')
              ));
      }
}
```

c) Continue configuring the form object by adding fields to the fieldset created in the previous step.


```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit_Form
       extends Mage_Adminhtml_Block_Widget_Form
{
       protected function _prepareForm()
       {
              $form = new Varien_Data_Form(array(
             'id'     => 'edit_form',
                           'action' => $this->getUrl('*/*/save',
                                        array('id' => $this-
>getRequest()->getParam('id'))),
             'method' => 'post'
        ));

              $fieldset = $form->addFieldset('general_form', array(
                     'legend' => $this->__('General Setup')
              ));

              if (Mage::registry('current_distributor')->getId()) {
                     $fieldset->addField('entity_id', 'label', array(
                           'label' => $this->__('Entity ID:'),
                     ));
              }

              $fieldset->addField('name', 'text', array(
                     'label'     => $this->__('Name'),
                     'class'     => 'required-entry',
                     'required'  => true,
                     'name'      => 'name',
              ));

              $fieldset->addField('email', 'text', array(
                     'label'     => $this->__('Distributor Email'),
                     'class'     => 'required-entry',
                     'required'  => true,
                     'name'      => 'email',
              ));

              $dateFormatIso = Mage::app()->getLocale()->getDateFormat(
                     Mage_Core_Model_Locale::FORMAT_TYPE_SHORT
              );

              $fieldset->addField('updated_at', 'date', array(
                     'label' => $this->__('Updated At'),
                     'name'  => 'updated_at',
                     'format' => $dateFormatIso,
                     'image'  => $this->getSkinUrl('images/grid-cal.gif'),
              ));
              $form->setUseContainer(true);
              $form->addValues($this->_getFormData());
              $this->setForm($form);
       }
}
```


d) Finally, create the `_getFormData()` method, which will be used to set the correct values for each of the form elements based on the entity currently being edited.

```
<?php

class Training_Distributor_Block_Adminhtml_Distributor_Edit_Form
      extends Mage_Adminhtml_Block_Widget_Form
{
      //snip...
      protected function _getFormData()
      {
            $data = Mage::getSingleton('adminhtml/session')->getFormData();

            if (! $data && Mage::registry('current_distributor')-
>getData()) {
                  $data = Mage::registry('current_distributor')->getData();
            }

            return (array) $data;
      }
}
```

e) In the Manage Distributors view, click the "Add Distributor" button or on a row in the grid to verify that the form is rendering.

Step 5 . Create the actions to receive and process posted form data

a) Define a `saveAction()` method in the `Training_Distributor_Adminhtml_DistributorController` class.


```
<?php

class Training_Distributor_Adminhtml_DistributorController
       extends Mage_Adminhtml_Controller_Action
{
       //snip...
       public function saveAction()
       {
              if ($data = $this->getRequest()->getPost())
              {
                     $this->_getSession()->setFormData($data);
                     $model = Mage::getModel('training_distributor/distributor');
                     $id = $this->getRequest()->getParam('id');

                     try {
                            if ($id) {
                                   $model->load($id);
                            }
                            $model->addData($data)->save();

                            $this->_getSession()->addSuccess(
                                          $this->__('Distributor was successfully
saved'));

                            $this->_getSession()->setFormData(null);

                            if ($this->getRequest()->getParam('back')) {
                                   $params = array('id' => $model->getId());
                                   $this->_redirect('*/*/edit', $params);
                            }
                            else {
                                   $this->_redirect('*/*/list');
                            }
                     }
                     catch (Exception $e) {
                            $this->_getSession()->addError($e->getMessage());
                            if ($model && $model->getId()) {
                                   $this->_redirect('*/*/edit', array('id' =>
$model->getId()));
                            }
                            else {
                                   $this->_redirect('*/*/new');
                            }
                     }

                     return;
              }

              $this->_getSession()->addError($this->__('No data found to
save'));
              $this->_redirect('*/*');
       }
}
```


b) Define a `deleteAction()` method in the `Training_Distributor_Adminhtml_DistributorController` class. This method will attempt to delete the entity data based on the ID present in the request.

```
<?php

class Training_Distributor_Adminhtml_IndexController
      extends Mage_Adminhtml_Controller_Action
{
      //snip...
      public function deleteAction()
      {
            $model = Mage::getModel('training_distributor/distributor');
            $id = $this->getRequest()->getParam('id');

            try {
                  if ($id) {
                        if (!$model->load($id)->getId()) {
                              Mage::throwException(
                                    $this->__('No record with ID "%s"
found.', $id)
                              );
                        }
                        $name = $model->getName();
                        $model->delete();

                        $this->_getSession()->addSuccess($this->__('"%s"
(ID %d) was successfully deleted', $name, $id));

                        $this->_redirect('*/*');
                  }
            }
            catch (Exception $e) {
                  Mage::logException($e);
                  $this->_getSession()->addError($e->getMessage());
                  $this->_redirect('*/*');
            }
      }
}
```


Exercise 3: Add Tabs to the Custom Distributor Entity Form Page

Step 1. Add a layout update for the edit action that adds a tabs block to the left block

- Edit the `Training_Distributor` adminhtml layout update file, adding the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout>
      <!-- snip... -->
      <adminhtml_distributor_edit>
            <reference name="content">
<block type="training_distributor/adminhtml_distributor_edit"
name="training_distributor_form" />
            </reference>
            <reference name="left">
                  <block
type="training_distributor/adminhtml_distributor_edit_tabs"
name="training_distributor_tabs" />
            </reference>
      </adminhtml_distributor_edit>
</layout>
```

Step 2. Create the tabs block and add a single tab containing the form contents

a) Create the file *Training/Distributor/Block/Adminhtml/Edit/Tabs.php* and define the class and `_construct()` method as follows.

```
class Training_Distributor_Block_Adminhtml_Distributor_Edit_Tabs
      extends Mage_Adminhtml_Block_Widget_Tabs
{
      protected function _construct()
      {
            parent::_construct();
            $this->setId('distributor_tabs');
            $this->setDestElementId('edit_form');
            $this->setTitle($this->__('Distributor Setup'));
      }
}
```


b) Add the tab definitions in the `_beforeToHtml()` method to the class.

```
class Training_Distributor_Block_Adminhtml_Distributor_Edit_Tabs
      extends Mage_Adminhtml_Block_Widget_Tabs
{
      // snip...
      protected function _beforeToHtml()
      {
            $this->addTab('general_section', array(
                        'label'   => $this->__('General'),
                        'title'   => $this->__('General'),
                        'content' => $this->getLayout()

-
>createBlock('training_distributor/adminhtml_distributor_edit_tab_general')
                              ->toHtml(),
            ));

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

Step 3. Create the block for the form contents

a) Copy the content from *Training/Distributor/Block/Adminhtml/Edit/Form.php* (from the previous exercise) to a new block class definition at *Training/Distributor/Block/Adminhtml/Edit/Tab/General.php*, adjust the class name accordingly so that it matches the filename, and remove the `setUseContainer()` call from `_prepareForm()`, as this will still be part of the original content form block.

b) Next, change the *Training/Distributor/Block/Adminhtml/Edit/Form.php* class.

```
class Training_Distributor_Block_Adminhtml_Distributor_Edit_Form
      extends Mage_Adminhtml_Block_Widget_Form
{
      protected function _prepareForm()
      {
            $form = new Varien_Data_Form(array(
                        'id'      => 'edit_form',
                        'action'  => $this->getUrl('*/*/save',
                                    array('id' => $this->getRequest()-
>getParam('id'))),
                        'method'  => 'post',
                        'enctype' => 'multipart/form-data',
            ));
            $form->setUseContainer(true);
            $this->setForm($form);
            return parent::_prepareForm();
      }
}
```

Note: All fieldset and field related method calls have been removed, as well as the `_getFormData()` method.

c) Create a new distributor or edit an existing distributor and verify that the page is now a tabbed display.

