

Exercise Solutions

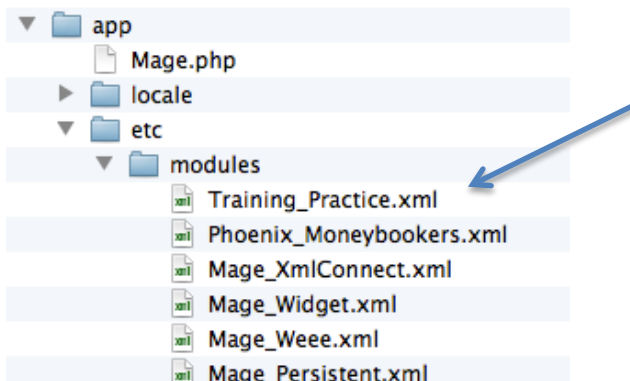
Section 2 Lesson 6. Configuration XML

Exercise 1. Register and Create an Empty Module

Use your IDE to do this exercise.

Step 1. Create a module registration file named *Training_Practice.xml*

- a) Create a new XML file named `app/etc/modules/Training_Practice.xml`.



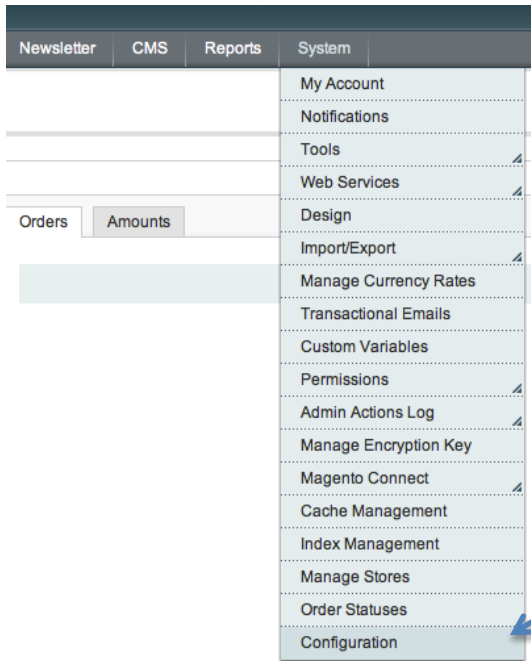
- b) Add the following content to the file `app/etc/modules/Training_Practice.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <modules>
        <Training_Practice>
            <active>true</active>
            <codePool>local</codePool>
        </Training_Practice>
    </modules>
</config>
```

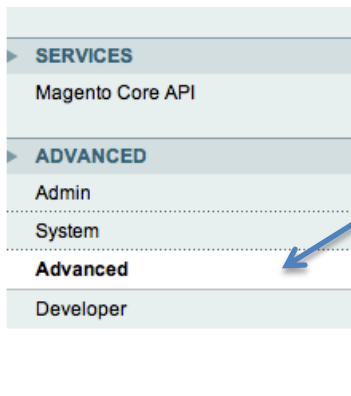
Step 2. Check that your module is listed in the system configuration section Advanced > Advanced

- a) Log in to the Admin interface.

b) From the top navigation menu, choose System > Configuration.

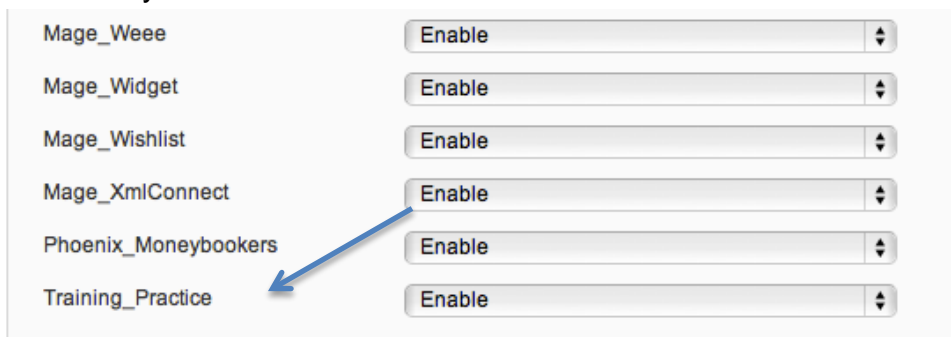


c) Click the “Advanced” tab on the bottom left.



d) Click on the blue title bar “Disable module output” to expand it.

e) Check that your module is listed.



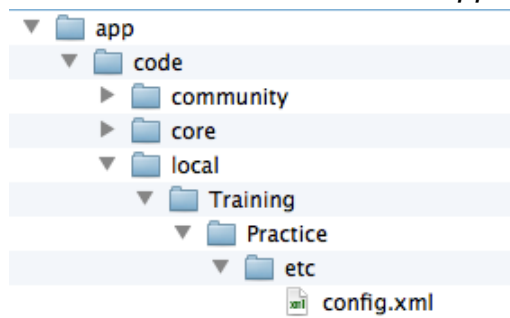
- f) If your module isn't listed, check the following:
- Check that your configuration cache is disabled.
 - Verify that the XML, the file name, and the file location are correct.

Step 3. Create the module directory and an *etc/config.xml* file and specify a version 0.1.0 for your module

- a) Create the directory *app/code/local/Training/Practice/etc/*

If intermediate directories don't exist, go ahead and create them.

- b) Create a new XML file named *app/code/local/Training/Practice/etc/config.xml*.



- c) Add the following content to the file *app/code/local/Training/Practice/etc/config.xml*.

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <modules>
    <Training_Practice>
      <version>0.1.0</version>
    </Training_Practice>
  </modules>
</config>
```

Step 4. Make sure your config.xml file is being read by creating some invalid markup to trigger an XML parse error

- a) Remove the > character from the closing *</config>* node in the file *app/code/local/Training/Practice/etc/config.xml* and save the change. The last line now should read *</config*

b) Reload any Magento page, frontend or Admin interface, to display the parse error.

```
Warning: simplexml_load_string() [function.simplexml-load-string]: Entity: line 8: parser error : expected '>' in  
#0 [internal function]: mageCoreErrorHandler(2, 'simplexml_load...', '/Volumes/Work/w...', 510, Array)  
#1 /Volumes/Work/workspace-ssd/EE_Lite/lib/Varien/Simplexml/Config.php(510): simplexml_load_string('loadString('loa  
#4 /Volumes/Work/workspace-ssd/EE_Lite/app/code/core/Mage/Core/Model/Config.php(318): Mage_Core_Model_Config->loadM  
#5 /Volumes/Work/workspace-ssd/EE_Lite/app/code/core/Mage/Core/Model/App.php(405): Mage_Core_Model_Config->loadModu  
#6 /Volumes/Work/workspace-ssd/EE_Lite/app/code/core/Mage/Core/Model/App.php(336): Mage_Core_Model_App->_initModule  
#7 /Volumes/Work/workspace-ssd/EE_Lite/app/Mage.php(640): Mage_Core_Model_App->run(Array)  
#8 /Volumes/Work/workspace-ssd/EE_Lite/index.php(80): Mage::run('', 'store')  
#9 {main}
```

c) If the error isn't displayed, check the following:

- Check that the configuration cache is disabled.
- Make sure that the path to the *config.xml* file matches the module name and code pool specified in the registration file.
- Check that the folder name and file name are typed in the correct case.

Step 5. When you are sure that Magento is merging your configuration into the configuration DOM tree, fix the XML

a) Add the > character to the closing </config> node again.

b) Reload and confirm that the simplexml parse error is no longer displayed.

Exercise 2. Configuration & Stores

Step 1. Specify a default value for the system configuration option contacts/email/recipient_email in your etc/config.xml

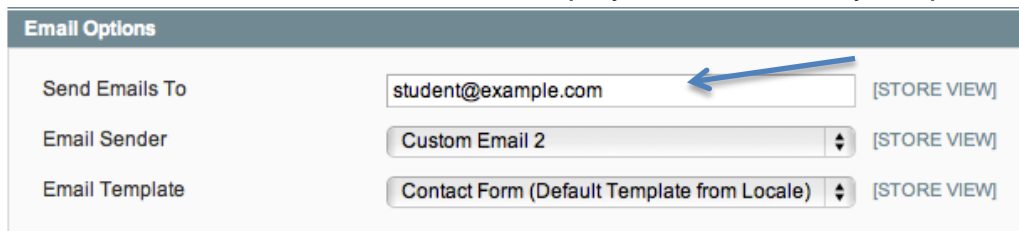
a) Open the file *app/code/local/Training/Practice/etc/config.xml* that you created in the previous exercise.

b) Add the following code section to the <config> node in that file.

```
<default>  
    <contacts>  
        <email>  
            <recipient_email>student@example.com</recipient_email>  
        </email>  
    </contacts>  
</default>
```

Step 2. Check that it is displayed in the Admin system configuration under Contacts > Email Options

- a) Log in to the Magento Admin interface.
- b) Navigate to System > Configuration.
- c) In the left navigation, click the Contacts tab.
- d) Click on the blue bar titled “Email Options” to expand it.
- e) Check that the field “Send Emails to” displays the value that you specified.



Email Options	
Send Emails To	<input type="text" value="student@example.com"/> [STORE VIEW]
Email Sender	<select value="Custom Email 2"></select> [STORE VIEW]
Email Template	<select value="Contact Form (Default Template from Locale)"></select> [STORE VIEW]

- f) If the value isn't displayed, check the following:
 - Check that the configuration cache is disabled.
 - Make sure that the XML is correct. The `<default>` node must be located inside the `<config>` node.

Step 3. Specify a store view scope value in the config file and check that it is shown in the Admin system configuration for the store scope

- To be able to specify a store scope value, first you need to find out a store code. These codes are installation dependent.
- Navigate to System > Manage Stores.
- Click on a Store View name to view its details.

Manage Stores [Create Website](#) [Create Store](#) [Create Store View](#)

Page of 1 pages | View per page | Total 3 records found [Reset Filter](#) [Search](#)

Website Name	Store Name	Store View Name
Main Website	Main Website Store	English
Main Website	Main Website Store	French
Main Website	Main Website Store	German

If your list of stores looks different, just choose any store view that is associated with the main website.

- Take note of the exact store view code.

Store View Information

Store *

Name *

Code *

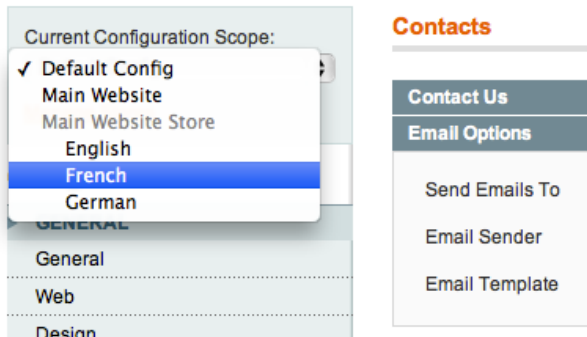
Status *

Sort Order

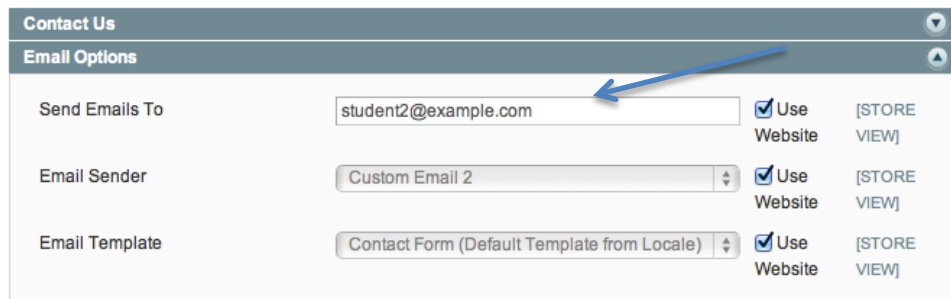
- Add the following code to the `<config>` node in your modules `config.xml` file. The first node inside of `<stores>` needs to exactly match the store code.

```
<stores>
  <french>
    <contacts>
      <email>
        <recipient_email>student2@example.com</recipient_email>
      </email>
    </contacts>
  </french>
</stores>
```

- f) In the Admin interface, navigate to the contact email configuration again. Notice that the displayed contact email still matches the default configuration.
- g) Switch the store view scope using the dropdown menu in the top left corner.



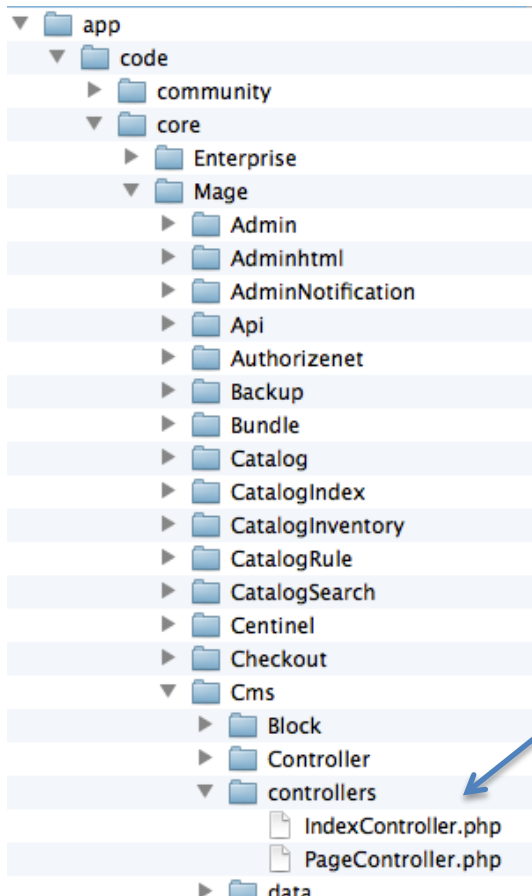
- h) Notice that the contact email now matches the store view scope value that you specified in the configuration XML.



Exercise 3. Reading Configuration Values

Step 1. Open `Mage_Cms_IndexController::indexAction()` (the home page) and display `contacts/email/recipient_email` using the first two methods discussed in this lesson

a) Open the file `app/code/core/Mage/Cms/controllers/IndexController.php`.



b) Navigate to the method `public function indexAction()`.

c) Insert this code at the beginning of the method.

```
public function indexAction($coreRoute = null)
{
    echo Mage::getStoreConfig('contacts/email/recipient_email');
```

d) Reload the home page, and notice that it displays the option value at the top of the screen.



- e) Switch to the store you specified as a store scope value, and notice that Magento now displays a store scope value.



- f) Switch back to the default store view.
- g) Specify a store code as the second parameter to the `Mage::getStoreConfig()` method call in `indexAction()`.

```
public function indexAction($coreRoute = null)
{
    echo Mage::getStoreConfig(
        'contacts/email/recipient_email', 'french'
    );
}
```

- h) Reload the frontend, and notice that now the store scope value is displayed regardless of the selected store view.



- i) Change the code in the method as follows.

```
public function indexAction($coreRoute = null)
{
    $store = Mage::app()->getStore();
    echo $store->getConfig('contacts/email/recipient_email');
}
```

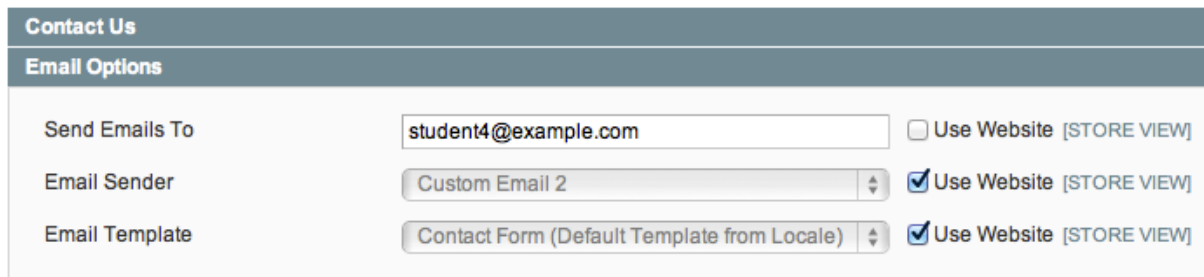
- j) Reload the home page, and notice that this matches the `Mage::getStoreConfig()` call without specifying a store.

```
public function indexAction($coreRoute = null)
{
    $store = Mage::app()->getStore('french');
    echo $store->getConfig('contacts/email/recipient_email');
}
```

- k) Change the `Mage::app()->getStore()` call to return a specific store view by passing the store code as an argument.
- l) Reload the home page, and notice that this matches the call to `Mage::getStoreConfig()` with a specified store view.

Step 2. Save a different contact email value through the Admin interface, first in default scope, then in a store scope, and notice how it overrides the values from the config files

- a) In the Magento Admin interface, navigate to the Contact Email options and switch to the default scope (if that isn't the currently selected one).
- b) Specify a different email address for the contact email recipient and click Save.
- c) In the frontend, switch to the default store scope. Notice that the new default value saved in the Admin interface overrides the default value specified in your *config.xml* file.
- d) In the frontend, switch to the store for which you specified a store scope value in your *config.xml* file, and notice that it still takes precedence over the saved default value.
- e) In the Admin interface, switch to store view scope, clear the "Use Website" checkbox, specify another email address, and save it.



Contact Us	
Email Options	
Send Emails To	student4@example.com <input type="checkbox"/> Use Website [STORE VIEW]
Email Sender	Custom Email 2 <input checked="" type="checkbox"/> Use Website [STORE VIEW]
Email Template	Contact Form (Default Template from Locale) <input checked="" type="checkbox"/> Use Website [STORE VIEW]

- f) In the frontend, switch to that store view, reload, and notice that now the store view scope value overrides the one specified in the config XML.

Step 3. Find that option value in the `core_config_data` table and notice how Magento stores different scope values

- a) Access phpMyAdmin or any other MySQL interface of your choice (for example, MySQL Workbench, NetBeans, or Sequel Pro).
- b) Select the Magento database and then select the table `core_config_data`. If you specified a table prefix during the Magento installation, adjust the table name accordingly.

- c) Filter the table to display only records where the path column matches the value `contacts/email/recipient_email`

```
mysql> select * from core_config_data where path = 'contacts/email/recipient_email';
```

config_id	scope	scope_id	path	value
12	default	0	contacts/email/recipient_email	student3@example.com
16	stores	2	contacts/email/recipient_email	student4@example.com

2 rows in set (0.00 sec)

- d) Notice that the `scope_id` for the row with the store scope value matches the ID of the store view.

```
mysql> select * from core_store where store_id = 2;
```

store_id	code	website_id	group_id	name	sort_order	is_active
2	french	1	1	French	0	1

1 row in set (0.00 sec)

Step 4. Access `contacts/contacts/enabled` (a Boolean option) with `Mage::getStoreConfigFlag()`

- a) Change your code in the `indexAction()` method call as follows.

```
public function indexAction($coreRoute = null)
{
    var_dump(Mage::getStoreConfigFlag('contacts/contacts/enabled'));
}
```

- b) Reload the frontend and notice that the returned value is a Boolean.

boolean true

ister.

Language: English (Cha

Step 5. Display the value of the `<codePool>` node for your module's configuration using `Mage::getConfig()->getNode()`

- a) Change the code in the `indexAction()` method call as follows.

```
public function indexAction($coreRoute = null)
{
    echo Mage::getConfig()
        ->getNode('modules/Training_Practice/codePool');
}
```

Notice that the full XPath inside of the `<config>` node is specified.

- b) Reload the home page and notice that the configuration node value is displayed.



Step 6 (Optional). Display the default contact email recipient and store scope values using `Magento::getConfig()->getNode()`

- a) Change the code in the `indexAction()` method call as follows.

Again, notice that the full XPath inside of the `<config>` node is specified.

```
public function indexAction($coreRoute = null)
{
    echo Magento::getConfig()
        ->getNode('default/contacts/email/recipient_email');
```

- b) Reload the home page. Notice that the default scope value is displayed regardless of the selected store view.
- c) Change the code to read the store scope value:
`->getNode('stores/french/contacts/email/recipient_email')`
- d) Reload the frontend and notice that the specified store view value is displayed regardless of the store view selected in the store switcher.

Step 7. Revert your changes in core files

- Remove all your additions to the `indexAction()` method in the file `app/code/core/Mage/Cms/controllers/IndexController.php`