Exercise Solutions

Section 5 Lesson 2. Setup Scripts & Update Scripts

Exercise 1: Declare a Setup Resource & Version Number

Use your IDE to do this exercise.

Step 1. Declare a setup resource `training_distributor_setup` and a version number of 0.0.1 in your *config.xml*

a) Add the following bold content to the existing content in *app/code/local/Training/Distributor/etc/config.xml.*

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
      <modules>
            <Training_Distributor>
                  <version>0.0.1</version>
            </Training_Distributor>
      </modules>
      <global>
            <resources>
                  <training_distributor_setup>
                        <setup>
                              <module>Training_Distributor</module>

      <class>Mage_Core_Model_Resource_Setup</class>
                        </setup>
                  </training_distributor_setup>
            </resources>
      </global>
      <!-- snip -->
</config>
```


b) Clear the configuration cache if necessary and check that the `core_resource` table is updated with your setup resource handle and version number.

```
mysql> SELECT * FROM `core_resource` WHERE `code` = 'training_distributor_setup';
+----------------------------+---------+---------------+
| code                       | version | data_version |
+----------------------------+---------+---------------+
| training_distributor_setup | 0.0.1   | 0.0.1         |
+----------------------------+---------+---------------+
1 row in set (0.00 sec)
```

Exercise 2: Create a Table with an Install Script for the `Training_Distributor` Module

Use your IDE to do this exercise.

Step 1. Create the `sql/training_distributor_setup` folders under your *module directory app/code/local/Training/Distributor/*

- Create the directory containing the setup scripts according to your configuration setup resource name.

Step 2. Create an install script *install-0.0.1.php* in the `training_distributor_setup` directory

- Create the install script according to your module's version number. Be sure to use dashes and no underscores to separate the version number `0.0.1` from the prefix in the file name.

Step 3. Create a `training_distributor_entity` table using DDL methods

a) In the install script, add the best-practice skeleton: alias `$this` with `$installer` and call both the `startSetup()` and `endSetup()` methods.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

//DDL operations will be added here

$installer->endSetup();
```


b) Next, retrieve the table name from the module configuration and assign it to a variable for use in the table operations.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

$installer->endSetup();
```

This will read the value from the table node inside `distributor` node under the Training_Distributor module's `resourceModel` node, i.e., *global/models/training_distributor_resource/entities/distributor/table*.

c) The initial database operation is a check to see if the table is already present. If it is, remove it.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

if ($installer->getConnection()->isTableExists($tableName))
{
    $installer->getConnection()->dropTable($tableName);
}

$installer->endSetup();
```

This is approach will ease rerunning the script during development. For production, once the module is registered, the script will not run again.

d) Use the `newTable()` method to instantiate a table creation DDL object and then add column, index, and comment syntax using method chaining.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

if ($installer->getConnection()->isTableExists($tableName)){
    $installer->getConnection()->dropTable($tableName);
}

$table = $installer->getConnection()->newTable($tableName)
       ->addColumn('entity_id', Varien_Db_Ddl_Table::TYPE_INTEGER, null,
              array(
                     'unsigned' => true,
                     'nullable' => false,
                     'primary' => true,
                     'identity' => true
              ), 'ID')
       ->addColumn('name', Varien_Db_Ddl_Table::TYPE_TEXT, '255',
              array(
                     'nullable' => false,
                     'default' => ''
              ), 'Distributor Name')
       ->addColumn('email', Varien_Db_Ddl_Table::TYPE_TEXT, '255',
              array(
                     'nullable' => false,
                     'default' => ''
              ), 'Distributor Email')
       ->addColumn('created_at', Varien_Db_Ddl_Table::TYPE_DATETIME, null,
              array(
                     'nullable' => false,
              ), 'Created At')
       ->addColumn('updated_at', Varien_Db_Ddl_Table::TYPE_DATETIME, null,
              array(
                     'nullable' => false,
              ), 'Modified At')
       ->addIndex(
              $installer->getIdxName($tableName, array('name')),
              array('name'),
              array('type' => Varien_Db_Adapter_Interface::INDEX_TYPE_UNIQUE)
       )
       ->addIndex(
              $installer->getIdxName($tableName, array('email')),
              array('email'),
              array('type' => Varien_Db_Adapter_Interface::INDEX_TYPE_UNIQUE)
       )
       ->setComment('Distributor Training Example Entity');

$installer->endSetup();
```


e) Finally, call the `createTable()` method to create the table according to the configured DDL table creation object.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

if ($installer->getConnection()->isTableExists($tableName)){
    $installer->getConnection()->dropTable($tableName);
}

//snip...

$installer->getConnection()->createTable($table);

$installer->endSetup();
```

Step 4. Remove the existing record for your setup resource from the `core_resource` table

- Remove the module setup resource record from the `core_resource` table so that the install script will be run again on the next reload.

```
DELETE FROM `core_resource` WHERE `code` = 'training_resource_setup';
```

Step 5. Clear the cache if necessary, browse any page, and check that the table was created

---

Note: In the event that the entry in `core_resource` is *not* created during the reload, the culprit is one of the following: configuration, folder structure, filename, or syntax.

- **Configuration:** Ensure that the setup resource node contains a `<setup>` node inside of it.
- **Folder structure:** Ensure that the setup resource node name `training_distributor_setup` under `global/resources` matches the folder name under the *sql* folder.

- **Filename:** Ensure that the filename is correctly formatted and ends in *.php*, and that a dash, *not* an underscore, is used in the file names. For install scripts, ensure that the version number is less than or equal to the version number specified in the filename.
- **Syntax:** An excellent way to debug is to add a `die()`statement to the top to ensure that it's being processed. If it is being processed, the issue must be with syntax.

Exercise 3: Create a Regular Upgrade Script to Add an Additional Table Column

Use your IDE to do this exercise.

Step 1. Create an upgrade script *upgrade-0.0.1-0.0.2.php*

- The file should be created adjacent to the install script that was created in the previous exercise.

Step 2. Add an additional column to the `training_distributor_entity` table using DDL adapter methods

a) Follow the same best practices as in the previous exercise: alias `$this` with `$installer` and add the `startSetup()` and `endSetup()` method calls.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

$installer->endSetup();
```


b) Using the table adapter, `$this->getConnection()`, call the `addColumn()` method and pass in the configuration parameters to add a text column for comments.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$tableName = $installer->getTable('training_distributor/distributor');

$installer->getConnection()->addColumn($tableName, 'comment', array(
      'type' => Varien_Db_Ddl_Table::TYPE_TEXT,
      'nullable' => true,
      'length' => '1k',
      'comment' => 'Comment Field'
));

$installer->endSetup();
```

Step 3. Upgrade the version number in the config.xml file to 0.0.2

- Update the `<version>` tag in the modules *etc/config.xml* file to `0.0.2`.

Step 4. Clear the configuration cache and hit any page

- Reload any Magento page to trigger the upgrade script.

Step 5. Verify that it works

a) Check the schema for your table (add table prefix if necessary). Your columns should be present, similar to this example.

```
mysql> DESCRIBE `training_distributor_entity`;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| entity_id  | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name       | varchar(255)     | NO   | UNI |         |                |
| email      | varchar(255)     | NO   | UNI |         |                |
| created_at | datetime         | NO   |     | NULL    |                |
| updated_at | datetime         | NO   |     | NULL    |                |
| comment    | text             | YES  |     | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```


b) If the changes are not present, check the `core_resource` table to determine if the configuration change (version number change) was registered. If the updated version number is registered in the database:

    1. Edit the values in the database in order to trigger the script to run again.
    2. Verify that the filename is correct.
    3. Add a `die()` statement to verify that the file is being processed.
    4. Check syntax.

Exercise 4: Create a Data Upgrade Script to Set Configuration Data

Use your IDE to do this exercise.

Step 1. Create the directory *data/training_distributor_setup*

- Create the directory *data/training_distributor_setup* in the module directory.

Step 2. Create an upgrade script *data-upgrade-0.0.2-0.0.3.php*

- Create the file data/training_distributor_setup/data-upgrade-0.0.2-0.0.3.php.

Step 3. For each website scope, set the configuration value to `0` for the option `shipping/option/checkout_multiple` using `$installer->setConfigData()`

a) As in the previous exercises, employ best practices.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

$installer->endSetup();
```


b) Next, retrieve all of the available website scopes and set up a `foreach` loop.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

foreach (Mage::app()->getWebsites() as $website) {

}

$installer->endSetup();
```

c) Finally, set the configuration value for each website in the loop.

```
<?php

/* @var $installer Mage_Core_Model_Resource_Setup */

$installer = $this;
$installer->startSetup();

foreach (Mage::app()->getWebsites() as $website) {
      $installer->setConfigData(
            'shipping/option/checkout_multiple',
            0,
            'websites',
            $website->getId()
      );
}

$installer->endSetup();
```

Step 4. Upgrade the version number in your *config.xml* to `0.0.3`

- Upgrade the `<version>` tag value in your modules *etc/config.xml* to `0.0.3`.

Step 5. Clear the configuration cache and hit any page

- Clear the configuration cache if necessary and reload any Magento page to trigger execution of the setup script.

Step 6. Verify that it worked

- Check that the configuration values you added are present in the `core_config_data` table.

> **Note:** Troubleshooting procedures are same as in previous exercises.

