# Exercise Solutions

# Section 3 Lesson 5. Module Initialization

## Exercise 1: Recap Modules & Class Groups

Use your IDE to do this exercise.

### Step 1. Create a new module named `Training_Recap` (including folders, two config files)

- Refer to the exercise solutions for Section 2, Lesson 6, Exercise 1 – Configuration XML for instructions on how to create a new module. Adjust the module name so that the module is named `Training_Recap`. After you complete that exercise you will have a skeleton module to work with for this exercise.

### Step 2. Register a class group `training` for models, blocks, and helpers

a) Open the *config.xml* file of the `Training_Recap` module.

b) Add the XML configuration to configure the class groups inside the `config` node.

```
<global>
  <models>
    <training_recap>
      <class>Training_Recap_Model</class>
    </training_recap>
  </models>
  <blocks>
    <training_recap>
      <class>Training_Recap_Block</class>
    </training_recap>
  </blocks>
  <helpers>
    <training_recap>
      <class>Training_Recap_Helper</class>
    </training_recap>
  </helpers>
</global>
```

This will map the class group `training_recap` to the *Model*, *Block,* and *Helper* directories inside of the `Training_Recap` module, with the help of the Magento autoloader.

### Step 3. Create an empty model file *Sample.php* in the *Model* folder

a) Create the *Model* directory for the module *app/code/local/Training/Recap/Model/*.

b) Create the file *app/code/local/Training/Recap/Model/Sample.php*.

### Step 4. Add a class declaration that extends from `Mage_Core_Model_Abstract`

- Open the file *app/code/local/Training/Recap/Model/Sample.php* and add the following code.

```
<?php

class Training_Recap_Model_Sample
      extends Mage_Core_Model_Abstract
{
}
```

### Step 5. Register a frontend route `recap` for your module

```
public function indexAction()
{

}
```

- Refer to the exercise solutions for Section 3, Lesson 4, Exercise 1 – Request Routing for instructions on how to create a new frontend route. Adjust the module name so that the module is named `Training_Recap`, change the controller name to `IndexController`, and change the front name so it is named `recap`.

### Step 6. Create an index controller with an `indexAction()` method in your module

- These steps are also covered in the exercise solutions for Section 3, Lesson 4, Exercise 1.

### Step 7. In your action method, instantiate the model using `Mage::getModel()`

- Change the contents of the `indexAction()` method in the controller to match the following code.

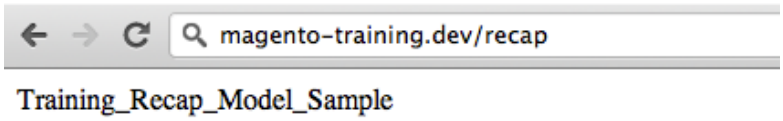
```
public function indexAction()
{
  $model = Mage::getModel('training_recap/sample');
}
```

### Step 8. Display the model class by using `get_class()` and set it on the response body

a) Change the contents of the `indexAction()` method in the controller to match the following code.

```
public function indexAction()
{
  $model = Mage::getModel('training_recap/sample');
  $this->getResponse()->setBody(get_class($model));
}
```

b) Open the browser and call the front name `recap` of your module. You do not need to specify a controller or action, because `index` is the default for both. However, if you choose to specify the full path, it is `recap/index/index`.




Training_Recap_Model_Sample

You should see the `Training_Recap_Model_Sample` output on an otherwise empty page.

## Exercise 2: Module Dependency & Activation

You will be working with the `Training_Recap` module from the previous exercise.

### Step 1. Make your module `Training_Recap` dependent on `Mage_Log`

a) Open the file *app/etc/modules/Training_Recap.xml.*

b) Add the XML to declare the dependency on `Mage_Log`.

```
<config>
  <modules>
    <Training_Recap>
      <active>true</active>
      <codePool>local</codePool>
      <depends>
        <Mage_Log/>
      </depends>
    </Training_Recap>
  </modules>
</config>
```

**Step 2. Disable the `Mage_Log` module and verify that your module doesn't work anymore**

a) Open the file *app/etc/modules/Mage_All.xml* and go to the declaration of `Mage_Log`.

b) Change the `<active>` node to `0` and save the file.

```
</Mage_GoogleCheckout>
<Mage_Log>
  <active>0</active>
  <codePool>core</codePool>
  <depends>
    <Mage_Core/>
    <Mage_Customer/>
  </depends>
</Mage_Log>
<Mage_Backup>
```

c) Clear the configuration cache, and on the next reload Magento will display an unmet dependency exception.


d) Core modules can be disabled without changing core files like *Mage_All.xml*. For now, revert all changes in the file by setting the `<active>` node back to `true`.

e) Reload the frontend to check that the site works again.

f) Add the following XML to your module's registration file *app/etc/modules/Training_Recap.xml* into the `<modules>` node.

```
<Mage_Log>
  <active>0</active>
</Mage_Log>
```

g) Reload the site to confirm that once again, the dependency isn't satisfied.

### Step 3. Re-enable the `Mage_Log` module and disable your own module

a) Comment out or remove the lines deactivating the `Mage_Log` module from your module registration file.

b) Set the `<active>` node inside of `<Training_Recap>` to `false`.

c) Confirm that your module configuration is no longer loaded by reloading the `recap` route of your site. Magento displays a 404 page.

d) Enable your `Training_Recap` module again, by setting the `<active>` node to `true`.

e) Confirm that the `Training_Recap` module is loaded again by reloading the `recap` route.

### Step 4. Enable your module but disable all local modules using the *app/etc/local.xml* file

a) Open the file *app/etc/local.xml*.

b) Set the node `config/global/disable_local_modules` to `true` and save the change.

c) Reload the `recap` route to confirm that once again, the `Training_Recap` module is not active.

d) There is no way to fulfill the objective while the `Training_Recap` module is in the local code pool. You need to put the module into the community code pool. To do so, move the directory *app/code/local/Training/Recap* to *app/code/community/Training/Recap*.

e) In the module registration file *app/etc/modules/Training_Recap.xml*, set the value of the node `config/modules/Training_Recap/codePool` to `community`.

f) Confirm that the `Training_Recap` module is loaded again by reloading the `recap` route.

g) Move your module back into the local code pool.