How To Create A Simple Module in Magento 2.0

A publication of

magestore
Magento Marketing Extensions

Mageno 2.0 was officially launched in 18th December 2014, which is a hot event to Magento community.

To understand more about which will be changed compared with the previous version, Magestore published a tutorial about creating a simple module in Magento 2.0.

1. Create a simple module in Magento 1.0
2. Magento 1.0 vs. Magento 2.0
3. Create a simple module in Magento 2.0

Create A Simple Module in Magento 1.0

Firstly, I want to mention about way to create a simple module in Magento 1.0 so that you can make a comparison between this version with the upcoming one: Magento 2.0

1

Declare your module in folder: app/etc/

```
<?xml version=”1.0″?>
<config>
        <modules>
                <Magento_Hello>
                        <active>true</active>
                        <codePool>local</codePool>
                </Magento_Hello>
        </modules>
</config>
```

2 Module Configuration

- Creat a controller : app/code/local/Magento/Hello/controllers/IndexController.php

```
class Magento_Hello_IndexController extends Mage_Core_Controller_Front_Action
{
    public function indexAction()
    {
     $this->loadLayout();
     $this->renderLayout();
    }
}
```

2 Module Configuration

- Creat a block : app/code/local/Magento/Hello/Block/Hello.php

```
class Magento_Hello_Block_Hello extends Mage_Core_Block_Template
{
  // write function here
}
```

2 Module Configuration

-- Creat a configuration file config.xml : app/code/local/Magento/Hello/etc/config.xml

```
<?xml version=”1.0″?>
<config>
    <modules>
        <Magento_Hello>
            <version>0.1.0</version>
        </Magento_Hello>
    </modules>
    <global>
    </global>
    <frontend>
        <routers>
            <magento_hello>
                <use>standard</use>
                <args>
                    <module>Magento_Hello</module>
                    <frontName>hello</frontName>
                </args>
            </magento_hello>
        </routers>
```

2 Module Configuration (continued)

-- Creat a configuration file config.xml : app/code/local/Magento/Hello/etc/config.xml

```
<layout>
        <updates>
            <hello>
                <file>hello.xml</file>
            </hello>
        </updates>
    </layout>
  </frontend>
  <global>
 <blocks>
        <hello>
            <class>Magento_Hello_Block</class>
        </hello>
    </blocks>
  </global>
</config>
```

3 Create a frontend template

- Write a file layout : app/design/frontend/default/default/layout/hello.xml

```
<layout version=”0.1.0″>
    <hello_index_index>
      <reference name=”root”>
<action method=”setTemplate”><template>page/1column.phtml</template></action>
      </reference>
      <reference name=”content”>
 <block type=”hello/hello” name=”hello” template=”hello/hello.phtml”/>
      </reference>
   </hello_index_index>
</layout>
```

3 Create a frontend template

- Push/Place content to file template hello.phtml

"This is a simple module in Magento 1.0"

When you access url http://localhost/magento20/hello/index/index, the browser will display: *"This is a simple module in Magento 1.0"*

Magento 1.0 vs. *Magento 2.0*

By looking at differences between Magento 1.0 & Magento 2.0, you can easily visualize the folder structure in Magento 2.0. Thus, making a simple module in Magento 2.0 is just a breeze. For deeper understand, move to the next part & practice.

Magento 1.0	Magento 2.0
Folder *app/code* includes subfolders: *core, community, local*	Folder *app/code* includes subfolders *Magento* and *Zend*. In *Magento 1.0*, *Magento* and*Zend* are subfolders of the folder *core*
Codes created by developers are located at *app/code/local or app/code/community*	Codes created by developers are written directly in *app/code*. There is no difference between local and community
Module declaration file is a xml file in *app/etc/modules* *Eg: Module Checkout in Magento Core is declared in app/etc/modules/Mage_All.xml*	Module declaration file is always *module.xml* which is written directly to folder etc in the module *Eg: module Checkout in Magento Core is declared in app/code/Magento/Checkout/etc/module.xml*
Layout and template are saved in folder app/design *Eg: app/design/frontend/default/default/layout*	Layout and template are saved in the folder “View” in the module. The folder is the same level with some folders like: Block, Controller, Helper, Model, etc. in the module *Eg: app/code/Magento/Hello/view/frontend/layout*

Create a simple module in Magento 2.0

By looking at differences between Magento 1.0 & Magento 2.0, you can easily visualize the folder structure in Magento 2.0. Thus, making a simple module in Magento 2.0 is just a breeze. For deeper understand, move to the next part & practice.

1

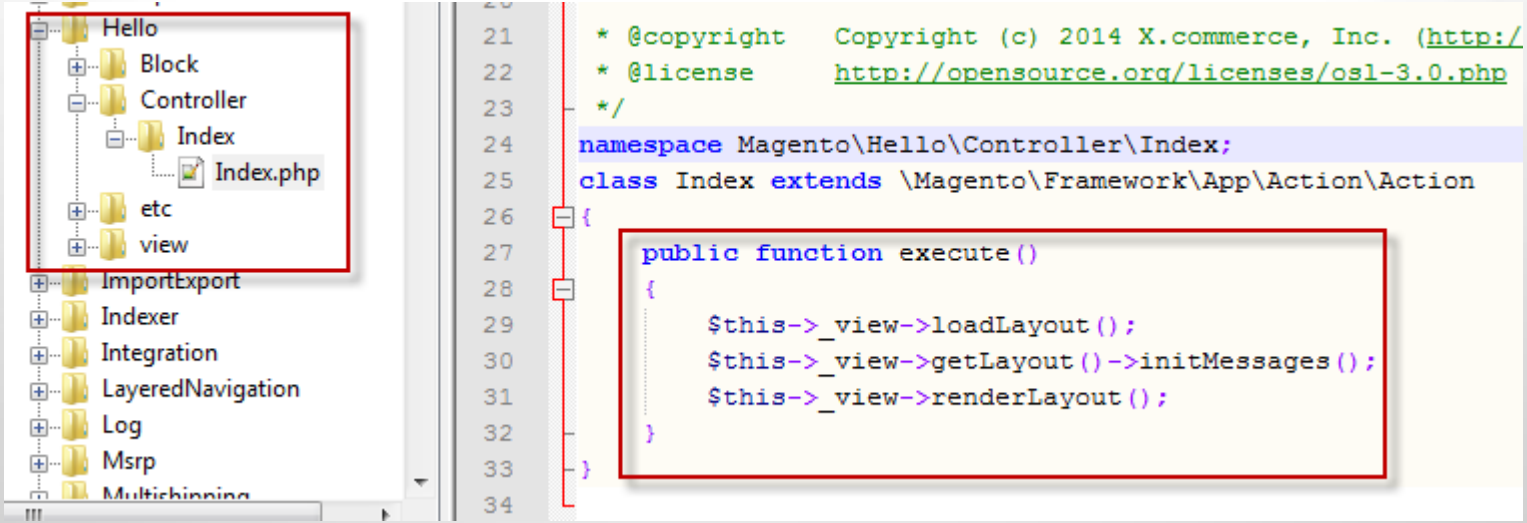
- Write the file module.xml in app/code/Magento/Hello/etc/module.xml to declare the module.

```
<config xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:noNamespaceSchemaLocation=”../../../../../lib/internal/Magento/Framework/Module/
etc/module.xsd”>
   <module name=”Magento_Hello” schema_version=”0.0.1″/>
</config>
```

2 Create controller and action

- Create the file Index.php in app/code/Magento/Hello/Controller/Index/Index.php

Folder Index plays the role of controller, while Index.php is action. The executive function of action Index is execute()



2 Create controller and action

- Create the file Index.php in app/code/Magento/Hello/Controller/Index/Index.php

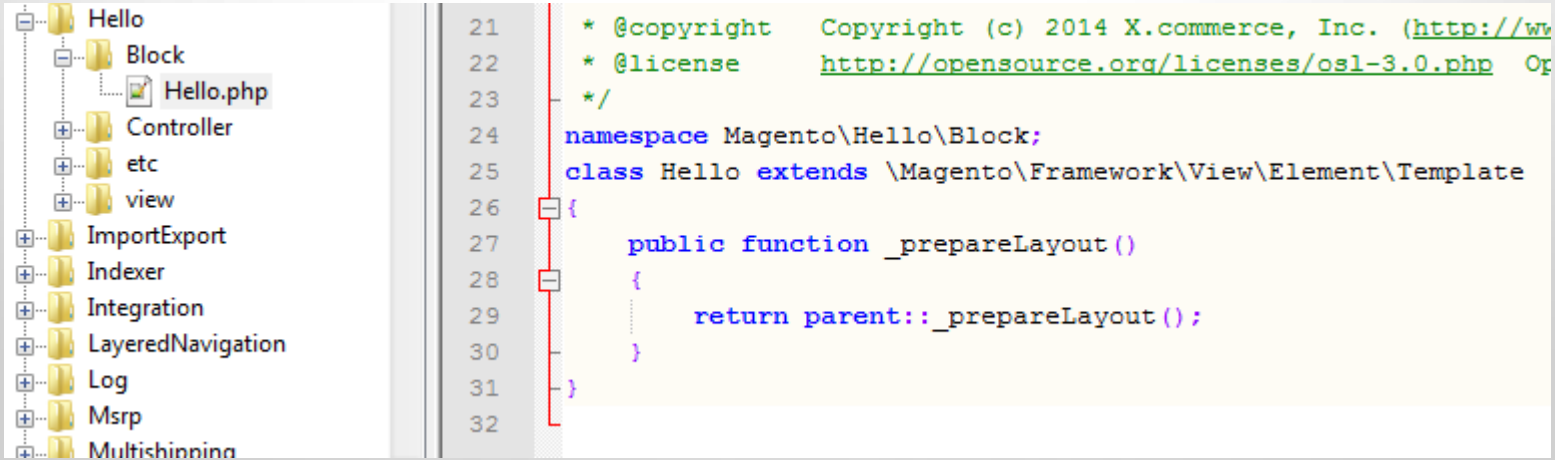
Folder Index plays the role of controller, while Index.php is action. The executive function of action Index is execute()

```
<config xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:noNamespaceSchemaLocation=”../../../../../lib/internal/Magento/Framework/Module/
etc/module.xsd”>
   <module name=”Magento_Hello” schema_version=”0.0.1″/>
</config>
```

2 Create a block

app/code/Magento/Hello/Block/Hello.php

```
namespace Magento\Hello\Block;
class Hello extends \Magento\Framework\View\Element\Template
{
public function _prepareLayout()
{
    return parent::_prepareLayout();
}
}
```



2 - Write configuration file: /app/code/Magento/Hello/etc/frontend/routes.xml

- In Magento 1.0, every information about router, events of frontend and backend is declared in Magento/Hello/etc/config.xml. However, in Magento 2.0, file config.xml only configures the default configuration value in tag <default>

+) Information about router of frontend will be reported in: Magento/Hello/etc/frontend/routes.xml (it is similar to backend)

+) Event of frontend will be declared in: Magento/Hello/ect/frontend/events.xml (it is similar to backend)

In the scope of a simple module, we only declare routers in Magento/Hello/etc/frontend/routes.xml

2

- Write configuration file: /app/code/Magento/Hello/etc/frontend/routes.xml

```
<config xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:noNamespaceSchemaLocation=”../../../../../../lib/internal/Magento/Framework/App/etc/routes.xsd”>
<router id=”standard”>
    <route id=”hello” frontName=”hello”>
        <module name=”Magento_Hello” />
    </route>
</router>
</config>
```

3 - Create a frontend template

- **Write a layout file: app\code\Magento\Hello\view\frontend\layout\hello_index_index.xml**

Name of layout file is really important in this step. It will be named after the structure:

router name_controlle namer_action name

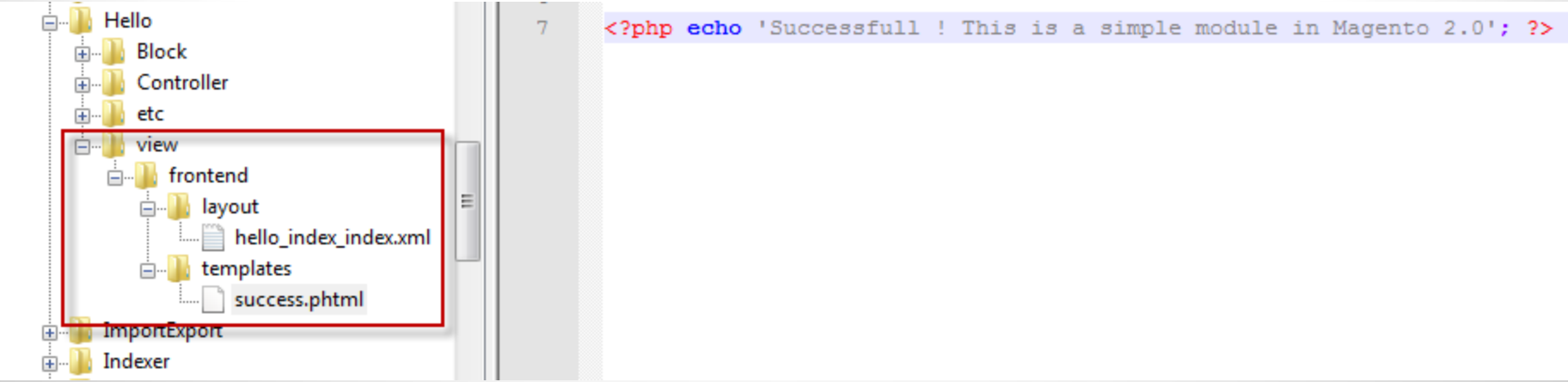
```
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../../../../../lib/internal/Magento/Framework/View
/Layout/etc/page_configuration.xsd">
<body>
    <referenceContainer name="content">
        <block class="Magento\Hello\Block\Hello" name="hello"
template="success.phtml">
        </block>
    </referenceContainer>
</body>
</page>
```

3 - Create a frontend template

3

Then, we create a file success.phtml as reporting in layout file:
app\code\Magento\Hello\view\frontend\templates\success.phtml

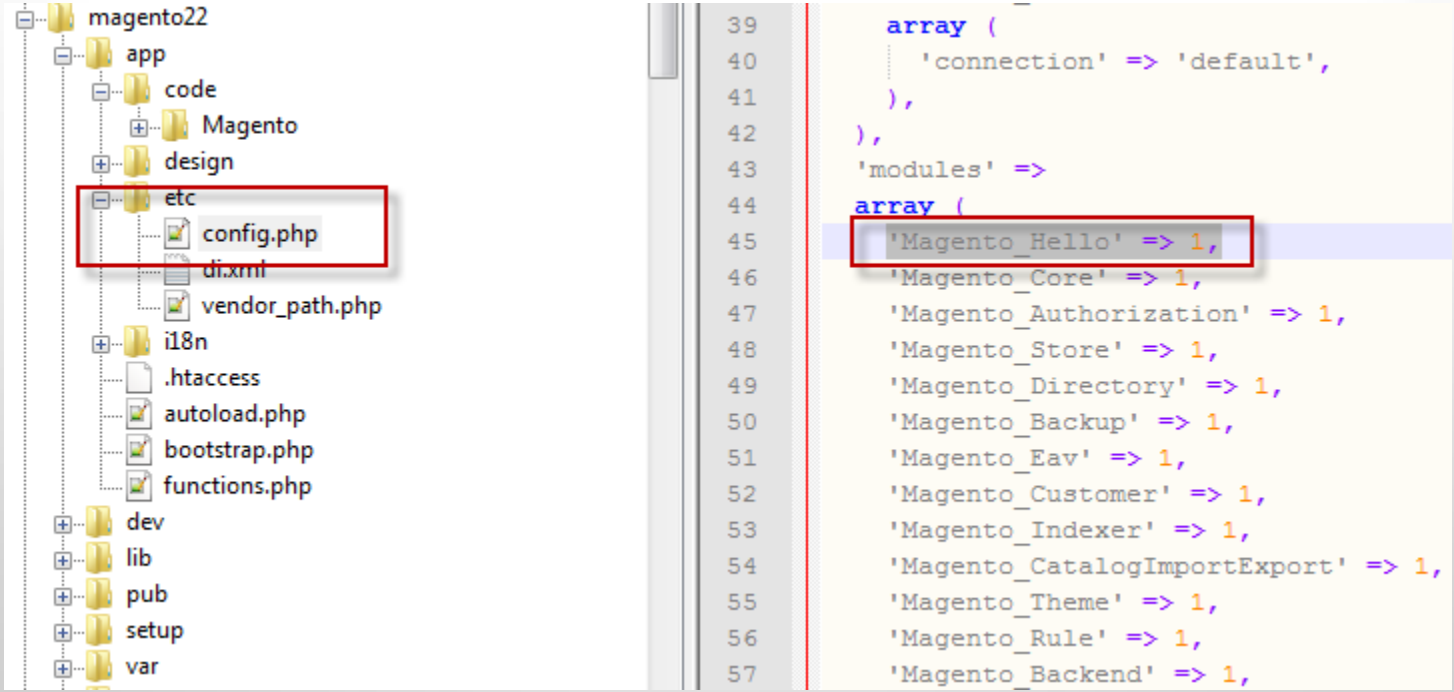
<?php echo 'Successfull ! This is a simple module in Magento 2.0'; ?>



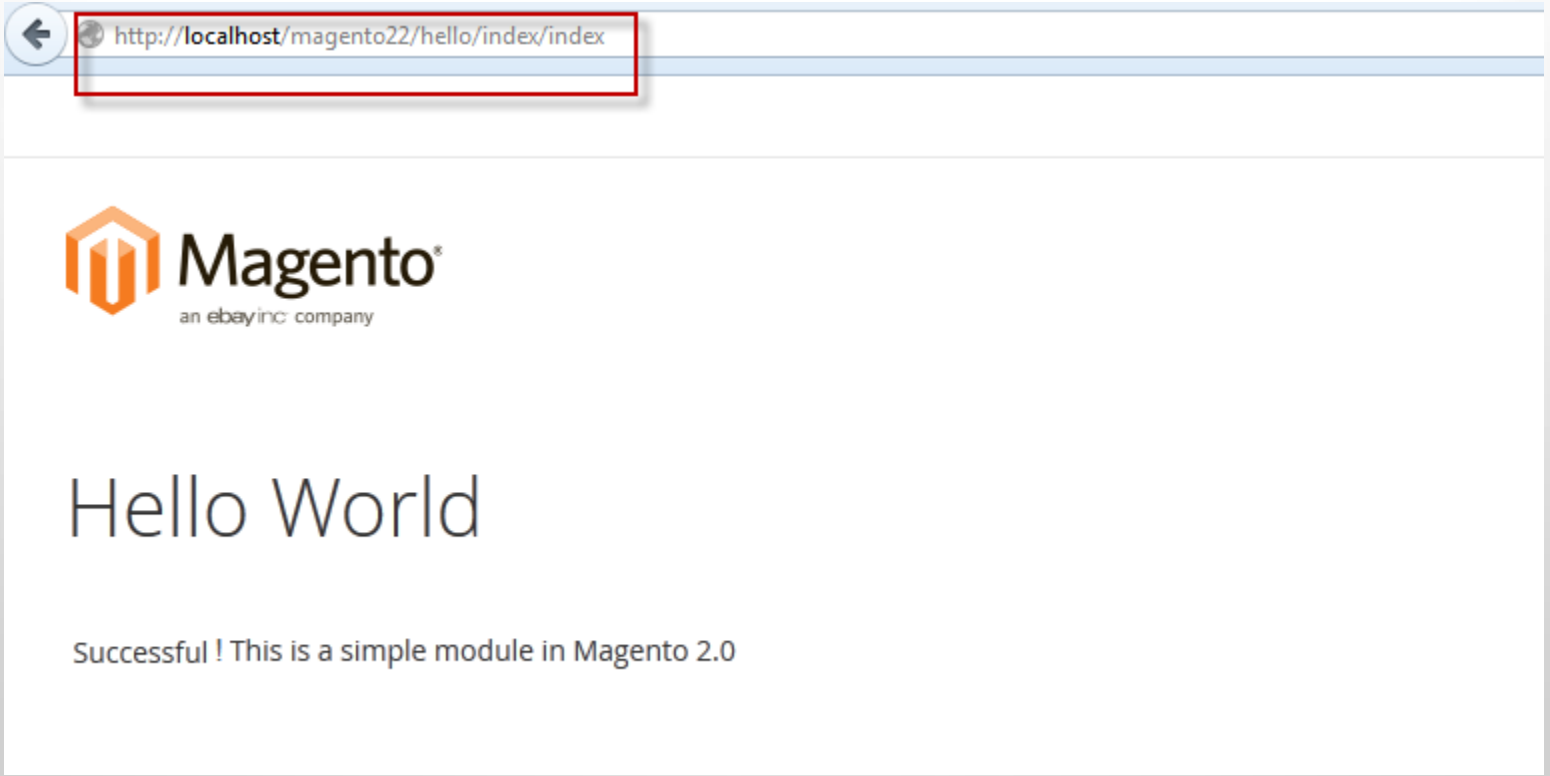
4 Activate Magento_Hello extension in configuration file

- Open the file **app/etc/config.xml**

- In the array **‘module’**, add the element: **‘Magento_Hello’ => 1,**



You have known all the steps to write a simple module in Magento 2.0. When you run the link: http://localhost/magento20/hello/index/index the result will be shown as the following:



... AND YOU'RE DONE!

- To download full pdf version, visit our blog post here
- To never miss any updates or tutorial about Magento 2.0, subscribe to our blog right now.

magestore
Magento Marketing Extensions