



How To Create A Simple Module in Magento 2.0

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

- Create 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

- Create 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

– Create 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>
        <routes>
            <magento_hello>
                <use>standard</use>
                <args>
                    <module>Magento_Hello</module>
                    <frontName>hello</frontName>
                </args>
            </magento_hello>
        </routes>
```

2

Module Configuration (continued)

– Create 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>
```



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 <i>app/code</i> includes subfolders: <i>core</i> , <i>community</i> , <i>local</i>	Folder <i>app/code</i> includes subfolders <i>Magento</i> and <i>Zend</i> . In <i>Magento 1.0</i> , <i>Magento</i> and <i>Zend</i> are subfolders of the folder <i>core</i>
Codes created by developers are located at <i>app/code/local</i> or <i>app/code/community</i>	Codes created by developers are written directly in <i>app/code</i> . There is no difference between local and community
Module declaration file is a xml file in <i>app/etc/modules</i> <i>Eg: Module Checkout in Magento Core is declared in app/etc/modules/Mage_All.xml</i>	Module declaration file is always <i>module.xml</i> which is written directly to folder <i>etc</i> in the module <i>Eg: module Checkout in Magento Core is declared in app/code/Magento/Checkout/etc/module.xml</i>
Layout and template are saved in folder <i>app/design</i> <i>Eg: app/design/frontend/default/default/layout</i>	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 <i>Eg: app/code/Magento/Hello/view/frontend/layout</i>

Create a simple module in Magento 2.0

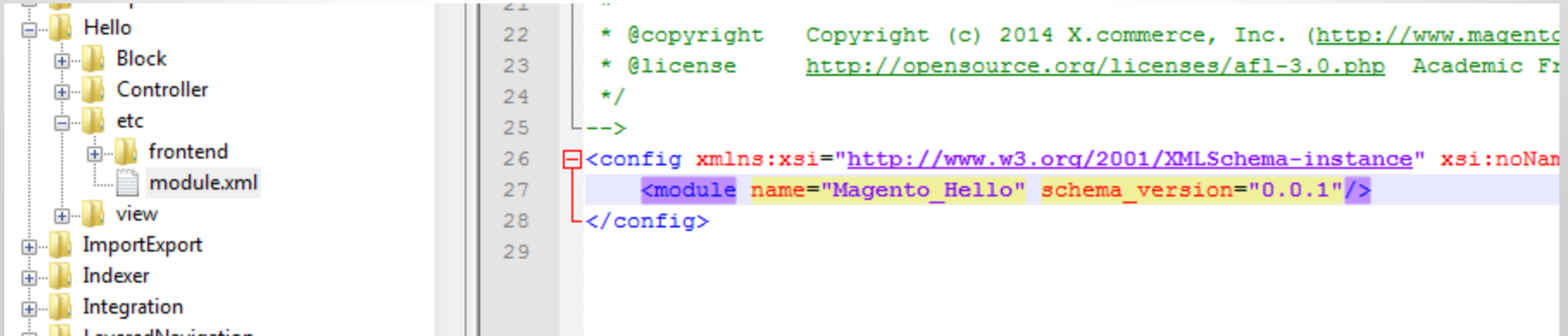


By looking at differences between Magento 1.0 & Magento 2.0, you can easily visualize the folder structure in Magento 2.0. Thus, making a simple module in Magento 2.0 is just a breeze. For deeper understand, move to the next part & practice.

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()



The screenshot displays a code editor with two panels. The left panel shows a file explorer with a tree structure. The 'Hello' folder is expanded, showing subfolders 'Block', 'Controller', 'etc', and 'view'. The 'Controller' folder is further expanded, showing an 'Index' folder which contains the 'Index.php' file. The right panel shows the code for 'Index.php'. The code starts with a copyright notice and a license link. It then defines a namespace 'Magento\Hello\Controller\Index' and a class 'Index' that extends '\Magento\Framework\App\Action\Action'. The class has a public function 'execute()' which calls three methods on the view object: 'loadLayout()', 'getLayout()->initMessages()', and 'renderLayout()'.

```
21 * @copyright Copyright (c) 2014 X.commerce, Inc. (http://)
22 * @license http://opensource.org/licenses/osl-3.0.php
23 */
24 namespace Magento\Hello\Controller\Index;
25 class Index extends \Magento\Framework\App\Action\Action
26 {
27     public function execute()
28     {
29         $this->_view->loadLayout();
30         $this->_view->getLayout()->initMessages();
31         $this->_view->renderLayout();
32     }
33 }
34
```

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
```

```
{
```

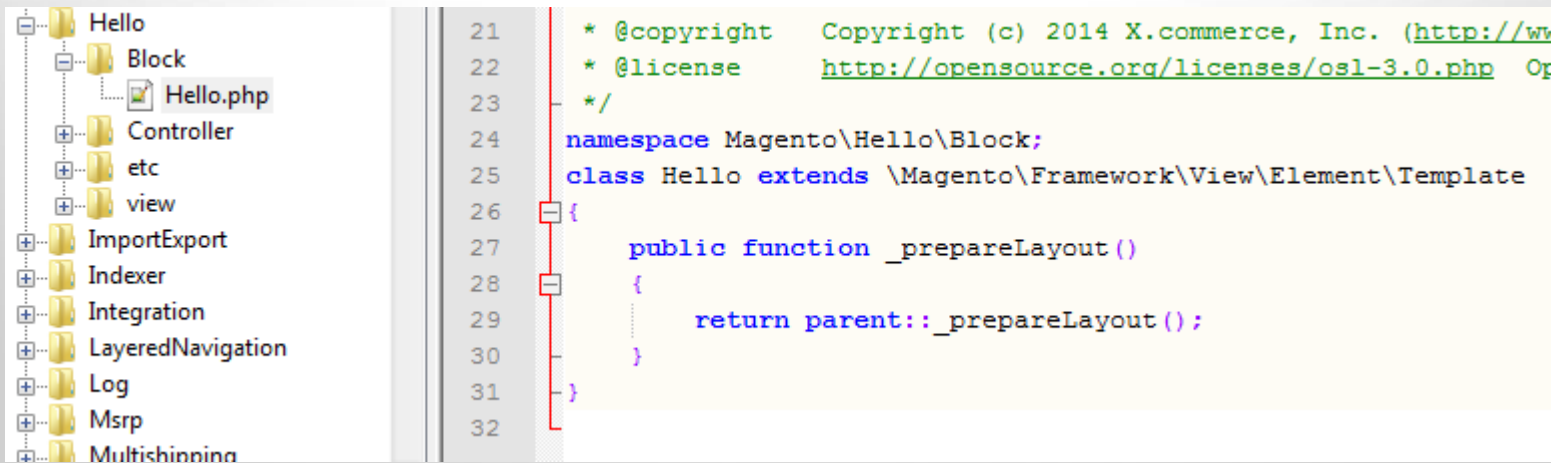
```
public function _prepareLayout()
```

```
{
```

```
    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/etc/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



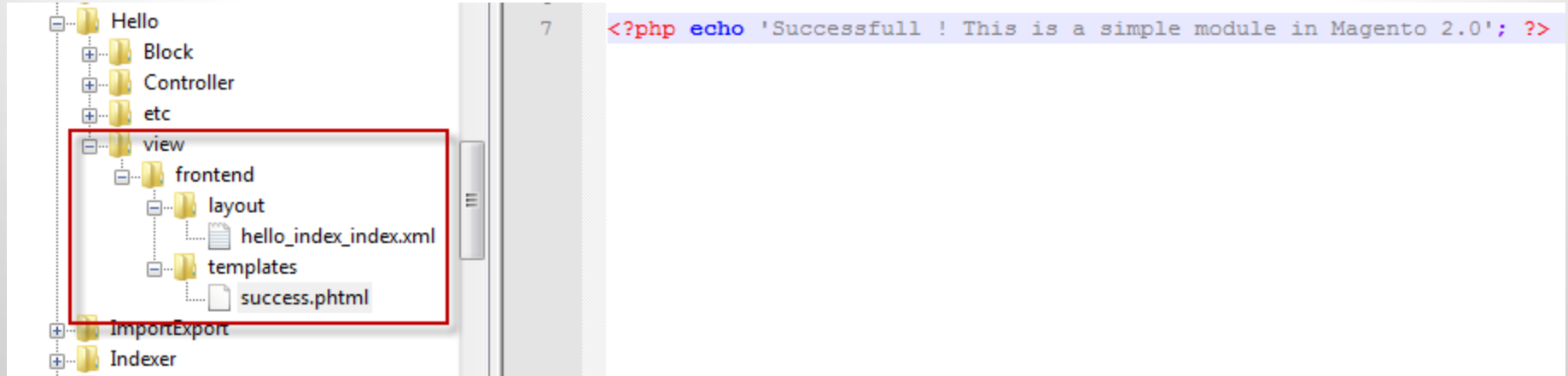
The screenshot displays a Magento IDE interface. On the left, a file explorer shows a project structure with folders like Block, Controller, etc, view, frontend, layout, and templates. The 'templates' folder is selected, showing a file named 'hello_index_in'. On the right, a code editor shows the XML content of the 'hello_index_in' file. The code is as follows:

```
19 * versions in the future. If you wish to customize Magento for your
20 * needs please refer to http://www.magentocommerce.com for more information.
21 *
22 * @copyright Copyright (c) 2014 X.commerce, Inc. (http://www.magentocommerce.com)
23 * @license http://opensource.org/licenses/afl-3.0.php Academic Free License (AFL 3.0)
24 */
25 -->
26 <page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=".."
27 <head>
28 <title>Hello World</title>
29 </head>
30 <body>
31 <referenceContainer name="content">
32 <block class="Magento\Hello\Block\Hello" name="hello" template="success.phtml">
33 </block>
34 </referenceContainer>
35 </body>
36 </page>
37
```

3

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

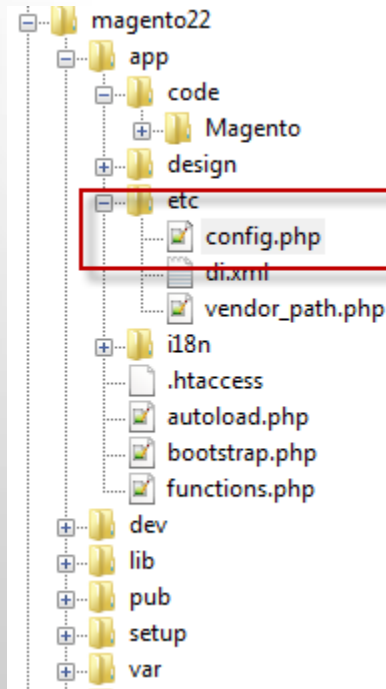
<?php echo 'Successful! This is a simple module in Magento 2.0'; ?>



4

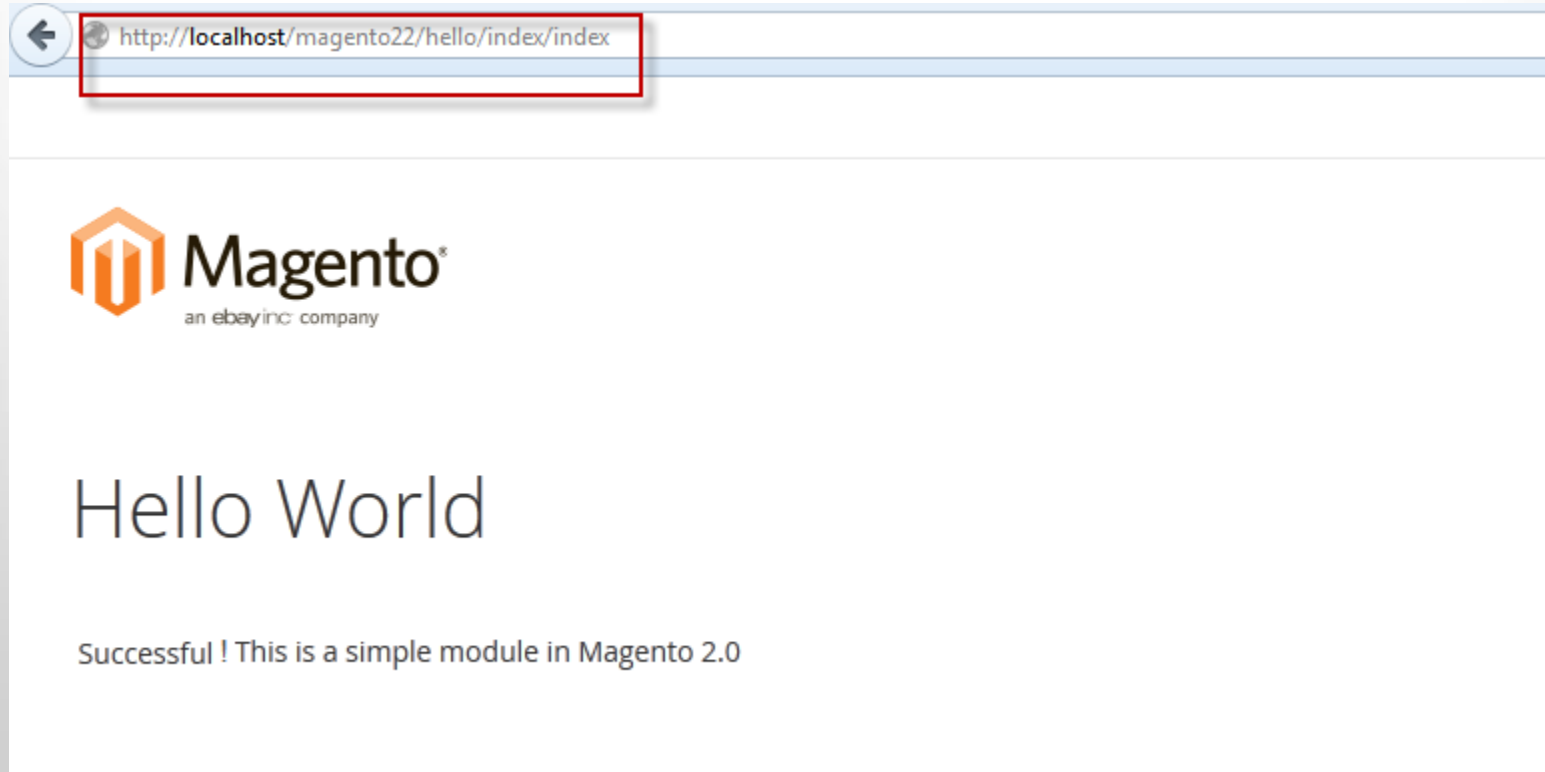
Activate Magento_Hello extension in configuration file

- Open the file **app/etc/config.xml**
- In the array **'module'**, add the element: **'Magento_Hello' => 1,**



```
39     array (  
40         'connection' => 'default',  
41     ),  
42 ),  
43 'modules' =>  
44 array (  
45     'Magento_Hello' => 1,  
46     'Magento_Core' => 1,  
47     'Magento_Authorization' => 1,  
48     'Magento_Store' => 1,  
49     'Magento_Directory' => 1,  
50     'Magento_Backup' => 1,  
51     'Magento_Eav' => 1,  
52     'Magento_Customer' => 1,  
53     'Magento_Indexer' => 1,  
54     'Magento_CatalogImportExport' => 1,  
55     'Magento_Theme' => 1,  
56     'Magento_Rule' => 1,  
57     'Magento_Backend' => 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.