# How to enable/disable Magento 2 profiler?

[Magento 2](https://magehit.com/magento-2-extensions.html) has in-built profiler which is used to identify performance problems on the server side. The profiler allows you to see the execution time of certain chunks of code. There are 3 types of profilers supported such as HTML, CSV file, and Firebug. They are used for analyzing performance task, displaying dependency graphs on a Magento page and revealing unused dependencies to reduce the memory and time.

## **Enable Profiler in Magento 2**

* Go to Magento **Admin Panel> System> Configuration**
* Scroll down to **Advanced> Developer**
* In **Debug> Profilers> Yes**
* Tap **Save Config**
* Open file php in root install
* Find the line #Varien\_Profiler::enable() then uncomment it. In case you cannot find it, you can place this line before the line Mage::run($mageRunCode, $mageRunType.);

## **Disable Profiler in Magento 2**

* Go to Magento **Admin Panel> System> Configuration**
* Scroll down to **Advanced> Developer**
* In **Debug> Profilers> No**
* Tap **Save Config**

# How to Enable Template Path Hints in Magento 2?

Many Magento 2 users are confused of enabling template path hints between for storefront and for admin. Therefore, in this post we will show you how to perform this action for both frontend and admin from admin panel obviously.

**Enable template path hints for frontend**

You just need to follow guiding below:

Navigate to **Admin Panel→ Store→ Configuration→ Advanced→ Developer→ Debug→ Enabled Template Path Hints for Storefront→ Yes**

**Enable template path hints for admin**

Go along following guiding:

Go to **Admin Panel→ Store→ Configuration→ Advanced→ Developer→ Debug→ Enabled Template Path Hints for Admin→ Yes**

That’s all about **enabling template path hints in Magento 2**. In fact, this is a very simple tutorial but you have to pay attention to the 7th option because Magento users usually make mistake at this step. Hope that you can perform this tutorial successfully. In case you need more [Magento tutorial](http://magehit.com/blog/magento-tutorial/), you can visit [our blog](http://magehit.com/blog/) to consult more.

### **Disable CSS & JS merge/ bundle/ minify**

* Navigate to **Stores > Configuration > Advanced > Developer > javaScript Settings and CSS Settings**
* Set **Enable javaScript Bundling, Merge javaScript Files, Minify javaScript, Merge CSS Files**into No

# How to Enable/Disable Maintenance Mode in Magento 2?

Maintenance mode is important mode in Magento 2 because you need to give your website this maintenance mode when you want to temporarily disable your website to update or fix bug.

## **In this post we will share you the way to Enable/Disable maintenance mode in Magento 2.**

If **var/.maintenance.flag** file doesn’t exist, it means the maintenance mode is off and vice verse.

However, there is another factor influencing on the status of the maintenance mode. **var/.maintenance.ip** file can make some exceptions to the maintenance so maintenance mode may be off even the **var/.maintenance.flag** file still exists.

### **Enable maintenance mode**

* Login your account using SSH
* Enter following command at the command prompt:

cd ~/public\_html

**\*Note:**

If Magento 2 was installed in a subdirectory, you need to change to that directory instead.

* To permit access to the front-end store from a specific IP address while Magento is in maintenance mode, you need to enter command:

bin/magento maintenance:allow-ips IP address

(IP address is which you want to allow access. You can see your own IP address at [http://ipfinder.us](http://ipfinder.us/))

* Enter following command to enable maintenance mode:

bin/magento maintenance:enable

Once maintenance mode is on, visitors can see a **Service Temporarily Unavailable** message in their web browsers instead of the store. But the authorized address can still view the store.

### **Disable maintenance mode**

* To disable maintenance mode and re-enable the store for all visitors, use command:

bin/magento maintenance:disable

\*You can view the maintenance mode status in your website by using command bin/magento maintenance:status

# How to disable extensions in Magento 2?

In case you no longer need an extension, you can disable it to remove its function from your Magento 2 store. In this post, we show you 3 easy ways to **disable extensions the unnecessary extension in your Magento 2 store**.

## **The ways to disable extensions in Magento 2**

**1st way:**

* **Admin panel> Stores> Settings> Configuration**
* After the configuration screen opens, choose **Advanced option**
* Then disable the Magento 2 extensions as you want
* Tap **Save Config**

**2nd way:**

* Navigate to **Stores> Configuration> M-connect Media**
* After that you will see a list of extensions. Click to any extensions you want to disable them
* Tap **Save Config**

**3rd way:**

* **Admin Panel> System> Web Set up Wizard> Component Manager**
* Then you can see Component Manager screen opened
* Click on the **Select option** to disable any extension you want
* Tap **Save Config**

# How to disable Magento 2 Compiler from command?

It is simple to **disable Compiler in your Magento 2**. We will show you command to do this.

You need to check the current compilation status:

$ php -f shell/compiler.php -- state

Compiler Status: Disabled

Compilation State: Compiled

Collected Files Count: 6959

Compiled Scopes Count: 4

Use command below to disable Compiler:

php -f shell/compiler.php -- disableCompiler include path disabled

Use following command to clear current Magento compilation:

php -f shell/compiler.php -- clearCompilation successfully cleared

Beside, you can apply the way below:

**Admin Interfaces> System> Tools> Compilation> Disable** button

# How to Set Magento 2 Mode Using CLI?

As you know, Magento 2 owns outstanding features that give developers as well as customer the ease of using. One of those is the switch Magento 2 modes from developer mode to production mode. With this feature, not only the ease of use but also the security of your store is improved. Moreover, you can set the ownership and file permissions appropriately.

There are some differences between production mode and developer mode like:

* **Production mode** has better execution because static view files are located in the bar/ static directory due to code compilation.
* **Developer mode** is used for internal development on a system already secured behind the firewall. When you are customizing your Magento store, developer mode will be of real help.

Now, let’s determine this tutorial!

### **Cleared Directory**

When you alter developer mode to production mode or contrast, Magento system clears the contents of directories below:

var/cachevar/divar/generationvar/view\_preprocessedpub/static

Cache, logs and compiled code are stored by [Magento 2](http://magehit.com/blog/magento-2/) by default with the help of var directories. In addition, you are allowed to customize these directories.

### **Ownership and Permissions for Developer Mode**

When you switch to developer mode, only directories are cleared. And no specific permissions on existing files are set by Magento system. There are some permission below are set one the directories and subdirectories:

* Directories: 770

With 770 permissions and support full control to an owner and the group.

* Files: 660

With 660 permissions and support write and read permissions to owner and the group.

### **Display Current Mode**

Apply command **magento deploy:mode:show** to display of current mode

Message will be like **current application mode: developer**

### **Change Modes**

Use command **magento deploy:mode:set {mode} [-s|–skip-compilation]** to change modes in Magento 2. You can alter {mode} if you want to change to other mode.

-skip-compilation is optional. You can use it for skipping code compilation when we change to production mode.

### **Change Production Mode**

Apply command **magento deploy:mode:set production**to change to production mode.

After that, this message can be seen like:

Enabled maintenance mode

Requested languages: en\_US

=== frontend -> Magento/luma -> en\_US ===

… more …

Successful: 1884 files; errors: 0

—

=== frontend -> Magento/blank -> en\_US ===

… more …

Successful: 1828 files; errors: 0

—

=== adminhtml -> Magento/backend -> en\_US ===

… more …

—

=== Minify templates ===

… more …

Successful: 897 files modified

—

New version of deployed files: 1440461332

Static content deployment complete Gathering css/styles-m.less sources. Successfully processed LESS and/or SASS files CSS deployment complete Generated classes:

Magento\Sales\Api\Data\CreditmemoCommentInterfacePersistor

Magento\Sales\Api\Data\CreditmemoCommentInterfaceFactory

Magento\Sales\Api\Data\CreditmemoCommentSearchResultInterfaceFactory

Magento\Sales\Api\Data\CreditmemoComment\Repository

Magento\Sales\Api\Data\CreditmemoItemInterfacePersistor

… more …

Compilation complete

Disabled maintenance mode

Enabled production mode.

### **Change Developer Mode**

When you switch to developer mode, you should clean generated classes and Object Manager entities like proxies to unexpected errors. To change to developer mode, follow steps below:

* Clear the contents of var/generation and var/di directories by using command:

**rm -rf <your Magento install dir>/var/di/\* <your Magento install dir>/var/generation/\***

* Set the mode by command **magento deploy:mode:set developer**

You will be getting a message like: **Switched to developer mode**.

With our guide above, you can easily change modes for your Magento application by applying command lines. Contact us if you have any problem when working with this tutorial. Don’t forget to consult our website blog to get more useful [tutorial for Magento 2](http://magehit.com/blog/magento-2-tutorial/).

# How to Configure Sales Emails in Magento 2?

There are many sections to configure with the sales emails like Credit Memo Comments and Credit Memo, Order Comments and Order, Invoice Comments and Invoice, Shipment Comments and Shipment. You can decide the email contact as the sender of the email, the email template to be used and each customer can receive a copy of the message.

## **Now, let’s move the way to configure Sales Email!**

### **Step 1: Refresh the Email Templates in Magento 2**

Make sure that you have refreshed each email template

### **Step 2: Configure the Sales Emails**

* **Admin Panel→ Stores→ Settings→ Configuration**
* On the lest panel, select **Sales Emails**
* In **Order**field:
* In **Enable**, set “Yes”
* In **Send Order Email Copy To** field, enter the email address. In case there are many email address added, you have to separate them by comma.
* In **New Order**configuration, you should select the email template **Template for Guest**. This email is sent to those who don’t have an account on your website.
* In **New Order Confirmation Template**, select the desire email template which is sent to registered clients
* In **New Order Confirmation Email**, choose the store contact of the email
* In **Send Order Email Copy Method**, choose one of sending methods:
  + Bcc: In the header of the same sales email, you might send a blind courtesy copy by including the recipient that is sent to the client. Customer doesn’t know that a copy is sent to many emails, so the BCC recipient is not visible to the client.
  + Separate email: you might send a copy email as separate email
* Open **Order Comments, Credit Memo section, Shipment, Invoice** and then complete them as same as Order
* Tap **Save Config**

# How to integrate PayPal into Magento 2 store?



PayPal is one of the most famous and professional payment service companies for online business. **Integrating PayPal into your Magento 2 store** will give your customers more convenience when purchasing in your store. With PayPal setup in your store, payment will become simpler and easier.

Before integrate PayPal for your Magento 2 store, you need to create a PayPal business account. You can choose 1 in 3 plans such as PayPal Payments Advanced, PayPal Payment Pro or PayPal Payment Standard based on your demand.

After owning a PayPal account, let’s start integrate into your [Magento 2](https://magehit.com/magento-2-extensions.html) store!

### **Step 1: Work with PayPal**

* In PayPal business account, go to **Service Settings > Hosted Checkout Pages > Set Up Menu**
* Then set:
  + **AVS**: No
  + **CSC**: No
  + **Enable** **Secure**: Yes
* Click **Save**
* Press Customize button on the top of page and then use **Layout C to customize the layout**
* **Tap Save** **and publish**
* **Sign up a new account for your current account for more security**

### **Step 2: Configure in Magento back-end**

* Go to **Magento Admin Panel > Store > Setting > Configuration > Sale > Payment method**
* In **Merchant Country** box, enter the country your store is located in
* Expand **PayPal All-in-one payment Solution** section
* Click to **Configure** button if you want to enable PayPal payment is a correct payment method in your store.
* After that, you will see 3 menus which you need to configure:

#### **Require PayPal Setting**

* + - Complete all the boxes such as **Email Associated with PayPal Merchant Account (optional), Partner, Vendor, User, Password, Text Mode**, and **Use Proxy**.

#### **Basic Settings – PayPal Payments Advanced**

* + - In **Title** box, you can choose Credit or Debit card for PayPal checkout
    - In **Sort** **Order**, you can set the order for the payment method when customer check out
    - In **Payment Action,** you can have 2 options:
      * **Authorization**: agree with the order and receive money by agreeing manually in the backend
      * **Sale**: get money from clients account automatically

#### **Advanced settings - PayPal Payments Advanced**

* + - At the bottom of the **Basic Settings**section, click to expand the **Advanced Settings** Then, complete the following as needed for your store:
    - Click**Save to apply.**

#### **Basic Settings – PayPal Express Checkout**

* + - In **Title**: enter the payment method to checkout
    - In **Sort Order**: enter a number to determine the sequence in which Express Checkout appears when listed with the other methods
    - In **Set Payment Action**: choose Authorization or Sale
    - In **Display on product Details Page**: Yes

#### **Advanced Settings - PayPal Express Checkout**

* + - In Display on Shopping Cart: Yes
    - Set Payment Applicable From to one of the following:  
      **All Allowed Countries**
    - Accepts payment from the**countries already specified**in your configuration.
    - Specific Countries
    - Accepts payments from only the countries you specify. **Hold the Ctrl**key down and in the **Payment Applicable From** the list, click each country where you accept payment.
    - In **Debug Mode**: Yes
    - In **Enable SSL Verification**: Yes
    - In **Transfer Cart Line Items**: Yes
    - In **Skip Order Review Step**: Yes
    - Tap **Save**

# Magento 2 Authorize.net setup



**Authorize.net** is an improved feature in Magento 2 which helps your customers ignore going directly to third party site to complete their payment when making purchase and all data of customer will be transferred through a security gateway. This feature brings the convenience for your customer like saving their time and effort as well as creates the satisfaction from your customer to your service. Today, we show you the way to **configure Authorize.net for Magento 2**.

### **Step 1: set up direct post for Authorize.net**

* **Store→ Configuration→ Sale→ Payment Methods**
* Turn on **net Direct Post:**
  + Set “Yes” for **Enable**box
  + Choose a option for Payment Action:
    - Authorize Only: Funds on the customer’s card are authorized by Authorize.Net, and an order is created in your store’s Admin. You can later create an invoice and capture the funds.
    - Authorized and Capture: Funds on the customer’s card are authorized and captured by Authorize.Net, and an order and invoice are created in your store’s Admin.

Create a name which can remind you about this option for **Title**box.

### **Step 2: Connect to Authorized.net account**

You have to fill in 2 information of your account like **API Login ID**and **Transaction Key.**

In merchant MD5 box, fill the data from your Authorize.net account at **Account→ Setting→ Security Settings→ MD5-Hash**

Choose one in two options Processed Ogone Payment and Processing in New order status box

You can test the performance more than one time and turn it off when you sure that everything is ok and your system is ready for running.

In Gateway URL box, you need to fill in the default link http://secure.authorize.net/gateway/transact.dll from [Authorize.net](https://www.authorize.net/).

### **Step 3: Configure the system**

* Set **Accept Currency** to “US dollar”
* Set **Debug**to “Yes” to save messages transmitted between your store and the **net Direct Post System**
* **Configure Payment** option:
  + In**Credit Card Types** list, chose each credit card that is accepted in your store
  + Set **Credit Card Verification** to “Yes” in order to require customers to enter a card verification value
  + Select one in two options I payment from Applicable:
    - All Allowed Countries: Customers from all countries can use this payment
    - Specific Countries: Only chosen countries in the list can make purchase from your store
  + Set the Minimum Order Total and Maximum Order Total for Direct Post transactions. (the order total must exactly match the minimum or maximum threshold or in between)
  + Enter **Sort Order** number to define the position of **Direct Post** in the list of payment methods during checkout
* Tap **Save Config**

It’s all things you have to do for **configuring Magento 2 Authorize.net** successfully. These steps are very simple so that you totally set up easily.

# How to setup CyberSource payment for Magento 2?



CyberSource is one of the first companies in online payment. Now, CyberSource becomes more and more popular so setup this payment for Magento 2 will help your customers achieve payment easily and conveniently.

## **The steps to setup CyberSource payment for Magento 2**

### **Step 1: Get the CyberSource credentials**

* Go to <http://www.cybersource.com/>, signup a CyberSource merchant account
* Get your credentials

### **Step 2: Enable CyberSource**

* Enable: set Yes
* Accept **Default Payment**action of “Authorized Only” to approve the purchase and puts a hold on the funds which will not withdraw from the bank account of customer till they make the next purchase.
* Fill in a title in **Title** box to identify CyberSource when checking

### **Step 3: Fill in your CyberSource credentials**

You need to enter all credentials below from your CyberSource account:

* Profile ID
* Merchant ID
* Transaction Key
* Access Key
* Secret Key

### **Step 4: Fill in payment information**

* Choose Processing or suspected Fraud status setting for **New Order Status**
* **Test Mode:**set to Yes to test CyberSource before put it live
* After that, set **Test Mode**to No
* **Debug:**set to Yes
* Set **Credit Card Types**for the payment you accepted. If you want to select multiple credit card, you can press the **Ctrl**key then click the option you want

### **Step 5: Fill in remaining information**

* Choose All Allowed Countries or Specific Countries for **Payment from Applicable Countries**. If you choose Specific Countries option, press **Ctrl** key then choose the countries you accept payment.
* Insert a number for **Minimum Order Total** box and **Maximum Order Total** box to limit the total amount for the order
* Enter a number in **Sort Order** box to determine the sequence where CyberSource appears when listed with other payment methods during checkout
* Click **Save Config**

# How to configure Returns in Magento 2?

Configuring Returns allows store owners to manage the returned products and amount of product in stores easily and time-saving. In this post, we will show you a simple way to configure Returns for your [Mageno 2](https://magehit.com/magento-2-extensions.html) store.

## **The steps to configure Returns in Magento 2**

1. **Admin Panel > Stores > Settings > Configuration**
2. Under **Sales** on the left panel, click **Sales**
3. Expand **RMA Settings** then:
   * **Enable RMA on Storefront:** set to Yes
   * **Enable RMA on Product Level:** set to Yes
   * **Use Store Address:**set to Yes if you want to have returned products sent to the store address or set to No then add an alternate address for returned products to be sent.
4. Tap **Save Config**

# Configuring Attribute sets in Magento 2

Working with Attributes sets is one of the first steps for creating a product. The attribute sets will be used as a template for the product record each time when a product is created. Magento 2 supports store owners to configure the default attribute set. In this post, we will show you the way to **create default sets for Magento 2**.

## **Create default attribute sets**

### **Create an attribute set**

1. **Magento Admin sidebar > Stores > Attributes > Attribute Set > Add New Set**
2. Enter a name in **Name** box and set the **Based On**
3. Click **Save** Then you will see next page displaying:

* **Edit Attribute Set Name** column: the name of attribute set
* **Groups** column: list of the current selection of attribute groups
* **Unassigned Attributes** column: list of the selection of attributes currently not assigned to the attribute set

1. Drag the attribute from **Unassigned Attributes** column to an appropriate folder in **Groups** column
2. Click **Save** button

### **Create a new attribute group**

1. In **Groups** column, click to **Add New** button
2. In **Name** box, fill in a name then tap **OK** button
3. Do one in two following performances:

* Drag unassigned attributes to the new group
* Drag attributes from any other group to the new group

### **Create custom attribute sets**

1. Insert any table related data after the module installation via upgradeData. Add following code in php under app/code/naespace/module/setup:
2. /\*
3. \* We declare the namespace of our module to avoid conflicts around multiple modules and introduce more flexibility
4. \*/
5. namespace Apptha\Blog\Setup;
6. /\*
7. \*The below are the namespaces and and classes to be included in order to create new custom attribute set programmatically
8. \*/
9. use Magento\Eav\Setup\EavSetup;
10. use Magento\Eav\Setup\EavSetupFactory;
11. use Magento\Framework\Setup\UpgradeDataInterface;
12. use Magento\Framework\Setup\ModuleContextInterface;
13. use Magento\Framework\Setup\ModuleDataSetupInterface;
14. use Magento\Catalog\Setup\CategorySetupFactory;
15. use Magento\Eav\Model\Entity\Attribute\SetFactory as AttributeSetFactory;
16. /\*
17. \* UpgradeDataInterface brings the ‘upgrade’ method which must be implemented
18. \*/
19. class UpgradeData implements UpgradeDataInterface
20. {
21. private $eavSetupFactory;
22. private $attributeSetFactory;
23. private $attributeSet;
24. private $categorySetupFactory;
25. public function \_\_construct(EavSetupFactory $eavSetupFactory, AttributeSetFactory $attributeSetFactory, CategorySetupFactory $categorySetupFactory )
26. {
27. $this->eavSetupFactory = $eavSetupFactory;
28. $this->attributeSetFactory = $attributeSetFactory;
29. $this->categorySetupFactory = $categorySetupFactory;
30. }
32. public function upgrade(ModuleDataSetupInterface $setup, ModuleContextInterface $context)
33. {
34. $categorySetup = $this->categorySetupFactory->create(['setup' => $setup]);
35. /\*\*
36. \*checking the current version of the module
37. \*this function is implemented from ModuleContextInterface
38. \*/
39. if(version\_compare($context->getVersion(), '1.1', '<')) {
40. $attributeSet = $this->attributeSetFactory->create();
41. $entityTypeId = $categorySetup->getEntityTypeId(\Magento\Catalog\Model\Product::ENTITY);
42. $attributeSetId = $categorySetup->getDefaultAttributeSetId($entityTypeId);
43. $data = [
44. 'attribute\_set\_name' => 'Apptha\_Custom\_Attribute\_Set', // define custom attribute set name here
45. 'entity\_type\_id' => $entityTypeId,
46. 'sort\_order' => 200,
47. ];
48. $attributeSet->setData($data);
49. $attributeSet->validate();
50. $attributeSet->save();
51. $attributeSet->initFromSkeleton($attributeSetId);
52. $attributeSet->save();
53. }
54. }

} ?>

1. Use following code to update the version in xml under app/code/namespace/etc:

<?xml version="1.0"?><config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">   <module name="Namespace\_ModuleName" setup\_version="1.1" /></config>

1. Upgrade the module setup using upgrade command via CLI

\*Note: You can upgrade the module via CLI by using command: php bin/magento setup upgrade

# Configuring Magento 2 Sitemap

A Sitemap is a plain text XML file including link to all the pages on your website. The Search Engines will use Sitemap to properly crawl and index your website. In [Magento 2](http://magehit.com/magento-2-extensions.html), some things related to Sitemap are different from [Magento 1](http://magehit.com/magento-extensions.html) so in this post, we will bring you all tutorials related to **Magento 2 Sitemap**.

## **Enable a Sitemap**

* Login to **Admin> Stores > Settings > Configuration > Catalog> XML Sitemap**
* Adjust settings for the options **Category Options, Product Options**, and **CMS page Options** as you want.
* Set **Generation Settings** then adjust:
* **Enable**: Yes
* **Start Time**: Fill in a time after 00:00
* **Frequency**: Choose Daily or Hourly depending on the products on your store
* **Error Email Recipient**
* **Error email Sender**
* **Error Email Template**
* Open **Search Engine Submission**> set **Enable Submission to Robot.txt** to Yes> Tap **Save Config**

## **Configure Sitemap**

### **Configure a sitemap for all stores**

* Create mkdir –p /data/web/magento2/sitemaps to store your sitemap
* **Admin Panel> Marketing> Seo & Search> Sitemap> Add a new sitemap**
* Insert all needed information like:
* **Filename**: xml
* **Path**: /
* **Store View**: based on the quantity of your stores
* Tap **Save & Generate**

### **Configure a sitemap for each store**

* Create mkdir –p /data/web/magento2/sitemaps to store your sitemap ( You have to ensure that you have no sitemap in /data/web/public/sitemap.xml in order to avoid the incorrect sitemap
* **Admin Panel> Marketing> Seo & Search > Sitemap> Add new sitemap**
* Fill in the required information:
* **Filename**: name your sitemap with convention: sitemap\_&striecode.xml
* **Path**: /sitemap/
* **Store View**: choose an option you want
* Tap **Save & Generate**

Add your sitemap location to your Robots.txt

* After completing to request your sitemap.xml, you need add it to your Robots.txt:Sitemap:http://www.example.com/sitemap.xml

## **Configure nginx**

### **Configure nginx to use a sitemap for all stores**

* Login on your hypernode then open /data/web/nginx/server.sitemap
* Create snippet below:
* location /sitemap.xml {
* alias /data/web/magento2/sitemaps/sitemap.xml;

}

### **Configure nginx to use a sitemap for each store**

* Login on your hypernode then open /data/web/nginx/server.sitemap
* Create snippet below:
* location /sitemap.xml {
* alias /data/web/magento2/sitemaps/sitemap\_$storecode.xml;

}

# Manage Magento 2 cache



**Magento 2 cache management** is one of the important tasks you should know if you want to run a good website. In this post, we will show you all tutorials related to managing cache in [Magento 2](https://magehit.com/magento-2-extensions.html)website.

### **Check the Cache Status**

Use following command:

magento cache:status

### **Enable/disable cache types**

* Enable/disable separate cache type
* magento cache:enable [type]

magento cache:disable [type]

* Enable/diable all cache types
* magento cache:enable

magento cache:disable

* Disable the DDL cache and the full page cache:

Magento cache:disable full\_page db\_ddl

### **Clean and flush cache types**

* Clean, flush separate cache type:
* magento cache:clean [type]

magento cache:flush [type]

* Clean all cache types:

magento cache:clean

# How to use REST API in Magento 2?

REST API plays an important role in web apps and is one of the most popular API designs used recently. But for using **REST API in Magento 2**, you have to know the flow to call APIs in PHP.

In case you want to use token-based REST API in [Magento 2](http://magehit.com/magento-2-extensions.html), you will need authenticate, get the token then pass it in the header of every request you perform. Additionally, you also need to create a User Role web service then register the role to a new Admin User in Magento 2.

## **We will show you all the tutorials to help you use REST API in Magento 2.**

**At first**, we need to create Web Service Role and Web Service User in Magento 2

* Create Web Service Role:
* Login **Admin Panel> System> User Roles> Add New Role**
* Add the Role Name and your current password of Admin in **Your Password** field
* Tap **Role Resources**
* Choose what are required for service of your web in **Resource Access**
* Tap **Save Role**
* Create Web Service User in Magento 2

This user is used for the role you’ve created

* Go to **System> All Users> Add New User**
* Fill in all the necessary information
* Tap **User Role** then choose which you’ve created
* Tap **Save User**

The user above will used to REST API web service in Magento 2.

**Next**, we will get starting with Magento 2 REST API.

* Authenticate REST API through Token authentication:Use following code:
* <?php
* //API URL for authentication
* $apiURL="http://magento-91647-257956.cloudwaysapps.com/index.php/rest/V1/integration/admin/token";
* //parameters passing with URL
* $data = array("username" => "apiaccess", "password" => "cloudways123");
* $data\_string = json\_encode($data);
* $ch = curl\_init($apiURL);
* curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, "POST");
* curl\_setopt($ch, CURLOPT\_POSTFIELDS, $data\_string);
* curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);
* curl\_setopt($ch, CURLOPT\_HTTPHEADER, array("Content-Type: application/json","Content-Length: ".strlen($data\_string)));
* $token = curl\_exec($ch);
* //decoding generated token and saving it in a variable
* $token= json\_decode($token);

?>

This code will help pass username and password with API URL in order to authenticate Magento 2 REST API as well as saved the token.  
\*Note: The username and password are of what you’ve created

* Get modules in Magento by REST API: By using REST API, you can get most of things in Magento 2. You can check in list of REST APIs for Magento EE and CE.
* Use following code to demonstrate the API:
* <?php
* //Using above token into header
* $headers = array("Authorization: Bearer ".$token);
* //API URL to get all Magento 2 modules
* $requestUrl='http://magento-91647-257956.cloudwaysapps.com/index.php/rest/V1/modules';
* $ch = curl\_init($requestUrl);
* curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);
* curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);
* $result = curl\_exec($ch);
* //decoding result
* $result= json\_decode($result);
* //printing result
* print\_r($result);

?>

With the code above, you passed the token in the header with the API URL in order to get all the modules installed on Magento 2 store.

* Use following code to print the result:
* <?php
* //API URL for authentication
* $apiURL="http://magento-91647-257956.cloudwaysapps.com/index.php/rest/V1/integration/admin/token";
* //parameters passing with URL
* $data = array("username" => "apiaccess", "password" => "cloudways123");
* $data\_string = json\_encode($data);
* $ch = curl\_init($apiURL);
* curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, "POST");
* curl\_setopt($ch, CURLOPT\_POSTFIELDS, $data\_string);
* curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);
* curl\_setopt($ch, CURLOPT\_HTTPHEADER, array("Content-Type: application/json","Content-Length: ".strlen($data\_string)));
* $token = curl\_exec($ch);
* //decoding generated token and saving it in a variable
* $token= json\_decode($token);
* //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//
* //Using above token into header
* $headers = array("Authorization: Bearer ".$token);
* //API URL to get all Magento 2 modules
* $requestUrl='http://magento-91647-257956.cloudwaysapps.com/index.php/rest/V1/modules';
* $ch = curl\_init($requestUrl);
* curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);
* curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);
* $result = curl\_exec($ch);
* //decoding result
* $result= json\_decode($result);
* //printing result
* print\_r($result);

?>

We’ve brought you many of [tutorials](https://magehit.com/blog/magento-2-tutorial/) related to **Magento 2 REST API**. Hope that they are helpful for you.

Implementing Magento 2 SOAP API

Today we will introduce you how to install SOAP API for Magento 2. Before using **Magento 2 SOAP API**, you should use SoapUI to test SOAP calls then starting implementing SOAP API.

You will start with WSDL file which is a XML Document with all available calls you can use to get data from the store. You can find SOAP V2 WSDL with the URL **http://www.domain.com/index.php/api/v2\_soap?wsdl=1**

Replace **www.domain.com** with your domain

Then you will see a long xml file in browser like:

[](http://magehit.com/wp/wp-content/uploads/2017/10/Magento2-SOAP-API-xml.jpg)It is quite difficult when working with this WSDL file, so you need a tool to help. A recommended one is SoapUI which help speed up your development. This tool is written in Java so it can run on Linux, Windows and Mac OSX.

**Step 1**: Download SoapUI and install

**Step 2**: Login SoapUI

* Open login Request 1, then you will see a window opened with soap xml code but you just need to care about 2 tags *username*and *apiKey*.
* At the beginning of 2 tags are “?” so you need to change to avoid getting “Access denied” error when logging in SoapUI

**Step 3**: Define SOAP API users

* Go to ***System> Web Services >SOAP/XML-RPC Users*** then you will see a list of all configured users for your API
* Go to ***System> Web Services> SOAP/XML-RPC Roles*** to define roles for users
* Create admin roles by selecting **All** in ***Role Resources*** tab
* Create users
* Set created admin role in ***User Role*** tab

Now you can fill all tags in your SoapUI Request window then run this request.

After that, you need come back a long number of digits and characters, copy your session id and store it.

We’ve introduced you the way to use SoapUI with SOAP API like login to get session id and creating a new API user role for Magento 2 backend. Hope you guys can understand and perform them.

# How to add a custom discount in Magento 2?

You know that discount id one of the most effective ways to attract customers making purchases. However, it is difficult to have suitable promotion for your product because you still need profit from the sale off product. In Magento 2, it has a **feature - custom discount** that helps you custom the discount for your product to have suitable price.

Follow steps below to add this feature to your Magento 2 store.

**Step 1: Register a total**

You need to register a total in the file **sale.xml app/code/Magento2/Webpos/etc/sales.xml:**

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento\_Sales:etc/sales.xsd">

<section name="quote">

<group name="totals">

<item name="customer\_discount" instance="MageHit\Webpos\Model\Total\Quote\Custom" sort\_order="420"/>

</group>

</section>

</config>

**Step 2: Add discount to change the grand total**

You need to add discount to change the grand total in the file **app/code/Magento2/Webpos/Model/Total/Quote/Custom.php**:

<?php

namespace MageHit\Webpos\Model\Total\Quote;

/\*\*

\* Class Custom

\* @package MageHit\Webpos\Model\Total\Quote

\*/

class Custom extends \Magento\Quote\Model\Quote\Address\Total\AbstractTotal

{

/\*\*

\* @var \Magento\Framework\Pricing\PriceCurrencyInterface

\*/

protected $\_priceCurrency;

/\*\*

\* Custom constructor.

\* @param \Magento\Framework\Pricing\PriceCurrencyInterface $priceCurrency

\*/

public function \_\_construct(

\Magento\Framework\Pricing\PriceCurrencyInterface $priceCurrency

){

$this->\_priceCurrency = $priceCurrency;

}

/\*\*

\* @param \Magento\Quote\Model\Quote $quote

\* @param \Magento\Quote\Api\Data\ShippingAssignmentInterface $shippingAssignment

\* @param \Magento\Quote\Model\Quote\Address\Total $total

\* @return $this|bool

\*/

public function collect(

\Magento\Quote\Model\Quote $quote,

\Magento\Quote\Api\Data\ShippingAssignmentInterface $shippingAssignment,

\Magento\Quote\Model\Quote\Address\Total $total

)

{

parent::collect($quote, $shippingAssignment, $total);

$baseDiscount = 10;

$discount = $this->\_priceCurrency->convert($baseDiscount);

$total->addTotalAmount('customdiscount', -$discount);

$total->addBaseTotalAmount('customdiscount', -$baseDiscount);

$total->setBaseGrandTotal($total->getBaseGrandTotal() - $baseDiscount);

$quote->setCustomDiscount(-$discount);

return $this;

}

}

After that, grand total will be changed.

**Step 3: Add the total**

You need to add the total in the layout file ***app/code/MageHit/Webpos/view/frontend/layout/checkout\_cart\_index.xml***. Because Maento 2 uses knockout js to show the total so there is no total discount information.

<?xml version="1.0"?>

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page\_configuration.xsd">

<body>

<referenceBlock name="checkout.cart.totals">

<arguments>

<argument name="jsLayout" xsi:type="array">

<item name="components" xsi:type="array">

<item name="block-totals" xsi:type="array">

<item name="children" xsi:type="array">

<item name="custom\_discount" xsi:type="array">

<item name="component" xsi:type="string">MageHit\_Webpos/js/view/checkout/summary/customdiscount</item>

<item name="sortOrder" xsi:type="string">20</item>

<item name="config" xsi:type="array">

<item name="custom\_discount" xsi:type="string" translate="true">Custom Discount</item>

</item>

</item>

</item>

</item>

</item>

</argument>

</arguments>

</referenceBlock>

</body>

</page>

Then it will call view model knockout in ***app/code/MageHit/Webpos/view/frontend/web/js/view/checkout/summary/customdiscount.js***

define(

[

'jquery',

'Magento\_Checkout/js/view/summary/abstract-total'

],

function ($,Component) {

"use strict";

return Component.extend({

defaults: {

template: 'MageHit\_Webpos/checkout/summary/customdiscount'

},

isDisplayedCustomdiscount : function(){

return true;

},

getCustomDiscount : function(){

return '$10';

}

});

}

);

After all, the total discount will be shown in the template knockout ***app/code/MageHit/Webpos/view/frontend/web/template/checkout/summary/customdiscount.html*:**

<!-- ko if: isDisplayedCustomdiscount() -->

<tr class="totals customdiscount excl">

<th class="mark" colspan="1" scope="row" data-bind="text: custom\_discount"></th>

<td class="amount">

<span class="price" data-bind="text: getCustomDiscount(), attr: {'data-th': custom\_discount}"></span>

</td>

</tr>

<!-- /ko -->

Follow exactly all steps above, you complete adding custom discount for your Magento 2 store. Hope that this post is useful for you. There are more effective [tutorials for Magento 2](http://magehit.com/blog/magento-2/) in our [MageHit blog](http://magehit.com/blog/), visit it to get these tutorials.

# How to Get Value of Custom Attribute on Magento 2 Rest API?

## **In this post, we will show you how to get value of custom attribute on Magento 2 Rest API.**

### **Step 1: Create new column and set value for the existing order**

Firstly, you need to create a new column in table sales\_order and name for it set a value for the existing orders. In this post, we named it “tip\_and\_trick\_attribute”.

### **Step 2: Create a specific new Life**

Next, you create a file **\app\code\Magento\TipandTrick\etc\extension\_attributes.xml** in the extension folder with following content:

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Api/etc/extension\_attributes.xsd">

<extension\_attributes for="Magento\Sales\Api\Data\OrderInterface">

<attribute code="tip\_and\_trick\_attribute" type="string" />

</extension\_attributes>

</config>

You have to create file extension\_attributes.xml to add the custom attribute because Magento 2 doesn’t accept to add a new field in the response on the Rest API

### **Step 3: Create observe**

In this step, you need to create observe for event ‘sales\_order\_load\_after’ through the file \app\code\Magento\TipandTrick\etc\events.xml to add the custom attribute into extension\_attributes.xml.

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Event/etc/events.xsd">

<event name="sales\_order\_load\_after">

<observer name="sales\_order\_load\_tip\_and\_trick\_attribute" instance="Magestore\TipAndTrick\Observer\Sales\OrderLoadAfter" />

</event>

</config>

**Step 4: Create a file a handle the event**

Create a file app\code\Magento\TipandTrick\Observer\Sales\OrderLoadAfter.php to handle the event ‘sales\_order\_load\_after’ which is declared above

<?php

namespace Magestore\TipAndTrick\Observer\Sales;

use Magento\Framework\Event\ObserverInterface;

class OrderLoadAfter implements ObserverInterface

{

public function execute(\Magento\Framework\Event\Observer $observer)

{

$order = $observer->getOrder();

$extensionAttributes = $order->getExtensionAttributes();

if ($extensionAttributes === null) {

$extensionAttributes = $this->getOrderExtensionDependency();

}

$attr = $order->getData('tip\_and\_trick\_attribute');

$extensionAttributes->setTipAndTrickAttribute($attr);

$order->setExtensionAttributes($extensionAttributes);

}

private function getOrderExtensionDependency()

{

$orderExtension = \Magento\Framework\App\ObjectManager::getInstance()->get(

'\Magento\Sales\Api\Data\OrderExtension'

);

return $orderExtension;

}

}

In execute function, you get the custom attribute from order data and set it into extension attributes of the orders.

### **Step 5: Delete folder to apply function**

Delete folder var\generation to apply function setTipandTrickAttribute’ and ‘getTipandTrickAttribute’ in file‘var\generation\Magento\Sales\Api\Data\OrderExtension.php’. This file will be auto-created.

After getting order on Magento 2 Rest API, you can see the custom attribute in the response.

# Create Payment methods in Magento 2

Payment methods are one of important part in a Magento store as they relate directly to purchases in your store. [Magento 2](https://magehit.com/blog/magento-2-tutorial/) supports multiple Payment methods for Magento store so in this post, we will show you the way to create a payment method for Magento store generally to help you generate any Payment method you want.

Let’s start!

## **Step 1: Create Payment method module**

* Create registration.php file with following lines:
* <?php
* \Magento\Framework\Component\ComponentRegistrar::register(
* \Magento\Framework\Component\ComponentRegistrar::MODULE,
* 'Mageplaza\_Payment',
* \_\_DIR\_\_

);

* Declare file module.xml
* <?xml version="1.0"?>
* <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">
* <module name="Magehit\_Payment" setup\_version="0.1.0">
* <sequence>
* <module name="Magento\_Sales"/>
* <module name="Magento\_Payment"/>
* <module name="Magento\_Checkout"/>
* <module name="Magento\_Directory" />
* <module name="Magento\_Config" />
* </sequence>
* </module>

</config>

You have to run module above after Magento\_Sales, Magento\_Checkout, Magento\_Payment, Magento\_Config and Magento\_Directory.

## **Step 2: Declare Payment method module**

* Create payment.xml file in folder etc
* <?xml version="1.0" ?>
* <payment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
* xsi:noNamespaceSchemaLocation="urn:magento:module:Magento\_Payment:etc/payment.xsd">
* <groups>
* <group id="offline">
* <label>Offline Payment Methods</label>
* </group>
* </groups>
* <methods>
* <method name="simple">
* <allow\_multiple\_address>1</allow\_multiple\_address>
* </method>
* </methods>
* </payment>
* Create file config.xml in folder etc:
* <?xml version="1.0" ?>
* <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
* xsi:noNamespaceSchemaLocation="urn:magento:module:Magento\_Store:etc/config.xsd">
* <default>
* <payment>
* <simple>
* <active>1</active>
* <model>Magehit\Payment\Model\Payment\Simple</model>
* <order\_status>pending</order\_status>
* <title>Simple</title>
* <allowspecific>0</allowspecific>
* <group>Offline</group>
* </simple>
* </payment>
* </default>

</config>

Model of the Payment method is declared in this file and called Simple

* Create file Simple model: We will create Model/Payment/Simple.php file with following lines:
* <?php
* namespace Magehit\Payment\Model\Payment;
* class Simple extends \Magento\Payment\Model\Method\Cc
* {
* protected $\_isGateway = true;
* protected $\_canCapture = true;
* protected $\_canCapturePartial = true;
* protected $\_canRefund = true;
* protected $\_canRefundInvoicePartial = true;
* protected $\_stripeApi = false;
* protected $\_countryFactory;
* protected $\_minAmount = null;
* protected $\_maxAmount = null;
* protected $\_supportedCurrencyCodes = array('USD');
* protected $\_debugReplacePrivateDataKeys
* = ['number', 'exp\_month', 'exp\_year', 'cvc'];
* public function \_\_construct(\Magento\Framework\Model\Context $context,
* \Magento\Framework\Registry $registry,
* \Magento\Framework\Api\ExtensionAttributesFactory $extensionFactory,
* \Magento\Framework\Api\AttributeValueFactory $customAttributeFactory,
* \Magento\Payment\Helper\Data $paymentData,
* \Magento\Framework\App\Config\ScopeConfigInterface $scopeConfig,
* \Magento\Payment\Model\Method\Logger $logger,
* \Magento\Framework\Module\ModuleListInterface $moduleList,
* \Magento\Framework\Stdlib\DateTime\TimezoneInterface $localeDate,
* \Magento\Directory\Model\CountryFactory $countryFactory,
* array $data = array()
* ) {
* parent::\_\_construct(
* $context, $registry, $extensionFactory, $customAttributeFactory,
* $paymentData, $scopeConfig, $logger, $moduleList, $localeDate, null,
* null, $data
* );
* $this->\_countryFactory = $countryFactory;
* $this->\_minAmount = $this->getConfigData('min\_order\_total');
* $this->\_maxAmount = $this->getConfigData('max\_order\_total');
* }
* /\*\*
* \* Authorize payment abstract method
* \*
* \* @param \Magento\Framework\DataObject|InfoInterface $payment
* \* @param float $amount
* \* @return $this
* \* @throws \Magento\Framework\Exception\LocalizedException
* \* @api
* \* @SuppressWarnings(PHPMD.UnusedFormalParameter)
* \*/
* public function authorize(\Magento\Payment\Model\InfoInterface $payment, $amount)
* {
* if (!$this->canAuthorize()) {
* throw new \Magento\Framework\Exception\LocalizedException(\_\_('The authorize action is not available.'));
* }
* return $this;
* }
* /\*\*
* \* Capture payment abstract method
* \*
* \* @param \Magento\Framework\DataObject|InfoInterface $payment
* \* @param float $amount
* \* @return $this
* \* @throws \Magento\Framework\Exception\LocalizedException
* \* @api
* \* @SuppressWarnings(PHPMD.UnusedFormalParameter)
* \*/
* public function capture(\Magento\Payment\Model\InfoInterface $payment, $amount)
* {
* if (!$this->canCapture()) {
* throw new \Magento\Framework\Exception\LocalizedException(\_\_('The capture action is not available.'));
* }
* return $this;
* }
* /\*\*
* \* Refund specified amount for payment
* \*
* \* @param \Magento\Framework\DataObject|InfoInterface $payment
* \* @param float $amount
* \* @return $this
* \* @throws \Magento\Framework\Exception\LocalizedException
* \* @api
* \* @SuppressWarnings(PHPMD.UnusedFormalParameter)
* \*/
* public function refund(\Magento\Payment\Model\InfoInterface $payment, $amount)
* {
* if (!$this->canRefund()) {
* throw new \Magento\Framework\Exception\LocalizedException(\_\_('The refund action is not available.'));
* }
* return $this;
* }

}

There are 3 main functions created in this model:

* authorize(): Authorize the payment
* capture(): Capture money from a customer
* refund(): Chargeback money to the customer

## **Step 3: Show Payment method in Checkout page**

* Create layout file view/frontend/layout/checkout\_index\_index.xml:
* <?xml version="1.0" ?>
* <pagepage layout="1column" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
* xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page\_configuration.xsd">
* <body>
* <referenceBlock name="checkout.root">
* <arguments>
* <argument name="jsLayout" xsi:type="array">
* <item name="components" xsi:type="array">
* <item name="checkout" xsi:type="array">
* <item name="children" xsi:type="array">
* <item name="steps" xsi:type="array">
* <item name="children" xsi:type="array">
* <item name="billing-step" xsi:type="array">
* <item name="children" xsi:type="array">
* <item name="payment" xsi:type="array">
* <item name="children" xsi:type="array">
* <item name="renders" xsi:type="array">
* <item name="children" xsi:type="array">
* <item name="offline-payments" xsi:type="array">
* <item name="component" xsi:type="string">Magento\_OfflinePayments/js/view/payment/offline-payments</item>
* <item name="methods" xsi:type="array">
* <item name="simple" xsi:type="array">
* <item name="isBillingAddressRequired" xsi:type="boolean">true</item>
* </item>
* </item>

</item>

* Create js files in order to load template KO: We will create file /view/frontend/web/js/view/payment/simple.js
* define(
* [
* 'uiComponent',
* 'Magento\_Checkout/js/model/payment/renderer-list'
* ],
* function (Component,
* rendererList) {
* 'use strict';
* rendererList.push(
* {
* type: 'simple',
* component: 'Magehit\_Payment/js/view/payment/method-renderer/simple-method'
* }
* );
* return Component.extend({});
* }

);

Another component js/view/method-renderer/simple-method is called in file above.

Then create file /view/frontend/web/js/view/payment/method-renderer/simple-method.js

define(

[

'Magento\_Checkout/js/view/payment/default'

],

function (Component) {

'use strict';

return Component.extend({

defaults: {

template: 'Magehit\_Payment/payment/simple'

},

getMailingAddress: function () {

return window.checkoutConfig.payment.checkmo.mailingAddress;

},

});

}

);

* Create KO template file:
* <div class="payment-method" data-bind="css: {'\_active': (getCode() == isChecked())}">
* <div class="payment-method-title field choice">
* <input type="radio"
* name="payment[method]"
* class="radio"
* data-bind="attr: {'id': getCode()}, value: getCode(), checked: isChecked, click: selectPaymentMethod, visible: isRadioButtonVisible()"/>
* <label data-bind="attr: {'for': getCode()}" class="label"><span data-bind="text: getTitle()"></span></label>
* </div>
* <div class="payment-method-content">
* <!-- ko foreach: getRegion('messages') -->
* <!-- ko template: getTemplate() --><!-- /ko -->
* <!--/ko-->
* <div class="payment-method-billing-address">
* <!-- ko foreach: $parent.getRegion(getBillingAddressFormName()) -->
* <!-- ko template: getTemplate() --><!-- /ko -->
* <!--/ko-->
* </div>
* <div class="checkout-agreements-block">
* <!-- ko foreach: $parent.getRegion('before-place-order') -->
* <!-- ko template: getTemplate() --><!-- /ko -->
* <!--/ko-->
* </div>
* <div class="actions-toolbar">
* <div class="primary">
* <button class="action primary checkout"
* type="submit"
* data-bind="
* click: placeOrder,
* attr: {title: $t('Place Order')},
* css: {disabled: !isPlaceOrderActionAllowed()},
* enable: (getCode() == isChecked())
* "
* disabled>
* <span data-bind="i18n: 'Place Order'"></span>
* </button>
* </div>
* </div>
* </div>

</div>

Finally, you need to download Payment method you‘ve created.

# Magento 2 logging

Logging variables or customizing messages are sometimes appeared in every development process. Basing on this necessary, we will explain the way to create a custom log. Magento 2 comes with built-in log facility which is based on Monolog library. This package can be found in the following location: “MAGENTO2\_ROOT/vendor/monolog“.

The main **Magento 2 log** facility class is “Magento\Framework\Logger\Monolog**“**, and this is defined in “MAGENTO2\_ROOT/app/etc/di.xml” as:

<preference for="Psr\Log\LoggerInterface" type="Magento\Framework\Logger\Monolog" />

This class extends class “Monolog\Logger” can be seen from monolog package.

<?php

/\*\*

\* Copyright © 2015 Magento. All rights reserved.

\* See COPYING.txt for license details.

\*/

namespace Magento\Framework\Logger;

use Monolog\Logger;

class Monolog extends Logger

{

}

Class “Monolog\Logger” includes a couple of interesting methods for creating logs.  
These methods accept two arguments, in which the first one is message (string) and the second one is optional array parameter (you can pass instance of object).

Some methods:

$this->\_logger->addDebug($message); // log location: var/log/system.log

$this->\_logger->addInfo($message); // log location: var/log/exception.log

$this->\_logger->addNotice($message); // log location: var/log/exception.log

$this->\_logger->addError($message); // log location: var/log/exception.log

$this->\_logger->critical($e); // log location: var/log/exception.log

One useful example for logging php exception:

**For Magento 1**: we are using static method

Mage::logException($e);

**For Magento 2**: we are using an instance of “Magento\Framework\Logger\Monolog” and method “critical” for logging exception from try-catch

$this->\_logger->critical($e);

// instance of $e will be converted to string (magic metod \_\_toString() will be called).

Let’s start with an example of how to get instance of Magento\Framework\Logger\Monolog in your class.

Magento 2 uses dependency injection and all instances of classes come through class constructor.

If you want to use object “Magento\Framework\Logger\Monolog” then the instance should be passed through constructor of your class.

Example of this code is shown below:

<?php

namespace Inchoo\Test\Model;

class Example{

protected $\_logger;

public function \_\_construct(

\Psr\Log\LoggerInterface $logger, //log injection

array $data = []

) {

$this->\_logger = $logger;

parent::\_\_construct($data);

}

public function someExampleMethod() {

/\*

some logic of method

\*/

//accessing to logger instance and calling log method

$this->\_logger->addDebug('some text or variable');

}

}

You can see that we passed “\Psr\Log\LoggerInterface $logger” in class through constructor in order to be able to use log object in this class.  
After that we can use instance “$this->\_logger” in class “Inchoo\Test\Model\Example“.

If you want to write a log in your custom file name, in a custom location, then the process is not that simple, you will have to create a custom Log handler.

Main **Magento 2 log** class has three handlers which are defined in the same “MAGENTO2\_ROOT/app/etc/di.xml” file. These handlers are: exception, system and debug.

<type name="Magento\Framework\Logger\Monolog">

<arguments>

<argument name="name" xsi:type="string">main</argument>

<argument name="handlers" xsi:type="array">

<item name="exception" xsi:type="object">Magento\Framework\Logger\Handler\Critical</item>

<item name="system" xsi:type="object">Magento\Framework\Logger\Handler\System</item>

<item name="debug" xsi:type="object">Magento\Framework\Logger\Handler\Debug</item>

</argument>

</arguments>

</type>

In some classes, passing log object through constructor is not needed, because the log object already exists.

For example, we have this situation on every html block class which extends “\Magento\Framework\View\Element\Template” or in model class which extends “\Magento\Framework\Model\AbstractModel“.

These “parent” classes have already property “$\_logger” instance of: Magento\Framework\Logger.

We have finished this tutorial and we hope it’s useful for you. Other helpful guides about [Magento 2.0](http://magehit.com/blog/magento-2/) will be implemented in the subsequent posts. We are looking forward to receiving all feedbacks as well as queries about Magento 2.0. Don't forget to share us your ideas or leave your comments!

Now, let’s try and experience new things from [**Magento 2 extensions**](http://magehit.com/magento-2-extension.html)!

How to create a new offline payment method in Magento 2?

As you know, it is easier when using online payment method for your Magento store. However, sometimes project may require a custom codes payment method. Therefore, you need to consider about programmatic product import and order creation script that might specialize in some specific labeled payment method. Thus, the payment method will be controlled.

In this post, we will introduce another way for payment-offline payment method- for your Magento store.

1. Create ***xml*** file in ***app/code/local/[name\_space]/[your\_module]/etc/adminhtml*** and then use code below:

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../../Config/etc/system\_file.xsd">

<system>

<section id="payment">

<group id="newpayment" translate="label" type="text" sortOrder="100" showInDefault="1" showInWebsite="1" showInStore="1">

<label>Newpayment</label>

<field id="active" translate="label" type="select" sortOrder="1" showInDefault="1" showInWebsite="1" showInStore="0">

<label>Enabled</label>

<source\_model>Magento\Config\Model\Config\Source\Yesno</source\_model>

</field>

<field id="title" translate="label" type="text" sortOrder="10" showInDefault="1" showInWebsite="1" showInStore="1">

<label>Title</label>

</field>

<field id="order\_status" translate="label" type="select" sortOrder="20" showInDefault="1" showInWebsite="1" showInStore="0">

<label>New Order Status</label>

<source\_model>[name\_space]\[your\_module]\Model\Config\Source\Order\Status\Pendingpayment</source\_model>

</field>

<field id="allowspecific" translate="label" type="allowspecific" sortOrder="40" showInDefault="1" showInWebsite="1" showInStore="0">

<label>Payment from Applicable Countries</label>

<source\_model>Magento\Payment\Model\Config\Source\Allspecificcountries</source\_model>

</field>

<field id="specificcountry" translate="label" type="multiselect" sortOrder="41" showInDefault="1" showInWebsite="1" showInStore="0">

<label>Payment from Specific Countries</label>

<source\_model>Magento\Directory\Model\Config\Source\Country</source\_model>

<can\_be\_empty>1</can\_be\_empty>

</field>

<field id="instructions" translate="label" sortOrder="50" showInDefault="1" showInWebsite="1" showInStore="1">

<label>Instructions</label>

</field>

<field id="sort\_order" translate="label" type="text" sortOrder="100" showInDefault="1" showInWebsite="1" showInStore="0">

<label>Sort Order</label>

<frontend\_class>validate-number</frontend\_class>

</field>

</group>

</section>

</system>

</config>

1. Create ***xml*** file in ***app/code/local/[name\_space]/[your\_module]/etc***

And then apply following command:

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../Store/etc/config.xsd">

<default>

<payment>

<newpayment>

<active>1</active>

<title>Newpayment</title>

<order\_status>pending\_payment</order\_status>

<instructions>Instruction.</instructions>

<payment\_action>true</payment\_action>

<model>[name\_space]\[your\_module]\Model\Newpayment</model>

<group>offline</group>

</newpayment>

</payment>

</default>

</config>

1. Create ***xml*** file in ***app/code/local/[name\_space]/[your\_module]/etc*** and then enter following code:

<payment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="../../../Magento/Payment/etc/payment.xsd">

<groups>

<group id="offline">

<label>Offline Payment Methods</label>

</group>

</groups>

<methods>

<method name="newpayment">

<allow\_multiple\_address>1</allow\_multiple\_address>

</method>

</methods>

</payment

1. Create ***php*** in ***app/code/local/[name\_space]/[your\_module]/Model*** and then apply code below:

<?php

namespace [name\_space]\[your\_module]\Model;

/\*\*

\* Pay In Store payment method model

\*/

class Newpayment extends \Magento\Payment\Model\Method\AbstractMethod

{

/\*\*

\* Payment code

\*

\* @var string

\*/

protected $\_code = 'newpayment';

/\*\*

\* Availability option

\*

\* @var bool

\*/

protected $\_isOffline = true;

}

1. Create ***php*** file in ***app/code/local/[name\_space]/[your\_module]/Model/Config/Source/Order/Status***and use following command:

<?php

namespace [name\_space]\[your\_module]\Model\Config\Source\Order\Status;

use Magento\Sales\Model\Order;

use Magento\Sales\Model\Config\Source\Order\Status;

/\*\*

\* Order Status source model

\*/

class Pendingpayment extends Status

{

/\*\*

\* @var string[]

\*/

protected $\_stateStatuses = [Order::STATE\_PENDING\_PAYMENT];

}

1. Create ***layout:*** create file ***xml*** in ***app/code/local/[name\_space]/[your\_module]/view/frontend/layout***and then apply code below:

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="1column" xsi:noNamespaceSchemaLocation="../../../../../../../lib/internal/Magento/Framework/View/Layout/etc/page\_configuration.xsd">

<body>

<referenceBlock name="checkout.root">

<arguments>

<argument name="jsLayout" xsi:type="array">

<item name="components" xsi:type="array">

<item name="checkout" xsi:type="array">

<item name="children" xsi:type="array">

<item name="steps" xsi:type="array">

<item name="children" xsi:type="array">

<item name="billing-step" xsi:type="array">

<item name="children" xsi:type="array">

<item name="payment" xsi:type="array">

<item name="children" xsi:type="array">

<item name="renders" xsi:type="array">

<!-- merge payment method renders here -->

<item name="children" xsi:type="array">

<item name="newpayment-payments" xsi:type="array">

<item name="component" xsi:type="string">[name\_space]\_[your\_module]/js/view/payment/newpayment</item>

<item name="methods" xsi:type="array">

<item name="newpayment" xsi:type="array">

<item name="isBillingAddressRequired" xsi:type="boolean">true</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</item>

</argument>

</arguments>

</referenceBlock>

</body>

</page>

1. Create ***js*** file in ***app/code/local/[name\_space]/[your\_module]/view/frontend/web/js/view/payment***and then apply command below:

define(

[

'uiComponent',

'Magento\_Checkout/js/model/payment/renderer-list'

],

function (

Component,

rendererList

) {

'use strict';

rendererList.push(

{

type: 'newpayment',

component: '[name\_space]\_[your\_module]/js/view/payment/method-renderer/newpayment-method'

}

);

/\*\* Add view logic here if needed \*/

return Component.extend({});

}

);

1. Create ***newpayment-method.js*** file in ***app/code/local/[name\_space]/[your\_module]/view/frontend/web/js/view/payment/method-renderer***and then enter command:

define(

[

'Magento\_Checkout/js/view/payment/default'

],

function (Component) {

'use strict';

return Component.extend({

defaults: {

template: '[name\_space]\_[your\_module]/payment/newpayment'

},

/\*\* Returns send check to info \*/

getMailingAddress: function() {

return window.checkoutConfig.payment.checkmo.mailingAddress;

},

});

}

);

1. Create template file ***html*** in ***app/code/local/[name\_space]/[your\_module]/view/frontend/web/template/payment***and then apply following code:

<div class="payment-method" data-bind="css: {'\_active': (getCode() == isChecked())}">

<div class="payment-method-title field choice">

<input type="radio"

name="payment[method]"

class="radio"

data-bind="attr: {'id': getCode()}, value: getCode(), checked: isChecked, click: selectPaymentMethod, visible: isRadioButtonVisible()"/>

<label data-bind="attr: {'for': getCode()}" class="label"><span data-bind="text: getTitle()"></span></label>

</div>

<div class="payment-method-content">

<!-- ko foreach: getRegion('messages') -->

<!-- ko template: getTemplate() --><!-- /ko -->

<!--/ko-->

<div class="payment-method-billing-address">

<!-- ko foreach: $parent.getRegion(getBillingAddressFormName()) -->

<!-- ko template: getTemplate() --><!-- /ko -->

<!--/ko-->

</div>

<div class="checkout-agreements-block">

<!-- ko foreach: $parent.getRegion('before-place-order') -->

<!-- ko template: getTemplate() --><!-- /ko -->

<!--/ko-->

</div>

<div class="actions-toolbar">

<div class="primary">

<button class="action primary checkout"

type="submit"

data-bind="

click: placeOrder,

attr: {title: $t('Place Order')},

css: {disabled: !isPlaceOrderActionAllowed()},

enable: (getCode() == isChecked())

"

disabled>

<span data-bind="i18n: 'Place Order'"></span>

</button>

</div>

</div>

</div>

</div>

# How to join data between two tables in Magento 2?

All users need to get data from two or more table regularly when working with Magento 2 project. However, almost users have not known how to join data between some tables. So, in this post, we will cover the way to do this. In this tutorial, we will get all order that created with a specific payment method - “Check Money Order”.

Now, let’s start determine this tutorial!

### **Step 1: Create a Collection class that extends:**

\Magento\Framework\Model\ResourceModel\Db\Collection\AbstractCollection

class Collection extends \Magento\Framework\Model\ResourceModel\Db\Collection\AbstractCollection

In Collection class, you need to have a construction function with two parameters your model name and Sale order resource model from Magento sales module

protected function \_construct()

{

$this-&gt;\_init('YourCompany\YourModule\Model\YourModel', 'Magento\Sales\Model\ResourceModel\Order');

}

### **Step 2: Create your own function**

You have to build your own function to get data

protected function filterOrder($payment\_method)

{

$this-&gt;sales\_order\_table = "main\_table";

$this-&gt;sales\_order\_payment\_table = $this-&gt;getTable("sales\_order\_payment");

$this-&gt;getSelect()

-&gt;join(array('payment' =&gt; $this-&gt;sales\_order\_payment\_table), $this-&gt;sales\_order\_table . '.entity\_id= payment.parent\_id',

array('payment\_method' =&gt; 'payment.method',

'order\_id' =&gt; $this-&gt;sales\_order\_table.'.entity\_id'

)

);

$this-&gt;getSelect()-&gt;where("payment\_method=".$payment\_method);

}

Then, in your model you just need to get the collection and call to filterOrder function above

$collection = $this-&gt;YourCollectionFactory-&gt;create();

$collection-&gt;filterOrder("checkmo");

foreach ($collection as $item)

{

//do what you want with the data here.

}

# How to set Developer mode in Magento 2?

[Magento 2](https://magehit.com/magento-2-extensions.html) added a command for switching Magento modes from Developer to Production and vice versa in order to increase security in website.

**When changing to developer mode, Magento 2 clears contents of directories below:**

var/cache

var/di

var/generation

var/view\_preprocessed

pub/static

But **.htaccess**files and files specifying the version of static content in **pub/static** are not removed.

## **Let’s start setting Developer mode for Magento 2!**

### **Step 1: Display current mode**

You should run the command as the Magento file system owner. Use following command:

magento deploy:mode:show

If command above doesn’t work, use following command under Magento root folder:

php bin/magento deploy:mode:set developer

### **Step 2: Change mode**

When changing to Developer mode, you need delete generated classes and Object Manager entitles in order to protect errors.

Then us following command to change to Developer mode:

magento deploy:mode:set {mode} [-s| --skip-compilation]

In case it doesn’t work, use this command:

php bin/magento deploy:mode:set {mode} [-s| --skip-compilation]

# Overriding a block in Magento 2

In each Magento project, there is always a certain level of customization required, which can involve adding entirely new elements or overriding the existing ones. Customizing your Magento store according to your preferences not only gives you a personalized feel and ease of use while handing tasks but also improves the customer user experience in several levels that can benefit your business with increased conversions and revenues. Besides completely modifying core files, Magento recommends to override the core files if needed.

## **There are few steps to override a block in Magento 2**

### **Step 1: Build a Magento 2 extension structure**

* Build directories:

magento2 --- app --- code

|--- Sample --- HelloWorld

| --- Block

| --- etc

| --- module.xml

| --- di.xml

* Create module.xml to define a Magento 2 extension:

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../../../../lib/internal/Magento/Framework/Module/etc/module.xsd">

<module name="Sample\_HelloWorld" setup\_version="0.0.1"/>

</config>

### **Step 2: Set preference in di.xml**

* Create di.xml to refer the overriding block class:

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../../../../../lib/internal/Magento/Framework/ObjectManager/etc/config.xsd">

<preference for="Magento\Theme\Block\Html\Title" type="Sample\HelloWorld\Block\HelloTitle" />

</config>

### **Step 3: Define an overriding class**

* Define HelloTitle.php under **magento2/app/code/Sample/HelloWorld/Block**:

<?php

namespace Sample\HelloWorld\Block;

use Magento\Framework\View\Element\Template;

class HelloTitle extends \Magento\Theme\Block\Html\Title

{

public function getPageTitle()

{

return 'haha9999';

}

protected function \_toHtml()

{

$this->setModuleName($this->extractModuleName('Magento\Theme\Block\Html\Title'));

return parent::\_toHtml();

}

}

# How to Import and Export products in Magento 2?

In this post, we introduce the way to **import and export product in Magento 2**. This [tutorial](https://magehit.com/blog/magento-2-tutorial/) is written in detail so it is easy for all merchants to follow.

1. **Get a Magento formatted CSV file**

Manually create your own CSV file

If you decide to start with a blank CSV file, ensure your file is formatted correctly for a [Magento 2](https://magehit.com/blog/magento-2/)site. Here are a few tips:

* Your CSV must be UTF-8 format
* Make sure the attribute code and not attribute label is used when populating the headings in your spreadsheet
* Ensure to save the file in .CSV format
* Stick with default options

1. **Add a product to your spreadsheet**

* I would highly recommend using Open Office as your spreadsheet of choice. Once installed, right click on CSV file you’ve exported from Magento and open it with Open Office

1. **Upload your data to your Magento 2 store**

* To import products from your CSV file, navigate **System> Data Transfer> Import**
* Choose **Products**
* In **Import Behavior**, there are 3 options:
* **Add/Update**: New data will be added to existing products or new products will be created if they don’t already exist in store. the only field that can’t be updated in SKU
* **Replace**: Existing data will be replaced with the new data from spreadsheet
* **Delete**: If a product in the spreadsheet matches a product in store, this product will be deleted from database
* Choose your **Field Separator** and **Multiple Value Separators**. Although by default Magento use a comma as a **Field Separator**, you can use a pipe instead. If you are using pipe as the field separator, you need to configure your CSV file to work with a pipe
* Tap **Select** your file then select **Check Data**
* Once receiving an Ok go message, tap **Import**
* When you complete setting, you will receive the following message Import successfully done

1. **Upload product images and final tweaks**

Make any final amendments to your products

1. **Test**
2. **Enable/Disable your products**

To Import/Export products in your Magento 2 store, click on the **Enable Product** toggle.

# How to import product with images in Magento 2?

When **importing products in Magento 2** store, there is Images File Directory file of the path leading to your product image file. In this post, we will show you two ways to import your product images as with Local server and with External Server.

## **Import Images from the local server**

* Upload the image files to the default folder for importing product images pub/media/import. The different folder is also acceptable as long as the path to the folder is specified during the import process.
* Fill the name of each image file to be imported on the correct row, by sku, and column based on image type in CSV data. You have to ensure that the relative path to the import directory is included before each file name
* Do following the instruction in importing products till the filling in the **Images File Directory**field
* Enter the path to the folder that you upload your images in the beginning in **Images File Directory** fields and continue importing products

## **Import images from External Server**

* Upload the images need importing to the selected folder on the external server
* In CSV data, enter the full URL for each image file in correct column by image type
* Do following the instruction in importing products till the filling in the **Images File Directory**field
* Enter the path to the folder that you upload your images in the beginning in **Images File Directory** fields and continue importing products

# How to get media url, static url, link url and base url in Magento 2?

Today we'll show how to get get Media URL, Static URL, Link URL and Base URL in Magento 2. To get URLs in Magento 2, the first you have to define Block file with \_\_construct() method with define storeManagerInterface in construct method.

**Create instance of \Magento\Framework\App\ObjectManager to get current store:**

$\_objectManager = \Magento\Framework\App\ObjectManager::getInstance(); //instance of\Magento\Framework\App\ObjectManager

$storeManager = $\_objectManager->get('Magento\Store\Model\StoreManagerInterface');

$currentStore = $storeManager->getStore();

**Getting Base URL:**

$this->\_storeManager->getStore()->getBaseUrl()

**Getting Media URL:**

$mediaUrl = $currentStore->getBaseUrl(\Magento\Framework\UrlInterface::URL\_TYPE\_MEDIA);

**Getting Link URL:**

$this->\_storeManager->getStore()->getBaseUrl(\Magento\Framework\UrlInterface::URL\_TYPE\_LINK);

**Getting Static URL:**

$this->\_storeManager->getStore()->getBaseUrl(\Magento\Framework\UrlInterface::URL\_TYPE\_STATIC);

# Add top link in Magento 2 website

**Top links** are default block generally used to create a personal area like My account, My Wishlist, Log in, etc. to add top link in Magento 2 website, you just need to perform 2 simple steps.

## **Step 1: Create/ custom default.xml file**

You need to create/ custom default.xml file in **app/code/[Name\_Space]/[Your\_Module]/view/frontend/layout**by applying command below:

<?xml version="1.0"?>

<!--

/\*\*

\* Copyright © 2015 Magento. All rights reserved.

\* See COPYING.txt for license details.

\*/

-->

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page\_configuration.xsd">

<body>

<referenceBlock name="header.links">

<block class="[Name\_Space]\[Your\_Module]\Block\Link" name="add-new-header-link">

<arguments>

<argument name="label" xsi:type="string" translate="true">New Link</argument>

<argument name="path" xsi:type="string" translate="true">new-link</argument>

</arguments>

</block>

</referenceBlock>

</body>

## **Step 2: Create Link.php file**

You need to create Link.php file in app**/code/[Name\_Space]/[Your\_Module]/Block**by using following command:

<?php

namespace [Name\_Space]\[Your\_Module]\Block;

class Link extends \Magento\Framework\View\Element\Html\Link

{

/\*\*

\* Render block HTML.

\*

\* @return string

\*/

protected function \_toHtml()

{

if (false != $this->getTemplate()) {

return parent::\_toHtml();

}

return '<li><a ' . $this->getLinkAttributes() . ' >' . $this->escapeHtml($this->getLabel()) . '</a></li>';

}

}

# How to remove Default Magento 2 footer links?

As you can see, there are some various links like Privacy and Cookie Policy, Search terms, Contact Us, Orders and Returns, [Advanced search](https://magehit.com/magento-advanced-search-extension.html) etc on the top of your [**Magento 2**](http://magehit.com/magento-2-extensions.html) website. They are sometimes creating a few inconveniences so some Magento store owners want to remove them. In this post, we support **a very simple way to ignore the unnecessary footers by some easy steps**.

Now, let’s start!

### **Step 1: Contact us**

You just need to go to Stores→ Configuration→ General tab→ Contact Link  
After that, select Enable Contact Us Link to NO

### **Step 2: Remove footer links**

**For Privacy and Cookie Policy**

* Go to \app\code\Magento\Cms\view\frontend\Layout\default.xml
* Comment block Line no- 13

<!–referenceBlock name=”footer\_links”>

<block class=”Magento\Framework\View\Element\Html\Link\Current” name=”privacy-policy-link”>

<arguments>

<argument name=”label” xsi:type=”string”>Privacy and Cookie Policy</argument>

<argument name=”path” xsi:type=”string”>privacy-policy-cookie-restriction-mode</argument>

</arguments>

</block>

</referenceBlock–>

**For Search Terms**

* Go to app\code\Magento\Search\view\frontend\Layout\default.xml
* Comment on block Line no- 13

<!–referenceBlock name=”footer\_links”>

<block class=”Magento\Framework\View\Element\Html\Link\Current” ifconfig=”catalog/seo/search\_terms” name=”search-term-popular-link”><arguments>

<argument name=”label” xsi:type=”string” translate=”true”>Search Terms</argument>

<argument name=”path” xsi:type=”string”>search/term/popular</argument></arguments>

</block>

</referenceBlock–>

**For Orders and Returns**

* Go to app\code\Magento\Sales\view\frontend\Layout\default.xml
* Comment block Line no- 16

<!–referenceBlock name=”footer\_links”>

<block class=”Magento\Sales\Block\Guest\Link” name=”sales-guest-form-link”>

<arguments>

<argument name=”label” xsi:type=”string”>Orders and Returns</argument>

<argument name=”path” xsi:type=”string”>sales/guest/form</argument></arguments>

</block>

</referenceBlock–>

**For Advanced Search**

* Go to app\code\Magento\CatalogSearch\view\frontend\Layout\default.xml
* Comment block Line no- 13

<!–referenceBlock name=”footer\_links”>

<block class=”Magento\Framework\View\Element\Html\Link\Current” name=”catalog-search-advanced-link”>

<arguments>

<argument name=”label” xsi:type=”string” translate=”true”>Advanced Search</argument>

<argument name=”path” xsi:type=”string”>catalogsearch/advanced</argument><argument name=”attributes” xsi:type=”array”>

<item name=”data-action” xsi:type=”string”>advanced-search</item></argument>

</arguments>

</block>

</referenceBlock–>