**Quick Overview Of Object Manager In Magento 2**

## **Mechanism**

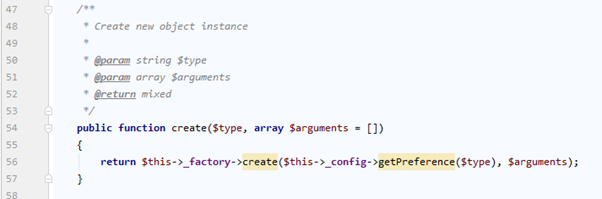
The Object Manager is mainly responsible for the instantiation and configuration of an Object through two main methods: GET, CREATE.

* The GET method returns a singleton object (an instance of the class shared among components when running Magento).
* **vendor/magento/framework/ObjectManager/ObjectManager.php::get()**



* The CREATE method returns an entirely new object (a new class instance).

**vendor/magento/framework/ObjectManager/ObjectManager.php::create()**



Therefore, if you call the GET method from 2 places, the same result will be generated. Otherwise, you will get a new object using the CREATE method.

## **Object Manager configuration**

The di.xml file configures the object manager and tells it how to handle dependency injection

This file specifies the preferred implementation class the object manager creates for the interface declared in a constructor class. The file also determines whether the object manager will create an object for every request or the object is a singleton.

## **Object Manager’s goal**

* Use the object manager to instantiate and insert the declaration class in the constructor.
* public function \_\_construct(
* \Magento\Framework\ObjectManagerInterface $objectManager
* ) {
* $this->\_objectManager = $objectManager;
* }
* Implement the singleton pattern (learn more at https://en.wikipedia.org/wiki/Singleton\_pattern)
* Manage dependencies
* Automatically initialize parameters

According to Magento’s core group, you should not use Object Manager in modules because it makes the class lose dependency injection.

## **You can use Object Manager in the following exceptions:**

* Use Object manager in static magic methods such as \_\_wakeup (), \_\_sleep (), …
* Use Object manager to maintain backward compatibility for a constructor
* The Object manager can depend on classes used to create objects like factories or proxies.

# <https://magefan.com/blog/magento-2-object-manager>

# Singleton pattern

 the **singleton pattern** is a [software design pattern](https://en.wikipedia.org/wiki/Software_design_pattern) that restricts the [instantiation](https://en.wikipedia.org/wiki/Instantiation_(computer_science)) of a [class](https://en.wikipedia.org/wiki/Class_(computer_programming)) to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system.

# Dependency Injection in Magento 2

In Magento 2 Dependency Injection is one of the most useful design pattern.

<https://www.sparsh-technologies.com/blog/dependency-injection-in-magento-2>

### Factory and proxies

Factory and Proxies are some kinds of exceptions for the direct call of Object manager because they need Object manager to generate new objects. As an example, you can overview any kind of DTO factory.

<https://www.atwix.com/magento/design-patterns-in-magento-2-object-manager/>

<https://www.sparsh-technologies.com/blog/dependency-injection-in-magento-2>

## Magento 2 Clean Cache vs. Flush:

You can run both the commands to remove played out items from your website: clean and flush Magento 2 cache. However, it is important that you learn the difference:

### Magento 2 Clean cache:

Since it deletes all enabled cache types, disabled cache types are not cleaned from Magento 2

### Flush Cache

Flushing a cache type purges the cache storage, which might affect other processes applications that are using the same storage. e.g. other websites. Flush cache types if you’ve already tried cleaning the cache and you’re still having issues that you cannot isolate.

* **By Default create() and get() method is responsible for generate new object and retrieve obejct in magento 2.**

<https://magento.stackexchange.com/questions/176018/how-magento2-creates-object-of-factory-classes>

**Object Manager:**

object manager that is responsible for the creation of all the objects.Object Manager resides here  
“Magento\Framework\ObjectManager\ObjectManager”  
it has three methods two of them are responsible for the creation of the objects get() and create(), get is like Mage::getSingleton(“ClassName”) and create is like Mage::getModel(“ClassName”) , **get creates sharable object and create will create new objects** .

<https://webkul.com/blog/magento2-code-generation-and-factory-design-pattern/>

* you must use Factory classes to inject non-injectable objects.

<https://symphisys.com/blog-patterns>

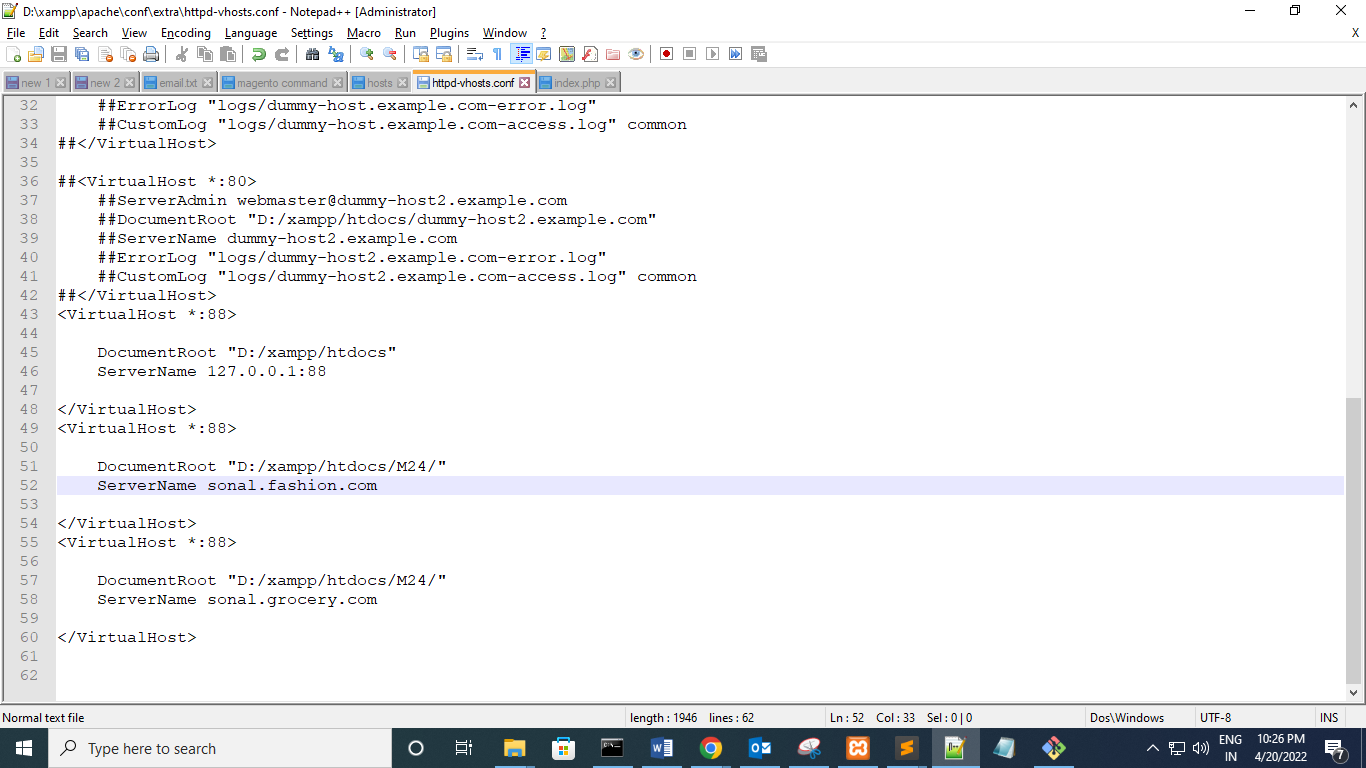
**create multiple website and multiple store:**

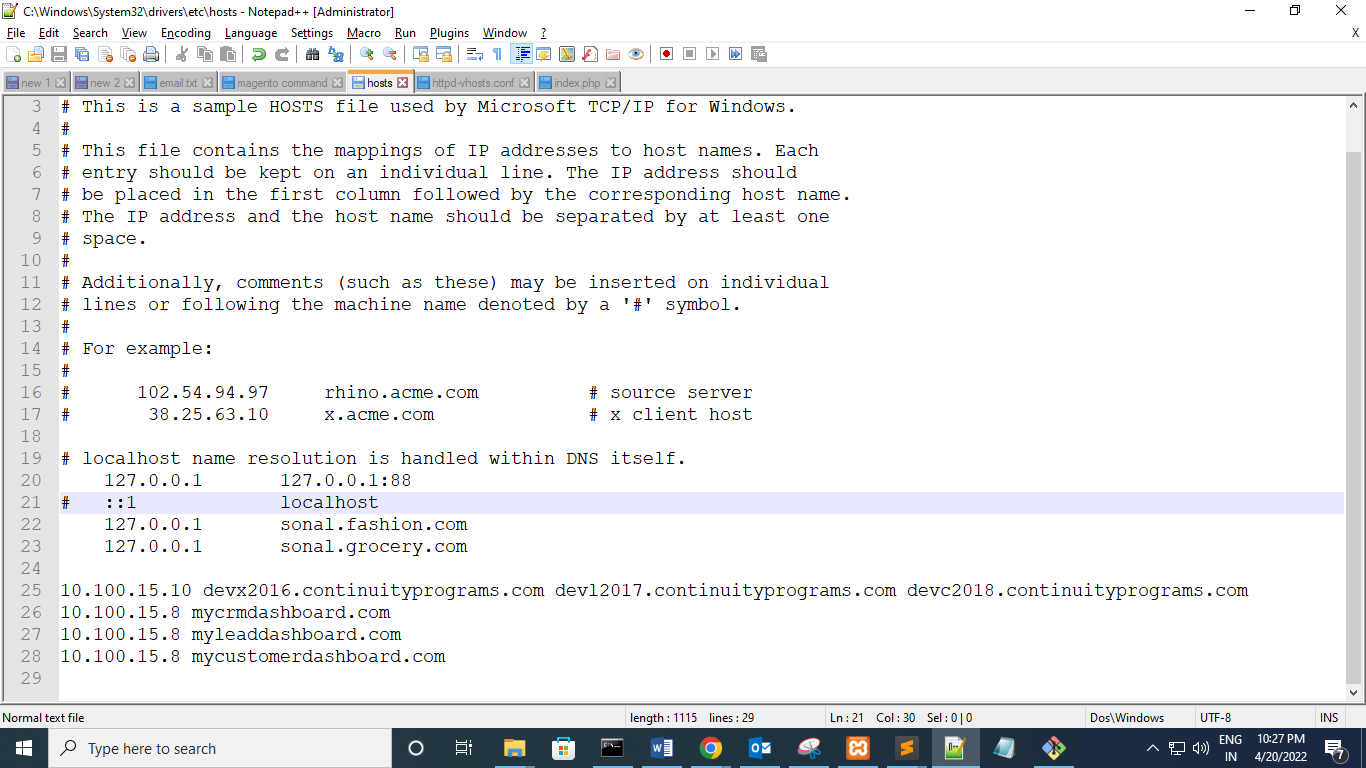
[**https://www.thecoachsmb.com/multiple-website-and-multistore-in-magento-2-on-localhost-xampp-windows/**](https://www.thecoachsmb.com/multiple-website-and-multistore-in-magento-2-on-localhost-xampp-windows/)

**Create virtual Host on local**

**step1:>** D:\xampp\apache\conf\extra\httpd-vhosts.conf

**step2:>** C:\Windows\System32\drivers\etc\hosts





* When we delete default store view

1. Edit store table
2. Store\_group
3. Store\_website

**Table in magento 2**

[**https://www.thecoachsmb.com/how-to-perform-different-operations-on-table-in-magento2-2022/**](https://www.thecoachsmb.com/how-to-perform-different-operations-on-table-in-magento2-2022/)

<table>

**-> Engine:** SQL engine, this value must be InnoDB or memory.

**-> Resource:** The database shard on which to install the table. This value must be default, checkout, or sales

<Column>

**-> identity:** Indicates whether a column is auto incremented.

**-> Type:** blob,blob (includes blob, mediumblob, longblob), boolean, date, datetime, int (includes smallint, bigint, tinyint), real (includes decimal, float, double, real), text (includes text, mediumtext, longtext), timestamp, varbinary, and varchar.

**-> Padding:** indicates the size of the integer column

**-> Length:** indicates the length of a column

**-> precision:** The number of allowed digits in a real data type.

**-> scale:** The number of digits after the decimal in a real data type.

<Constraints>

**-> Type:** primary, unique, or foreign

**-> Referrence\_id:** a custom identifier that is used only for relation mapping in the scope of db\_schema.xml files

**For Remove**

**-> disabled:** Disables or deletes the declared table, column, constraint, or index.

**Please note that migrating data from another table and renaming columns at the same time is not supported.**

[**https://www.thecoachsmb.com/how-to-include-third-party-javascript-libraries/**](https://www.thecoachsmb.com/how-to-include-third-party-javascript-libraries/)

[**https://vrajeshpatel.in/2021/01/24/declarative-schema-in-magento-2-3/**](https://vrajeshpatel.in/2021/01/24/declarative-schema-in-magento-2-3/)

# What is the RequireJS configuration map vs paths vs shim in Magento 2?

[**https://www.rakeshjesadiya.com/require-js-config-map-vs-paths-vs-shim-in-magento/**](https://www.rakeshjesadiya.com/require-js-config-map-vs-paths-vs-shim-in-magento/)

A **requirejs-config.js** file contains the root config object, under the root config object you have to declare all the configuration properties.

*Require js used AMD (Asynchronous module definition) pattern to load the js file in the page.*

* **paths config used to declare third party js file also.**

[**https://www.thecoachsmb.com/what-is-the-requirejs-configuration-map-vs-paths-vs-shim-in-magento-2/**](https://www.thecoachsmb.com/what-is-the-requirejs-configuration-map-vs-paths-vs-shim-in-magento-2/)

# Unlock 11 Necessary Design Patterns for Every Magento 2 Developer

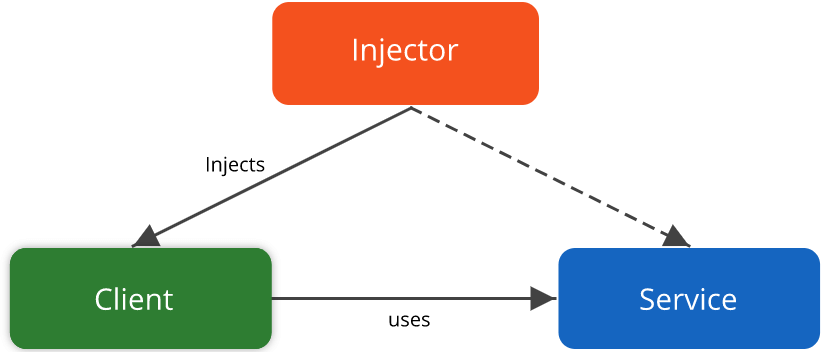
[**https://bsscommerce.com/confluence/magento-2-design-patterns/**](https://bsscommerce.com/confluence/magento-2-design-patterns/)

* Magento 2 Design Patterns can be very helpful if you know the right way to do it.
* design patterns are dispensable in software programming. Thanks to design patterns, code becomes more organized and easier to develop and work with.
* In Magento 1, almost all objects are implemented and called via the Mage class.
* In Magento 2, Object Manager is used to replace the Mage class. This helps to solve some problems when you instantiate parameters by creating a close relationship between 3 patterns: object management, dependency injection, and plugins.
* Object Manager takes the primary responsibility in instantiating and configuring Objects via the two main methods: GET and CREATE.
* GET returns a singleton object (an instance of the class is used to share between components when running Magento, on the other hand).
* CREATE returns an entirely new object (a new class instance).

## **Dependency Injection**

Dependency Injection (DI) is a design pattern to solve programming dependencies and replace the Mage class in Magento 1.

* Service object: is used to declare dependencies
* Client object: depends on service object and inherits from dependencies
* Interface object: defines methods that client can use the service’s dependencies
* Injector object: implements the service’s dependencies and gives them to the client object.
* Dependency is also called a function of services,
* and injection is an action that gives dependence to its dependent object (client). The client can now use the service without building a new one



**Magento coding standared:**

Run the command on magento root directory

1. composer create-project --repository=https://repo.magento.com magento/marketplace-eqp magento-coding-standard

3. Cmd for check errors: magento-coding-standard/vendor/bin/PHPCS --standard=PEAR app/code/Pharmacy/AdditionalPayment/

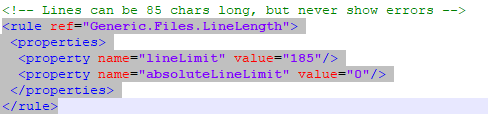
4. Automatic fix the some error: magento-coding-standard/vendor/bin/phpcbf --standard=PEAR app/code/Pharmacy/AdditionalPayment/

-------------

1. Change continue to continue 2 in some file

For fix “Line exceeds 85 characters; contains 107 characters” error

1. D:\xampp\htdocs\ReceiptPharmacy\magento-coding-standard\vendor\squizlabs\php\_codesniffer\src\Standards\PEAR\ruleset.xml

****

1. Past the code in all file in custom module

**/\*\***

**\* PHP\_CodeSniffer tokenises PHP code and detects violations of a**

**\* defined set of coding standards.**

**\***

**\* PHP version 7.2**

**\***

**\* @category PHP**

**\* @package PHP\_CodeSniffer**

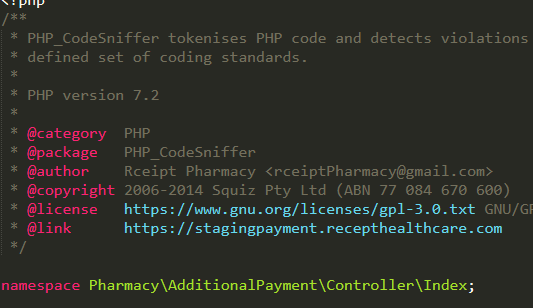
**\* @author Rceipt Pharmacy <rceiptPharmacy@gmail.com>**

**\* @copyright 2006-2014 Squiz Pty Ltd (ABN 77 084 670 600)**

**\* @license https://www.gnu.org/licenses/gpl-3.0.txt GNU/GPLv3**

**\* @link https://stagingpayment.recepthealthcare.com**

**\*/**

****