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

## **Mechanism**

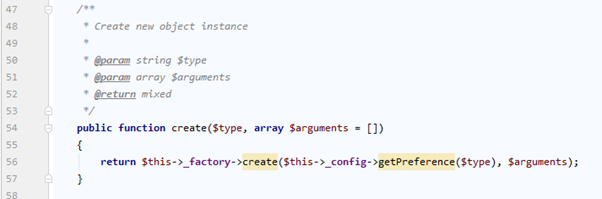
The Object Manager is mainly responsible for the instantiation and configuration of an Object through two main methods: GET, CREATE.

* 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.

# 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/