Aim Of The Singleton Pattern

- Ensure a class has only one instance and provide a global point to retrieve it

Things To Know About Singleton Pattern

- This pattern forms part among one of the patterns originating from the Famous Gang Of Four (aka GoF)
- The singleton pattern in itself is an implementation of the Responsibility Pattern
- The responsibility pattern advocates the use of one object which is solely responsible for doing one specific task
- The singleton is used to create only one single point of entry or functionality within a system or application

Singleton In PHP5

Thoughts Before Implementing Singleton Pattern

1. Will your system need only one instance of this class **throughout**?
2. Will the different component of your system use this class **exactly** the same way

Possible Usage Of Singleton Pattern

- When creating a database layer – you would want one instance to communicate with your database. Thus helping in maintaining only one connection instance instead of multiple one which might otherwise be resource consuming.
- When you have a big/complex system going on; singleton helps in maintaining a synchronized state within your system. For example full-stack framework like Zend and Symfony make a great deal use of singleton within their respective application.
- When you observe that most of the methods you are accessing within your projects, come from one recurring object. This is a candidate for being a singleton.

Anatomy Of A Singleton Pattern In A PHP Context

You need to have these conditions fulfilled:

1. Presence of a static member variable – as a placeholder for the instance of that class
2. Locked down the constructor of that class – simply by making it’s visibility private
3. Prevent any object or instance of that class to be cloned, that is to prevent any entity within the system to make copy of this object – make it private.
4. Have a single globally accessible static method to access/retrieve the instance of that class – a public static method

```php
class MySingletonClass
{
    ///Condition 1 - Presence of a static member variable
    private static $_instance = null;

    ///Condition 2 - Locked down the constructor
    private function  __construct() { } //Prevent any oustide instantiation of this class

    ///Condition 3 - Prevent any object or instance of that class to be cloned
    private function  __clone() { } //Prevent any copy of this object

    ///Condition 4 - Have a single globally accessible static method
    public static function getInstance()
    {
        if( !is_object(self::$_instance) )  //or if( is_null(self::$_instance) ) or if( self::$_instance == null )
            self::$_instance = newMySingletonClass();
        return self::$_instance;
    }

    ///Now we are all done, we can now proceed to have any other method logic we want

    //a simple method to echo something
    public function GreetMe()
    {
        echo '<br />Hello, this method is called by using a singleton object..';
    }
}//END Class

///Testing some calls to that class
$obj1 = MySingletonClass::getInstance();
$obj2 = MySingletonClass::getInstance();
$obj3 = MySingletonClass::getInstance();

$obj1->GreetMe();
$obj2->GreetMe();
$obj3->GreetMe();
```

NOTE: In the PHP world, methods __construct() and __clone() forms part of the famous ‘Magic Methods’.

Possible Cons Of Singleton

- While Singleton can sound very sexy, too much of it will definetly pause some performance issues. Use it sparingly. For e.g, Zend 1.x versions suffer a slight performance issue due to so many singletons (as compared to Symfony)
- Might be a problem for Unit Testing; since it tests individual objects
- If not carefully designed/planned, might hinder flexibility within your classes

Singleton Pattern And The Static Context

A Singleton pattern is not the same thing as a static context.
Doru Moisa made a comparison about the performance of Static call versus Singleton call in PHP, you might to check it out.
Based on his observation, it’s observed that Static calls are faster than Singletons, but also for number of calls more than 100 0r 1000, Singleton is surely desirable.