

In software engineering, the singleton pattern is a design pattern used to implement only one instantiation of a class to one object. This pattern might not be used often but it might be very helpful in database classes for example. We do not need to connect to a database every time a script is called if we use a persistent connection.

The singleton concept is simple: A class that can be instantiated only once. To achieve this, we cannot declare the constructor as public, but as private. Now, since a private method cannot be called from outside of the class, we need to create a static public method that has access to the constructor.

The concept is not that hard to understand and neither to implement. Here's a sample code:

```
class Singleton {

    private static $instance = NULL;

    private function __construct() {
        // use my mysql_pconnect()
    }

    public function __destruct() {
        // have some useful stuff here
    }

    public static function getInstance() {
        if(!is_object(self::$instance)) {
            self::$instance = new Singleton();
        }
        return self::$instance;
    }
}
```

As you can see, the constructor is declared private. Therefore, you cannot do something like that:

```
$Singleton = new Singleton();
```

You will get something like:

```
Fatal error: Call to private Singleton::__construct() from invalid context in /home/eric/test.php on line xx
```

In order to make it work, you simply need to do it like that:

```
$Singleton = Singleton::getInstance();
```

