Date
23/03/2017

Reference: devdocs.magento.com/guides/v2.1/ext-best-practices/extension-coding/common-programming-bp.html

Programming Best Practices

Follow a set of coding standards: Following a set of coding standards will help make your code consistent and maintainable.

Magento's Coding Standards are based on the following:

1. Zend Coding standards.
   https://framework.zend.com/manual/1.12/en/coding-standard.html
2. PSR-2
   http://www.php-fig.org/psr/psr-2/
3. PSR-4
   http://www.php-fig.org/psr/psr-4/

$\rightarrow$ a. PHP File Formatting
b. Naming Conventions
c. Coding Style

Date
29/03/2017

Design Patterns in PHP

Singleton Pattern: There are several scenarios in which you might want to make your constructor private. The common reason is that in some cases, you don't want outside code to call your constructor directly, but force it to use another method to get an instance of your class:

or

The purpose of using a private constructor ensures that there can be only one instance of a class and provides a global access point to that instance and this is common with The singleton Pattern.

We only ever want a single instance of our class to exist:

```
class Singleton
{
    private static $instance = null;

    private function __construct()
    {
    }

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

Date
30/03/2017.

3rd Party API integration (use) in our application:
Look : hotelogex/application/models/CurrencyConverter

Payment Gateway integration into our application:
Look : hotelogex/application/models/Gateway

3rd Party library use in our application
To use third-party library in our application, we should use adapter.
Adapter: Adapters are the classes follow the adapter pattern (a software design pattern also known as 'wrapper') and wrap-around (- रैपराउन्ड - चारों तरफ से ढकना/लपेटना/cover करना) classes from third-party libraries. These classes allow you to use functionality from third-party libraries in your code by converting the third-party class interfaces into an interface that is expected by your native code.
When to use Adapter: You should always use adapter classes instead of directly using classes from third-party libraries. This reduces the change impact on your code when the API changes in a third-party library.
We recommend using adapter classes to get access to the functionality provided by third-party classes.
How to write Adapter: A common approach in developing an adapter is to create an interface

Reference: devdocs.magento.com/guides/V2.1/
Date 30/03/2017 [redacted] extension-dev_guide/adapters.html

[redacted]
named 'AdapterInterface' to describe the functionality the third-party class provides. This class is typically found in a directory labeled 'Adapter'. Classes implementing this adapter interface use the third-party class directly to provide indirect functionality.

This approach allows you to update or substitute different implementations provided by other third-party classes without the need to update code that uses your adapter.

Example of Adapters: [redacted]

1. magento2/vendor/magento/framework/Code/Minifier [redacted]
   Here, magento has created 2 adapters. First Minifier/Adapter/Css/CSSmin.php to minify css & second Minifier/Adapter/Js/JShrink.php to [redacted] minify javascript.
2. magento2/vendor/magento/framework/Image
   Here, magento has created 2 adapters. First Adapter/Gd2.php & second Adapter/ImageMagick.php. Both adapter classes provides the concrete implementation using the third-party libraries.
   Gd2: It is a library used to create & manipulate [redacted] image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP & XPM. It is enabled by default in PHP7.
   ImageMagick: It is a native php extension to [redacted] create & modify images using the ImageMagick API.

Date
31/03/2017

Plugin/Module/Component

Plug-in applications are programs that can easily be installed and used as part of your application.

Example:

magento2/vendor/magento/module-customer
magento2/vendor/magento/module-cms
are different magento2 components and you can also create a new one for yourself.
In same way, Drupal has a lot of modules & Joomla has a lot of plugins.

Wrappers or Wrapper Classes

Wrap - रैप - ढकना/लपेटना/कवर करना
Magento has several wrapper classes.
Example: For superglobal variables
Reference: devdocs.magento.com/guides/v2.1/ext-best-practices/extension-coding/security-performance-data-bp.html
Magento recommends to not directly use any PHP superglobals such as:
$GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, $_ENV
Instead use the
Magento\Framework\HTTP\PhpEnvironment\Request
i.e.
magento2/vendor/magento/framework/HTTP/PhpEnvironment/Request.php
wrapper class to safely access these values.