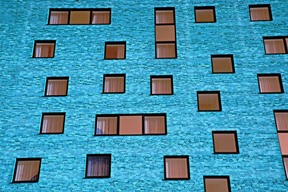
How to simplify a PHP code with the help of the façade pattern?

Published November 26, 2015

We need to consider the use of the **façade pattern** in those cases that the code that we want to use consists of too many classes and methods, and all we want is a simple interface, preferably one method, that can do all the job for us.



The problem: complicated code with too many classes and methods

A real life example can be a code that shares the newest posts in our blog with several social networks. Each social network has its own class, and a set of methods to share our posts.

* A CodeTwit class to tweet on twitter.
* A Googlize class to share our posts on Google plus.
* And a Reddiator class to share in reddit.

That's the code for the three classes:

// Class to tweet on Twitter.

class CodeTwit {

  function tweet($status, $url)

  {

    var\_dump('Tweeted:'.$status.' from:'.$url);

  }

}

// Class to share on Google plus.

class Googlize {

  function share($url)

  {

    var\_dump('Shared on Google plus:'.$url);

  }

}

// Class to share in Reddit.

class Reddiator {

  function reddit($url, $title)

  {

    var\_dump('Reddit! url:'.$url.' title:'.$title);

  }

}

The problem is that every time that we want to share our posts, we need to call to all of the methods. It's too much work for us! We want to simplify the system, and instead of calling to all the methods, call to only one method.

The solution: a Façade class

We can simplify the code by using a **Façade class** with the following characteristics:

1. It holds references to the classes that it uses (in our case, to the CodeTwit, Googlize, and the Reddiator classes).
2. It has a method that calls all of the methods that we need.

*A****façade class****enables us to call only one method instead of calling to many methods. By doing so, it simplifies the work with the system, and allows us to have a simpler and more convenient interface.*

In our example, the shareFacade class gets the social networks objects injected to its constructor, holds these objects by reference, and has the ability to call to all of the share methods from a single share method.

That's the code:

// The Facade class

class shareFacade {

  // Holds a reference to all of the classes.

  protected $twitter;

  protected $google;

  protected $reddit;

  // The objects are injected to the constructor.

  function \_\_construct($twitterObj,$gooleObj,$redditObj)

  {

    $this->twitter = $twitterObj;

    $this->google = $gooleObj;

    $this->reddit = $redditObj;

  }

  // One function makes all the job of calling all the share methods

  // that belong to all the social networks.

  function share($url,$title,$status)

  {

    $this->twitter->tweet($status, $url);

    $this->google->share($url);

    $this->reddit->reddit($url, $title);

  }

}

// Create the objects from the classes.

$twitterObj = new CodeTwit();

$gooleObj = new Googlize();

$redditObj = new Reddiator();

// Pass the objects to the class facade object.

$shareObj = new shareFacade($twitterObj,$gooleObj,$redditObj);

// Call only 1 method to share your post with all the social networks.

$shareObj->share('[https://myBlog.com/post-awsome](https://myblog.com/post-awsome)','My greatest post','Read my greatest post ever.');

And that's all! We got what we wanted. We shared our post with three social networks by calling only a single share method.

In conclusion

We use the **Façade class** in those cases that we need to simplify a complex code that has too many classes and too many functions, and all that we need is a simple interface that allows us to work with only a single class, most often, with a single method.