The template pattern — the power of abstraction

Published November 4, 2015

The **template design pattern** is easily the easiest pattern to learn. In fact, even if you have just a little experience with object oriented PHP, you have probably invented it yourself.



The problem: code duplication

Let's start with the following two classes that can generate an eBook and a paper book.

1 2 3 4 5 6 7 8 91011121314151617181920// Class to generate paperback books.

class Paperback {

  private $title = "";

  private $content = "";

  function setTitle( $str )

  {

    $this->title = $str;

  }

  function setContent( $str )

  {

    $this->content = $str;

  }

  function printBook()

  {

    var\_dump("The book '{$this->title}' was printed.");

  }

}

1 2 3 4 5 6 7 8 91011121314151617181920// Class to generate eBooks.

class Ebook {

  private $title = "";

  private $content = "";

  function setTitle( $str )

  {

    $this->title = $str;

  }

  function setContent( $str )

  {

    $this->content = $str;

  }

  function generatePdf()

  {

    var\_dump("A PDF was generated for the eBook '{$this->title}'.");

  }

}

You are probably concerned with the code duplication since both classes contain the private variables $title and $content as well as the methods that set them, setTitle and setContent.

The solution: abstract class

The **template pattern** suggests that we avoid the code duplication by creating an [abstract class](https://phpenthusiast.com/object-oriented-php-tutorials/abstract-classes-and-methods) that the classes need to extend. Within this [abstract class](https://phpenthusiast.com/object-oriented-php-tutorials/abstract-classes-and-methods) we convene the code instead of duplicating it. Since the subclasses inherit the code of the abstract class they are able to re-use the methods of the [parent class](https://phpenthusiast.com/object-oriented-php-tutorials/inheritance-in-object-oriented-php), instead of each sub class having to duplicate the exact same code.

Let's create an [abstact](https://phpenthusiast.com/object-oriented-php-tutorials/abstract-classes-and-methods) Book class and make it convene the duplicated code of the original classes. Be sure to make the variables [protected](https://phpenthusiast.com/object-oriented-php-tutorials/inheritance-in-object-oriented-php) so they can be accisble from the subclasses.

1 2 3 4 5 6 7 8 910111213141516171819202122232425262728293031// The abstract parent class.

abstract class Book {

**protected** $title;

**protected** $content;

  function setTitle( $str )

  {

    $this->title = $str;

  }

  function setContent( $str )

  {

    $this->content = $str;

  }

}

class Paperback **extends** Book {

  function printBook()

  {

    var\_dump("The book '{$this->title}' was printed.");

  }

}

class Ebook **extends** Book {

  function generatePdf()

  {

    var\_dump("A PDF was generated for the eBook '{$this->title}'.");

  }

}

Let's test the code:

1 2 3 4$paperback = new Paperback;

$paperback -> setTitle("The greatest paperback ever");

$paperback -> printBook();

**And the result is:**

string "The book 'The greatest paperback ever' was printed." (length=51)

In conclusion

As I said, the **template pattern** is the simplest design pattern ever. All we need to do is to convene the duplicated code from the classes into an [abstract](https://phpenthusiast.com/object-oriented-php-tutorials/abstract-classes-and-methods) parent class and make the original classes into subclasses that inherit the code of the parent abstract class.