

---

# **DesignPatternsPHP Documentation**

*Release 1.0*

**Dominik Liebler and contributors**

**Sep 12, 2018**



<b>1</b>	<b>Patterns</b>	<b>3</b>
1.1	Creational	3
1.1.1	Abstract Factory	3
1.1.2	Builder	6
1.1.3	Factory Method	12
1.1.4	Multiton	16
1.1.5	Pool	18
1.1.6	Prototype	21
1.1.7	Simple Factory	24
1.1.8	Singleton	26
1.1.9	Static Factory	28
1.2	Structural	31
1.2.1	Adapter / Wrapper	31
1.2.2	Bridge	37
1.2.3	Composite	40
1.2.4	Data Mapper	43
1.2.5	Decorator	48
1.2.6	Dependency Injection	51
1.2.7	Facade	55
1.2.8	Fluent Interface	58
1.2.9	Flyweight	61
1.2.10	Proxy	64
1.2.11	Registry	68
1.3	Behavioral	70
1.3.1	Chain Of Responsibilities	71
1.3.2	Command	75
1.3.3	Iterator	79
1.3.4	Mediator	84
1.3.5	Memento	88
1.3.6	Null Object	93
1.3.7	Observer	96
1.3.8	Specification	98
1.3.9	State	103
1.3.10	Strategy	107
1.3.11	Template Method	111
1.3.12	Visitor	115

1.4	More . . . . .	119
1.4.1	Service Locator . . . . .	119
1.4.2	Repository . . . . .	123
1.4.3	Entity-Attribute-Value (EAV) . . . . .	131
<b>2</b>	<b>Contribute</b>	<b>137</b>

This is a collection of known [design patterns](#) and some sample code how to implement them in PHP. Every pattern has a small list of examples (most of them from Zend Framework, Symfony2 or Doctrine2 as I'm most familiar with this software).

I think the problem with patterns is that often people do know them but don't know when to apply which.

[Click here to download full PDF material](#)