



Expert

Design Patterns en PHP

Les 23 modèles de conception :
descriptions et solutions illustrées
en **UML2** et **PHP**

Téléchargement
www.editions-eni.fr



Laurent DEBRAUWER
Yannick EVAÏN

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **EIPHDES** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Partie 1 : Introduction

Chapitre 1

Introduction aux patterns de conception

- 1. Design Patterns ou patterns de conception 15
- 2. La description des patterns de conception 17
- 3. Le catalogue des patterns de conception 18
- 4. Comment choisir et utiliser un pattern de conception pour résoudre un problème ? 20
- 5. Organisation du catalogue des patterns de conception 23
 - 5.1 Aspects spécifiques des exemples de code PHP 23

Chapitre 2

Une étude de cas : la vente en ligne de véhicules

- 1. Description du système 27
- 2. Cahier des charges 27
- 3. Prise en compte des patterns de conception 29

Partie 2 : Patterns de construction

Chapitre 3

Introduction aux patterns de construction

1. Présentation	31
2. Les problèmes liés à la création d'objets	32
2.1 Problématique	32
2.2 Les solutions proposées par les patterns de construction	33

Chapitre 4

Le pattern Abstract Factory

1. Description	35
2. Exemple	35
3. Structure	38
3.1 Diagramme de classes	38
3.2 Participants	39
3.3 Collaborations	39
4. Domaines d'utilisation	39
5. Exemple en PHP	40

Chapitre 5

Le pattern Builder

1. Description	51
2. Exemple	51
3. Structure	53
3.1 Diagramme de classes	53
3.2 Participants	53
3.3 Collaborations	54
4. Domaines d'utilisation	55

5. Exemple en PHP 55

Chapitre 6
Le pattern Factory Method

1. Description 63
2. Exemple 63
3. Structure 65
 3.1 Diagramme de classes 65
 3.2 Participants 66
 3.3 Collaborations 66
4. Domaines d'utilisation 66
5. Exemple en PHP 67

Chapitre 7
Le pattern Prototype

1. Description 73
2. Exemple 73
3. Structure 76
 3.1 Diagramme de classes 76
 3.2 Participants 77
 3.3 Collaboration 77
4. Domaines d'utilisation 77
5. Exemple en PHP 78

Chapitre 8
Le pattern Singleton

1. Description 85
2. Exemple 85

4 _____ Design Patterns en PHP

Les 23 modèles de conception

3. Structure	86
3.1 Diagramme de classe	86
3.2 Participant	86
3.3 Collaboration	87
4. Domaine d'utilisation	87
5. Exemples en PHP	87
5.1 La liasse vierge	87
5.2 La classe Vendeur	89

Partie 3 : Patterns de structuration

Chapitre 9

Introduction aux patterns de structuration

1. Présentation	93
2. Composition statique et dynamique	94

Chapitre 10

Le pattern Adapter

1. Description	97
2. Exemple	97
3. Structure	99
3.1 Diagramme de classes	99
3.2 Participants	99
3.3 Collaborations	100
4. Domaines d'application	100
5. Exemple en PHP	101

Chapitre 11
Le pattern Bridge

- 1. Description 107
- 2. Exemple..... 107
- 3. Structure 110
 - 3.1 Diagramme de classes..... 110
 - 3.2 Participants..... 111
 - 3.3 Collaborations..... 111
- 4. Domaines d'application 111
- 5. Exemple en PHP 112

Chapitre 12
Le pattern Composite

- 1. Description 119
- 2. Exemple..... 119
- 3. Structure 122
 - 3.1 Diagramme de classes..... 122
 - 3.2 Participants..... 122
 - 3.3 Collaborations..... 123
- 4. Domaines d'application 124
- 5. Exemple en PHP 125

Chapitre 13
Le pattern Decorator

- 1. Description 129
- 2. Exemple..... 129

6 --- Design Patterns en PHP

Les 23 modèles de conception

3. Structure	134
3.1 Diagramme de classes	134
3.2 Participants	135
3.3 Collaborations	135
4. Domaines d'application	135
5. Exemple en PHP	136

Chapitre 14

Le pattern Facade

1. Description	141
2. Exemple	142
3. Structure	144
3.1 Diagramme de classes	144
3.2 Participants	145
3.3 Collaborations	145
4. Domaines d'application	146
5. Exemple en PHP	147

Chapitre 15

Le pattern Flyweight

1. Description	153
2. Exemple	153
3. Structure	156
3.1 Diagramme de classes	156
3.2 Participants	156
3.3 Collaborations	157
4. Domaine d'application	157
5. Exemple en PHP	157

Chapitre 16
Le pattern Proxy

- 1. Description 163
- 2. Exemple..... 163
- 3. Structure 167
 - 3.1 Diagramme de classes..... 167
 - 3.2 Participants..... 168
 - 3.3 Collaborations..... 168
- 4. Domaines d'application 168
- 5. Exemple en PHP 169

Partie 4 : Patterns de comportement

Chapitre 17
Introduction aux patterns de comportement

- 1. Présentation 173
- 2. Distribution par héritage ou par délégation 174

Chapitre 18
Le pattern Chain of Responsibility

- 1. Description 177
- 2. Exemple..... 177
- 3. Structure 181
 - 3.1 Diagramme de classes..... 181
 - 3.2 Participants..... 181
 - 3.3 Collaborations..... 182
- 4. Domaines d'application 182
- 5. Exemple en PHP 182

8 _____ Design Patterns en PHP

Les 23 modèles de conception

Chapitre 19

Le pattern Command

1. Description	189
2. Exemple	189
3. Structure	193
3.1 Diagramme de classes	193
3.2 Participants	194
3.3 Collaborations	194
4. Domaines d'application	195
5. Exemple en PHP	196

Chapitre 20

Le pattern Interpreter

1. Description	205
2. Exemple	205
3. Structure	208
3.1 Diagramme de classes	208
3.2 Participants	209
3.3 Collaborations	209
4. Domaines d'application	210
5. Exemple en PHP	210

Chapitre 21

Le pattern Iterator

1. Description	219
2. Exemple	219

- 3. Structure 221
 - 3.1 Diagramme de classes..... 221
 - 3.2 Participants..... 222
 - 3.3 Collaborations..... 222
- 4. Domaines d’application 222
- 5. Exemple en PHP 223

Chapitre 22
Le pattern Mediator

- 1. Description 229
- 2. Exemple..... 229
- 3. Structure 233
 - 3.1 Diagramme de classes..... 233
 - 3.2 Participants..... 233
 - 3.3 Collaborations..... 234
- 4. Domaines d’application 234
- 5. Exemple en PHP 234

Chapitre 23
Le pattern Memento

- 1. Description 245
- 2. Exemple..... 245
- 3. Structure 248
 - 3.1 Diagramme de classes..... 248
 - 3.2 Participants..... 248
 - 3.3 Collaborations..... 249
- 4. Domaines d’application 249
- 5. Exemple en PHP 249

Chapitre 24

Le pattern Observer

1. Description	259
2. Exemple	259
3. Structure	262
3.1 Diagramme de classes	262
3.2 Participants	263
3.3 Collaborations	263
4. Domaines d'application	263
5. Exemple en PHP	264

Chapitre 25

Le pattern State

1. Description	269
2. Exemple	269
3. Structure	272
3.1 Diagramme de classes	272
3.2 Participants	272
3.3 Collaborations	273
4. Domaines d'application	273
5. Exemple en PHP	273

Chapitre 26

Le pattern Strategy

1. Description	283
2. Exemple	284

- 3. Structure286
 - 3.1 Diagramme de classes.....286
 - 3.2 Participants.....286
 - 3.3 Collaborations.....287
- 4. Domaines d'application287
- 5. Exemple en PHP288

Chapitre 27
Le pattern Template Method

- 1. Description295
- 2. Exemple.....295
- 3. Structure300
 - 3.1 Diagramme de classes.....300
 - 3.2 Participants.....300
 - 3.3 Collaborations.....300
- 4. Domaines d'application301
- 5. Exemple en PHP301

Chapitre 28
Le pattern Visitor

- 1. Description305
- 2. Exemple.....305
- 3. Structure309
 - 3.1 Diagramme de classes.....309
 - 3.2 Participants.....310
 - 3.3 Collaborations.....310
- 4. Domaines d'application311
- 5. Exemple en PHP311

Partie 5 : Application des patterns

Chapitre 29

Compositions et variations de patterns

1. Préliminaire	319
2. Le pattern Pluggable Factory	320
2.1 Introduction	320
2.2 Structure	325
2.3 Exemple en PHP	326
3. Reflective Visitor	337
3.1 Discussion	337
3.2 Structure	341
3.3 Exemple en PHP	343
4. Le pattern Multicast	353
4.1 Description et exemple	353
4.2 Structure	356
4.3 Exemple en PHP	357
4.4 Discussion : comparaison avec le pattern Observer	366

Chapitre 30

Le pattern composite MVC

1. Introduction au problème	367
2. Le pattern composite MVC	368
3. Exemple en PHP	375
3.1 Introduction	375
3.2 Architecture	377
3.3 Étude du code	378

Chapitre 31

Les patterns dans la conception de logiciels

- 1. Modélisation et conception avec les patterns de conception397
- 2. Autres apports des patterns de conception. 400
 - 2.1 Un référentiel commun 400
 - 2.2 Un ensemble récurrent de techniques de conception. 400
 - 2.3 Un outil pédagogique de l’approche à objets 400

Annexe

Exercices

- 1. Énoncés des exercices 401
 - 1.1 Création de cartes de paiement 401
 - 1.1.1 Création en fonction du client. 401
 - 1.1.2 Création à l’aide d’une fabrique. 402
 - 1.2 Autorisation des cartes de paiement 402
 - 1.3 Système de fichiers 402
 - 1.4 Browser graphique d’objets 403
 - 1.5 États de la vie professionnelle d’une personne 404
 - 1.6 Cache d’un dictionnaire persistant d’objets 404
- 2. Correction des exercices 407
 - 2.1 Création de cartes de paiement 407
 - 2.1.1 Création en fonction du client. 407
 - 2.1.2 Création à l’aide d’une fabrique. 408
 - 2.2 Autorisation des cartes de paiement 408
 - 2.3 Système de fichiers 409
 - 2.4 Browser graphique d’objets 418
 - 2.5 États de la vie professionnelle d’une personne 420
 - 2.6 Cache d’un dictionnaire persistant d’objets 421

- Index 423

Chapitre 4

Le pattern Abstract Factory

1. Description

Le but du pattern `Abstract Factory` est la création d'objets regroupés en familles sans devoir connaître les classes concrètes destinées à la création de ces objets.

2. Exemple

Le système de vente de véhicules gère des véhicules fonctionnant à l'essence et des véhicules fonctionnant à l'électricité. Cette gestion est confiée à l'objet `Catalogue` qui crée de tels objets.

Pour chaque produit, nous disposons d'une classe abstraite, d'une sous-classe concrète décrivant la version du produit fonctionnant à l'essence et d'une sous-classe décrivant la version du produit fonctionnant à l'électricité. Par exemple, à la figure 4.1, pour l'objet scooter, il existe une classe abstraite `Scooter` et deux sous-classes concrètes `ScooterÉlectricité` et `ScooterEssence`.

L'objet `Catalogue` peut utiliser ces sous-classes concrètes pour instancier les produits. Cependant si, par la suite, de nouvelles familles de véhicules doivent être prises en compte par la suite (diesel ou mixte essence-électricité), les modifications à apporter à l'objet `Catalogue` peuvent être assez lourdes.

Le pattern `Abstract Factory` résout ce problème en introduisant une interface `FabriqueVehicule` qui contient la signature des méthodes pour définir chaque produit. Le type de retour de ces méthodes est constitué par l'une des classes abstraites de produit. Ainsi, l'objet `Catalogue` n'a pas besoin de connaître les sous-classes concrètes et reste indépendant des familles de produit.

Une sous-classe d'implantation de `FabriqueVehicule` est introduite pour chaque famille de produit, à savoir les sous-classes `FabriqueVehiculeÉlectricité` et `FabriqueVehiculeEssence`. Une telle sous-classe implante les opérations de création du véhicule appropriée pour la famille à laquelle elle est associée.

L'objet `Catalogue` prend alors pour paramètre une instance répondant à l'interface `FabriqueVehicule`, c'est-à-dire soit une instance de `FabriqueVehiculeÉlectricité`, soit une instance de `FabriqueVehiculeEssence`. Avec une telle instance, le catalogue peut créer et manipuler des véhicules sans devoir connaître les familles de véhicules et les classes concrètes d'instanciation correspondantes.

L'ensemble des classes du pattern Abstract Factory pour cet exemple est détaillé à la figure 4.1.

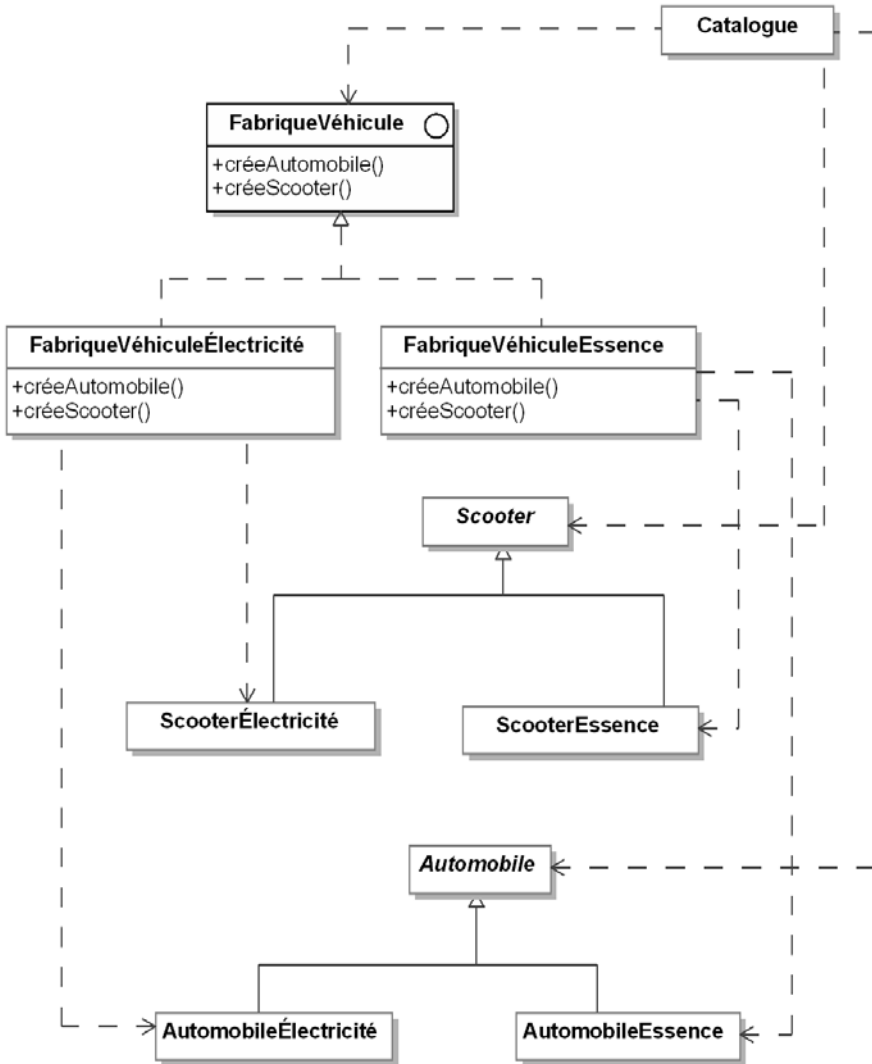


Figure 4.1 - Le pattern Abstract Factory appliqué à des familles de véhicules

3. Structure

3.1 Diagramme de classes

La figure 4.2 détaille la structure générique du pattern.

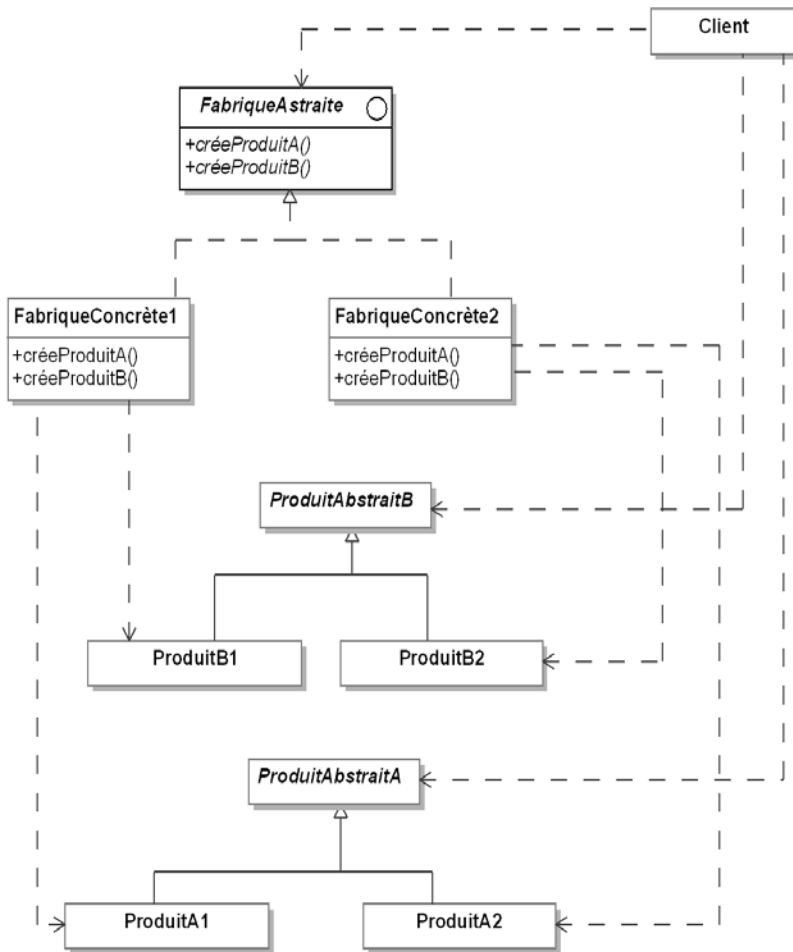


Figure 4.2 - Structure du pattern Abstract Factory

3.2 Participants

Les participants au pattern sont les suivants :

- `FabriqueAbstraite` (`FabriqueVehicule`) est une interface spécifiant les signatures des méthodes créant les différents produits.
- `FabriqueConcret1`, `FabriqueConcret2` (`FabriqueVehiculeElectricite`, `FabriqueVehiculeEssence`) sont les classes concrètes implantant les méthodes créant les produits pour chaque famille de produits. Connaissant la famille et le produit, elles sont capables de créer une instance du produit pour cette famille.
- `ProduitAbstraitA` et `ProduitAbstraitB` (`Scooter` et `Automobile`) sont les classes abstraites des produits indépendamment de leur famille. Les familles sont introduites dans leurs sous-classes concrètes.
- `Client` est la classe qui utilise l'interface de `FabriqueAbstraite`.

3.3 Collaborations

La classe `Client` utilise une instance de l'une des fabriques concrètes pour créer ses produits au travers de l'interface de `FabriqueAbstraite`.

■ Remarque

Normalement, il ne faut créer qu'une seule instance des fabriques concrètes, celle-ci pouvant être partagée par plusieurs clients.

4. Domaines d'utilisation

Le pattern est utilisé dans les domaines suivants :

- Un système utilisant des produits a besoin d'être indépendant de la façon dont ces produits sont créés et regroupés.
- Un système est paramétré par plusieurs familles de produits qui peuvent évoluer.

5. Exemple en PHP

Nous introduisons maintenant un petit exemple d'utilisation du pattern écrit en PHP. Le code PHP correspondant à la classe abstraite `Automobile` et ses sous-classes est donné à la suite. Il est très simple, décrit les quatre attributs des automobiles ainsi que la méthode `afficheCaracteristiques` qui permet de les afficher.

```
<?php
namespace AbstractFactory;

abstract class Automobile
{
    /**
     *
     * @var string
     */
    protected $modele;
    /**
     *
     * @var string
     */
    protected $couleur;
    /**
     *
     * @var int
     */
    protected $puissance;
    /**
     *
     * @var double
     */
    protected $espace;

    /**
     *
     * @param string $modele
     * @param string $couleur
     * @param int $puissance
     * @param double $espace
     */
    public function __construct($modele, $couleur, $puissance,
                               $espace)
    {
```