Design Patterns

Design Patterns
Patrons de conception
Expérience en boîte

Un métier difficile !

- La conception orienté objet est une activité difficile il faut trouver les bons objets les factoriser en classes de la bonne granularité, définir des interfaces et des hiérarchies de classes et surtout établir des relations entre tout cela.
- Écrire des éléments modifiables est difficile, des éléments réutilisables est encore plus difficile.

À l'origine...

- Une conception flexible et réutilisable est rarement obtenue du premier jet.
- Avec l'expérience on tend a produire de la bonne conception, comment capitaliser cette expérience, c'est l'objectif des Design Patterns.

Architecture logicielle

- L'Art de l'architecture logicielle est un art difficile car il ne peut se baser sur des loi physiques (quasi)immuables
- Le principe des Design Patterns
  - Des techniques éprouvées pour résoudre des problèmes récurrents
  - Il ne reste *presque* qu'à brancher...

« Dans un contexte, solution éprouvée à un problème récurrent »

Référence ...

- GoF *Gang of Four*
  - Gamma Helm Johnson Vlisside
  - Design patterns : elements of reusable object-oriented software
  - Design patterns : catalogue de modèles de conception réutilisables
  - 005.12 DES à la bibliothèque Copernic (10ex)

1995 !

23 DPs

Bibliographie
> 250 DPs

capitalisation de l'expérience

- Un Design Pattern décrit une situation fréquente et une réponse éprouvée à cette situation
- Le Design Pattern peut s'utiliser *tel quel* ou être adapté ou combiné avec d'autre pour répondre aux besoins
- Les Design Patterns décrits ici sont ceux fournis par le *Gof* qui sont devenus une référence pour l'ensemble de la communauté des développeurs objet

Design Pattern outil de communication

- Les DPs du GOF permettent de définir un nouveau langage pour l'ensemble de la communauté des développeurs objet
- Pour les concepteurs/développeurs, ce "langage"
  - Établit une terminologie sans ambiguïté
  - Simplifie souvent la conception technique
  - Et permet de se concentrer sur la conception "métier", "fonctionnelle"

DP : description

- Nom !
  - produit un tas de questions intéressantes
- Les problèmes : QUAND l'appliquer
- Solution : description (UML), responsabilités, collaborations entre classes/objets – COMMENT
- Conséquences
  - Résultats
  - Impacts
  - Compromis

3 Types de Design Pattern

- Patrons de création/construction
  - liés au problème de choisir la classe responsable des création, de choisir le type créé
  - permet l'encapsulation des classes effectives
- Patrons structurels
  - liés au problèmes d'organisation des objets dans un logiciel
    - Composition des classes et des objets
- Patrons comportementaux (behavioral)
  - liés au problèmes de communication entre les objets
    - Distribution des responsabilités

Portée (Scope)

- Portée (Scope) : à quel niveau s'applique un DP ? Au niveau des classes ou des objets ?
  - Les patrons au niveau des classes s'occupent des relations entre les classes et les sous-classes. Ces relations sont fixées *statiquement* à la compilation.
    - Exemple: Les patrons de création au niveau des classes délèguent une partie de la création des objets aux sous-classes.
  - Les patrons au niveau des objets s'occupent des relations entre les objets, lesquels sont plus dynamiques et peuvent être modifiées à l'exécution
    - Les patrons de création au niveau des objets délèguent la création à d'autres objets.

Classe / Objet

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method (107)	Adapter (139)	Interpreter (243) Template Method (325)
	Object	Abstract Factory (87) Builder (97) Prototype (117) Singleton (127)	Adapter (139) Bridge (151) Composite (163) Decorator (175) Facade (185) Proxy (207) Flyweight (195)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Observer (293) State (305) Strategy (315) Visitor (331)

Copyright GoF

Purpose	Design Pattern	Aspect(s) That Can Vary
Creational	Abstract Factory (87)	families of product objects
	Builder (97)	how a composite object gets created
	Factory Method (107)	subclass of object that is instantiated
	Prototype (117)	class of object that is instantiated
	Singleton (127)	the sole instance of a class
Structural	Adapter (139)	interface to an object
	Bridge (151)	implementation of an object
	Composite (163)	structure and composition of an object
	Decorator (175)	responsibilities of an object without subclassing
	Facade (185)	interface to a subsystem
	Flyweight (195)	storage costs of objects
	Proxy (207)	how an object is accessed; its location
Behavioral	Chain of Responsibility (223)	object that can fulfill a request
	Command (233)	when and how a request is fulfilled
	Interpreter (243)	grammar and interpretation of a language
	Iterator (257)	how an aggregate's elements are accessed, traversed
	Mediator (273)	how and which objects interact with each other
	Memento (283)	what private information is stored outside an object, and when
	Observer (293)	number of objects that depend on another object; how the dependent objects stay up to date
	State (305)	states of an object
	Strategy (315)	an algorithm
	Template Method (325)	steps of an algorithm
	Visitor (331)	operations that can be applied to object(s) without changing their class(es)

Copyright GoF

Table 1.2: Design aspects that design patterns let you vary

A votre service
A votre service

- Les DP sont là pour VOUS aider
  - Pas d'obligation de les utiliser
  - Ils doivent vous apporter un bénéfice dont vc conscients !
  - Le droit de les "tordre"

- Les DP sont là pour VOUS aider (l'équipe, l'entreprise)
  - Réfléchir ensemble
  - Partager
  - Transmettre

Démarche avec les DPs

- La démarche d'apprentissage des patrons de conception :
  - comprendre leur utilité
  - accepter leur utilité
  - lire et relire les patrons
  - les intérioriser pour les utiliser
  - LES UTILISER ! *PRESQUE* PARTOUT !
    - Attention risque « Overdesign » !