
Design Patterns
overview

What is a Design Pattern?(1)

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

Christopher Alexander

What is a Design Pattern?(2)

- Have you had a design déjà-vu-that?
  - (feeling that you've solved a problem before but not knowing exactly where or how)
- Design Patterns (DP) are recorded experience in designing object-oriented software.
- Each **DP** systematically **names, explains, and evaluates an important and recurring design** in object-oriented systems.

Essential elements(1)

- The **pattern name**
  - used to describe a design problem,
  - its solutions, and consequences
- The **problem**
  - describes when to apply the pattern.
  - It explains the problem and its context.

Essential elements(2)

- The **solution**
  - describes the elements that make up the design, their relationships, responsibilities, and collaborations.
  - provides an abstract description independent from a particular concrete design or implementation
- The **consequences**
  - the results and trade-offs of applying the pattern
  - often concern space and time trade-offs
  - may address language and implementation
  - include its impact on a system's flexibility, extensibility, or portability.

Describing Design Patterns(1)

- **Pattern Name and Classification**
- **Intent**
  - What does the design pattern do?
  - What is its rationale and intent?
  - What particular design issue or problem does it address?
- **Also Known As**
- **Motivation**
  - A scenario that illustrates a design problem and the proposed solution.

Describing Design Patterns(2)

- **Applicability**
- **Structure**
  - A graphical representation of the involved classes
- **Participants**
  - The classes participating in the design pattern and their responsibilities.
- **Collaborations**
  - How the participants collaborate to carry out their responsibilities.

Describing Design Patterns(3)

- **Consequences**
  - How does the pattern support its objectives?
  - What are the trade-offs and results of using it?
  - What aspect of system structure does it let you vary independently?
- **Implementation**
  - What pitfalls, hints, or techniques should you be aware of when implementing the pattern?
  - Are there language-specific issues?
- **Sample Code, Known Uses, and Related Patterns**

Pattern Classification

- **Purpose**
  - Creational
    - about object creation
  - Structural
    - about composition of classes or objects
  - Behavioral
    - about interaction and distribution of responsibility
- **Scope** (the pattern applies primarily to)
  - Class or Object

		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)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Flyweight (195) Observer (293) State (305) Strategy (315) Visitor (331)

Table 1.1: Design pattern space

From: Gang of Four - Design Patterns, Elements of reusable Object Oriented Software

How to Select a DP

- Consider how design patterns solve design problems.
- Scan Intent sections.
- Study how patterns interrelate.
- Study patterns of like purpose.
- Examine a cause of redesign.
- Consider what should be variable in your design.

The most common patterns

- Abstract Factory
- Adapter
- Composite
- Decorator
- Factory Method
- Observer
- Strategy
- Template Method