

Home › Design Patterns › Creational patterns

Builder Design Pattern

Intent

- 01 Separate the construction of a complex object from its representation so that the same construction process can create different representations.
- Parse a complex representation, create one of several targets.

Problem

- 02 An application needs to create the elements of a complex aggregate. The specification for the aggregate exists on secondary storage and one of many representations needs to be built in primary storage.

Discussion

- 03 Separate the algorithm for interpreting (i.e. reading and parsing) a stored persistence mechanism (e.g. RTF files) from the algorithm for building and representing one of many target products (e.g. ASCII, TeX, text widget). The focus/distinction is on creating complex aggregates.

[+] Feedback

BOOKMARK

Search

×

contents

Design Patterns

§ Creational patterns

- Creational patterns
- Abstract Factory Design Pattern
- Builder Design Pattern
- Factory Method Design Pattern
- Object Pool Design Pattern
- Prototype Design Pattern
- Singleton Design Pattern

§ Structural patterns

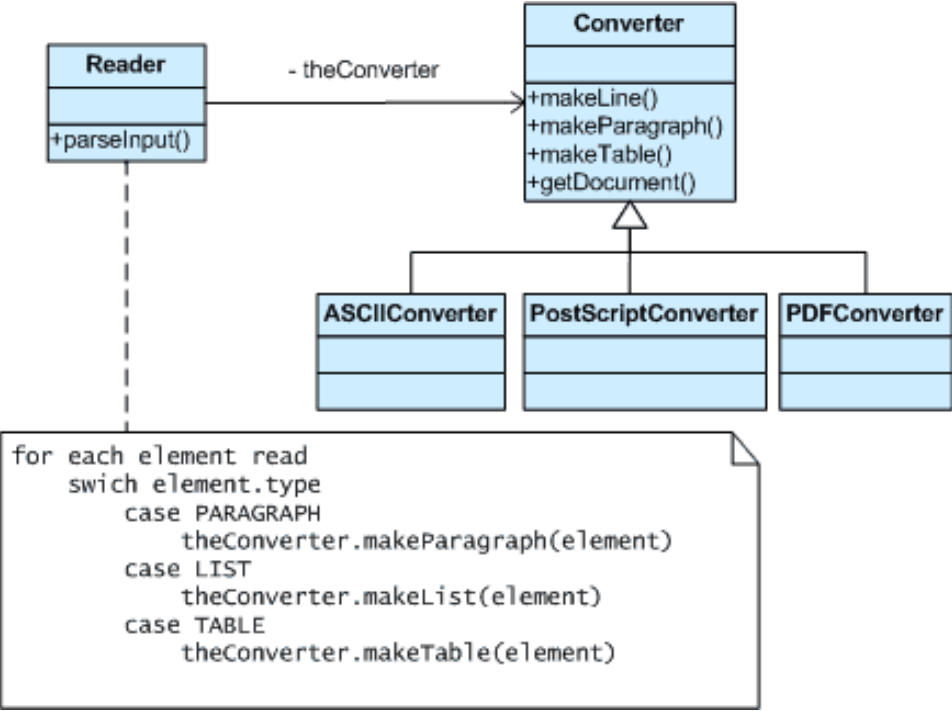
§ Behavioral patterns

04 The “director” invokes “builder” services as it interprets the external format. The “builder” creates part of the complex object each time it is called and maintains all intermediate state. When the product is finished, the client retrieves the result from the “builder”.

05 Affords finer control over the construction process. Unlike creational patterns that construct products in one shot, the Builder pattern constructs the product step by step under the control of the “director”.

Structure

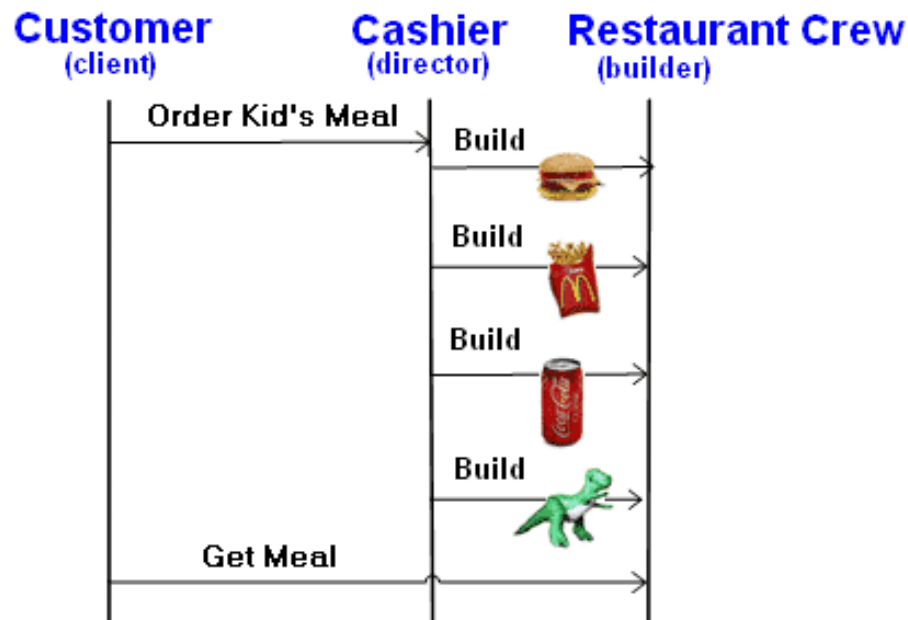
06 The Reader encapsulates the parsing of the common input. The Builder hierarchy makes possible the polymorphic creation of many peculiar representations or targets.



Example

07

The Builder pattern separates the construction of a complex object from its representation so that the same construction process can create different representations. This pattern is used by fast food restaurants to construct children's meals. Children's meals typically consist of a main item, a side item, a drink, and a toy (e.g., a hamburger, fries, Coke, and toy dinosaur). Note that there can be variation in the content of the children's meal, but the construction process is the same. Whether a customer orders a hamburger, cheeseburger, or chicken, the process is the same. The employee at the counter directs the crew to assemble a main item, side item, and toy. These items are then placed in a bag. The drink is placed in a cup and remains outside of the bag. This same process is used at competing restaurants.



Check list

08

1. Decide if a common input and many possible representations (or outputs) is the problem at hand.
2. Encapsulate the parsing of the common input in a Reader class.
3. Design a standard protocol for creating all possible output representations. Capture the steps of this protocol in a Builder interface.
4. Define a Builder derived class for each target representation.
5. The client creates a Reader object and a Builder object, and registers the latter with the former.
6. The client asks the Reader to “construct”.
7. The client asks the Builder to return the result.

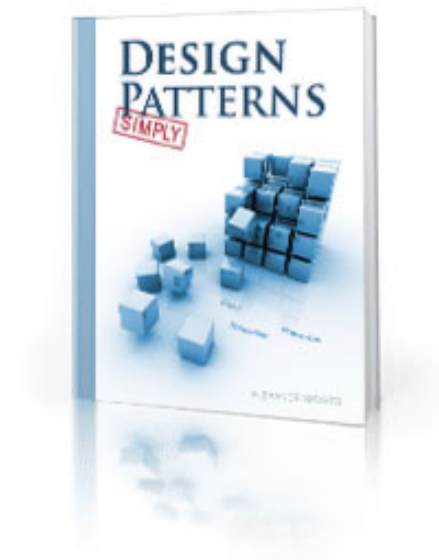
Rules of thumb

09

- Sometimes creational patterns are complementary: Builder can use one of the other patterns to implement which components get built. Abstract Factory, Builder, and Prototype can use Singleton in their implementations.
- Builder focuses on constructing a complex object step by step. Abstract Factory emphasizes a family of product objects (either simple or complex). Builder returns the product as a final step, but as far as the Abstract Factory is concerned, the product gets returned immediately.
- Builder often builds a Composite.
- Often, designs start out using Factory Method (less complicated, more customizable, subclasses proliferate) and evolve toward Abstract Factory, Prototype, or Builder (more flexible, more complex) as the designer discovers

where more flexibility is needed.

Read next



This article is taken from the book **Design Patterns - Simply**.

ALL design patterns are compiled there. The book is written in a clear and simple language that makes it easy to read and understand (just like this article).

It is a part of our Design Patterns Course. We distribute it in PDF format, so it will be available for downloading in 10 seconds!

[Learn more...](#)

10

Builder code examples

C# examples

- [Builder in C#](#)

C++ examples

- [Builder in C++](#)

Delphi examples

- [Builder in Delphi](#)

Java examples

- [Builder in Java: Before and after](#)
- [Builder in Java](#)

PHP examples

- [Builder in PHP](#)

◀ Abstract Factory Design Pattern

Factory Method Design Pattern ▶



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 3.0 Unported License](#)