CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2002
Doc 7 Builder

Contents

Builder .................................................................................................................. 2
Applicability ........................................................................................................... 2
Example – XML Parser ........................................................................................ 4
Consequences ........................................................................................................ 7

References

Design Patterns: Elements of Reusable Object-Oriented Software,
Gamma, Helm, Johnson, Vlissides, Addison-Wesley, 1995, pp. 97-106

The Design Patterns Smalltalk Companion, Alpert, Brown, Woolf, 1998, pp. 47-62

Copyright ©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San
Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license
defines the copyright on this document.
Builder Intent

Separate the construction of a complex object from its representation so that the same construction process can create different representations

Applicability

Use the Builder pattern when

• The algorithm for creating a complex object should be independent of the parts that make up the object and how they're assembled

• The construction process must allow different representations for the object that's constructed
Collaborations

The client creates the Director object and configures it with the desired Builder object.

Director notifies the builder whenever a part of the product should be built.

Builder handles requests from the director and adds parts to the product.

The client retrieves the product from the builder.

```
(aClient) \[ new ConcreteBuilder \]
\[ new Director( aBuilder ) \]
\[ Add(aLineShape) \]
\[ BuildPartA() \]
\[ BuildPartB() \]
\[ BuildPartC() \]
\[ GetResult \]
(aDirector) \[ aConcreteBuilder \]
```
Example – XML Parser

Director
  XML Parser

Abstract Builder Class
  XML.SAXDriver (Smalltalk)
  org.xml.sax.helpers.DefaultHandler (Java)
  DefaultHandler (C++)

Concrete Builder Class
  Your subclass of the abstract builder

Client
  Your code that uses the tree built
Java Example

public static void main(String argv[])
{
    SAXDriverExample handler = new SAXDriverExample();

    // Use the default (non-validating) parser
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try
    {
        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse( new File("sample"), handler );
    }
    catch (Throwable t)
    {
        t.printStackTrace();
    }
    System.out.println( handler.root());
}
Smalltalk Example

| builder exampleDispatcher |

builder := SAXDriverExample new.
exampleDispatcher := SAXDispatcher new contentHandler: builder.
XMLParser
  processDocumentInFilename: 'page'
  beforeScanDo:
    [:parser |
    parser
      saxDriver:(exampleDispatcher);
      validate: true].
built root.
Consequences

- It lets you vary a product's internal representation
- It isolates code for construction and representation
- It gives you finer control over the construction process

Implementation

- Assembly and construction interface
  Builder may have to pass parts back to director, who will then pass them back to builder
- Why no abstract classes for products
- Empty methods as default in Builder