Open
TechSchool

Design Patterns: an Introduction

BENOÎT BLEUZÉ   HAIKO SCHOL

27/08/2014

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Introduction

Design Patterns, what is all the fuss about?

- ▶ Mantra uttered over and over by old bearded gurus
- ▶ Scary diagrams
- ▶ Abstract, mysterious names
- ▶ universal magical answer to the spaghetti code I end up having at the end of my project

Open
TechSchool

Introduction

Design Patterns, what is all the fuss about?

- ► Mantra uttered over and over by old bearded gurus
- ► Scary diagrams
- ► Abstract, mysterious names
- ► universal magical answer to the spaghetti code I end up having at the end of my project

<div align="center">Let's demystify. . .</div>

Outline

Engineering techniques

Categories

Concrete examples
    MVC Pattern
    Creational Patterns
    Structural Patterns
    Behavioral Patterns

WordS of Caution

Conclusions

Open
TechSchool

History

▶ **Design Patterns**: invented by software Architect, *Christopher Alexander*.

▶ Design Patterns: Elements of Reusable Object-Oriented Software 1994,
authored by the **Gang Of Four** (*Erich Gamma*, *Richard Helm*, *Ralph Johnson and John Vlissides*) popularised the concept in Computer science.

▶ Since then, countless books on how to apply many more patterns to many specific languages and problems.

Open
TechSchool

History

▶ **Design Patterns**: invented by ~~software~~ Architect, *Christopher Alexander*.

▶ Design Patterns: Elements of Reusable Object-Oriented Software 1994,
authored by the **Gang Of Four** (*Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides*) popularised the concept in Computer science.

▶ Since then, countless books on how to apply many more patterns to many specific languages and problems.

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

History

▶ **Design Patterns**: invented by ~~software~~ Architect, *Christopher Alexander*.

▶ **Design Patterns: Elements of Reusable Object-Oriented Software** 1994,
authored by the **Gang Of Four** (*Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides*) popularised the concept in Computer science.

▶ Since then, countless books on how to apply many more patterns to many specific languages and problems.

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

History

- **Design Patterns**: invented by ~~software~~ Architect, *Christopher Alexander*.

- **Design Patterns: Elements of Reusable Object-Oriented Software** 1994,
  authored by the **Gang Of Four** (*Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides*) popularised the concept in Computer science.

- Since then, countless books on how to apply many more patterns to many specific languages and problems.

Open
TechSchool

Reusable techniques solving recurring problems

They are **not**:

- ▶ tools (tools are editors or IDEs)
- ▶ ready made one size fits all solutions

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Reusable techniques solving recurring problems

A Pattern is only complete if it comes with:

- ▶ a typical situation along with why it causes trouble
- ▶ a solution and the reason it is a good one.
- ▶ a context, when to apply it.
- ▶ limitations

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

GoF Object Oriented Categories

In the book **Design Patterns**, the are the following categories:

Creational create resources instead of direct instantiation

Structural organise the structure of the data: hierarchy, composition

Behavioral drive the data flow, communication between objects

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

GoF Creational Patterns

Abstract Factory groups object factories that have a common
theme.

Builder constructs complex objects by separating construction and
representation.

Factory Method creates objects without specifying the exact class
to create.

Prototype creates objects by cloning an existing object.

Singleton restricts object creation for a class to only one
instance.

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

GoF Structural Patterns

Adapter allows classes with incompatible interfaces to work together by wrapping its own interface around that of an already existing class.

Bridge decouples an abstraction from its implementation so that the two can vary independently.

Composite composes zero-or-more similar objects so that they can be manipulated as one object.

Decorator dynamically adds/overrides behaviour in an existing method of an object.

Facade provides a simplified interface to a large body of code.

Flyweight reduces the cost of creating and manipulating a large number of similar objects.

Proxy provides a placeholder for another object to control access, reduce cost, and reduce complexity.

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

GoF Behavioral Patterns

Chain of responsibility delegates commands to a chain of processing objects.

Command creates objects which encapsulate actions and parameters.

Interpreter implements a specialized language.

Iterator accesses the elements of an object sequentially without exposing its underlying representation.

Mediator allows loose coupling between classes by being the only class that has detailed knowledge of their methods.

Memento provides the ability to restore an object to its previous state (undo).

Observer is a publish/subscribe pattern which allows a number of observer objects to see an event.

State allows an object to alter its behavior when its internal state changes.

Strategy allows one of a family of algorithms to be selected on-the-fly at runtime.

Template method defines the skeleton of an algorithm as an abstract class, allowing its subclasses to provide concrete behavior.

Visitor separates an algorithm from an object structure by moving the hierarchy of methods into one object.

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
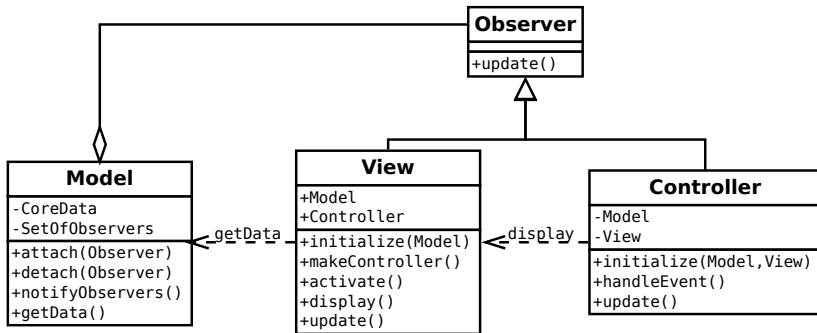Conclusions

Other known categories

Architectural Large application scale patterns: core/plugin, data stores. . .

Concurrency concurrent programming techniques, including locking mechanisms, thread pooling. . .

Testing code testing patterns: mock objects, set-up/tear-down. . .

Domain specific Some patterns are very much domain specific

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

MVC: Principle

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Factory Method Pattern: Principle

Open TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
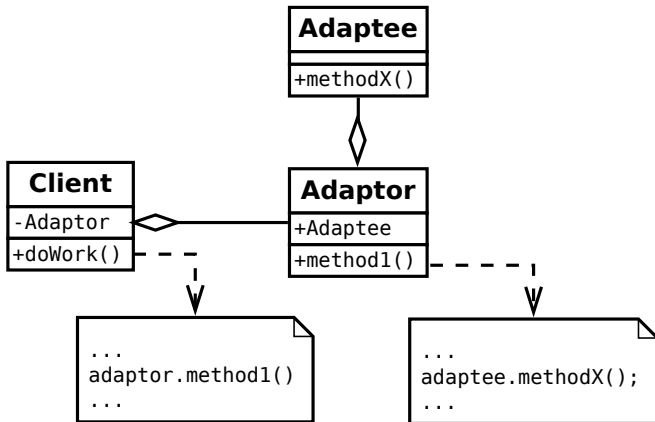Structural Patterns
Behavioral Patterns
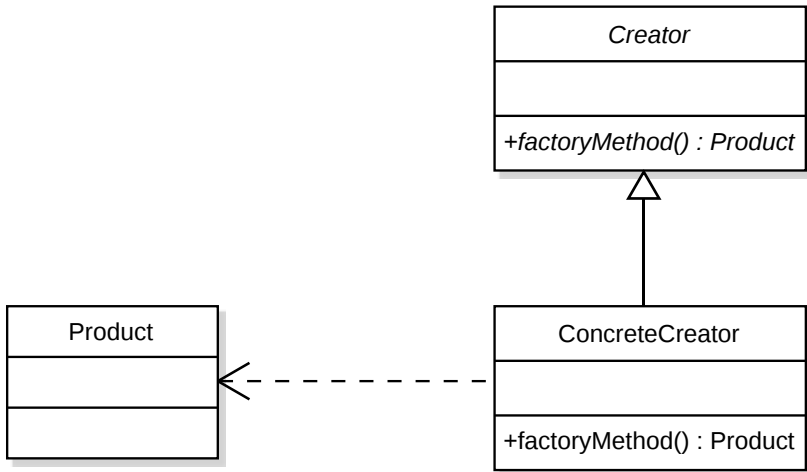
Factory Method Pattern: Code

```python
root = etree.Element('body')
h1 = etree.Element('h1')
h1.text = 'The Title'
root.append(h1)
p = etree.Element('p')
p.text = 'Always write Python'
root.append(p)

from lmxl.builder import E

doc = E('body',
        E('h1', 'The Title'),
        E('p', 'Always write Python'))
```
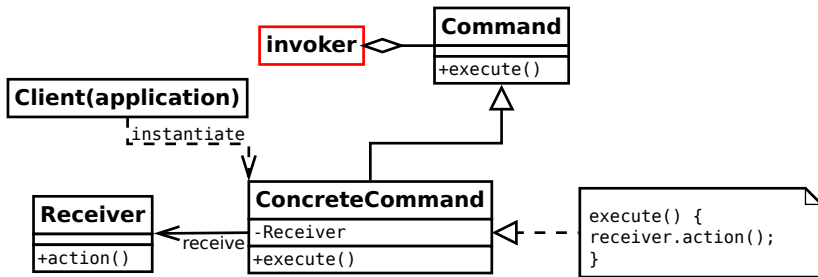
Adapter Pattern: Principle

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Adapter Pattern: Code

Python has the concept of a "file-like" object, which has read()
and write(). Examples are file and StringIO, but **not** socket
(it has send() and recv() instead). Luckily, we can get an
Adapter from socket with makefile():

```python
import socket

s = socket.socket(socket.AF_INET)
s.connect(...)
s.send('hello there')
file_like_socket = s.makefile()
file_like_socket.write('hi again')
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Command Pattern: Principle

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Command Pattern: Code

```python
def paint_line(x1, y1, x2, y2): pass

class PaintLineCommand(object):
  def __init__(self, x1, y1, x2, y2):
      self.x1, self.y1 = x1, y1
      self.x2, self.y2 = x2, y2

  def __call__(self):
      paint_line(self.x1, self.y1, self.x2, self.y2)

  def __str__(self):
      return 'paint_line from {}/{} to {}/{}'.format(
          self.x1, self.y1, self.x2, self.y2)

cmd = PaintLineCommand(5, 23, 42, 123)
cmd()
logging.debug(cmd)
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Command Pattern: Code

```python
class PaintLineCommand(object):
    # ...

    def undo(self):
        erase_line(self.x1, self.y1, self.x2, self.y2)


cmd = PaintLineCommand(5, 23, 42, 123)
cmd()
command_history.append(cmd)
command_history[-1].undo()
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

MVC Pattern
Creational Patterns
Structural Patterns
Behavioral Patterns

Python Perspective

Patterns don't have to be about object-oriented design. They can be something very simple.

```python
if 'page' in params:
    page = params['page']
else:
    page = 1
```

Can be written as

```python
page = params.get('page', 1)
```

Open
TechSchool

Limitations

- ▶ Language specific patterns
- ▶ Trends
- ▶ Technical advances, obsolescence

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: Principle

- One instance shared all over the code: "global object"
- Examples: config managers, Factories, Facade objects

Open
TechSchool

Singleton Pattern: Python

There are many ways to implement it in Python.

- ▶ a module instead of a class
- ▶ a class method instance() (does not enforce single instance)
- ▶ implement __new__()
- ▶ metaclass
- ▶ ...
- ▶ use the Monostate Pattern instead

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: IOLoop in Tornado

```python
@staticmethod
def instance():
    """Returns a global 'IOLoop' instance.
        ...
    """
    if not hasattr(IOLoop, "_instance"):
        with IOLoop._instance_lock:
            if not hasattr(IOLoop, "_instance"):
                # New instance after double check
                IOLoop._instance = IOLoop()
    return IOLoop._instance
```

Q: But is it thread-safe?

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: Metaclass

```python
class Singleton(type):
    def __init__(cls, name, bases, attrs):
        super(Singleton, cls).__init__(
            cls, bases, attrs)

        cls._instance = None

    def __call__(cls, *a, **k):
        if cls._instance is None:
            cls._instance = super(
                Singleton, cls).__call__(*a, **k)

        return cls._instance
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: Using the metaclass

```python
class EventCounter(object):
    __metaclass__ = Singleton

    def __init__(self):
        self.count = 0

    def register_event(self):
        self.count += 1
```

```python
>>> a, b = EventCounter(), EventCounter()
>>> b.count
0
>>> a.register_event()
>>> b.count
1
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: Alternative "Monostate"

```python
class Borg(object):
    _shared_state = {}

    def __new__(cls, *a, **k):
        obj = super(Borg, cls).__new__(cls, *a, **k)
        obj.__dict__ = cls._shared_state
        return obj


class SevenOfNine(Borg): pass
```

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Singleton Pattern: Controversies

- ▶ very much used, BUT
- ▶ introducing global state never encouraged
- ▶ tricky thread safety at initialisation (ultimately flawed in c++ due to uncertainty of dynamic loading order)
- ▶ overused: situations REALLY needing a singleton are rare

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Conclusions

There is so much more we can show...

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Links

- Many Python examples from here:
  https://www.youtube.com/watch?v=Er5K_nR5lDQ
- Good talk by Alex Martelli at EuroPython:
  https://www.youtube.com/watch?v=bPJKYrZjq10

Open
TechSchool

Engineering techniques
Categories
Concrete examples
WordS of Caution
Conclusions

Questions?