Refactoring to Design Patterns

PHP Unconference 2013

Qafoo GmbH

September 23, 2013

About Qafoo

Helping people to create high quality web applications.

`http://qafoo.com`

`@qafoo`

- Trainings
- Consulting
- Tools

Part I

Refactoring

Refactoring

- Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.
- Its heart is a series of small behavior preserving transformations.
- Each transformation (called a 'refactoring') does little, but a sequence of transformations can produce a significant restructuring.
- Since each refactoring is small, it's less likely to go wrong.

Part II

Design Patterns

Patterns are . . .

. . . names for proven ideas how a certain class of problems can be solved.

Patterns are **not** . . .

- . . . appliable to every problem.
- . . . directly transferable to code.
- . . . written in stone.
- . . . always the best solution.

Refactoring and Patterns

- Design Patterns are often target of a refactoring
- Refactoring to patterns to
  - . . . reduce complexity
  - . . . reduce duplication
  - . . . increase readability/comprehension
  - . . . reach SRP and DIP
- Helpful Refactorings
  - Extract Method
  - Extract Class
  - Move Method

Factory

A factory creates an object for you.

- Actually 4 patterns
  - Factory
  - Factory Method
  - Abstract Factory
  - Builder

Adapter

The Adapter converts between different APIs

- Integrate 3rd party code (libraries)
- Seamless integration into existing interfaces
- Avoid hard dependencies on 3rd party API
- Make 3rd party library replacable
- Strongly Related to the Bridge pattern
- Extremely important patterns for decoupling

Strategy/Policy Pattern

Strategy allows to exchange algorithms at run time.

- Object-oriented `switch` statement
- When calculations are changing frequently
- Or when they change based on state
- Construction of strategies often combined with a factory

THANK YOU

Rent a quality expert
qafoo.com