Decorator

Dmitri Nesteruk
QUANTITATIVE ANALYST

@dnesteruk http://activemesa.com

Overview

Motivation

Function Decorator

Wrapping Decorator

Mixin Inheritance

Usability Improvements

Motivation

Want to augment existing functionality

Do not want to rewrite or alter existing code (Open-Closed Principle)

Want to keep new functionality separate (Single Responsibility Principle)

Need to be able to interact with existing structures

Decorator

Allows for adding behavior to individual objects without affecting the behavior of other objects of the same class.

Summary

Functional decorators let you wrap functions with before/after code (for, e.g., logging)

An aggregate decorator does not give you the underlying object's features, but can be composed at runtime

A decorator based on mixin inheritance is more flexible, exposes underlying object's features, but is only constructible at compile time