Releasing your potential

Logica

Varianten des “Observer Pattern”

Das Observer Pattern ist ein GoF Entwurfmuster um Änderungen an einem Objekt in abhängigen Ansichten zu aktualisieren. Das Pattern bietet eine Lösung zur Nachrichtenverteilung innerhalb einer Applikation für eine beliebige Anzahl an Empfängern.

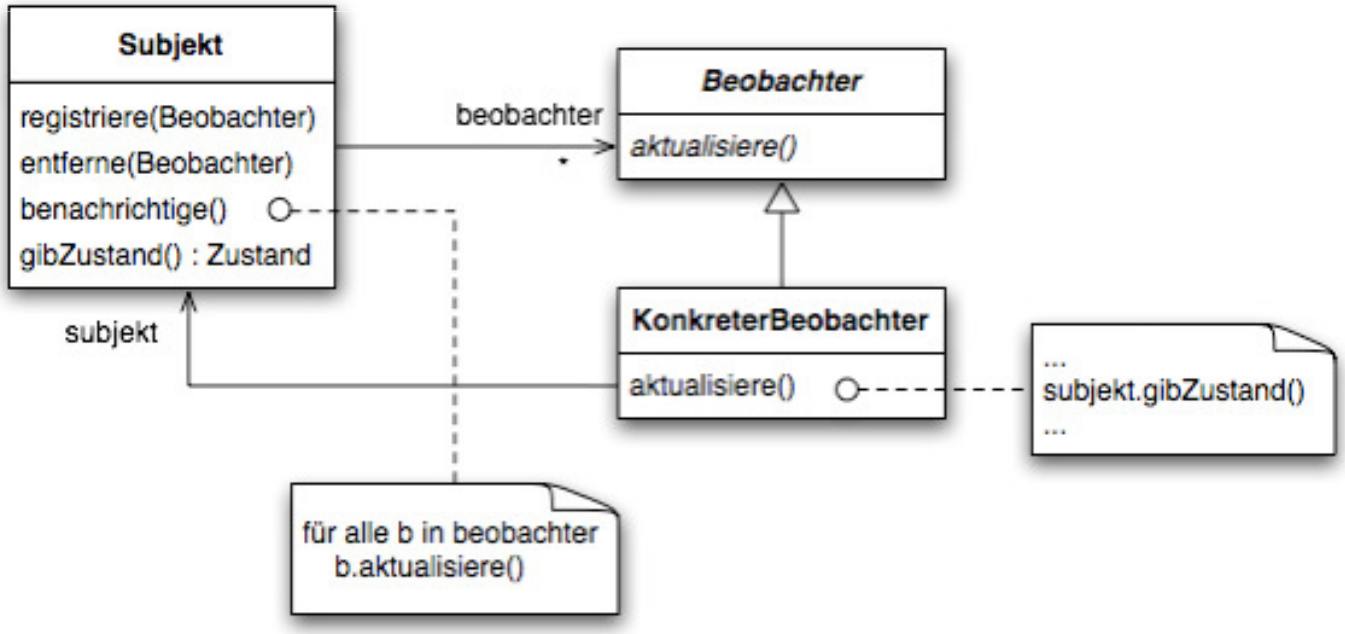
Dieser Vortrag stellt verschiedene Alternativen einer Observer Pattern Implementierung vor und zeigt die Vor- und Nachteile auf. Am Ende steht eine Observer-Komponente für eine Windows Forms Applikationen.

Agenda

- Observer Pattern nach GoF
  - Einleitung
  - Demo einer Implementierung
  - Vorteile / Nachteile
- Event-Basierter Observer
  - Einleitung
  - Demo einer Implementierung
  - Vorteile / Nachteile
- Observer-Komponente
  - Einleitung
  - Demo einer Implementierung
  - Vorteile / Nachteile

Observer Pattern

- Subjekt = Beobachtetes Objekt
- Konkreter Beobachter = Konkrete Implementierung eines Beobachters
- Beobachter = Schnittstelle für alle Beobachter Implementierungen
- Beobachter.aktualisiere = Aktualisieren der konkreten Beobachter
- Subjekt.benachrichtige = Aufrufen der Aktualisierung aller registrierten Beobachter



Observer Pattern (Demo)

- Demo
  - Fachobjekt „PersonSubjekt“
  - Schnittstelle „IBeobachter“
  - Beobachter „DatumBeobachter“
  - Beobachter „EigenschaftenBeobachter“

Observer Pattern (Vorteile / Nachteile)

- Pro: Es funktioniert ☺
- Pro: Saubere Architektur
- Contra: Erweiterbarkeit: Viele Schnittstellen oder Methoden
  - Attach / Detach für jede Schnittstelle im Subjekt
  - Leerimplementierung für nicht benutzte Methoden
- Contra: Verwaltungscode im Fachobjekt

Event Basierter Observer

- Subjekt mit Events
- Attach / Detach über Event und EventHandler
- EventHandler mit EventArgs als Schnittstelle

Event Basierter Observer (Demo)

- Demo
  - Fachobjekt „Subjekt“ + Schnittstelle „ISubjekt“
  - EventArgs „SubjektEventArgs“ (statt „IBeobachter“)

  - Beobachter „DatumBeobachter“
  - Beobachter „EigenschaftenBeobachter“

Event Basierter Observer (Vorteile / Nachteile)

- Pro: Es funktioniert ☺
- Pro: Einfache Erweiterbarkeit durch Events
- Pro/Contra: Kein Verwaltungscode im Fachobjekt
- Contra: Sehr viele EventHandler in großen Objektbäumen (Vater-Kind)
- Contra: Keine Visual Studio Unterstützung
- Contra: Auswahl eines Objektes nicht unterstützt

Observer Komponente

- Anforderung
  - Event Basierter Observer
  - Windows Forms Komponente

- Lösung
  - Observer Komponente, Subjekt als Assoziation
  - Statische Liste mit allen Instanzen der Observer Komponente
  - SubjektAktualisiert Event
  - Statische RaiseSubjektAktualisiert Methode (Auslösen des Events aller Instanzen)
  - Subjekt als Parameter der RaiseSubjektAktualisiert Methode

Observer Komponente (Demo)

- Observer Komponente „SubjektObserver“
  - Singleton Liste mit allen Instanzen
    - New() erweitern
    - Dispose() erweitern
  - Event „SubjektAktualisiert“
  - Statische Methode „RaiseSubjektAktualisiert“

  - Fachobjekt „Subjekt“ + Schnittstelle „ISubjekt“
  - EventArgs „SubjektEventArgs“

  - Beobachter „DatumBeobachter“
  - Beobachter „EigenschaftenBeobachter“

Observer Komponente (Vorteile / Nachteile)

- Pro: Es funktioniert ☺
- Pro: Visual Studio Unterstützung
- Contra: Fachobjekt ist abhängig von Controller
  - Pro: *Beobachter rufen die Raise* Methoden auf (Validierung etc.)
- Pro/Contra: Immer genau ein Subjekt im Fokus

Fragen

Es gibt keine dummen Fragen,

es gibt nur dumme Antworten!

E-Mail/MSN: thomas.mentzel@logica.com

Blog: http://thomas.mentzel.name

Twitter: http://twitter.com/ThomasMentzel