Clean Code & Design Pattern

Tag 2

Zeitplan

12:00

Start

Pause

13:15 & 14:45

16:00

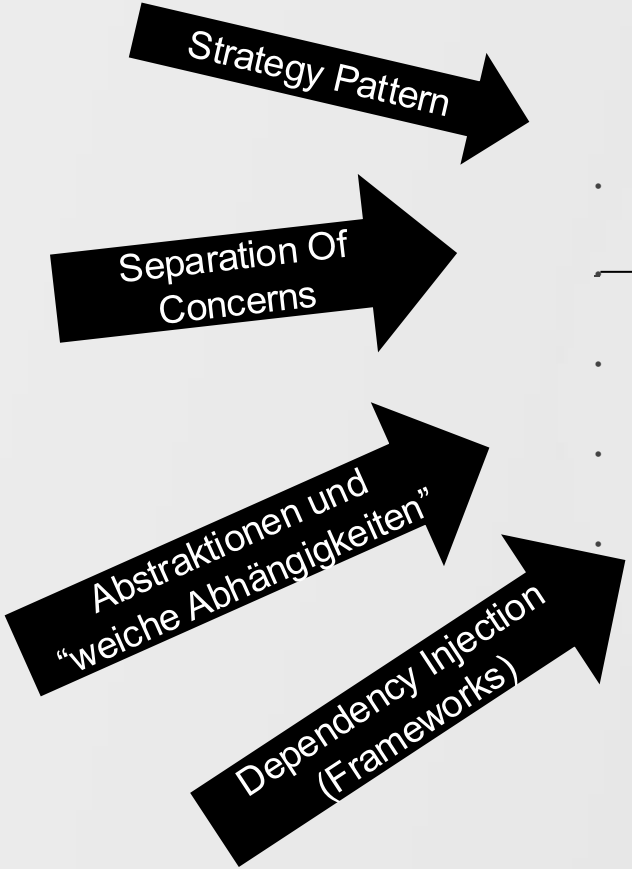
Ende

Review Tag 1

Design Pattern ∞ Clean Code Principles

- Design Pattern
    a. Problem → Lösung
    b. Kommunikation
- Clean Code Principles
    a. Strukturierung von Code
    b. Qualität von Code
    c. Dokumentation

SOLID Principles



- SRP (Single Responsibility Principle
- ~~OCP (Open Closed Principle)~~
- LSP (Liskov Substitution Principle)
- ISP (Interface Seggregation Principle)
- DIP (Dependency Inversion Principle)

Side Quest “Factory”

- Person ist “nur”eine Person
    a. PS: Eine IPerson hat auch eine IUniqueIdentity
- Ein “Converter” kann nur string <--> Person umwandeln
- Ein “Persister”kann nur string <--> Dateisystem
- Single Responsibility Principle mit Ravioli Code
    a. PS: Quick and Dirty now, clean later.
- Eine “Factory” in der Mitte verbindet diese Aspekte
    a. PS: Prinzip: Single Level of Abstraction

Preface

Architectural Decision Records

ADR - Architectural Decision Records

- Architekturentscheidungen bewusst machen
    a. Technologie
    b. Frameworks
- Pro und Contra abwägen
- Architekturentscheidungen können geändert werden,
- Entscheidungsänderungen werden auch Dokumentiert

MADR - Markdown Architectural Decision Records

- Markdown Template für Entscheidungen
- Architekturentscheidungen liegen nahe am Code
- Alter Code → Entscheidungen zu diesem Zeitpunkt
- Webseite, Docker Container, … bei Bedarf. Sogar im TFS/Azure DevOps

- https://adr.github.io/madr/

01

Command Pattern

Command Pattern

- Kommando Entwurfsmuster
- Verwendung z.B. Buttons einer UI

- Prüfen ob ein Kommando ausgeführt werden kann (CanExecute)
- Ausführen des Kommandos(Execute)
- Beide Methoden bekommen den gleichen Parameter “Context”

Aufgabe

Implementieren des Command Pattern

Projekt „PdfTools“

ICommand Interface
CanExecute(ctx)
Execute(ctx)

02
“Simple” Composite Pattern

“Simple” Composite Pattern

- Aggregieren von mehreren Implementierungen
- Aufrufen als wäre es eine Implementierung
- CompositeXYZ erbt von IXYZ
- CompositeXYZ kennt alle “regulären” XYZ
- CompositeXYZ leitet alle Aufrufe an XYZ weiter

Aufgabe

Implementiere das Composite Command

Projekt “PdfTools”

CompositeCommand erstellen
Commands und injizieren

Tagging Interface verwenden

03

Empty-Object Pattern

aka. Null-Object Pattern

Null-Object Pattern / Empty-Object Pattern

- Implementieren des Interface mit einer „leeren“ Funktionalität
- Funktionen geben „Default()“ zurück
- Methoden machen „nichts“

- Das Empty-Object Pattern darf den Zustand/das Verhalten des Systems nicht verändern

Aufgabe

Implementiere ein Empty-Object

Projekt “PdfTools”

EmptyStrategy,
Fallback-Strategy.

Resumee Command vs. Strategy

- Prüfen „CanExecute()“
- Mehrere „true“ für gleichen Context
- Verwendung: z.B. UI-Buttons

- N-Methoden per Strategie
- I.d.R. nur eine Strategie
- Verwendung: z.B. MSSQL vs. MongoDB

- Aggregierung über “Simple” Composite

- Leere Implementierung Empty-Object