SEMINARARBEIT

Im Studiengang Informatik

Mediator & Observer

Ausgeführt von: Florian Feka
Peter Öttl
Personenkennzeichen: 1910257104(Feka)
1910257146(Öttl)

BegutachterIn: DI Wolfgang Berger

Wien, den 6. Juni 2021

FH University of Applied Sciences
TECHNIKUM WIEN

Inhaltsverzeichnis

1 Umsetzung

Bei dieser Übung habe ich mich für das Angular Framework entschieden da es sehr viele Vorteile im Bereich UI Komponenten und State Management bietet.

Die Liste und die Buttons sind im HTML definiert worden. Die selektierten Items sind angebunden an dem ‘selectedItems‘ Array. Ob einer der drei Buttons disabled wird hängt jeweils von den drei Variablen ‘button1Disabled‘, ‘button2Disabled‘ und ‘button3Disabled‘ ab. Im Konstruktor wird die Observer function hinzugefügt zum Mediator die dann aufgerufen wird, wenn sich der Wert ’listCount’ ändert (mehr dazu dann beim Mediator). Die Liste führt die Funktion ‘listSelectionChanged()‘ aus wenn sich an der Selektion in der Liste was ändert.

```
export class AppComponent implements OnInit {
  selectedItems = [];
  mediator = new Mediator();
  button1Disabled = false;
  button2Disabled = false;
  button3Disabled = false;
  constructor() {
    this.mediator.addObserver('listCount', () => {
      const value = this.mediator.getValue('listCount');
      this.button1Disabled = true;
      this.button2Disabled = true;
      this.button3Disabled = true;
      if (value === 1) {
        this.button1Disabled = false;
      } else if (value >= 1) {
        this.button2Disabled = false;
      } else if (value === 0) {
        this.button3Disabled = false;
      }
    });
  }
  ngOnInit(): void {
    this.mediator.setValue('listCount', 0);
  }

  listSelectionChanged() {
    this.mediator.setValue('listCount', this.selectedItems?.length);
  }
}
```

Quellcode 1: app.component.ts

Der Mediator hat eine Storage map und eine Observer map. Im Storage wird die Funktion definiert ‘setValue‘ die bei einer Änderung des Wertes die Funktion ‘notifiyObservers‘ vom übergebenen Mediator aufruft. Die ‘setValue‘ Funktion von Storage wird vom Mediator in der eigenen ‘setValue‘ funktion aufgerufen, wo in der Storage map eine neue Storage Instanz gespeichert wird und gleich danach der übergebene Wert gesetzt wird mit der ‘setValue‘ Funktion von Storage (Zeile 20-22). Nach dem ‘notifyObserver‘ aufgerufen wurde (Zeile 10) kommt es dazu, dass die jeweilige Observer Funktion aufgerufen wird (die wir in unserem Beispiel in app.component.ts Zeile 8 hinzugefügt haben).

```
1 export class Storage {
2   private value: any;
3
4   getValue(): any {
5     return this.value;
6   }
7
8   setValue(mediator: Mediator, storageName: string, value: any) {
9     this.value = value;
10    mediator.notifyObservers(storageName);
11  }
12 }
13
14 export class Mediator {
15  private storageMap: { [key: string]: Storage; } = {};
16
17  private observers: { [key: string]: () => void; } = {};
18
19  setValue(storageName: string, value: any) {
20    const storage = new Storage();
21    this.storageMap[storageName] = storage;
22    storage.setValue(this, storageName, value);
23  }
24
25  getValue(storageName: string) {
26    return this.storageMap[storageName].getValue();
27  }
28  addObserver(storageName: string, observer: () => void): void {
29    this.observers[storageName] = observer;
30  }
31
32  notifyObservers(eventName: string): void {
33    const observer = this.observers[eventName];
34    observer();
35  }
36 }
```

Quellcode 2: mediator.ts

2 Fragen oder Erkenntnisse

Beim Mediator Pattern sehe ich den Bezug zur Praxis nicht so ganz. Aufgrund der Component Based Architecture von modernen Frontend Frameworks ist das Mediator Pattern nicht wirklich notwendig. Hierbei werden die benötigten Daten direkt in der Komponente gehalten. Somit werden die unterschiedlich benötigten Werte gut abstrahiert. Meiner Meinung nach hat das Mediator Pattern den aktuellen Code sogar komplizierter und anfänglich undurchschaubar gemacht. In Bezug auf dem aktuellen Beispiel hätte ich eine List Komponente gebaut welche mit der listSelectionChanged Methode nur den Wert in eine Variable gespeichert hätte. Im HTML Template hätte ich das diabled Feld der Buttons einfach abhängig vom Wert disabled bzw. enabled und der ChangeDetector von Angular hätte sich um den Rest gekümmert.

Ganz anders sieht es beim Observable Pattern aus. Das könnte man sich im Reative Programming gar nicht mehr wegdenken. Es war sehr interessant so etwas selbst zu entwickeln aber bei einem echten Projekt würde ich mich da eher auf rxJS zurückgreifen.

Quellcodeverzeichnis