WAP - Internetové aplikace

Projekt 1
Reťazec zodpovednosti

Patrik Németh
xnemet04

Popis

Pre ukážku využitia návrhového vzoru reťazca zodpovednosti (angl. chain of responsibility) bol implementovaný todo manažér pre príkazový riadok. Jedná sa o jednoduchú aplikáciu využívajúcu tento návrhový vzor pre plnenie požiadaviek, ktoré sú generované priamo užívateľom. Vytvárané položky todo sa ukladajú vo formáte JSON do špecifikovaného súboru.

Existujú rôzne prístupy k ukončeniu propagácie požiadavky cez reťazec zodpovednosti. V implementovanej knižnici sa využíva prístup ukončenia propagácie po splnení požiadavky jedným z článkov reťazca zodpovednosti. Dosiahnutie konca reťazca znamená nesplnenie požiadavky a vyúsťuje do výnimky.

Využitie návrhového vzoru reťazca zodpovednosti v tejto aplikácii otvára možnosti pre ľahkú rozšíriteľnosť, a to jednoducho tým, že sa vytvorí ľubovoľny počet nových článkov reťazca pre konkrétne typy požiadaviek a zadefinuje sa ich chovanie. Dynamické vlastnosti implementačného jazyka umožňujú jednoduché odovzdávanie parametrov bez redefnície alebo explicitného preťažovania metód definovaných v knižnici, čo z nej činí vysoko univerzálny framework pre tento návrhový vzor.

Použitie knižnice

Užívateľ vytvorí inštanciu enkapsulujúcej triedy `Chain`:

```
let chain = new Chain();
```

Následne vytvorí články reťazca v rámci enkapsulujúcej inštancie pomocou triedy `Link`, prípadne triedy z nej derivovanej, kde prvý argument konštruktora je identifikátor požiadavky:

```
chain.linkA = new Link(“A”);
chain.linkB = new Link(“B”);
chain.linkC = new Link(“C”);
```

Ďalej nastaví poradie článkov:

```
chain.setFirst(chain.linkA);
chain.linkA.setNext(chain.linkB);
chain.linkB.setNext(chain.linkC);
```

A nakoniec zadefinuje metódy `completeRequest` jednotlivých článkov:

```
chain.linkA.completeRequest = function() { console.log(“A”); }
chain.linkB.completeRequest = function() { console.log(“B”); }
chain.linkC.completeRequest = function() { console.log(“C”); }
```

Reťazec sa spustí cez enkapsulujúci objekt:

```
chain.handle(request);
```

Spustenie programu

Program využíva argumenty, ktorých prehľad je možné zobraziť pomocou prepínača `-h` alebo `--help`. Program sa používa nasledovne:

```
node model.js file OP [-h | --help]
```

`OP` môžu byť:

- `add {text}` Pridá todo položku s textom `text` do súboru `file`.
- `rem {id}` Odoberie todo položku identifikovanú číslom `id` zo súboru `file`.
- `cng {id} {text}` Zmení text todo položky `id` na `text` v súbore `file`.
- `sho` Zobrazí všetky todo položky v súbore `file`.

Viacslovný `text` je nutné uviesť v jednoduchých úvodzovkách (napr. ‘text so štyrmi slovami’). Súbor `file` sa vytvorí podľa potreby.

Implementácia

Tento projekt bol implementovaný a testovaný v Node.js verzie 15.9.

`library.js`

Jedná sa o jednoduchú obecnú knižnicu dvoch, užívateľom rozšíriteľných, tried. Umožňujú implementovať návrhový vzor reťazca zodpovednosti poskytovaním základného frameworku pre poskladanie takéhoto reťazca.

Trieda `Chain`

Slúži ako enkapsulácia reťazca zodpovednosti, kde užívateľ do jej inštancie pridáva jednotlivé články reťazca zodpovednosti. Trieda definuje metódu pre nastavenie prvého článku, a metódu pre zaslanie požiadavky prvému článku.

Trieda `Link`

Slúži ako článok v rámci reťazca zodpovednosti. Každý článok obsahuje informáciu o type požiadavky, ktorú má spracovať (atribút `request`) a o svojom následníkovi (atribút `next`). Obsahuje aj dve preddefinované metódy: `setNext()`, ktorá nastavuje následníka daného článku a `handle()`, ktorá rozhoduje o preposlaní požiadavky ďalšiemu článku v reťazci, alebo o prípadnom spracovaní tejto požiadavky súčasným článkom.

V tejto triede je namodelovaná aj “abstraktná” metóda `completeRequest()`, ktorej východisková implementácia iba hodí výnimku, že nie je implementovaná, čím sa simuluje abstraktnosť metódy. Implementácia tejto metódy totiž musí byť špecifická pre daný prípad použitia knižnice, pretože ona plní konkrétne požiadavky.

`model.js`

Pre demonštráciu knižnice library.js bol implementovaný jednoduchý todo manažér. Využitie reťazca zodpovednosti spočíva v odovzdávaní požiadavky na úkon manažéra, ako napríklad pridanie alebo odobratie todo položky. Manažér ukladá todo položky do špecifikovaného súboru vo formáte JSON.

Trieda `Todoer`

Jedná sa o rozširujúcu triedu knižnicovej triedy `Link`. Rozširuje ju o metódy špecifické pre konkrétny prípad použitia univerzálnej knižnice.

Kód v `model.js`

Zdrojový kód je možne vnímať ako rozdelený na štyri časti. V prvej prebieha spracovanie programovýh argumentov. Ďalej sa nachádza definícia triedy `Todoer` a zloženie reťazca zodpovednosti. Tretia časť obsahuje definíciu metódy `completeRequest()` pre jednotlivé články reťazca. Nakoniec je reťazec spustený s požiadavkou užívateľa a prípadnými parametrami.

UML diagram tried