WAP
Objekty v jazyku JS
Command

Peter Grofčík

Marec 2021

1 Popis návrhového vzoru

Cieľom návrhového vzoru Command[1] je zapúzdriť akcie/operácie/požiadavky ako objeky, čo umožňuje oddelenie častí systémov, ktoré požadujú vykonanie danej operácie od tých, ktoré ju vykonajú.

Návrhový vzor sa skladá z troch hlavných častí:

- Invoker
- Command
- Receiver

Požadovaná operácia je zabalená v objekte Command a posunutá objektu Invokera, ktorý riadi (invokuje) uskutočnenie commandov a tie po spustení predávajú akciu objektu Receivera, ktorý požadované akcie vykoná. Týmto spôsobom je oddelený objekt vydávajúci príkaz (Invoker) od objektu, ktorý daný príkaz vykoná (Receiver). Invoker tiež ukladá vykonané Commandy do stacku, čo umožňuje uchovávanie posledných vykonaných akcii zabalených vrámci objektu command a zároveň možnosť odvolania/návratu vykonaných akcii.

1.1 Využitie

Ako som už spomenul, tento návrhový vzor je vhodný v prípade, že potrebujeme implementovať multi-level "undo" operáciu, kedy si Invoker udržuje niekoľko posledných vykonaných operácii a je schopný zvrátiť ich účinok. Tento návrhový vzor je tiež vhodný vrámci sieťovej komunikácie, kde môžme odosielať objekt command vzdialenému sieťovému zariadeniu (napr. akcie hráčov v počítačových hrách). Z rovnakého dôvodu je tiež vhodný pre paralelné programovanie (resp. paralelný processing), kde commandy predstavujú úlohy na zdieľaných zdrojoch avšak spúšťané viacerými vláknami. Vzor je tiež vhodný vrámci webového

[1] https://archive.org/details/designpatternsel00gamm/page/233/mode/2u

backendu, kedy opäť akcie z rôznych tlačidiel sú zabalené do objektu command a každá z akcii môže mať ľubovoľný tvar, a teda je jednoduché doplniť novú funkcionalitu v prípade, že o ňu zákazník požiada.

2 Vytvorené API

Vytvorené API som testoval na node verzii 13.14 a 14.15. V najuniverzálnejšej implementácii by vzor Command obsahoval iba Command class, keďže Invoker a Receiver sú objektmi, ktoré vytvárajú či vykonávajú jednotlivé špecifické akcie zapúzdrené v Commande. Pre názornosť som však vytvoril všetky 3 v čo najuniverzálnejšej podobe.

2.1 Invoker

Invoker v konštruktore nepožaduje žiadny vstup a vytvára prázdne pole pre commandy. Nad objektom sú podporované dve funkcie. *ExecuteCommand* (očakáva na vstupe Command objekt) zavolá funkciu Execute na obdržanom commande, a zároveň daný command uloží do poľa (s ktorým pracuje ako so zásobníkom). *UndoCommand* (neočakáva žiadny vstup) vyberie posledný vykonaný command a zavolá nad ním funkciu Undo.

2.2 Command

Command v konštruktore očakáva 3 parametre. Objekt Receivera a dva objekty pre akciu a akciu undo. V náväznosti na Invokera podporuje funkcie Execute a Undo, ktorými po zavolaní pošle požiadavku na vykonanie jednej z nich na Receivera.

2.3 Receiver

Receiver v konštruktore očakáva objekt nad ktorý vykonáva akcie (mocking DB) a funkciou Action podporuje spustenie vykonávania akcie obdržanej z Commandu. Receiver class vyslovene spĺňa názornosť univerzálnej implementácie z definície commandu ako som už spomínal vyššie. Zároveň však obsahuje jedinu závyslosť voči implementácii problému (resp. jej ukážke). Objekt ktorý obdrží v konštruktore musí podporovať funkciu *actions*, ktorá spracúva a vykonáva obdržané akcie. Odstránením Receivera by sa táto závyslosť presunula do každej z funkcii objektu Command.

3 Príklad využitia

V príklade som vychádzal z implementácie informačného systému pre banku z predmetu IUS/IIS. Príklad obsahuje classu *bankDB*, ktorá predstavuje DB účtov, vrátane klientov/vlastníkov účtov. Účet podporuje jednoduché operácie výberu a vkladu financii, a objekt bankDB dodatočne ich kombináciu pri presúvaní

peňazí z účtu na účet. Tento objekt ako som už spomenul tiež musí podporovať funkciu *actions*, ktorá spracúva a vykonáva obdržané akcie. Daný objekt je posunutý pre Receivera z modelu v *library.mjs*.

Pre akcie je vytvorená clasa action, ktorá obsahuje potrebné informácie o požadovanej akcii (jej typ, hodnotu financii, čísla účtov). Objekt akcie je predaný spolu s akciou s reverzným účinkom objektu Command a každý takto vytvoreny command je posunutý objektu invoker.

V príklade samotnom vykonávam 3 akcie na presun, výber a vklad financii. Pred akciami a po nich je na consolu vypísaný aktuálny stav dvoch účtov v databáze banky. Následne dôjde k odvolaniu posledných dvoch akcii a finálnemu výpisu stavu účtov po zvrátení akcii.