Wzorce projektowe, cz. 11 – Observer

Głównym obszarem wykorzystania wzorca ***Obserwator*** jest stworzenie relacji typu jeden-do-wielu łączącej grupę obiektów. Dzięki zastosowaniu wzorca zmiana stanu (czyli zmiana aktualnych wartości pól) obiektu obserwowanego umożliwi automatyczne powiadomienie o niej wszystkich innych dołączanych elementów (obserwatorów).

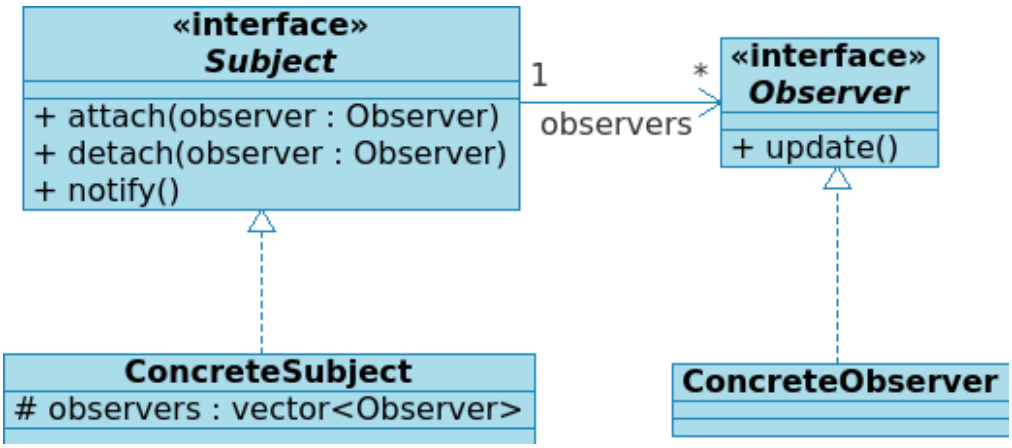

Diagram klas wzorca Observer

Interfejs ***Subject*** definiuje operacje ***attach()*** i ***detach()*** pozwalające odpowiednio na dołączanie i odłączanie obserwatorów. Ponadto zdefiniowana jest też operacja ***notify()***, służąca do powiadamiania wszystkich zarejestrowanych obserwatorów o zmianie stanu obiekt obserwowanego poprzez wywołanie w pętli metody ***update()***, która jest zadeklarowana w interfejsie ***Observer***. Operacja ta jest implementowana w klasie realizującej ten interfejs i służy do powiadamiania konkretnego obserwatora o zmianie stanu obiektu obserwowanego.

Przykładowa implementacja

```
<?php


class Observer1 implements SplObserver {


    public function update(SplSubject $subject) {
        echo 'Aktualizacja Observer1';
    }


}


class Observer2 implements SplObserver {
```

```
    public function update(SplSubject $subject) {
        echo 'Aktualizacja Observer2';
    }

}

class Subject implements SplSubject {

    protected $observers = array();

    public function attach(SplObserver $observer) {
        $this->observers[spl_object_hash($observer)] = $observer;
    }

    public function detach(SplObserver $observer) {
        unset($this->observers[spl_object_hash($observer)]);
    }

    public function notify() {
        foreach ($this->observers as $observer) {
            $observer->update($this);
        }
    }

}

$subject = new Subject();
$observer1 = new Observer1();
$observer2 = new Observer2();

$subject->attach($observer1);
```

```
$subject->attach($observer2);
$subject->notify();

?>
```

Przykład z życia wzięty

No i tu zaczynają się małe schody … Ze względu na specyfikę PHP (aplikacja „działa" raptem ułamki sekund) użyteczność tego wzorca jest nieco ogranicza w porównaniu z innymi językami (Java, C++ itp.). Jednak, mimo tego da się wykorzystać dobrodziejstwa ***Obserwatora*** w PHP.

```
<?php

class CacheObserver implements SplObserver {

    public function update(SplSubject $subject) {
        echo "Odswieza cache";
    }

}

class RSSObserver implements SplObserver {

    public function update(SplSubject $subject) {

        echo "Odswieza RSS";
    }

}

class NewsletterObserver implements SplObserver {

    public function update(SplSubject $subject) {

        echo "Wysylam maile z nowym newsem";
```

```
    }

}

class News implements SplSubject {

    private $observers = array();

    public function attach(SplObserver $observer) {
        $this->observers[spl_object_hash($observer)] = $observer;
    }

    public function detach(SplObserver $observer) {
        unset($this->observers[spl_object_hash($observer)]);
    }

    public function notify() {
        foreach ($this->observers as $observer) {
            $observer->update($this);
        }
    }

    public function add($data) {
        echo 'Dodaje news do bazy';
        $this->notify();
    }

}

$news = new News();

$news->attach(new RSSObserver());
```

```
$news->attach(new CacheObserver());
$news->attach(new NewsletterObserver());

$news->add(array(
    'title' => 'Tytuł',
    'content' => 'blablabla'
));

?>
```

Do obiektu klasy ***News***, który jest obserwowany dodajemy trzech obserwatorów. Dzięki temu zabiegowi przy dodawaniu nowego artykułu elementy systemu zostaną „poinformowane" i odświeżą dane w cache/RSS oraz wyślą odpowiednie maile do czytelników.

Zalety i wady

Zalety:

- Luźna zależność między obiektem obserwującym i obserwowanym. Ponieważ nie wiedzą one wiele o sobie nawzajem, mogą być niezależnie rozszerzane i rozbudowywane bez wpływu na drugą stronę.
- Relacja między obiektem obserwowanym a obserwatorem tworzona jest podczas wykonywania programu i może być dynamicznie zmieniana.
- Możliwość zablokowania klientowi drogi do bezpośredniego korzystania ze złożonego systemu, jeśli jest to konieczne.

Wady:

- Obserwatorzy nie znają innych obserwatorów, co w pewnych sytuacjach może wywołać trudne do znalezienia skutki uboczne.

Zastosowanie

Wzorzec Obserwatora sprawdza się wszędzie tam, gdzie stan jednego obiektu uzależniony jest od stanu drugiego obiektu.