Wrocław, 29. marca 2006

Projektowanie Systemów Informatycznych

Politechnika Wrocławska
Wydział Informatyki i Zarządzania
IV rok studiów

Wzorce projektowe związane z warstwą danych

Autor dokumentu: **STAWARZ Paweł**
Indeks: **125939**
Grupa: **wtorek/np 15.15**
Termin seminarium: **7. marca 2006**

Prowadzący: **dr Jan Kwiatkowski**

Spis treści

Abstrakt

W niniejszej pracy zostaną omówione elementarne wzorce projektowe związane z warstwą danych popularnego modelu warstwowego aplikacji. W pierwszej części pracy pojawi się niewielki wstęp przybliżający sens i działanie omawianego modelu, aby następnie skupić się szczegółowo na analizowanej warstwie danych. Zostaną tu dokładnie przedstawione cztery wzorce projektowe: *Table Data Gateway*, *Row Data Gateway*, *Active Record* oraz *Data Mapper*. Każdy z tych wzorców będzie wyczerpująco opisany i podparty przykładowymi diagramami UML jak również fragmentami kodu w języku Java.

Model warstwowy aplikacji

Wprowadzenie

Wady i zalety

Modelowanie aplikacji w sposób warstwowy stało się ostatnio podejściem bardzo modnym i często wykorzystywanym. Dzieje się to nie bez przyczyny, ponieważ podział aplikacji na logicznie wyodrębnione warstwy niesie ze sobą wiele korzyści. Przede wszystkim jasno sprecyzowany podział pozwala rozwijać wiele myśli niemalże równolegle. Wydzielone warstwy mogą odpowiadać za komunikację z użytkownikiem, za przechowywanie danych w odpowiednich strukturach, za udostępnianie gromadzonych danych, czy wreszcie za realizowanie pewnej określonej logiki dziedzinowej. Dzięki temu każda warstwa aplikacji może być traktowana jako pewna spójna całość prawie niezależna od innych. Pozwala to między innymi zastępować aktualne rozwiązania, w ramach jednej warstwy, implementacjami alternatywnymi, które de facto mogą realizować te same zadania, ale w nieco odmienny sposób. Wyszczególnione warstwy poddają się również bardzo łatwo standaryzacji, która dość często ułatwia życie programisty/projektanta, który biorąc czynny udział w tworzeniu nowych aplikacji może śmiało korzystać z dotychczasowych rozwiązań jak również praktykowanych podejść w często pojawiających się podobnych sytuacjach. Oczywiście model warstwowy nie jest pozbawiony kilku wad, ale nie wymyślono dotychczas rozwiązań idealnych do każdego typu zagadnienia. Jedną z nich jest pojawiający się czasem problem modyfikacji wielu warstw jednocześnie, np. w sytuacji gdy do przechowywanych danych w pewnej tabeli relacyjnej bazy danych dołączono nową kolumnę, która odpowiednio powinna być uwzględniona podczas prezentacji zwracanych wyników do użytkownika, czy też podczas wprowadzania nowych danych poprzez dostarczony do tego formularz. W tej sytuacji pojawia się małe niedociągnięcie wspomnianej wcześniej „prawie niezależności". Nie można również przesadzać z liczbą warstw wydzielonych w aplikacji. Nadmierna ich ilość może znacznie zmniejszyć wydajność działania końcowego oprogramowania, głównie ze względu na pojawiającą się między warstwami transformację danych pomiędzy różnymi ich reprezentacjami.

Działanie

Działanie modelu warstwowego aplikacji opiera się na kilku charakterystycznych cechach, które zostaną tu w skrócie omówione. Wyobraźmy sobie sytuację, w której wyodrębnionych zostało n warstw i załóżmy, że w celu ich rozróżnienia przyjęto uporządkowaną numerację od 1 dla warstwy najwyższej do n dla warstwy najniższej.

W większości przedsięwzięć przestrzegane są następujące zasady funkcjonowania takiego modelu:

- każda warstwa i korzysta z usług dostarczanych przez warstwę $i+1$,
- każda warstwa i nie wymaga obecności warstw $1,2,...,i-1$, a więc nie korzysta również z żadnych usług przez nie dostarczanych,
- każda warstwa i ukrywa obecność warstw niższych $i+1,i+2,...,n$ przed warstwami wyższymi od niej, tzn. $1,2,...,i-1$.

Podstawowe warstwy

Podział

W najprostszym modelu warstwowym aplikacji stosuje się podejście trójwarstwowe, w którym wyróżnia się trzy elementarne warstwy: prezentacji, logiki biznesowej oraz źródła danych. Podział ten w sposób najbardziej ogólny wyodrębnia trzy obszary logiczne: komunikacji z użytkownikiem, realizacji funkcjonalności związanej z zamodelowaną dziedziną oraz udostępniania i przechowywania danych

trwałych. Chociaż jest to minimalny podział warstwowy, to jednak nadal dość często (dla mało złożonych systemów) można ograniczyć się do realizacji jedynie dwóch warstw: prezentacji oraz źródła danych. Wyodrębnienie dodatkowej warstwy logiki biznesowej wynika w głównej mierze ze zwiększających się potrzeb użytkownika, który zwiększając własne wymagania co do projektowanego systemu informatycznego przyczynia się tym samym do znacznego zwiększania jego złożoności. Pojawiła się tym samym potrzeba oddzielenia kodu odpowiedzialnego za prezentację jak również dostępu do danych od kodu odpowiedzialnego za realizację wymagań funkcjonalnych postawionych przez użytkownika.

Omawiane tutaj podejście trójwarstwowe charakteryzuje się oczywiście wspomnianym wcześniej przestrzeganiem podstawowych zasad:

- w warstwie źródła danych nie znajdują się odwołania do usług realizowanych przez warstwę prezentacji oraz warstwę logiki biznesowej,
- w warstwie logiki biznesowej nie znajdują się odwołania do usług realizowanych przez warstwę prezentacji, ale znajdują się odwołania do usług realizowanych przez warstwę źródła danych,
- w warstwie prezentacji znajdują się odwołania do usług realizowanych przez warstwę logiki biznesowej, ale nie znajdują się odwołania do usług realizowanych przez warstwę źródła danych.

Warstwa prezentacji

Warstwa prezentacji, jak sama nazwa wskazuje, odpowiedzialna jest przede wszystkim za prezentację wyników żądanych przez użytkownika. Jest ona pewnego rodzaju interfejsem zewnętrznym pozwalającym użytkownikowi komunikować się z systemem poprzez wywoływanie dostępnych usług. Pełni ona również funkcję obsługi zdarzeń wynikających np. z obsługi myszki bądź klawiatury.

W najprostszej postaci warstwa prezentacji może być realizowana przez najzwyklejszy wiersz poleceń działający w trybie tekstowym. Współcześnie stosuje się jednak częściej tzw. graficzne interfejsy użytkownika, które wykorzystują możliwości dostarczane przez aktualne systemy operacyjne, a które w znaczny sposób ułatwiają poruszanie się użytkownika w jego aplikacji.

Warstwa logiki biznesowej

Warstwa logiki biznesowej odpowiedzialna jest za realizację właściwej logiki działania systemu związanej z zamodelowaną rzeczywistością. Stara się ona spełnić wszystkie wymagania funkcjonalne stawiane przez użytkownika wobec systemu – np. automatyzuje proces wyliczania pewnych wartości na podstawie danych trwałych (znajdujących się w bazie danych) jak również danych wprowadzanych przez użytkownika. Pośredniczy ona w obsłudze żądań użytkownika oraz w korzystaniu z usług dostarczanych przez warstwę źródła danych. Zajmuje się tym samym walidacją danych wprowadzanych przez użytkownika oraz określaniem właściwym dyspozycji wobec warstwy źródła danych.

Warstwa źródła danych

Warstwa źródła danych odpowiedzialna jest za organizację danych trwałych przechowywanych na serwerach plików bądź w bazach danych i odpowiednim udostępnianiu ich wyższym warstwom, tak aby mogły one korzystać z tych danych bez wiedzy dotyczącej struktury ich przechowywania. Sercem tej warstwy jest zazwyczaj relacyjna baza danych, za którą odpowiedzialna jest jedna z wielu implementacji systemu zarządzania relacyjną bazą danych opartego na powszechnie znanym standardzie SQL. Popularność wykorzystywania relacyjnych baz danych w większości przedsięwzięć bierze się właśnie z dobrze już opanowanego przez wielu programistów/projektantów standardu.

Uwagi

Każda z wymienionych wyżej warstw może być dalej dzielona na wiele modułów odpowiedzialnych za szczegółowo wyodrębnione obszary logiczne. Na przykład warstwa prezentacji może dostarczać zarówno modułu realizującego komunikację z systemem poprzez wiersz poleceń, jak również modułu realizującego tę komunikację poprzez graficzny interfejs użytkownika. Analogicznie w warstwie logiki biznesowej można wydzielić wiele modułów, z których każdy odpowiedzialny będzie za pewien wytyczony obszar rzeczywistości. Oczywiście w warstwie źródła danych może się również znajdować więcej modułów realizujących komunikację z odpowiednimi systemami przechowującymi interesujące nas dane.

Jasne wytyczenie granic pomiędzy opisywanymi warstwami okazuję się być dość trudne. Często podział ten będzie uzależniony od struktury fizycznej systemu. Może się też okazać, że pewne funkcjonalności realizowane np. przez warstwę logiki biznesowej można będzie przerzucić na warstwę prezentacji

(np. walidacja wprowadzanych przez użytkownika danych) bądź też warstwę źródła danych (np. pilnowanie spójności bazy danych i związane z nią sprawdzanie wszelkich ograniczeń i założeń dziedzinowych). Dlatego też pomimo swej przeważającej niezależności, omawianie wzorców projektowych związanych z warstwą źródła danych będzie prowadzone w kontekście pewnych rozwiązań dotyczących wyższej warstwy logiki biznesowej.

Wzorce projektowe warstwy źródła danych

Wzorzec projektowy

Coraz większą popularnością podczas projektowania nowych systemów informatycznych, cieszy się stosowanie tzw. wzorców projektowych. Wiąże się to ze zwiększaniem wydajności oraz efektywności tworzenia nowych aplikacji. Najogólniej rzecz biorąc wzorzec projektowy oznacza pewne konkretne rozwiązanie dotyczące często pojawiającego się problemu, bądź też często pojawiających się podobnych sytuacji. Jest to rozwiązanie typowe dla określonego zagadnienia, które w sposób efektywny realizuje swoje zadanie. Wzorce projektowe powstają w wyniku praktycznych doświadczeń, w których nowe pomysły świetnie się sprawdzają. Korzystanie ze sprawdzonych wzorców projektowych pozwala z kolei zaoszczędzić czas i pieniądze związane z realizacją własnych pomysłów, które na pewno nie pozbawione będą błędów. Wzorce projektowe dają pewną gwarancję jakości, ponieważ skoro są wykorzystywane w wielu przedsięwzięciach, to na pewno zostały już wystarczająco przetestowane. Niestety w ostatnich latach liczba sprawdzonych wzorców projektowych diametralnie wzrosła i w tej chwili niedoświadczony projektant łatwo może się zgubić w gąszczu oferowanych możliwości. Oczywiście osobiście uważam jednak, że ogromna liczba wzorców projektowych jest rewelacyjnym zjawiskiem, które w znacznym stopniu uelastycznia proces projektowania systemu informatycznego. Należy przy tym pamiętać, że powstały opis wzorca projektowego nie wymaga restrykcyjnego jego stosowania. W tym tkwi również wielka siła wzorców, gdyż nie są one sztywnym schematem rozwiązania czegoś, tylko pewną radą na ukierunkowanie dalszych działań projektowych. Można tutaj przy okazji zacytować trafną myśl architekta Alexandra związaną z wzorcami budowlanymi, która świetnie się sprawdza również w kontekście wzorców projektowych systemów informatycznych: „Każdy wzorzec opisuje pewien regularnie napotykany problem i łączy go z ogólnym opisem jego rozwiązania w sposób, który pozwala stosować to rozwiązania miliony razy, ale za każdym razem nieco inaczej”.

Zarys wzorców projektowych warstwy logiki biznesowej

Zanim szczegółowo zostaną omówione wzorce projektowe związane z warstwą źródła danych, warto chociaż ogólnie zarysować ideę trzech podstawowych wzorców projektowych związanych z warstwą logiki biznesowej: *Transaction Script*, *Domain Model* oraz *Table Module*. Po szczegóły odsyłam do książki Martina Fowlera [1]. Należy przy tym zwrócić uwagę, iż stosowanie niżej omówionych wzorców w żaden sposób nie wyklucza się wzajemnie – naturalnie można łączyć każde rozwiązanie z wydzieloną częścią logiki biznesowej.

Transaction Script

Wzorzec *Transaction Script* jest najprostszym podejściem do realizacji usług dostarczanych przez warstwę logiki biznesowej. Wzorzec ten, jak sama nazwa wskazuje, jest skryptem pewnej określonej transakcji. Transakcja utożsamiana jest tutaj odpowiednio z pojedynczą czynnością użytkownika, która realizowana jest przez procedurę rozpoczynającą się od otwarcia transakcji i kończącą się na jej zamknięciu. W ramach takiej transakcji mogą być np. pobierane dane wejściowe z warstwy prezentacji, które następnie podlegają weryfikacji, obliczeniom, utrwalaniu w bazie danych, itp. Transakcja może również sprowadzać się do zwrócenia żądanych danych do warstwy prezentacji. Naturalną rzeczą w tym podejściu staje się wyodrębnianie powtarzających się często podprocedur, które następnie wykorzystywane są przez wiele takich skryptów transakcji.

Wzorzec *Transaction Script* jest prostym modelem proceduralnym zrozumiałym dla wszystkich programistów, który dobrze współpracuje z omawianymi w dalszej części pracy wzorcami projektowymi warstwy źródła danych *Row Data Gateway* oraz *Table Data Gateway*. Jednakże podejście to nadaje się jedynie do systemów informatycznych z niewielką złożonością dziedzinową. Jeśli tylko złożoność systemu będzie odpowiednio wysoka, to stosowanie tego wzorca może doprowadzić do powstawania wymieszanego bełkotu zawiłych procedur o niejasnej strukturze i niezrozumiałym układzie.

Domain Model

Wzorzec *Domain Model* jest niejako odpowiedzią na zwiększającą się złożoność dziedzinową realizowanych systemów informatycznych. Jak sama nazwa wskazuje skupia on swoją uwagę na kształtowaniu modelu dziedzinowego realizującego logikę biznesową modelowanej rzeczywistości, które bazuje na ogólnej obiektowości. Podejście to w znaczny sposób przyczynia się do przejrzystego porządkowania coraz bardziej złożonego kodu. Ze względu na mocno rozpowszechniony współcześnie sposób programowania obiektowego zaleca się stosowanie tego wzorca nawet w najprostszych rozwiązaniach. Proponowane rozwiązanie sprowadza się do tworzenia klas, których obiekty reprezentują pojedyncze wiersze tabeli w relacyjnej bazie danych.

Wzorzec *Domain Model* może współpracować zarówno ze wzorcami typu *Gateway* jak i wzorcami *Active Record* czy *Data Mapper* szczegółowo opisanymi w dalszej części pracy. Zaleca się jednak stosowanie wzorca *Active Record* w przypadku gdy logika dziedziny jest prosta, a komunikacja między klasami i tabelami duża, natomiast *Data Mapper* w sytuacji gdy logika dziedziny staje się złożona i pojawiają się mocne powiązania pomiędzy modelem dziedziny a schematem bazy danych

Table Module

Wzorzec *Table Module* jest rozwiązaniem pośrednim między wzorcem *Transaction Script* a *Domain Model*, ponieważ porządkuje on logikę dziedzinową wokół pojedynczej tabeli. Z jednej strony przypomina wzorzec *Transaction Script* ze względu na tworzone procedury, z drugiej jednak strony zbliża się do wzorca *Domain Model* ze względu na zwiększoną strukturalizację – procedury grupowane są w klasy obsługujące całe tabele. Wzorzec *Table Module* przeznaczony jest do współpracy z typem danych *Record Set*, który umożliwia wykonywanie operacji na pobieranych zbiorach rekordów. Proponowane rozwiązanie sprowadza się do tworzenia klas, których obiekty reprezentują pojedyncze tabele relacyjnej bazy danych.

Wzorzec *Table Module* wykorzystywany jest najczęściej wraz ze wzorcem *Table Data Gateway*, który zostanie szczegółowo omówiony w dalszej części pracy.

Szczegółowy opis wzorców projektowych warstwy źródła danych

Omówione w niniejszej części wzorce projektowe warstwy źródła danych dotyczyć będą tzw. wzorców architektury, które określają sposób komunikacji logiki dziedziny z bazą danych. Jak się okazuje wybór odpowiedniego wzorca architektury w znaczny sposób wpływa na dalszy przebieg całego projektu głównie ze względu na ogromną trudność jaką sprawia ewentualna późniejsza zmiana określonego podejścia. Wybór ten wpływa również znacząco na projekt logiki dziedziny.

Ze względu na to, że język SQL stosowany w komunikacji z relacyjnymi bazami danych jest całkowicie niezależny od języków programowania, w których implementowana jest cała aplikacja, bardzo wygodne staje się wydzielenie fragmentu kodu, w którym znajdować się będą wszystkie definicje transakcji/operacji SQL. Rozwiązanie to ma wiele zalet: po pierwsze bardzo łatwo można odnaleźć miejsca odpowiedzialne za efektywność i wydajność określonych operacji bazodanowych, a po drugie usprawnieniu ulega współpraca z administratorem docelowej bazy danych, który w każdej chwili ma możliwość wglądu w aktualny zbiór wykorzystywanych operacji, dzięki czemu może on starać się zoptymalizować zarówno istniejące zapytania jak i strukturę samej bazy, np. poprzez tworzenie nowych indeksów. Jednym z rozwiązań oddzielających operacje SQL od logiki dziedziny jest umieszczenie ich w osobnych klasach, które oparte są na strukturze tabel bazy (jedna tabela – jedna klasa) – podejście to oparte jest na wzorcu *Gateway*. Wzorzec *Gateway* może być stosowany na dwa sposoby:

- *Row Data Gateway* – obiekt klasy reprezentuje pojedynczy wiersz tabeli,
- *Table Data Gateway* – obiekt klasy reprezentuje pojedynczą tabelę.

Kolejnym rozwiązaniem jest wzorzec *Active Record* wykorzystywany w ramach wzorca *Domain Model*. Operacje SQL wchodzące w interakcje z tabelami bazy danych wplecione zostają w klasy reprezentujące obiekty dziedziny.

Ostatnim często spotykanym rozwiązaniem jest wzorzec *Data Mapper*, który wykorzystywany jest również w ramach wzorca *Domain Model*. Podejście to stosowane jest w przypadkach dużej złożoności logiki dziedzinowej, w której pojawiają się typowe dla programowania obiektowego dziedziczenia, agregacje, kolekcje, itp. *Data Mapper* jest wzorcem mapowania obiektowego-relacyjnego (O/R), który rozdziela kod *Domain Model* od bazy danych poprzez utworzenie dodatkowej warstwy pośredniczącej pomiędzy warstwą

logiki biznesowej oraz warstwą źródła danych. Ze względu na to, iż zadanie mapowania O/R nie zawsze jest przedsięwzięciem prostym, aktualnie istnieje wiele narzędzi komercyjnych w pełni wspomagających ten proces. Niekiedy tworzenie i rozwijanie własnej takiej warstwy mapującej może okazać się o wiele bardziej kosztowne niż zakup, co prawda nietaniego, ale dobrze działającego gotowego narzędzia.

Table Data Gateway

Wzorzec *Table Data Gateway* umieszcza wszystkie operacje SQL w osobnych klasach, których pojedyncze egzemplarze reprezentują pojedyncze tabele relacyjnej bazy danych. Bardzo często wzorzec ten wykorzystywany jest na przykład tylko do opakowywania procedur składowanych zdefiniowanych w bazie danych. Jak sama nazwa wskazuje wzorzec ten pełni wówczas rolę bramy pośredniczącej w komunikacji pomiędzy logiką dziedzinową aplikacji, a operacjami wykonywanymi na danych trwałych przechowywanych w określonej tabeli bazy danych.

Literatura

Franciszek Grabski, Jerzy Jaźwiński, *Metody Bayesowskie w niezawodności i diagnostyce*, Warszawa 2001

Jacek Jakubowski, Rafał Sztencel, *Wstęp do teorii prawdopodobieństwa*, Warszawa 2001