POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2013/2014

PRACA DYPLOMOWA MAGISTERSKA

Michał Aniserowicz

[TYTUŁ] (Generator aplikacji opartych o architekturę CQRS?)

Praca wykonana pod kierunkiem
dra inż. Jakuba Koperwasa

Ocena: .....

.....

Podpis Przewodniczącego Komisji
Egzaminu Dyplomowego

Spis treści

Rozdział 1

Wstęp

TODO
- problem - w projektach często występuje duplikacja w wielu miejscach - w CQRS to w ogóle

Rozdział 2

Duplikacja

2.1 Wprowadzenie

TODO: Opisać co jest duplikacją, a co nie (użycie odnośnika (np. nazwy tabeli) w wielu miejscach nie jest). Odniesinie do Pragmatic Programmer.

TODO: zasada DRY

TODO: gdzie występuje duplikacja, rodzaje duplikacji

TODO: Wszystko jest o dziedzinie abplikacji - a co z resztą? BLL etc.

TODO: Sformułować konkretny przykład aplikacji.

W każdej aplikacji obiektowej korzystającej z bazy danych występuje duplikacja opisu dziedziny aplikacji. Występuje ona w co najmniej dwóch miejsach:

1. schemacie bazy danych (DDL),
2. definicjach klas w kodzie źródłowym aplikacji.

Kolejnym typowym miejscem, w którym umieszcza się informacje o dziedzinie aplikacji jest jej dokumentacja.

W miarę jak rozrasta się projekt informatyczny, pojawia się tendencja do duplikowania fragmentów dziedziny poruszanego przez niego problemu. Duplikacja ta rozprzestrzenia się pomiędzy modułami aplikacji, których zadaniem obróbka tych samych danych, ale w różny sposób. Przykładowo, aplikacja może udostępniać przechowywane dane na następujące sposoby:

- wyświetlać je na stronie WWW,
- wystawiać jako API,
- eksportować do arkusza kalkulacyjnego.

Jeśli moduły te posiadają osobne implementacje dziedziny aplikacji, to każda zmiana dziedziny wymaga zmodyfikowania implementacji dziedziny w każdym z modułów - co wiąże się z dużymi kosztami.

Z drugiej strony, identyczność zestawu danych udostępnianego przez różne moduły może nie być pożądana. Przykładowo, na stronie WWW wyświetlane mogą być jedynie

podstawowe dane danej encji, podczas gdy pełne dane dostępne są po wyeksportowaniu arkusza kalkulacyjnego. Takie wymaganie wymusza duplikację części dziedziny aplikacji pomiędzy różnymi implementacjami tej dziedziny.

2.2 Generyczna implementacja

2.2.1 Refleksja

TODO: Czy generatory to jedyne rozwiązanie? Duplikacji pozwala też uniknąć refleksja.

2.2.2 Serializacja (do JSONA)

TODO

2.2.3 Inne generyczne rozwiązania

TODO

2.3 Rozwiązanie: użycie licznych generatorów

2.3.1 Generator code-first / database-first

Rozwiązaniem problemu może być osiągnięcie implementacji, w której pełna dziedzina aplikacji definiowana jest w jednym miejscu, a jej implementacje są generowane automatycznie w odpowiednich modułach. Taka sytuacja sprawia, że duplikacja pomiędzy modułami przestaje być problemem - aby wprowadzić zmiany we wszystkich implementacjach, wystarczy wprowadzic pojedynczą modyfikację w definicji dziedziny aplikacji.

Najprostszym przykładem realizującym to rozwiązanie jest użycie generatora definicji klas na podstawie schematu bazy danych (tzw. podejście database-first) lub generatora schematu bazy danych na podstawie definicji klas (tzw. podejście code-first). Przykładami takich generatorów są:

- Hibernate,
- EntityFramework,
- LinqToSQL.

TODO: Wymienić więcej, dla różnych technolgii, dać przypisy.

Jednakże takie generatory eliminują tylko jeden rodzaj duplikacji, wymieniony na początku rozdziału.

2.3.2 Generator dokumentacji

Duplikacji w dokumentacji aplikacji można uniknąć poprzez zastosowanie narzędzi generujących tę dokumentację na podstawie kodu źródłowego aplikacji. Narzędzia takie pozwalają na wygenerowanie dokumentacji w kilku formatach, takich jak PDF czy HTML. Ich przykłady to:

- JavaDoc,
- ...

TODO: Wymienić więcej, dla różnych technolgii, dać przypisy.

W przypadku tych narzędzi występuje ten sam problem, co w przypadku generatorów schematu bazy danych lub definicji klas - eliminują one tylko jeden rodzaj duplikacji.

2.3.3 Inne generatory

Aby uniknąć pozostałych rodzajów duplikacji:

- duplikowania nazw i typów kolumn tabel w definicjach widoków bazy danych,
- duplikowania nazw i typów pól klas pomiędzy implementacjami dziedziny aplikacji w różnych jej modułach,

należałoby skorzystać z kolejnych generatorów.

Warto zauważyć, że różne typy generatorów za źródło danych obierają sobie różne definicje dziedziny: np. generator database-first bazuje na języku DDL, a generator dokumentacji - na kodzie źródłowym aplikacji. Używanie wielu różnych generatorów usuwających pojedyncze rozdzaje duplikacji w końcu doprowadziłoby do powstania “łańcucha” generatów, w którym wynik działania jedngo generatora byłby źródłem danych dla innego. Takie rozwiązanie może być bardzo trudne w utrzymaniu.

2.4 Jeden generator wszystkiego

Odpowiednim rozwiązaniem wydaje się być zastosowanie pojedynczego generatora potrafiącego wygenerować wszystkie potrzebne artefakty. Powstanie takiego generatora dającego się zastosować w każdym projekcie jest jednak bardzo mało prawdopodobne:

- każdy projekt ma inne wymagania odnośnie wygenerowanych artefaktów,
- prawdopodobne nie istnieje format, który pozwalałby zdefiniować każdą dziedzinę w najlepszy (najbardziej naturalny) dla niej sposób.

Rozdział 3

Implementacja

Celem niniejszej pracy jest próba stworzenia takiego generatora. Założenia funkcjonalne:

- generator ma być na tyle elastyczny, aby móc obsłużyć wiele sposobów zdefiniowania dziedziny aplikacji,
- generator powinien pozwalać na wygenerowanie dowolnych plików tekstowych (skryptów SQL, kodu źródłowego, dokumentancji HTML itd.),
- generator nie musi generować logiki biznesowej aplikacji - wystarczy dziedzina

Założenia niefunkcjonalne:

- generator zostanie stworzony w technolgii .NET Framework,
- ...

Szczególna uwaga zostanie poświęcona aplikacjom opartym o architekturę CQRS i wykorzystującym bazy danych typu NoSQL. Specyficzną cechą takich aplikacji jest to, że operują one na modelach o wysokim stopniu denormalizacji, co wiąże się z masowo występującą duplikacją danych.

TODO: Opisać CQRS. TODO: Opisać Event Sourcing. TODO: Opisac NoSQL. TODO: Opisać rodzaje baz NoSQL. TODO: Wybrać bazę NoSQL i dlaczego Cassandra. TODO: Opisać Cassandrę.

Bibliografia

[1] Tadeusiewicz R., Korohoda P. (Kraków 1997). *Komputerowa analiza i przetwarzanie obrazów.* Wydawnictwo Fundacji Postępu Telekomunikacji. ISBN 83-86476-15-X.

Oświadczenie

Oświadczam, że Pracę Dyplomową pod tytułem *"[TYTUŁ]"*, którą kierował dr inż. Jakub Koperwas, wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Michał Aniserowicz