Composite UI

Michał Padzik

padzikm@gmail.com

https://github.com/padzikm/CompositeUI

Założenia:

- Każdy serwis jest fizycznie i logicznie odpowiedzialny za swoje dane i operacje na nich
- Każdy serwis umie wyświetlić swoje dane w odpowiednim formacie
- Każdy serwis rozwijamy niezależnie

Wskazówki jak zacząć:

- Elementy na stronie to viewmodele, które oprócz zawierania danych umieją się same wyświetlić
- Tzw BrandingService odpowiedzialny tylko za stworzenie planszy dla elementów i umiejscowienie każdego z nich we właściwym miejscu
- Tzw IT/OpsService integrujący serwisy i zapewniający konfigurację

Potencjalne problemy integracji:

- Centralny punkt zależności
- Konieczna wiedza co, u kogo i kiedy wywołać
- Ewentualna potrzeba współpracy pomiędzy serwisami (np w kwestii wiązania danych, walidacji, nazewnictwa, etc)
- Zmiana w jednym serwisie może oznaczać zmiany w pozostałych
- Konflikty wstrzykiwanych zależności

Rozwiązanie krok 1:

- Nazwanie i zapisanie tego co ma się znajdować na danej stronie
- Mając nazwane elementy BrandingService wie, w którym miejscu umieścić dany viewmodel i może mu zlecić wyświetlenie się
- Viewmodele dostarcza serwis integracyjny – jak je zebrać?

Rozwiązanie krok 2:

- Elementy identyfikowane na stronie – HttpContext wie wszystko
- Zamiast wywoływać serwisy dla danej strony stwórzmy mechanizm mapujący żądania, czyli routing
- Routing jest jeden globalny, więc wprowadźmy wewnętrzny routing dla każdego serwisu jako kopię globalnego
- Jedyna zmiana – przestrzeń nazw na właściwą danemu serwisowi
- Tym sposobem zainteresowane serwisy same dostarczą modele
- Jak wyrenderować model skoro jego widoki i akcje są w oddzielnym routingu, niż jest on wywoływany?

Rozwiązanie krok 3:

- ViewModel musi umieć trafić do swoich zasobów – zarejestrujmy serwisy w globalnym routingu w unikalny sposób
- Każda ścieżka będzie zawierała klucz (np nazwę serwisu – unikalna), która będzie potrafiła znaleźć drogę tylko jeśli podamy ten klucz
- Trafiając z viewmodelu do widoku podaliśmy klucz, a ponieważ każde wywołanie głębiej dziedziczy dane, to automatycznie zostanie skopiowany nasz klucz
- Od widoku poruszamy się domyślnie w obrębie konkretnego serwisu
- Skąd wziąć fizyczny plik widoku?

Rozwiązanie krok 4:

- Dla każdego serwisu stwórzmy viewengine, który dzięki kluczowi w routingu wie czy pytanie jest adresowane do niego
- Wiele powtarzalnych implementacji różniących się wartością klucza – dodając odrobinę intuicyjnej konwencji niech T4 zrobi to za nas
- Komponenty wewnątrz serwisu chcą komunikować się między sobą np poprzez ajax – jak trafić bezpośrednio do adresata, bez zanieczyszczania głównej aplikacji?

Rozwiązanie krok 5:

- W globalnym routingu rejestrujemy serwisy jako areas z unikalnym przedrostkiem (np nazwą serwisu), a w głównej aplikacji zastrzegamy area o danym wzorze (np z przyrostkiem Service)

- Zyskujemy unikalne ścieżki przychodzące – dla kontrolerów w głównym folderze area: NazwaSerwisu/controller/action a dla dowolnej fizycznej area: NazwaSerwisu/area/controller/action

- Jak przesyłać dane od klienta i głównej aplikacji do serwisów?

Rozwiązanie krok 6:

- Ponownie HttpContext zna odpowiedź
- Każdy serwis wie co włożył, zatem także wie co i jak wyjąć
- Dzięki jednemu procesowi serwisy współdzielą sesję, ciasteczka, etc co pozwala na zostawianie sobie informacji (np główna aplikacja przydziela Id danemu zasobowi, a każdy serwis ma do niego dostęp)
- Co z transakcjami?

Rozwiązanie krok 7:

- Dzięki jednemu procesowi możemy objąć całość/część operacji transakcją/transakcjami
- Transakcja rozproszona, ale korzystając np z msmq komunikacja będzie tylko w obrębie lokalnej maszyny
- Co z kontenerami zależności dla serwisów?

Rozwiązanie krok 8:

- Graf zależności ma jeden wierzchołek początkowy – kontroler, który jest unikalny dla każdego serwisu
- Do głównego kontenera przekażmy zależnośći wymagane do sterowania aplikacją (np routing, viewengines, controleractivator), a dla każdego serwisu stwórzmy oddzielny kontener
- Ponieważ odpowiedzialności głównej aplikacji i serwisów są różne, to nie ma konfliktu pomiędzy poszczególnymi kontenerami

Rozwiązanie krok 9:

DEMO