Sprawozdanie

Programowanie w środowisku .NET

Dawid Gawiński

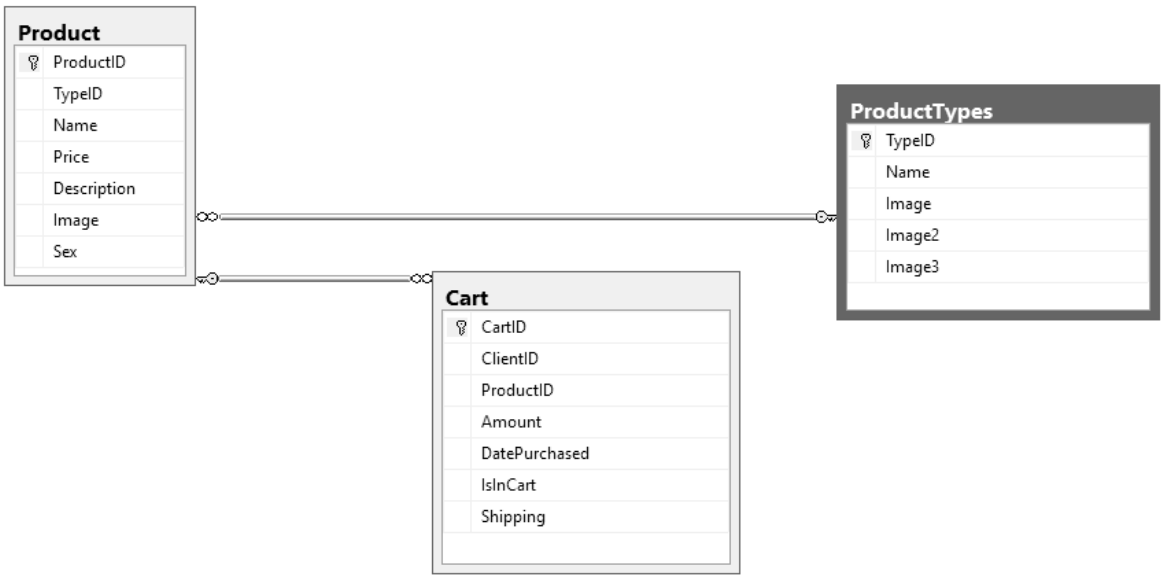
28/11/2016

Spis treści

1. Wstęp

Za zadanie do zrealizowania w trakcie zajęć projektowych było stworzenie dwóch aplikacji internetowych, jedną zrealizowaną za pomocą WebForms, a drugą za pomocą wzorca architektonicznego MVC. Temat projektu był sklep odzieżowy.
Oby dwie metodyki pozwoliły na napisanie atrakcyjnej strony internetowej, jednak aby wyłonić która technologia jest lepsza, zostanie przeprowadzona analiza porównawcza pod kryterium tworzenia nowego projektu, projektowania wyglądu strony, tworzenia bazy danych i jej połączenie z aplikacją, czasu obsługi stron przez serwer oraz refaktoryzacji.

1.1. Schemat użytej bazy danych



Rysunek 1 Schemat bazy danych

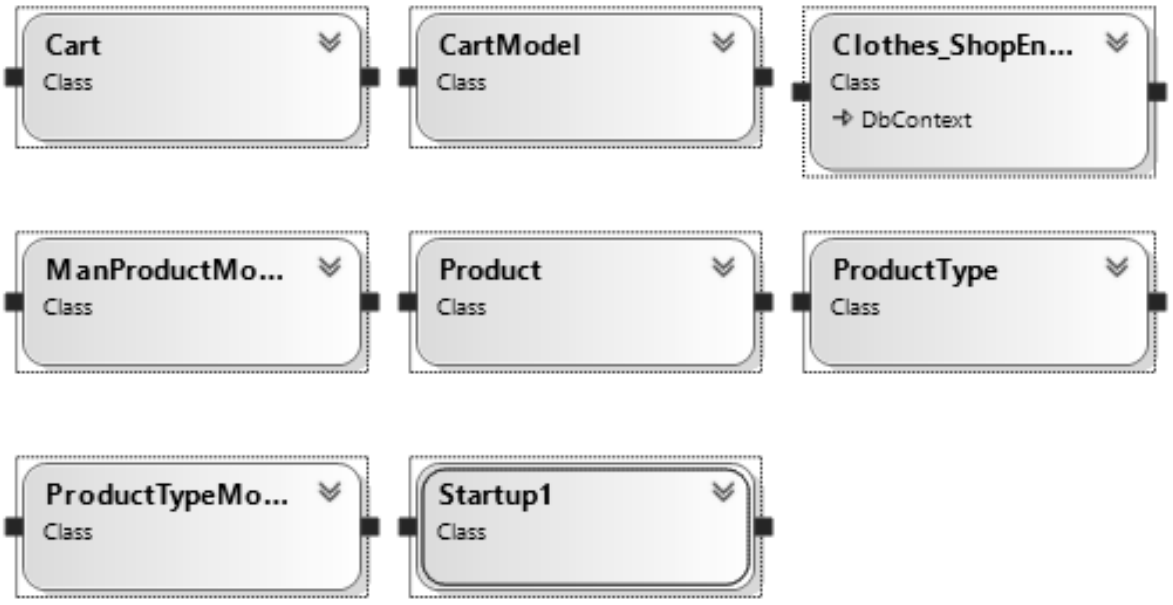
```
USE [Clothes_Shop]
GO
/****** Object:  Table [dbo].[Cart]    Script Date: 25.11.2016
13:31:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Cart](
	[CartID] [int] IDENTITY(1,1) NOT NULL,
	[ClientID] [varchar](50) NOT NULL,
	[ProductID] [int] NOT NULL,
	[Amount] [int] NOT NULL,
	[DatePurchased] [datetime] NULL CONSTRAINT
[DF_Cart_DatePurchased]  DEFAULT (getdate()),
	[IsInCart] [bit] NOT NULL,
	[Shipping] [int] NULL,
 CONSTRAINT [PK_Cart] PRIMARY KEY CLUSTERED
```

```
(
	[CartID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Product]    Script Date: 25.11.2016
13:31:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Product](
	[ProductID] [int] IDENTITY(1,1) NOT NULL,
	[TypeID] [int] NOT NULL,
	[Name] [varchar](100) NULL CONSTRAINT [DF_Product_Name]
DEFAULT ('Unknown'),
	[Price] [int] NULL,
	[Description] [text] NULL,
	[Image] [varchar](250) NULL,
	[Sex] [varchar](2) NULL,
 CONSTRAINT [PK_Product] PRIMARY KEY CLUSTERED
(
	[ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[ProductTypes]    Script Date:
25.11.2016 13:31:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ProductTypes](
	[TypeID] [int] IDENTITY(1,1) NOT NULL,
	[Name] [varchar](50) NOT NULL,
	[Image] [varchar](250) NULL,
	[Image2] [varchar](250) NULL,
	[Image3] [varchar](250) NULL,
 CONSTRAINT [PK_ProductTypes] PRIMARY KEY CLUSTERED
(
	[TypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[Cart]  WITH CHECK ADD  CONSTRAINT
[FK_Cart_Product] FOREIGN KEY([ProductID])
REFERENCES [dbo].[Product] ([ProductID])
GO
ALTER TABLE [dbo].[Cart] CHECK CONSTRAINT [FK_Cart_Product]
GO
ALTER TABLE [dbo].[Product]  WITH CHECK ADD  CONSTRAINT
[FK_Product_ProductTypes] FOREIGN KEY([TypeID])
REFERENCES [dbo].[ProductTypes] ([TypeID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Product] CHECK CONSTRAINT
[FK_Product_ProductTypes]
GO
```

Kod źródłowy SQL związanego z stworzeniem bazy danych

1.2. Diagram klas w WebForms a MVC



Rysunek 2 Diagram klas użytych w technologii WebForms

Rysunek 3 Diagram klas użytych w technologii MVC

2. Porównanie WebForms z MVC

2.1. Tworzenie nowego projektu, polucji

2.1.1. WebForms

W przypadku technologii WebForms do projektu dodaje się nowe pliki typu aspx. Te pliki można otworzyć w różnych widokach, m.in. wyglądu w formie formatki na którą można przeciągać kontrolki, kodu gdzie można samemu napisać dostępne kontrolki oraz widoku kodu, gdzie piszemy już prawidłowe zachowanie naszej aplikacji.

Według mnie różnorodność kontrolek jest bardzo wielką zaletą, gdyż do swojej strony możemy dodać praktycznie wszystko co chcemy, od pól tekstowych, tabel, list do różnego rodzaju kontrolek nawigacyjnych i okien związanych z logowaniem i rejestracją.

Przy przenoszeniu komponentów na formatkę spotkałem się z problemem ułożenia komponentów. Często wskakiwały mi one w miejsce, którego nie chciałam. Istniała możliwość korekty zachowania tych komponentów dzięki technologii bootstrap, jednak ten aspekt opiszę w kolejnym podpunkcie.

Czymś, co nie podobało mi się w WebForms, był fakt, że przy dodaniu nowych przycisków, list rozwijanych, od razu wymagane były metody dotyczących akcji tych kontrolek. Spowodowało to wiele pustego kodu, który nic nie wnosił do mojej aplikacji poza chaosem. Przy WebForms niemożliwe jest ustawienie przekierowania strony za pomocą kontrolek z poziomu widoku.

Według mnie trochę to utrudnia wdrożenie takiej aplikacji, gdyż należy napisać kolejne metody z poziomu kodu, co wpływa negatywnie na jego czytelność.
Przy korzystaniu z wersji Visual Studio do roku 2012 bardzo łatwe i przyjemne jest narzędzie WebAdministration Tool. Umożliwia ono zarządzanie użytkownikami, grupami oraz ich uprawnieniami. Przydało mi się to bardzo w momencie, kiedy musiałem zabronić niektóre strony dla niezalogowanych użytkowników. Zamiast tworzyć dużej ilości warunków wystarczyło, że utworzyłem regułę zabraniającą dostępu niezalogowanym użytkownikom do wybranych zasobów mojej strony i WebAdministation Tool zadbał o jej weryfikację. Dzięki temu narzędziu bardzo łatwe jest również proces uwierzytelniania użytkowników, tworzenia nowych kont, gdyż ta technologia sama tworzy potrzebne tabele i nimi zarządza.

2.1.2. MVC

W trakcie realizacji swojej aplikacji korzystałem z MVC w wersji 5.
Charakteryzuje się ono bardzo estetycznym, czytelnym szablonem oraz możliwością korzystania z nowych funkcjonalności silnika Razor. W przypadku tej technologii jest trochę odmienne postępowanie niż w WebForms, gdzie pierw zaczyna się od tworzenia interfejsu graficznego. W MVC po dodaniu modelu bazy danych, tworzymy kontroler dla danej tabeli lub stworzonego przez nas modelu zawierającego elementy z różnych tabel i opisujemy jego zachowanie. W swoim wypadku zaczęłem od pobrania obiektów z bazy danych, co zajęło mi w moim projekcie jedną linijkę kodu. Następnie jednym kliknięciem zostaje wygenerowany widok z tabelą. Jest to bardzo wygodne i przyjemne rozwiązanie.
Przy MVC nie występuje wygląd formatki, jednak jest pasek narzędziowy zawierający kilka komponentów takich jak pola tekstowe czy przyciski, jednak ich używanie nie jest zbyt popularne.
Generowanie widoków edytowania, tworzenia nowego obiektu lub jego usuwania również jest bardzo proste.
MVC umożliwia przyjemną opcję dodania przejścia do innego widoku dzięki kontrolkom od razu z widoku,
bez deklaracji tego w kodzie, co nie powoduje nam nieczytelnego kodu jak w przypadku WebForms.
Moduł rejestracji i logowania zostaje od razu utworzony przy generacji projektu. Nie musimy się martwić o tworzenie encji na przechowywanie na użytkowników. Te tabele jednak nie są tworzone w naszym projekcie bazy, a prawdopodobnie na innej instancji bazy. Niemożliwe jest więc pobranie użytkowników odwołując się do tabeli użytkowników, ale można korzystając z HttpContext pobrać aktualnie zalogowanego użytkownika. Zablokowanie dostępu użytkownikom do pewnych zasobów nie jest nam jednak ułatwiane jak w przypadku WebAdministation Tool i dlatego należy się sporo namęczyć przy zabronieniu dostępu do jakiegoś obszaru niezalogowanym użytkownikom.
Nowy system routingu - pozwala on tworzyć proste i przyjemne adresy URL dla naszej strony, które są bardzo istotne w procesie pozycjonowania naszej strony. Dzięki temu systemowi możemy zawrzeć słowa kluczowe w URL.

2.2. Projektowanie wyglądu strony

2.2.1. WebForms

Przy tworzeniu pustego projektu WebForms nie zostają dołączone pliki bootstrapu i należy dodać je ręcznie do projektu, a następnie do każdego wyglądu strony osobno. Komponenty przeciągane na kontrolkę nie zawierają odniesienia do boostrapu i należy dodać je ręcznie.

2.2.2. MVC

W przypadku projektu realizowanego w technologii MVC 5 pliki bootstrapowe są dołączane od razu do ścieżki projektu, a także do każdego widoku.
Widoki generowane z poziomu kontrolera są bardzo przyjemne dla oka, komponenty pochodzą z komponentów bootstrapowych. Bardzo łatwa jest ich zmiana estetyczna, gdy np. chcemy, żeby nasza tabela była w paski, to wystarczy tylko abyśmy dopisali „striped" do widoku tabeli.
Według mnie w MVC znacznie łatwiejsze jest podzielenie strony na master page i content page, gdyż te pliki są już wygenerowane. Wystarczy zmienić treść stopki, ewentualnie kolor paska nawigacyjnego i nasza strona jest gotowa.

2.3. Tworzenie bazy danych i jej połączenie z aplikacją

2.3.1. WebForms

W przypadku WebForms bardzo łatwe jest dołączenie do kontrolek takich jak ListView, DropDownList, TableView danych z bazy danych. Przeprowadza nas przez ten proces kreator, w którym nawet możliwe jest dodanie do zapytania SQL parametru WHERE, który pochodzi z innej kontrolki np. DropDownList. Według mnie jest to bardzo przydatne i użyłem tego w swoim projekcie.
Walidacja danych jest gotowa przy użyciu kontrolek LoginView lub CreateNewUser. Z poziomu właściwości tych kontrolek możliwa jest zmiana komunikatów błędów lub ustawienia innych kryteriów walidacji.
Walidacja w przypadku naszych własnych okienek dodawania nowych elementów do bazy musi być przeprowadzona przez nas, nic nie zostanie wygenerowane.

2.3.2. MVC

Według mnie lepszą techniką jest używanie obiektowej bazy danych. Używając MVC entity framework jest bardzo łatwy do obsługi. Modele są gotowe wygenerowane i wystarczy dodać do nich kontroler i widok. Co prawda dodawanie źródła bazy danych do konkretnych komponentów, jeśli chcielibyśmy to robić ręcznie, jest trudniejsze niż w WebForms, ale nie należy do skomplikowanych.
Gdy generujemy widoki edycji bądź tworzenia obiektów z bazy danych, nie musimy się martwić o walidację danych, gdyż została już ona zaimplementowana przy generowaniu widoku.
Walidacja danych przy tworzeniu nowego użytkownika również została wygenerowana przy tworzeniu projektu i mamy pewność, że nasi użytkownicy mają niepowtarzalne loginy i bezpieczne hasła.

2.4. Czas obsługi stron przez serwer

2.4.1. WebForms

ViewState - mechanizm do przekazywania stanów kontrolek pomiędzy requestami wysyłanymi przez stronę spowodował że podczas przeładowywania pomiędzy serwerem i klientem przesyłane były olbrzymie porcje danych. Konsekwencją tego było znaczne wydłużenie czasu ładowania strony, oraz zmniejszenie szybkości jej działania.

2.4.2. MVC

Brak ViewState - dzięki bezstanowej naturze ASP
.Net MVC strony tworzone w tej technologii mogą być nawet kilkaset kilobajtów mniejsze niż strony stworzone za pomocą WebForms. Dzięki czemu działanie aplikacji jest o wiele szybsze.

2.5. Refaktoryzacja, czyli zmiany w kodzie źródłowym

2.5.1. WebForms

Kod napisany przy pomocy WebForms według mnie nie jest łatwy do refaktoryzacji. Osoba, która po raz pierwszy zobaczy taki kod, musi włożyć sporo wysiłku, aby go zrozumieć. Duża część przypisania wartości do kontrolek typu TableView, DropDownList ma miejsce w widoku. Kod widoku jest bardzo nieczytelny i ciężko w nim cokolwiek znaleźć. Wydaje mi się, że gdyby kod odpowiadający za przypisanie kontrolek do danych byłby w klasie, to kod wydałby się znacznie bardziej czytelny.
Jako że nie ma podziału na kontrolery, modele, to nad kodem nie może pracować jednocześnie kilka osób. Nie ma tu podzielenia projektu na modułowość.
Przede wszystkich WebForms nie jest najłatwiejszą technologią do testowania, co jest bardzo ważne w przypadku, gdyby strona internetowa miała zostać wdrożona do Internetu.

2.5.2. MVC

Jako że zachowana jest architektura MVC, czyli podziału na widok, kontroler oraz model, kod jest bardzo czytelny. Nad kodem może pracować wiele osób, każda osoba nad innym kontrolerem i takie postępowania sprawi, że nikt nikomu nie będzie przeszkadzał.
W kontrolerze znajduje się najpotrzebniejszy kod, który od razu wysyła dane do widoku przez co wiadomo, czego dotyczy konkretny widok.
Jako że architektura MVC jest ogólno przyjęta, to każda inna osoba niż ta co napisała kod, będzie w stanie go zrozumieć.
Aplikacje tak napisane bardzo łatwo jest testować ze względu na ich architekturę.

3. Podsumowanie

Według mnie pod względem łatwości wdrożenia lepsze jest Mvc, gdyż generowane widoki są bardzo przejrzyste, zawierają całą funkcjonalność i nie należy wiele w nich zmieniać.
Pod względem wyglądu oraz łatwości dołączenia bootstrapa, który w dzisiejszych czasach jest podstawą, również przewagę mają strony napisane w MVC.
Pod względem szybkości działania oraz łatwości walidacji bezkonkurencyjnie również wygrywa MVC . Projekty napisane w MVC same generują nam walidację, co jest wielkim ułatwieniem dla programisty.
Jeśli chodzi o bazę danych, to oby dwa sposoby wydają mi się w porządku pod względem łatwości implementacji.
Pod względem łatwości refaktoryzacji zdecydowanie wygrywa MVC.
Coś, co mnie bardzo urzekło w tradycyjnych stronach ASP.NET, to narzędzie WebAdministration Tool, które pozwala na bardzo wiele możliwości i na pewno ułatwia życie administratorom stron internetowych w zarządzaniu użytkownikami.
Jeśli zamierzamy zbudować aplikację webową pracującą w Internecie, lub dużą aplikacje intranetową, w której przede wszystkim liczy się szybkość działania, niezawodność, kompatybilność z większą ilością przeglądarek oraz wysoka jakość wykonania to wybór powinien paść na MVC.
W przypadku jednak gdy mamy wykonać szybko aplikację internetową, w której nie będziemy przejmować się szybkością działania, pozycjonowaniem oraz testami lepiej posłużyć jest się zwykłym ASP.Net. Myślę, że ten sposób tworzenia aplikacji będzie bardzo dobry dla prototypów, gdy chcemy stworzyć szkielet strony internetowej, aby móc go pokazać swojemu klientowi.