MACIEJ WĘGRZYNOWSKI
Wydział Elektroniki i Informatyki Politechniki Koszalińskiej
Kierunek Informatyka, II rok studiów niestacjonarnych drugiego stopnia
Przedmiot PROGRAMOWANIE W ŚRODOWISKU .NET

WYKORZYSTANIE WZORCA MVC W ASP.NET

praktyczne ćwiczenie

ZAWARTOŚĆ ĆWICZENIA

- Przygotowanie środowiska do pracy z ASP.NET MVC
- Aplikacja bazodanowa z wykorzystaniem wzorca MVC w ASP.NET
- Nowy projekt
- Nowa baza danych
- Dodajemy model
- Dodajemy kontroler
- Listowanie rekordów z bazy danych
- Dodajemy widok
- Dodawanie rekordu
- Edycja rekordu
- Usuwanie rekordu
- Podgląd rekordu
- Modyfikacja szablonu
- Odinstalowanie Microsoft Visual Web Developer 2010 Express

Przygotowanie środowiska do pracy z ASP.NET MVC

Do pracy z ASP.NET MVC w wersji 3 będzie potrzebne Microsoft Visual Web Developer 2010 Express, .NET Framework 4.0, framework ASP.NET MVC 3 i dowolna przeglądarka internetowa. Serwer IIS nie jest wymagany do testowania aplikacji, ponieważ Visual Studio posiada wbudowany serwer (SQL Server Express 2008), na którym uruchamiane są aplikacje. Narzędzia te są w wersji bezpłatnej do zastosowania niekomercyjnego.

Microsoft przygotował dla osób chcących poznać środowisko .NET przydatne narzędzie – Microsoft Web Platform Installer[1], z poziomu którego można zainstalować wszystko co niezbędne do rozpoczęcia pracy z ASP.NET MVC. Do napisania niniejszej aplikacji bazodanowej wystarczą następujące składniki z Web Platform Installer:

- Visual Web Developer 2010 Express
- Microsoft .NET Framework 4
- ASP.NET MVC 3
- SQL Server Express 2008

Instalacja w/w składników trwa dość długo i trzeba uzbroić się w cierpliwość.

Aplikacja bazodanowa z wykorzystaniem wzorca MVC w ASP.NET

Aplikacja, którą stworzymy będzie zawierała uproszczoną bazę filmów wraz z możliwością jej rozbudowania, dodawania i edytowania rekordów. Taka aplikacja świetnie zapozna Cię ze wzorcem MVC w ASP.NET. Opiera się na materiałach zawartych na stronie http://www.asp.net/mvc.

Aby ułatwić budowanie aplikacji, skorzystamy z gotowych funkcji Visual Studio 2010. Pozwolimy mu wygenerować początkowy kod wraz z treścią dla naszych kontrolerów, modeli i widoków. Na tak utworzonych „fundamentach" rozbudujemy projekt.

Jeżeli programowałeś w ASP lub ASP.NET, to „przesiadka" na ASP.NET MVC nie sprawi Ci najmniejszego problemu. Podglądy w ASP.NET MVC są bardzo podobne do stron ASP. I tak jak tradycyjne ASP.NET Web Forms, ASP.NET MVC zapewnia dostęp do bogatego zestawu języków i klas przedstawionych przez .NET.

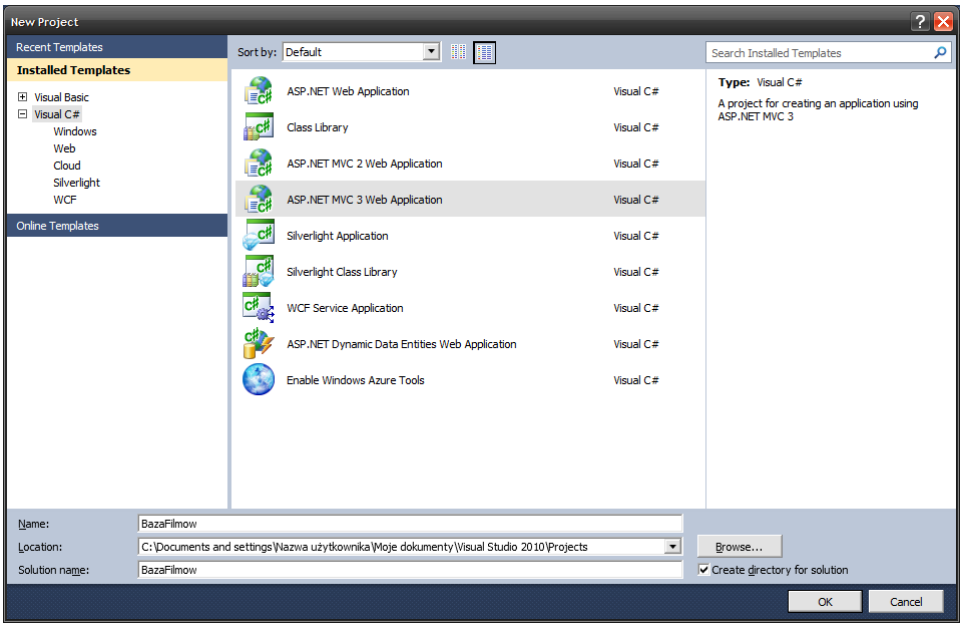
Zbudujemy bazę danych filmów, która będzie miała możliwość:

- wyświetlania filmów
- dodawania
- usuwania
- edytowania

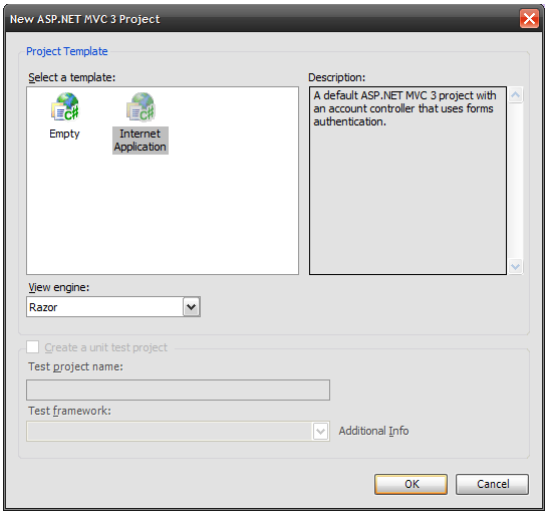
[1] Microsoft Web Platform Installer – http://www.microsoft.com/web/downloads/platform.aspx

Nowy projekt

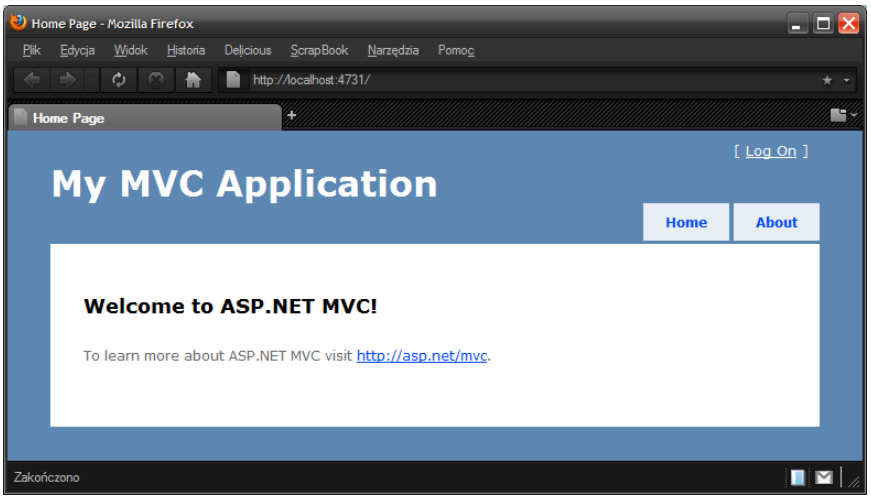
Po uruchomieniu środowiska Microsoft Visual Studio 2010 należy wybrać z menu **FILE › NEW PROJECT** a następnie **ASP.NET MVC 3 WEB APPLICATION** (piszemy w języku VISUAL C#), a nasz projekt nazywamy *BazaFilmow*.



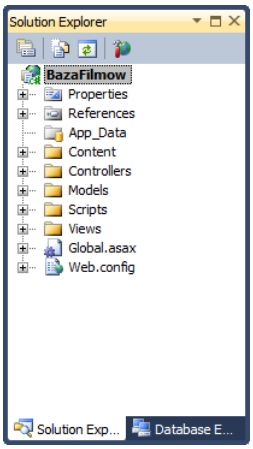
Następnie wybieramy **INTERNET APPLICATION (**opartego na silniku Razor).



I taka przykładowa aplikacja zawarta w pakiecie Microsoft Visual Studio 2010 jest gotowa do uruchomienia. Uruchamiamy wciskając klawisz **F5** lub zieloną strzałkę na pasku narzędzi (bądź z menu **DEBUG › START DEBUGGING**). Nasz projekt jest wówczas kompilowany i uruchomia lokalny serwer ASP.NET Development Server, który jest wbudowany w Visual Web Developer. Zgłoszenie uruchomienia serwera pojawi się w lewym dolnym rogu ekranu i będzie zawierać numer portu, pod którym uruchomiona jest aplikacja, np. http://localhost:1234 (numer portu będzie inny). W następnej kolejności Visual Web Developer automatycznie uruchomi naszą domyślną przeglądarkę internetową wraz z adresem wskazującym na lokalny serwer projektowanej aplikacji. Umożliwia to nam szybkie i wygodne przetestowanie aplikacji na naszym komputerze.



Wyłączamy debugowanie skrótem klawiszy **Shift+F5** bądź wyborem menu **DEBUG › STOP DEBUGGING**.



Utworzona w nowym projekcie struktura katalogów wygląda następująco:

`/Controllers`	Kontrolery
`/Views`	Szablony widoków
`/Models`	Modele przetwarzania danych
`/Content`	Zdjęcia, CSS, wszelkie inne treści statyczne
`/Scripts`	skrypty JavaScript
`/App_data`	pliki bazy danych

Katalogi te są zawsze tworzone, również w pustej aplikacji, ponieważ ASP.NET MVC z góry narzuca strukturę katalogów zgodną ze wzorcem MVC. Więc kontrolery mamy w katalogu CONTROLLERS, widoki w katalogu VIEWS, modele w katalogu MODELS itp.

Ponieważ projekt został wygenerowany jako przykładowa aplikacja, usuwamy plik i katalog:

- Controllers\HomeController.cs
- Views\Home

I zaczynamy…

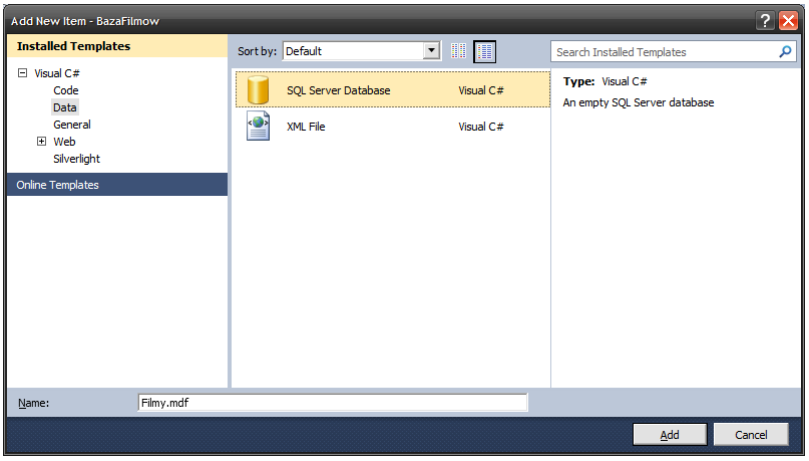
Nowa baza danych

Musimy stworzyć miejsce do przechowywania danych filmów. Skorzystamy z bezpłatnej bazy danych SQL Server Express. Aby utworzyć nową bazę danych:

1. Klikamy prawym przyciskiem myszy (PPM) na folder APP_DATA w oknie Solution Explorer i wybieramy opcję **ADD › NEW ITEM**.

2. Wybieramy kategorię **DATA** i zaznaczamy **SQL SERVER DATABASE**.

3. Nazywamy naszą bazę danych np. **Filmy.dbf** i klikamy **ADD**.

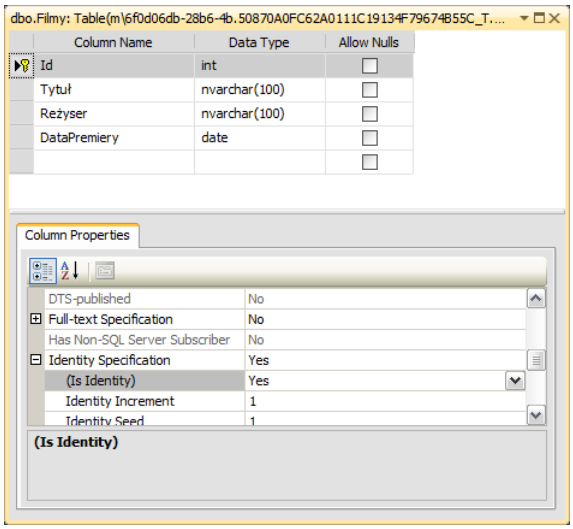


Aby połączyć się z nowoutworzoną bazą danych, klikamy dwukrotnie na plik **Filmy.dbf** znajdujący się w folderze **APP_DATA**. Otworzymy w ten sposób Database Explorer.

Następnie musimy utworzyć nową tabelę w bazie danych. Z poziomu okna Database Explorer, klikamy PPM katalog TABLES i wybieramy z menu opcję **ADD NEW TABLE**. Wybranie tej opcji otwiera okno do projektowania tabeli. Utworzymy poniższe kolumny danych:

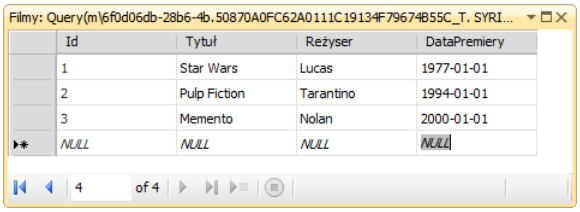
Nazwa kolumny	Typ danych	Zezwalaj na wartości Null
Id	Int	No
Tytuł	Nvarchar(100)	No
Reżyser	Nvarchar(100)	No
DataPremiery	DateTime	No

Następnie kolumnę **ID** zaznaczamy jako klucz pierwotny, klikając przycisk **SET PRIMARY KEY** (ikonka klucza) oraz we właściwościach kolumny (Column Properties) rozwijamy **IDENTITY SPECIFICATION**, a właściwości **IS IDENTITY** nadajemy wartość **YES**.



Zapisujemy nową tabelę jako **Filmy**, klikając przycisk **SAVE** (ikona dyskietki).

Teraz dodajemy kilka rekordów do tabeli **Filmy**. Klikamy **PPM** na tabelę **Filmy** w oknie Database Explorera i wybieramy z menu opcję **SHOW TABLE DATA**. I tam wprowadzamy listę filmów:



Dodajemy model

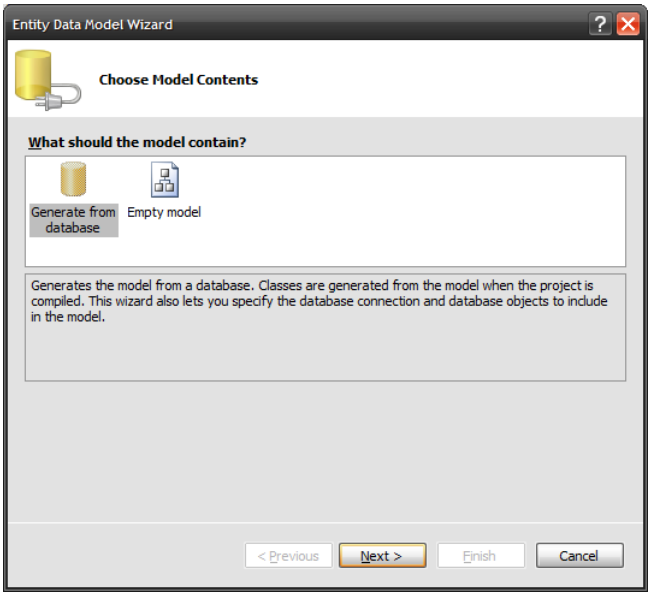
Następnie utworzymy zestaw klas do reprezentowania naszej bazy danych. Musimy stworzyć model bazy danych. Skorzystamy z Microsoft Entity Framework do automatycznego tworzenia klas dla naszego modelu bazy danych. Uruchamiamy Entity Data Model Wizard:

1. Klikamy **PPM** na folder MODELS w oknie Solution Explorer i wybieramy opcję z menu **ADD › NEW ITEM**.

2. Wybieramy kategorię **DATA** i szablon **ADO.NET ENTITY DATA MODEL**.

3. Nazywamy nasz model danych **FilmyModel.edmx** i klikamy **ADD**.

Po kliknięciu przycisku **ADD**, pojawi się kreator **Entity Data Model**.



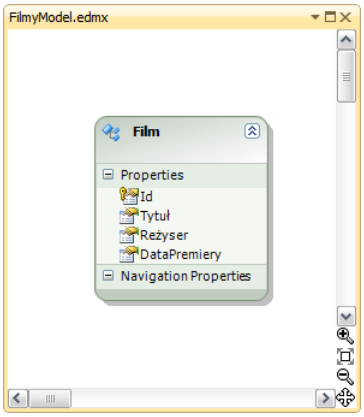
Wykonamy poniższe kroki:

1. W kroku **Choose Model Contents** wybieramy opcję **GENERATE FROM DATABASE** i klikamy przycisk **NEXT**.

2. W następny kroku **Choose Your Data Connection** określamy połączenie danych. Użyjemy **Filmy.mdf** do połączenia danych i nazwijmy je *FilmyEntities***.** Klikamy przycisk **NEXT**.

3. W kroku **Choose Your Database Objects** rozwijamy węzeł *Tables* i wybieramy tabelę *Filmy*. Wpisujemy nazwę **Filmy.Model** i klikamy przycisk **FINISH**.

Po zakończeniu kreatora otwiera się **Entity Data Model Designer**.



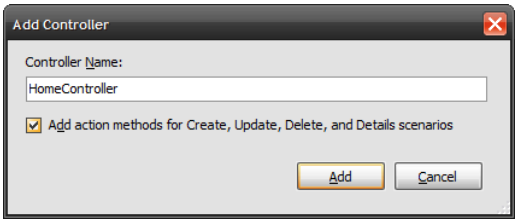
Zmieniamy nazwę klasy z *Filmy* na *Film*. Po dokonaniu tej zmiany klikamy przycisk **SAVE** (ikona dyskietki).

Dodajemy kontroler

Aby powołać naszą aplikację do życia, musimy stworzyć co najmniej jeden kontroler.

1. Klikamy **PPM** (prawym przyciskiem myszy) na katalogu CONTROLLERS i wybieramy z menu kontekstowego opcję **ADD › CONTROLLER**.

2. Nazywamy nowy kontroler *HomeController* oraz zaznaczamy checkbox **Add action methods for Create, Update, and Details scenarios**.



Po zaakceptowaniu utworzona zostanie nowa klasa reprezentująca kontroler, zawierająca metody tj. *Index, Details, Create, Delete* i *Edit*. W dalszej części dodamy kod niezbędny do użycia tych metod. Utworzony kompletny kod klasy znajduje się w pliku **HomeController.cs** w katalogu CONTROLLERS.

Dlaczego kontroler musi nazywać się *Home*, a akcja *Index*? Wynika to z domyślnych ustawień ASP.NET MVC, które musi wiedzieć jaką akcję i z jakiego kontrolera wywołać w momencie wejścia na stronę bez określania tych parametrów w adresie. Domyślna nazwa kontrolera i akcji zawarta jest w pliku **Global.asax**, którego jednym z zastosowań jest definiowanie tras dla routera.

Listowanie rekordów z bazy danych

Metoda *Index()* kontrolera jest metodą domyślną dla ASP.NET MVC. Po uruchomieniu aplikacji wywoływana jest właśnie ta metoda, jako domyślna na stronie głównej. Użyjemy tej metody, aby domyślnie listować zawartość filmów z bazy danych. Skorzystamy z klasy modelu bazy danych, którą wcześniej utworzyliśmy.

Do pliku **HomeController.cs** dołączamy bibliotekę naszych modeli, aby można było skorzystać z zawartej w niej klasy *FilmyEntities*.

```
using BazaFilmow.Models;
```

Zmodyfikujemy metodę *Index()*, aby używała metody klasy *FilmyEntities* do pobrania wszystkich rekordów z tabeli *Filmy*.

```
        private FilmyEntities _db = new FilmyEntities();

        //
        // GET: /Home/

        public ActionResult Index()
        {
            return View(_db.Filmy.ToList());
        }
```

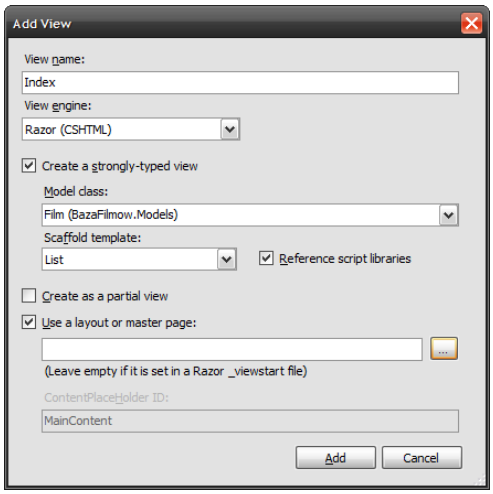
Wyrażenie *_db.MovieSet.ToList* zwraca wszystkie rekordy z bazy danych. Oczywiście, uruchomienie aplikacji spowoduje błędy, gdyż aplikacja nie ma określonego widoku. Przed dodaniem widoku trzeba „zbudować" projekt wybierając z menu **DEBUG › BUILD BAZAFILMOW**.

Dodajemy widok

Dodaliśmy kontroler i model. Przejdziemy teraz na poziom wyżej i dodamy widok, czyli „ubierzemy" nasze wyniki w szablony HTML.

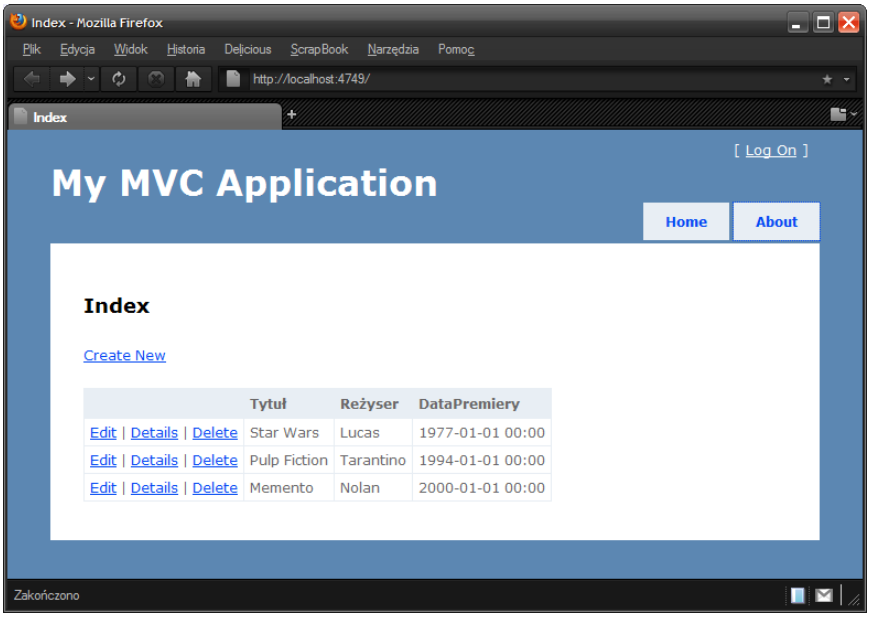
Czynność ta jest równie prosta jak tworzenie kontrolera i sprowadza się do kliknięcia **PPM** na nazwie metody akcji *Index()* i wyborze z menu kontekstowego opcji **ADD VIEW**. Wyświetlone zostanie okienko dialogowe dodania widoku.

Zaznaczamy ptaszkiem pole typu checkbox **Create a strongly-typed view**, natomiast z listy rozwijanej **Scaffold template** wybieramy opcję *List*.



W ten sposób możemy łatwo i szybko dodawać szablony widoku do odpowiednich metod akcji. Po kliknięciu **ADD** zostanie utworzony nowy szablon **Index.cshtml** w katalogu \VIEWS\HOME (katalog zostanie utworzony, jeśli nie istnieje). Widzimy, że tutaj narzucona jest struktura MVC – katalog HOME odpowiada kontrolerowi *HomeController*, natomiast nazwa szablonu jest taka sama jak metody. I mamy porządek w strukturze katalogów.

Uruchamiamy aplikację i widzimy, że szablon głównej strony pokazuje wszystkie rekordy z bazy Filmy w postaci tabeli HTML. Widok zawiera pętlę *foreach*, która przechodzi przez każdy film zawarty we właściwościach *ViewData.Model*.



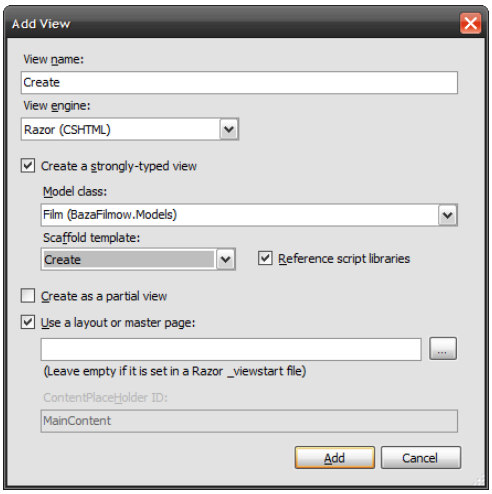
Zrobiliśmy szybko pierwszą, prostą i łatwą aplikację bazodanową w ASP.NET z wykorzystaniem wzorca MVC. Teraz ją rozbudujemy i dokończymy.

Dodawanie rekordu

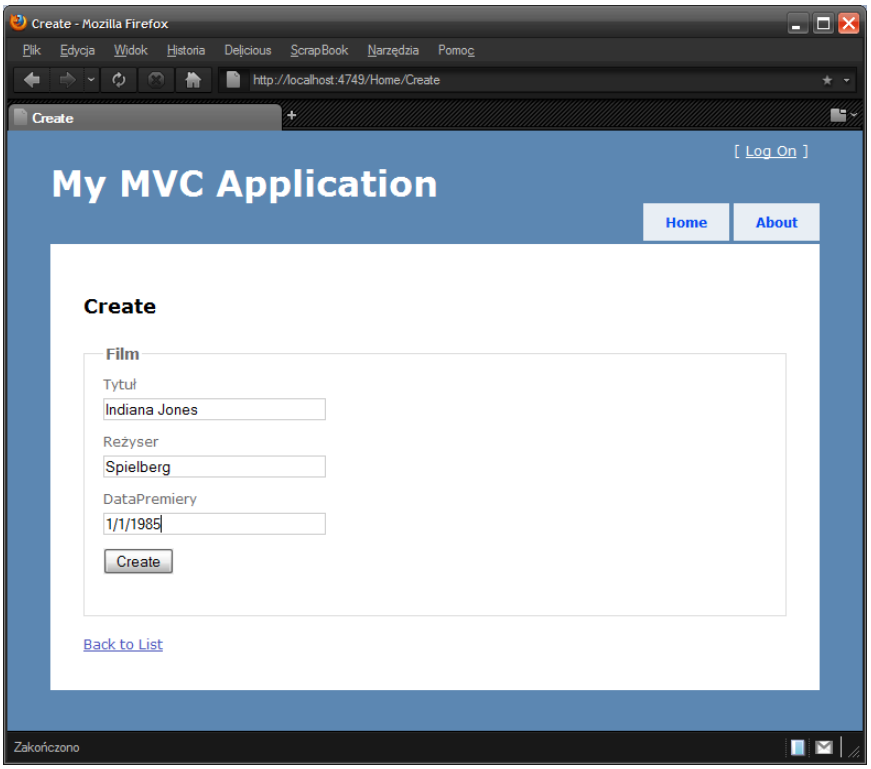
Modyfikujemy metodę *Create()* w pliku **HomeController.cs** według poniższego listingu.

```
        //
        // GET: /Home/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Home/Create

        [HttpPost]
        public ActionResult Create(Film newMovie)
        {
            if (ModelState.IsValid)
            {
                _db.Filmy.AddObject(newMovie);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            else
            {
                return View(newMovie);
            }
        }
```

I dla tej metody tworzymy widok, tak jak w przypadku metody *Index()*, tyle że z rozwijanej listy **Scaffold template** wybieramy opcję *Create*.



Utworzony został szablon **Create.cshtml** w katalogu VIEWS. Uruchamiamy aplikację skrótem **F5** i na stronie głównej klikamy w odnośnik Create New. Następnie wypełniamy odpowiednie pola i klikamy Create.



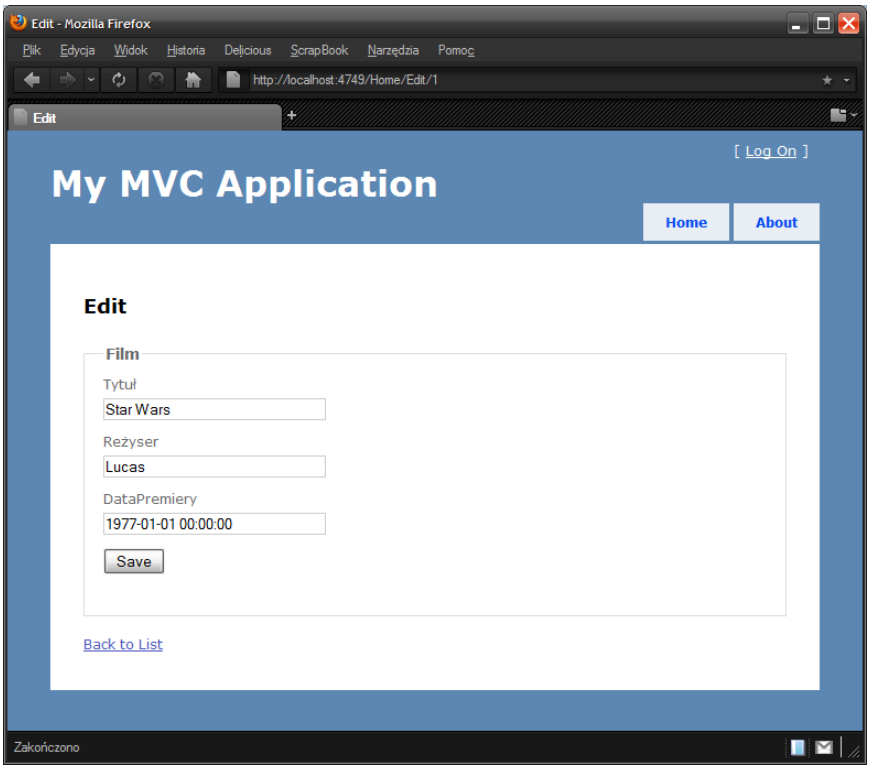
W efekcie dodamy nowy rekord do bazy danych.

Edycja rekordu

Implementujemy metodę *Edit()* według poniższego listingu.

```
        public ActionResult Edit(int id)
        {
            var movieToEdit = (from Film in _db.Filmy where Film.Id == id select Film).First();

            return View(movieToEdit);
        }

        //
        // POST: /Home/Edit/5
        [HttpPost]
        public ActionResult Edit(Film movieToEdit)
        {
            var originalMovie = (from Film in _db.Filmy where Film.Id == movieToEdit.Id select
Film).First();

            if (!ModelState.IsValid)
                return View(originalMovie);
            _db.ApplyCurrentValues(originalMovie.EntityKey.EntitySetName, movieToEdit);
            _db.SaveChanges();

            return RedirectToAction("Index");
        }
```

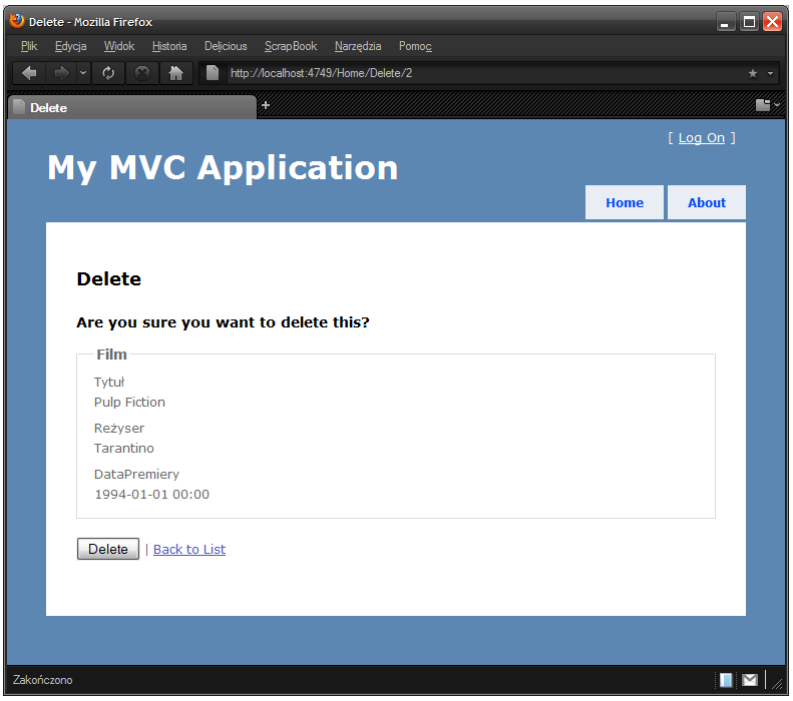
W następnej kolejności dodajemy widok do metody *Edit()* i mamy działającą opcję edycji istniejącego rekordu w bazie danych.



Usuwanie rekordu

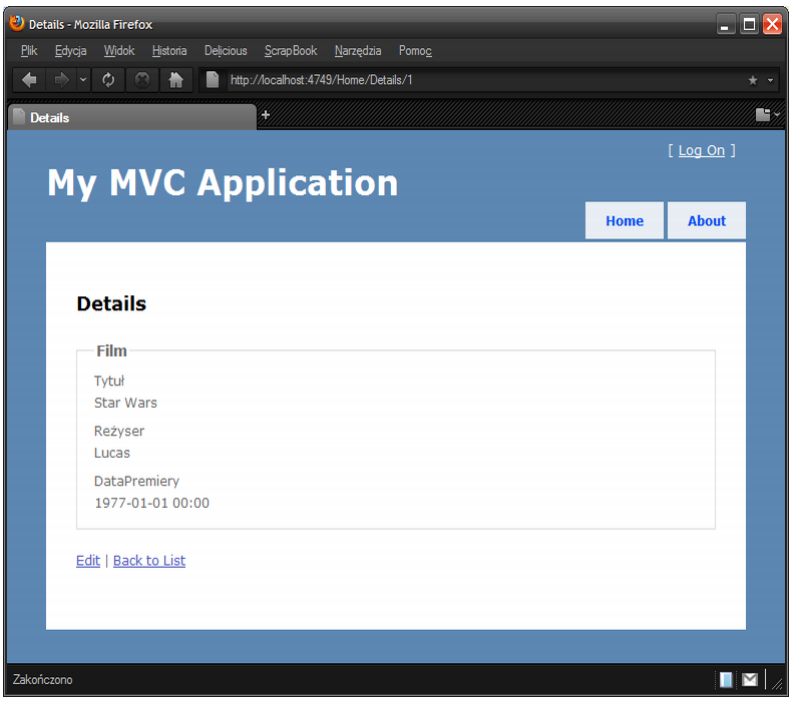
Modyfikacja metody akcji *Delete()*.

```
        //
        // GET: /Home/Delete/5

        public ActionResult Delete(int id)
        {
            var movieToDelete = (from Film in _db.Filmy where Film.Id == id select Film).First();

            return View(movieToDelete);
        }

        //
        // POST: /Home/Delete/5

        [HttpPost]
        public ActionResult Delete(Models.Film movieToDelete)
        {
            var SelMovie = (from Film in _db.Filmy where Film.Id == movieToDelete.Id select
Film).First();

            if (!ModelState.IsValid)
                return View(SelMovie);

            _db.DeleteObject(SelMovie);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
```



Podgląd rekordu

```
        //
        // GET: /Home/Details/5

        public ActionResult Details(int id)
        {
            var movieToDetails = (from Film in _db.Filmy where Film.Id == id select Film).First();

            return View(movieToDetails);
        }
```



Modyfikacja szablonu

W aplikacjach internetowych istnieją wspólne elementy na wielu stronach tj. nawigacja, nagłówek, stopka, logo, źródła do arkuszy styli itp. Silnik Razor zawarty w ASP.NET MVC pozwala nam łatwo zarządzać takimi elementami, przy tworzeniu nowego widoku automatycznie zostaje tworzony plik **_Layout.cshtml** w katalogu /VIEWS/SHARED.

Kaskadowe arkusze styli zawarte są w pliku **Site.css** w katalogu CONTENT. Tam też można sobie utworzyć katalog IMAGES i umieszczać grafikę potrzebną do szablonów.

Modyfikujemy **_Layout.cshtml**.

```
            <div id="title">
                <h1>Baza filmów</h1>
            </div>
```

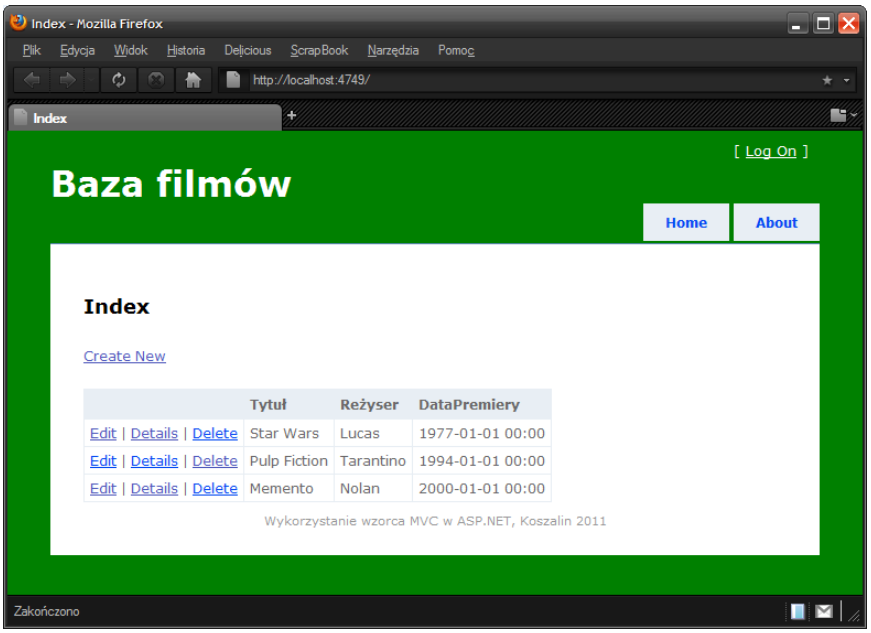
Dodamy też stopkę.

```
            <div id="footer">
            Wykorzystanie wzorca MVC w ASP.NET, Koszalin 2011
            </div>
```

Modyfikujemy **Site.css** (styl odpowiedzialny za body).

```
background-color: green;
```

I widzimy efekt.



Podsumowanie

Ćwiczenie miało za zadanie zapoznać Cię ze wzorcem MVC w ASP.NET. Nauczyliśmy się budować prostą aplikację ASP.NET wykorzystując model bazy danych, widok oraz kontroler. Pominięte zostały testowanie aplikacji Driven Test-Development oraz rozbudowa routingu. Dalsza rozbudowa aplikacji zależy już od Ciebie.

Odinstalowanie Microsoft Visual Web Developer 2010 Express

Kompleksowa instalacja Microsoft Visual Web Developer 2010 Express wraz z wymaganymi narzędziami do pracy (tj. MS SQL Server Express 2008 i .NET Framework 4.0) zawiera dużą ilość składników w systemie i ręczne odinstalowanie ich w **DODAJ/USUŃ PROGRAMY** może być frustrujące. Aby odinstalować narzędzia, za pomocą którego stworzyliśmy aplikację ASP.NET MVC (w celu zainstalowania nowszej wersji lub kompleksowego usunięcia środowiska) można skorzystać pomocniczego narzędzia Microsoft o nazwie Visual Studio 2010 Uninstall Utility[2]. Za pośrednictwem tego narzędzia możemy dokonać trzech typów odinstalowania (polecam opcję full zostawiając przydatne środowisko .NET Framework):

1. DEFAULT

   Usuwa wszystkie najnowsze narzędzia w wersji 2010 wraz ze wszystkimi składnikami. Nie usuwa starszych wersji Visual Studio (tj. Visual Studio 2008) lub środowiska Microsoft .NET Framework 4.0.

   Uruchamiamy program bez parametrów: `VS2010_Uninstall-RTM.ENU.exe`

2. FULL

   Usuwa Visual Studio 2010 ze składnikami oraz wcześniejsze wersje Visual Studio. Opcja ta nie usuwa Microsoft .NET Framework 4.0

   Uruchamiamy program z parametrem: `VS2010_Uninstall-RTM.ENU.exe /full`

3. COMPLETE

   Kompleksowe usunięcie całego zestawu Visual Studio 2010 ze składnikami, wcześniejsze wersje Visual Studio oraz Microsoft .NET Framework 4.0, zostawiając czysty system.

   Odpalamy z parametrem: `VS2010_Uninstall-RTM.ENU.exe /full /netfx`

[2] Microsoft Visual Studio 2010 Uninstall Utility – http://code.msdn.microsoft.com/vs2010uninstall