

ASP.NET MVC 5, Bootstrap i Knockout.js

TWORZENIE DYNAMICZNYCH I ELASTYCZNYCH
APLIKACJI INTERNETOWYCH

Tytuł oryginału: ASP.NET MVC 5 with Bootstrap and Knockout.js

Tłumaczenie: Piotr Pilch

ISBN: 978-83-283-2050-5

© 2016 Helion S.A.

Authorized Polish translation of the English edition of ASP.NET MVC 5 with Bootstrap and Knockout.js, ISBN 9781491914397 © 2015 Jamie Munro.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/aspboo.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/aspboo>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Przedmowa 9

Wprowadzenie 13

Część I. Pierwsze kroki 15

1. Wprowadzenie do wzorca architektury MVC 17

Tworzenie pierwszego projektu 17

Analizowanie kontrolera HomeController 19

Analizowanie widoków 21

Struktura adresu URL 23

Podsumowanie 24

2. Wprowadzenie do środowiska Bootstrap 25

Analizowanie menu domyślnego 25

Menu z elementami rozwijanymi i polem wyszukiwania 28

Przyciski 30

Alerty 32

Kompozycje 32

Podsumowanie 33

3. Wprowadzenie do biblioteki Knockout.js 35

Instalowanie biblioteki Knockout.js 35

Prosty przykład 36

Czym jest MVVM? 39

Tworzenie modeli widoku 40

Podsumowanie 42

4. Praca z bazą danych	43
Wprowadzenie do środowiska Entity Framework	43
Przeływ Code First	45
Przeływ Database First	48
Tworzenie danych testowych	51
Podsumowanie	54

Część II. Praca z danymi55

5. Tworzenie listy, sortowanie i stronicowanie tabel	57
Użycie mechanizmu scaffolding dla modelu Author	57
Sortowanie autorów	63
Stronicowanie autorów	68
Podsumowanie	72
6. Użycie formularzy	73
Integrowanie biblioteki Knockout.js z formularzem	73
Współużytkowanie widoku i modelu widoku	79
Usuwanie przy użyciu okna modalnego	85
Wyświetlanie pustych tabel	89
Podsumowanie	91
7. Serwerowe modele widoku	93
Dlaczego tworzone są serwerowe modele widoku?	93
Model widoku autorów	94
Aktualizowanie listingu autorów	95
Aktualizowanie formularza dodawania/edytowania	97
Aktualizowanie okna dialogowego usuwania	98
Podsumowanie	99
8. Wprowadzenie do komponentu Web API	101
Instalowanie komponentu Web API	101
Aktualizowanie listy autorów	103
Aktualizowanie formularza służącego do dodawania/edytowania danych autorów	111
Podsumowanie	114

Część III. Architektura kodu 115

9. Tworzenie filtrów globalnych	117
Filtry uwierzytelniania	117
Filtry autoryzacji	118
Filtry akcji	118
Filtry wyniku	118
Filtry wyjątku	118
Globalne sprawdzanie poprawności komponentu Web API	118
Automatyczne odwzorowywanie przy użyciu filtra wyniku	122
Obsługa błędów komponentu Web API	125
Obsługa błędów wzorca MVC	127
Podsumowanie	130
10. Dodawanie uwierzytelniania i autoryzacji	131
Przegląd uwierzytelniania	131
Przegląd procesu autoryzacji	132
Implementowanie filtra uwierzytelniania	133
Implementowanie filtra autoryzacji	140
Podsumowanie	143
11. Routing adresów URL przy użyciu atrybutów	145
Podstawy routingu za pomocą atrybutów	145
Prefiksy trasy	148
Ograniczenia routingu	149
Podsumowanie	152
12. Złożony model i prosty kontroler	153
Separacja zagadnień	153
Usługi i zachowania	156
Podsumowanie	162

Część IV. Praktyczny przykład 163

13. Tworzenie koszyka zakupów	165
Wymagania koszyka zakupów	165
Projekt koszyka zakupów	166
Tworzenie pakietów i minifikacja kodu JavaScript	166
Podsumowanie	168

14. Budowanie modelu danych	169
Modele przepływu Code First	169
Definiowanie kontekstu DbContext i inicjowanie danych	172
Modele widoku	175
Podsumowanie	177
15. Implementowanie układu	179
Układ współużytkowany	179
Podsumowanie koszyka zakupów	180
Menu kategorii	187
Podsumowanie	190
16. Listy książek	191
Strona główna	191
Wyróżnione książki	192
Książki filtrowane według kategorii	194
Podsumowanie	197
17. Dodawanie pozycji do koszyka	199
Szczegóły dotyczące książki	199
Komponenty i wiązania niestandardowe	203
Zapisywanie pozycji koszyka	209
Podsumowanie	211
18. Aktualizowanie i usuwanie pozycji koszyka	213
Szczegóły koszyka	213
Użycie biblioteki Knockout.js do obsługi szczegółów koszyka	216
Finalizowanie koszyka zakupów	220
Podsumowanie	223
Skorowidz	225

Wprowadzenie do wzorca architektury MVC

MVC to wzorec architektury. Skrót wywodzi się od terminu *Model-View-Controller*. Moją definicję wzorca MVC mogę podsumować w następujący sposób:

- Model zarządza danymi aplikacji. Każdy model reprezentuje przeważnie jedną tabelę lub większą liczbę tabel w bazie danych.
- Widok zawiera wizualną reprezentację aplikacji. W witrynach internetowych osiąga się to zwykle z wykorzystaniem języków HTML i JavaScript oraz arkuszy stylów CSS.
- Kontroler pełni funkcję pośrednika między modelem i widokiem. Typowy kontroler będzie żądać danych od modelu i przekazywać je widokowi, aby użyć ich podczas wyświetlania danych. W czasie zapisywania danych kolejność działań byłaby odwrotna. Kontroler odebrałby dane z widoku i przekazał je modelowi w celu ich zapisania.

ASP.NET MVC 5 to środowisko implementujące wzorec architektury MVC.

Termin **MVC** będzie wielokrotnie powtarzany w książce. W większości sytuacji będę się odwoływać do środowiska MVC implementującego wzorec MVC.

Tworzenie pierwszego projektu

Oprogramowanie Visual Studio oferuje różne szablony ułatwiające rozpoczęcie projektu. W książce skoncentrowałem się na dwóch konkretnych szablonach: MVC i Web API.

Szablon MVC umożliwia tworzenie aplikacji internetowych, które korzystają z wzorca architektury *Model-View-Controller*. Szablon ten zostanie bardziej szczegółowo omówiony w książce.

Szablon Web API pozwala na tworzenie aplikacji internetowych RESTful. REST to kolejny typ wzorca architektury oprogramowania, powszechnie wykorzystywany do tworzenia interfejsów API lub aplikacji klient-serwer. Szablon Web API z łatwością integrowany jest z wzorcem architektury MVC, co umożliwia ponowne wykorzystanie kodu w projektach opartych na szablonach MVC i Web API. Zademonstruję to w dalszej części książki.

W ramach pierwszego przykładu skupię się na szablonie MVC. W oknie *Visual Studio Start Page* lub z menu *File/New* wybierz opcję *Project*. Jak widać, dostępnych jest wiele różnych typów projektów do utworzenia. W menu *Templates* rozwiń pozycję *Visual C#*. Z menu wybierz opcję *Web*.

W swojej domyślnej instalacji dysponuję tylko jedną opcją aplikacji internetowej ASP.NET. Wybierz ją i wprowadź nazwę w udostępnionym polu. Wpisałem nazwę **BootstrapIntroduction** (wprowadzenie do środowiska Bootstrap).

Po wybraniu nazwy projektu kliknij przycisk OK. Zostanie zaprezentowanych kilka różnych szablonów aplikacji internetowych. Wybierz szablon wzorca MVC, a zauważysz dodatkowe opcje umożliwiające dołączenie interfejsu Web API oraz projektu testu jednostkowego. Na potrzeby przykładu opcje pozostawię niezaznaczone.

W wypadku szablonu MVC dostępne są też cztery różne opcje uwierzytelniania:

No Authentication

Wszystkie strony witryny internetowej będą publicznie dostępne.

Individual User Accounts

Opcja sprawia, że aplikacja internetowa umożliwi użytkownikom rejestrowanie się i logowanie za pomocą utworzonej nazwy użytkownika i hasła. Opcja zapewnia też kilka różnych wariantów logowania w serwisach społecznościowych, w tym na Facebooka i Twittera, na konto Google i konto Microsoft. W każdym z tych wypadków informacje będą przechowywane w automatycznie tworzonej bazie danych członków.

Work And School Accounts

Opcja powoduje, że aplikacja internetowa zostanie zintegrowana z usługami Active Directory, Microsoft Azure Active Directory lub Office 365 w celu zabezpieczenia stron w obrębie aplikacji.

Windows Authentication

Dzięki tej opcji aplikacja internetowa jest zabezpieczana za pomocą modułu Windows Authentication IIS. Taki wariant jest powszechnie stosowany w wypadku aplikacji intranetowych, w których wszystkie konta użytkowników istnieją na serwerze WWW obsługującym aplikację.

Na potrzeby omawianego przykładu opcję zabezpieczeń zmieniłem na *No Authentication* (rysunek 1.1).

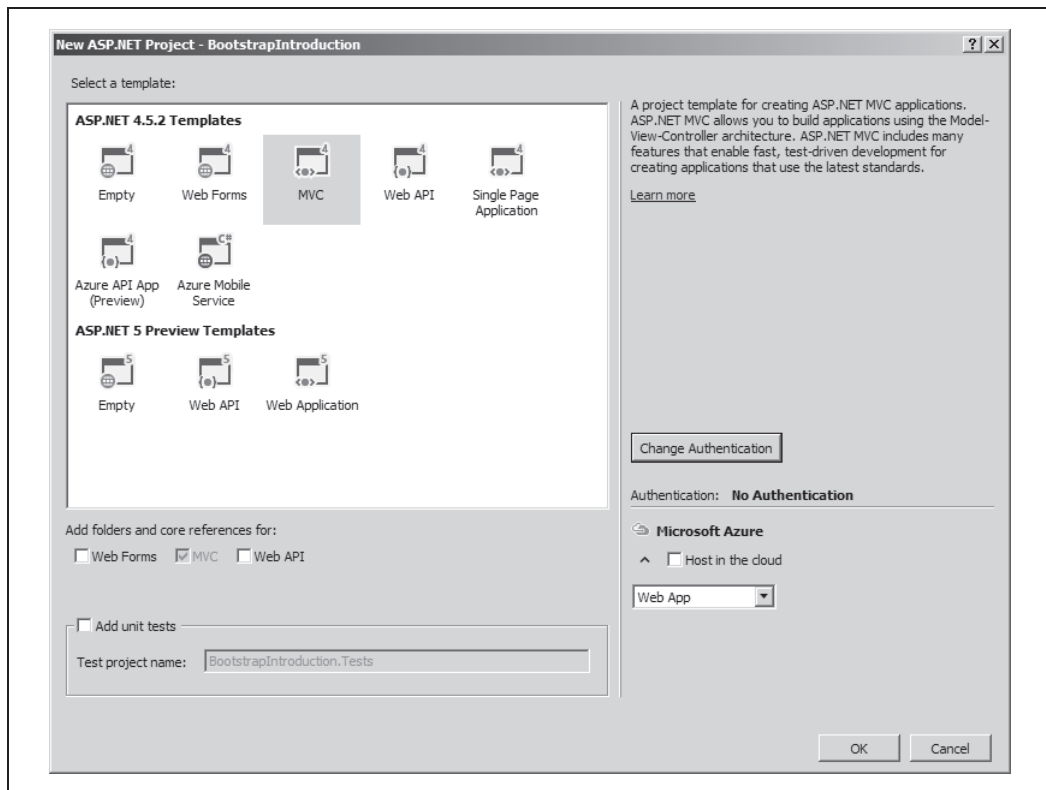
Gdy będziesz gotowy do zakończenia tworzenia projektu, kliknij przycisk OK. W zależności od wydajności komputera minutę albo dwie może potrwać dla projektu proces konfigurowania i wstępnego ładowania wielu przydatnych plików i folderów, które ułatwią rozpoczęcie pracy.

Po zakończeniu procesu możliwe będzie wybranie opcji *Start Debugging* z menu *Debug*. Spowoduje ona uruchomienie nowej aplikacji internetowej w domyślnej przeglądarce internetowej.

Po przyjrzeniu się folderom zauważysz, że z każdą rolą we wzorcu MVC powiązany jest osobny folder. Folder *Views* zawiera zwykle podfolder o nazwie zgodnej z nazwą kontrolera, ponieważ w folderze tym znajduje się zwykle kilka widoków, aby w prosty sposób umożliwić uporządkowanie plików.

Jeśli rozwiniesz folder *Views*, ujrzysz dwa podfoldery: *Home* i *Shared*. Folder *Home* dopasowany jest do klasy *HomeController* istniejącej w folderze *Controllers*. Folder *Shared* jest używany na potrzeby widoków zawierających wspólny kod innych widoków.

Widoki te obejmują układy lub widoki częściowe. Układy to specjalne widoki zawierające na każdej stronie widok wielokrotnego użycia. Widok częściowy jest widokiem do wielokrotnego zastosowania, posiadającym podzbiór danych, które mogą być dołączone do jednej strony lub ich większej liczby.



Rysunek 1.1. Tworzenie projektu

Rysunek 1.2 prezentuje zrzut ekranu domyślnej witryny internetowej tworzonej z wykorzystaniem nowego projektu MVC. Menu zawiera trzy odnośniki: *Strona główna*, *Informacje o* i *Kontakt*. Pamiętajmy o nich, rozpoczynając poznawanie kodu.

Analizowanie kontrolera HomeController

Zacznijmy od przyjrzenia się kontrolerowi. W folderze *Controllers* znajduje się plik o nazwie *HomeController.cs*, którego zawartość powinna być podobna do zaprezentowanej w przykładzie 1.1.

Przykład 1.1. Plik *HomeController.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace BootstrapIntroduction.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Rysunek 1.2. Domyślna witryna internetowa

```

public ActionResult About()
{
    ViewBag.Message = "Strona z opisem aplikacji";
    return View();
}
public ActionResult Contact()
{
    ViewBag.Message = "Strona z informacjami kontaktowymi";
    return View();
}
}
}

```

HomeController to klasa zawierająca trzy metody: Index, About i Contact. W terminologii wzorca MVC są one często określane mianem *akcji*. Akcja o nazwie Index to zwykle główny punkt wejścia kontrolera. Zauważ, że akcje About i Contact dopasowują nazwy utworzonych odnośników — wyjątkiem jest akcja Home, ponieważ odnosi się ona do akcji Index.

Wszystkie kontrolery w aplikacji MVC będą rozszerzać podstawową klasę Controller. Każda metoda w klasie zwraca typ o nazwie ActionResult. W większości sytuacji typ ten będzie zwracany przez wszystkie używane akcje. W zamieszczonych dalej przykładach zostaną objaśnione inne zwracane typy.

Akcje About i Contact wprowadzają też właściwość ViewBag klasy Controller. Właściwość ta umożliwia dynamiczne przekazywanie danych widokowi. Przykład 1.2 zademonstruje sposób jej użycia w widoku.



Właściwość ViewBag

Właściwość ViewBag pozwala na współużytkowanie danych między kontrolerami i widokami. Zmienna ta jest definiowana jako typ dynamiczny i nie posiada żadnych predefiniowanych właściwości. Dzięki temu możliwe jest podanie dowolnej nazwy właściwości, która może zawierać dowolny typ danych.

I wreszcie — każda akcja jest zwracana z wywołaniem funkcji `View`. Metoda ta istnieje w klasie `Controller`, rozszerzanej przez wszystkie kontrolery. W wyniku jej zastosowania ładowany jest widok i wykonywany jest kod Razor zawarty w pliku `.cshtml`. W przykładzie 1.1 nie są przekazywane funkcji żadne parametry, co oznacza, że domyślnie wzorzec MVC będzie szukać widoku o takiej samej nazwie jak nazwa funkcji.

Analizowanie widoków

Po rozwinięciu zawartości folderu `Views` widoczny jest podfolder o nazwie `Home`. Znajdują się w nim trzy pliki: `About.cshtml`, `Contact.cshtml` i `Index.cshtml`. Nazwa każdego pliku jest zgodna z nazwą jego akcji w kontrolerze. Rozszerzenie `.cshtml` to skrót od C# HTML. Widoki te umożliwiają użycie składni silnika Razor (<http://bit.ly/razor-syntax>), która stanowi kombinację języków HTML i C#. Daje to możliwość implementowania typowych technik tworzenia kodu, takich jak instrukcje warunkowe, pętle i zwracanie danych dynamicznych (na przykład wcześniej wspomniana właściwość `ViewBag`).

W przykładzie 1.2 zaprezentowano zawartość pliku `About.cshtml` domyślnej strony tworzonej wraz z projektem. Elementy korzystające ze składni silnika Razor zaczynają się od symbolu `@`. W połączeniu ze znakiem nawiasu klamrowego `{` pozwala to na stosowanie wielowierszowego kodu C#. W tym przykładzie dla właściwości `ViewBag` ustawiany jest tytuł strony (wartość "Informacje o"). Właściwość `Title` powiązana z właściwością `ViewBag` jest powszechnie używana w układzie współużytkowanym do ustawiania tytułu strony wyświetlanej przez przeglądarkę. W omawianym przykładzie właściwość `Title` jest też umieszczona w znaczniku `h2`.

Przykład 1.2. Plik `About.cshtml`

```
@{
    ViewBag.Title = "Informacje o";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>
<p>Użyj tego obszaru do udostępnienia dodatkowych informacji.</p>
```

Oprócz właściwości `Title` widoczna jest właściwość `Message`, powiązana w znaczniku `h3` z właściwością `ViewBag`. Jak już wspomniano, właściwość `Message` została ustawiona w akcji kontrolera dla metod `About` i `Contact`.

Jest to znakomity przykład tego, jak kod Razor jest wstawiany do standardowego kodu HTML w celu wygenerowania treści dynamicznej w momencie wykonywania operacji renderowania w przeglądarce.

W czasie przeglądania strony *Informacje o mozesz zauważyć, że zawiera ona znacznie więcej kodu HTML niż tylko kilka elementów. Jest to wynikiem użycia układu współużytkowanego. Domyślnie wszystkie widoki są umieszczane w obrębie szablonu domyślnego, który znajduje się w innym folderze, o nazwie `Shared`, umieszczonym w folderze `Views`. Jeśli rozwiniesz folder `Shared`, ujrzysz plik `_Layout.cshtml` (przykład 1.3).*

Przykład 1.3. Plik `_Layout.cshtml`

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ViewBag.Title – Moja aplikacja ASP.NET</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse"
          data-target=".navbar-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        @Html.ActionLink("Nazwa aplikacji", "Index", "Home",
          new { area = "" }, new { @class = "navbar-brand" })
      </div>
      <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li>@Html.ActionLink("Strona główna", "Index", "Home")</li>
          <li>@Html.ActionLink("Informacje o", "About", "Home")</li>
          <li>@Html.ActionLink("Kontakt", "Contact", "Home")</li>
        </ul>
      </div>
    </div>
  </div>
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year – Moja aplikacja ASP.NET</p>
    </footer>
  </div>
  @Scripts.Render("~/bundles/jquery")
  @Scripts.Render("~/bundles/bootstrap")
  @RenderSection("scripts", required: false)
</body>
</html>

```

Układ domyślny zawiera kod HTML wielokrotnego użytku, pojawiający się na każdej stronie w obrębie witryny (są to takie elementy jak tytuł strony, nagłówek, stopka, arkusze CSS czy kod JavaScript).

Renderowany widok wstawiany jest do tego układu w momencie wywołania metody `RenderBody` za pośrednictwem kodu Razor.

W wypadku tego układu współużytkowanego ma miejsce wiele innych rzeczy. Układ korzysta z kilku klas pomocniczych zapewnianych przez środowisko MVC, takich jak klasa `HtmlHelper` (<http://bit.ly/htmlhelper>) — do tworzenia odnośników, `Scripts` (<http://bit.ly/scripts-render>) — do dołączania plików JavaScript, `Styles` (<http://bit.ly/styles-render>) — do dołączania plików CSS, a także klasa `RenderSection`, która umożliwia widokom wskazywanie konkretnej treści do wstawienia w określonym miejscu układu współużytkowanego. Zostanie to zaprezentowane w rozdziale 2.

Struktura adresu URL

W momencie uruchomienia domyślnej witryny internetowej zostały utworzone następujące trzy odnośniki:

- *Strona główna*. Kieruje do głównego katalogu witryny (/).
- *Informacje o*. Kieruje do katalogu */Home/About*.
- *Kontakt*. Kieruje do katalogu */Home/Contact*.

Odnośniki te działają, ponieważ na początku tworzenia projektu MVC odpowiednio skonfigurowano trasę domyślną. Trasy umożliwiają witrynie internetowej informowanie wzorca MVC o tym, jak powinien odwzorować adres URL na konkretny kontroler i akcję w projekcie.

Trasy są konfigurowane w pliku *App_Start/RouteConfig.cs*. Przykład 1.4 prezentuje trasę domyślną skonfigurowaną z wykorzystaniem nowego projektu MVC.

Przykład 1.4. Trasa domyślna

```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}",  
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }  
);
```

W przykładowej konfiguracji trasy definiowane są następujące trzy istotne elementy:

- Nazwa. Każda trasa musi mieć unikatową nazwę.
- Adres URL. Jest to względny adres URL występujący po nazwie domeny witryny internetowej. Adres URL może zawierać kombinację statycznego tekstu ze zmiennymi.
- Konfiguracja domyślna. Jeśli nie zapewniono dowolnej ze zmiennych w adresie URL, mogą być dla nich ustawione wartości domyślne.

Jeśli ponownie przeanalizujemy wymienione wcześniej odnośniki, to dojdziemy do wniosku, że ich działanie wynika z następujących powodów:

- W wypadku przejścia do katalogu / adres URL nie zawiera żadnego kontrolera, akcji ani identyfikatora. Ustawiane wartości domyślne wskazują, że zostanie użyty kontroler *HomeController* i akcja *Index*, a identyfikator nie musi istnieć.
- W wypadku przejścia do katalogów */Home/About* i */Home/Contact* nie są używane żadne wartości domyślne, ponieważ w adresie URL są określone zarówno kontroler, jak i akcja.

Okazuje się, że odnośnik *Strona główna* może być też dostępny za pośrednictwem katalogów */Home* i */Home/Index*. Gdy jednak adresy URL są tworzone w obrębie widoku, wzorzec MVC wybiera spośród nich najkrótszy i najwłaściwszy adres URL, czyli po prostu katalog /.

W wypadku takiej trasy możesz utworzyć nowy kontroler i (lub) akcję, a ponadto trasa może być automatycznie dostępna przy użyciu jej nazwy i nazwy akcji.

Podsumowanie

Jeśli dopiero rozpoczynasz przygodę z wzorcem architektury MVC, treść tego rozdziału może sprawiać na Tobie przytłaczające wrażenie. Predefiniowane szablony MVC w środowisku Visual Studio są dość rozbudowane i zapewniają projektantom sporą „zaliczkę” w momencie realizowania projektów.

Dlatego też trudno omówić każdy szczegół dotyczący tego, co jest tworzone. W tym wprowadzającym rozdziale przedstawiłem wiele podstawowych funkcji, które umożliwią rozpoczęcie pracy.

Istnieje znacznie więcej szczegółów do omówienia, związanych z wzorcem architektury *Model-View-Controller*. W kolejnych rozdziałach książki, prezentujących bardziej zaawansowane zagadnienia, szczegóły te będą w znacznym stopniu przybliżane.

A

adres
 IP
 sprawdzanie poprawności, 141
 wyodrębnianie, 141
 URL, 23
akcja Index, 149
aktualizowanie
 formularza, 97, 111
 listingu, 95
 listy, 103
 okna dialogowego usuwania, 98
 pozycji koszyka, 213
 środowiska Bootstrap, 73
alerty, 32
analizowanie
 kontrolera HomeController, 19
 menu domyślnego, 25
 widoków, 21
architektura kodu, 115
ASP.NET, 131
automatyczne
 generowanie klasy, 51
 odwzorowywanie, 122
autoryzacja, 131, 132

B

baza danych, 43
biblioteka
 jQuery, 168
 Knockout.js, 35
błąd
 klienta, 114
 serwera, 114

Bootstrap, 25
budowanie modelu danych, 169

C

CRUD, 12

D

DAL, Data Access Layer, 46
dane, 55
 testowe, 51
definiowanie
 kontekstu, 172
 menu, 25
 odnośników menu, 26
 trasy, 146
diagram
 cyklu życia żądania, 133
 funkcji OnAuthentication, 132
dodawanie
 autoryzacji, 142
 pozycji do koszyka, 199
dołączanie biblioteki Knockout.js, 36
dostęp do
 bazy danych, 43
 właściwości obserwowalnych, 77

E

EF, Entity Framework, 43
element
 @RenderSection, 37
 li, 29
elementy rozwijane, 28
Entity Framework, 43
enumerators SortOrder, 64

F

filtrowanie według kategorii, 194
filtry

- akcji, 118
- autoryzacji, 118, 140
- globalne, 117
- uwierzytelniania, 117, 131, 133
- wyjątku, 118, 125
- wyniku, 118, 122

finalizowanie koszyka zakupów, 220
format JSON, 205
formularz, 73

- dodawania/edytowania, 97, 111
- wyszukiwania, 29
- z obsługą błędów, 76

funkcja

- AddToCart, 210
- Application_Start, 53
- Delete, 99
- foreach, 61
- Get, 103, 189
- GetByCartIdAndBookId, 211
- Index, 105, 214
- OnActionExecuted, 120
- OnActionExecutedAsync, 120
- OnActionExecuting, 119
- OnActionExecutingAsync, 119
- OnAuthentication, 131, 134
- OnAuthenticationChallenge, 131, 136
- pureComputed, 186
- SaveChanges, 222
- setTimeout, 76
- sortEntitiesBy, 108, 110
- successfulDelete, 219
- ToList, 61
- UpdateCartItem, 222
- upsertCartItem, 205
- validateAndSave, 111

funkcje niestandardowe, 186

G

generowanie klasy, 51

H

hasło, 135

I

IDE, Integrated Development Environment, 13
implementowanie

- filtru autoryzacji, 140
- filtru uwierzytelniania, 133
- układu, 179

informacje konfiguracyjne, 57
inicjowanie danych, 172
instalowanie

- biblioteki Knockout.js, 35
- komponentu Web API, 101
- narzędzia Fiddler, 121

K

karta Composer, 121
klasa

- AuthenticatedUsers, 138, 139
- AuthorsController, 60
- AuthService, 151, 158
- BasicAuthorizationAttribute, 140
- BasicChallengeActionResult, 137
- BookContext, 46
- BookInitializer, 53
- BookService, 192, 196, 200
- BundleConfig, 167
- CartItemService, 222
- CartService, 181
- CategoryService, 188
- DataInitialization, 172
- DbContext, 46
- ExceptionHandlerAttribute, 125
- FilterAttribute, 130
- FilterConfig, 129
- GenerateResultListFilterAttribute, 122, 123
- HtmlHelper, 107
- HtmlHelperExtensions, 67–70
- kontekstu danych, 57
- modal, 89
- modelu, 57
- MvcHtmlString, 67
- navbar, 28
- navbar-nav, 27
- nav-pills, 27
- OnApiExceptionAttribute, 125
- OnExceptionAttribute, 127, 128
- PagingService, 108, 110
- QueryOptions, 64–66

- QueryOptionsCalculator, 157
- ResultList, 124
- RouteConfig, 145
- ShoppingCartContext, 172
- ValidationActionFilterAttribute, 119
- WebApiConfig, 120, 126
- Knockout.js, 35
 - formularze, 73
 - instalacja, 35
 - obsługa szczegółów koszyka, 216
- kody statusu HTTP, 114
- komponent Web API, 101
- komponenty, 203
- kompozycje, 32
- komunikaty alertów, 32
- konfiguracja domyślna, 23
- kontekst DbContext, 172
- kontroler, 17, 154
 - AuthorsController, 58, 103, 124, 142, 150, 161
 - BooksController, 147, 192, 195, 199
 - CartItemController, 209
 - CartsController, 181, 213
 - CategoriesController, 188
 - HomeController, 19, 38, 146
 - wzorca MVC, 104
- koszyk zakupów, 165
 - aktualizowanie pozycji, 213
 - dodawanie pozycji, 199
 - finalizowanie, 220
 - lista kategorii, 190
 - listy książek, 191
 - podsumowanie, 180
 - projekt, 166
 - usuwanie pozycji, 213
 - użycie biblioteki Knockout.js, 216
 - wymagania, 165
 - zapisywanie pozycji, 209

L

- leniwe ładowanie, 46
- lista, 57
 - autorów, 63
- listing, 95
- listy książek, 191
 - filtrowanie według kategorii, 194
 - strona główna, 191
 - wyróżnione książki, 192
- logika biznesowa, 154

Ł

- łańcuch
 - ograniczeń, 149
 - połączenia BookContext, 47

M

- mechanizm scaffolding, 57
- menu
 - domyślne, 25
 - elementy rozwijane, 28
 - kategorii, 187
 - pole wyszukiwania, 28
 - rozwijane, 28
 - zwijane, 26
- metoda Post, 210
- minifikacja kodu JavaScript, 166
- model, 17
 - model Author, 45, 57, 74, 84, 170
 - AuthorFormViewModel, 74
 - Book, 45, 170
 - Cart, 171
 - CartItem, 171
 - Category, 170
 - Person, 41
 - QueryOptions, 64
 - ViewModel, 40
 - widoku, 40, 79, 93, 175
 - AuthorFormViewModel, 82
 - AuthorIndexViewModel, 87, 111
 - AuthorViewModel, 94, 175
 - BookDetailViewModel, 202
 - BookViewModel, 175
 - CartDetailViewModel, 216, 217
 - CartItemViewModel, 177, 204
 - CartSummaryViewModel, 184, 208, 219
 - CartViewModel, 176
 - CategoryViewModel, 176
 - ResultList, 104
 - ReturnData, 126
- modele przepływu Code First, 169
- MVC, Model-View-Controller, 17
- MVVM, Model-View-ViewModel, 35, 39

N

narzędzie

Fiddler, 121

NuGet Package Manager, 35

nazwa użytkownika, 135

O

obiekt

BundleConfig, 187

ViewModel, 62

obiekty inicjujące bazę danych, 53

obsługa

błędów, 76

komponentu Web API, 125

wzorca MVC, 127

szczegółów koszyka, 216

oczyszczanie danych, 154

oddzielanie kodu, 153

odnośniki

menu, 26

przycisków, 71

ograniczenia routingu, 149

ograniczenie, 150

okno

Add Connection, 48

Add Controller, 57

modalne, 85

opcja Use a layout page, 57

opcje uwierzytelniania, 18

operacje CRUD, 12

optymalizacja SEO, 145

orkiestracje, 155

ORM, Object-Relational-Mapper, 43

P

pakiet NuGet, 37

pasek postępu, 77

pierwszy projekt, 17

plik

_CartItemForm.cshtml, 205

_Layout.cshtml, 21, 25, 36

_List.cshtml, 193

About.cshtml, 21

App_Start/RouteConfig.cs, 23

Basic.cshtml, 37

BookInitializer.cs, 51

BookContext.cs, 46

bootstrap.css, 33

BundleConfig.cs, 167

Contact.cshtml, 21

Delete.cshtml, 98

Details.cshtml, 201

EDMX, 44, 50

Global.asax.cs, 102, 174

HomeController.cs, 19

Index.cshtml, 21, 191, 214

knockout.custom.js, 186, 206, 217

PagingService.js, 108

RouteConfig, 102

Summary.cshtml, 182

WebApiConfig, 102

Web.config, 47

podsumowanie koszyka, 187

pole wyszukiwania, 28

prefiksy trasy, 148

procedura obsługi błędów, 129

proces CRUD, 58

programowanie obiektowe, 40

projekt koszyka zakupów, 166

projektant aplikacji internetowych, 11

przegląd

autoryzacji, 132

uwierzytelniania, 131

przekształcanie danych, 154

przepływy zadań

Code First, 44, 169

Database First, 44, 48

Model First, 44

przestrzeń nazw modeli widoku, 94

przycisk, 30

wysyłania, 77

menu

zwijanego, 26

rozwijanego, 31

pusty listing, 90

R

refaktoryzacja, 154

repozytoria, 155

REST, 17

RESTful, 101

routing

adresów URL, 145

atrybuty, 145

- ograniczenia, 149
- prefiksy trasy, 148
- rozszerzenie
 - HtmlHelper, 62, 107
 - HtmlHelperExtension, 184
 - subTotal, 186

S

- scaffolding, 57
- sekcja Scripts, 77
- SEO, Search Engine Optimization, 145
- separacja zagadnień, 153
- serwerowe modele widoku, 93, 95
- silnik Razor, 61
- sortowanie, 63, 108
- sprawdzanie
 - dotyczące autoryzacji, 134
 - poprawności, 74
 - adresu IP, 141
 - modelu, 119
 - użytkownika, 140
- strona główna, 191
- stronicowanie, 68
- struktura adresu URL, 23
- szablon
 - MVC, 17
 - Web API, 17
- szczególne dotyczące książki, 199

Ś

- środowisko
 - ASP.NET, 131
 - Bootstrap, 25
 - Entity Framework, 43
 - Visual Studio, 101

T

- tabela, 89
 - Author, 49
 - Book, 49
- trasa, 146
 - domyślna, 23
 - domyślna kontrolera, 148
 - opcjonalna, 147

- tworzenie
 - danych testowych, 51
 - filtrów globalnych, 117
 - filtru autoryzacji, 140
 - filtru uwierzytelniania, 133
 - filtru wyjątku, 125
 - kontrolera AuthorsController, 58
 - koszyka zakupów, 165
 - listy, 57
 - łańcucha ograniczeń, 149
 - modeli widoku, 40
 - odnośników przycisków, 72
 - pakietów, 166
 - projektu, 17, 19
 - przycisków, 31
 - tabeli, 61
 - właściwości obserwowalnych, 84

U

- układ współużytkowany, 179
- usługa CartItemService, 210
- usługi, 154, 156
- usuwanie
 - autora, 85
 - pozycji koszyka, 213
- uwierzytelnianie, 18, 131, 135
 - podstawowe, 137, 139
- użycie
 - atrybutów, 145
 - formularzy, 73

V

- Visual Studio, 101

W

- warstwa
 - dostępu do danych, 46
 - orkiestracji, 155
 - repozytoriów, 155
 - usług, 154
 - zachowań, 154
- Web API, 101
 - obsługa błędów, 125
 - sprawdzanie poprawności, 118

- wiązanie
 - danych value, 76
 - danych wysyłania, 76
 - foreach, 215
 - isDirty, 206, 207
 - niestandardowe, 203, 206
 - textInput, 206
 - visible, 206, 207
- widok, 17, 40, 79
 - Create, 78
 - Delete, 87
 - Edit, 81
 - Error.cshtml, 128
 - formularza, 81
 - Index, 65, 84, 106
 - usuwania danych, 98
- witryna domyślna, 20
- właściwości
 - obserwowalne, 84
 - wirtualne, 46
- właściwość ViewBag, 20
- współużytkowanie
 - logiki biznesowej, 153
 - widoku i modelu widoku, 79
- wyodrębnianie
 - adresu IP, 141
 - nazwy, 135
- wyświetlanie pustych tabel, 89

- wzorzec
 - MVC, 17
 - MVVM, 35, 39
 - REST, 17
 - Unit of Work, 155, 156

Z

- zachowania, 154, 156
- zapisywanie pozycji koszyka, 209
- zmienne
 - obserwowalne, 75
 - prywatne, 60
- znacznik
 - div, 89
 - tbody, 62
- znak pytajnika, 149

Ż

- żądanie AJAX, 110, 218

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Praktyczne wprowadzenie do świata AngularJS!

Budowa aplikacji internetowych często wymaga integracji różnych technologii. Praca programisty staje się wówczas nieco trudniejsza, ale za to utworzone aplikacje mogą działać na wielu nowoczesnych urządzeniach bez potrzeby pisania kodu dla każdego sprzętu z osobna. W tej książce przedstawiono wyjątkowo udane połączenie trzech technologii, czyli środowiska ASP.NET MVC 5 umożliwiającego budowę zaawansowanych aplikacji internetowych, interakcję z bazą danych oraz dynamiczne renderowanie kodu HTML, środowiska Bootstrap pozwalającego na tworzenie ładnych i elastycznych widoków, a także biblioteki Knockout.js, która łączy te technologie, a jednocześnie rozszerza elastyczny projekt aplikacji internetowej dzięki dynamicznym interakcjom po stronie klienta, sterowanym przez serwerową aplikację internetową.

Książka, którą trzymasz w dłoniach, to podręcznik, który umożliwi Ci płynne tworzenie aplikacji zgodnych z wzorcem MVC (Model-View-Controller) za pomocą trzech technologii: ASP.NET MVC 5, środowiska Bootstrap i biblioteki Knockout.js. Poznasz sposoby, które pozwolą Ci napisać bardzo dobrze zorganizowane i łatwe w utrzymaniu projekty.

Dowiedz się, jak:

- tworzyć dobrze zorganizowane i łatwe w utrzymaniu aplikacje internetowe
- budować serwerowe aplikacje internetowe na platformie ASP.NET MVC 5, korzystać z baz danych i w dynamiczny sposób renderować strony HTML
- tworzyć elastyczne widoki za pomocą środowiska Bootstrap i umożliwiać ich renderowanie na przeróżnych nowoczesnych urządzeniach
- ulepszać projekt elastycznej aplikacji internetowej za pomocą biblioteki Knockout.js z wykorzystaniem szybkich interakcji po stronie klienta

Jamie Munro — od ponad 15 lat zajmuje się projektowaniem witryn i aplikacji internetowych. Od blisko dekady jest mentorem młodszych programistów: przekazuje im swoją wiedzę i pozwala korzystać ze swojego wieloletniego doświadczenia.

sięgnij po WIĘCEJ



KOD KORZYŚCI

Helion

41501 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

☎ 0 801 339900

📞 0 601 339900

Sprawdź najnowsze promocje:
 ● <http://helion.pl/promocje>
 Książki najchętniej czytane:
 ● <http://helion.pl/bestsellery>
 Zamów informacje o nowościach:
 ● <http://helion.pl/novosci>

Helion SA
 ul. Kościuszki 1c, 44-100 Gilwice
 tel.: 32 230 98 63
 e-mail: helion@helion.pl
<http://helion.pl>

ISBN 978-83-283-2050-5



9 788328 320505

Informatyka w najlepszym wydaniu

cena: 49,00 zł