

Spis treści

przedmowa	ix
podziękowania	xi
o tej książce	xiii
o autorze	xvii
o ilustracji na okładce	xviii

CZĘŚĆ 1 UŻYWANIE C# I .NET1

1 Przedstawiamy C# i .NET 3

1.1 Dlaczego warto pracować w C#?	5
Powód 1: C# jest ekonomiczny	6
Powód 2: C# jest łatwy w utrzymaniu	6
Powód 3: C# jest przyjazny dla dewelopera i łatwy w użyciu	7
1.2 Kiedy lepiej nie pracować w C#?	8
Tworzenie systemu operacyjnego	8
Tworzenie wbudowanych systemów czasu rzeczywistego w C#	9
Przetwarzanie numeryczne a C#	10
1.3 Przełączanie się na C#	10
1.4 Czego można się nauczyć z tej książki	13
1.5 Czego nie nauczymy się z tej książki	13
Podsumowanie	15

2 .NET i proces kompilacji 17

2.1 Czym jest .NET Framework?	18
2.2 Czym jest .NET 5?	19
Ćwiczenia	20
2.3 Jak kompilowane są języki zgodne z CLI	21
Krok 1: Kod C# (wysokiego poziomu)	22
Krok 2: Common Intermediate Language (poziom asemblera)	25
Krok 3: Kod natywny (poziom procesora)	32
Ćwiczenia	34
Podsumowanie	35

CZĘŚĆ 2 ISTNIEJĄCA BAZA KODU..... 37

3 Jak zły jest ten kod? 39

- 3.1 Przedstawiamy Flying Dutchman Airlines 40
- 3.2 Kawałki układanki: Spojrzenie na wymagania 42
 - Mapowanie obiektowo-relacyjne 42
 - Punkt końcowy GET /flight: Pobieranie informacji o wszystkich lotach 43
 - Punkt końcowy GET /flight/{flightNumber}: Pobieranie informacji o konkretnym locie 43
 - Punkt końcowy POST /booking/{flightNumber}: Rezerwowanie lotu 45
- 3.3 Uzgadnianie wymagań z istniejącą bazą kodu 47
 - Ocena istniejącego schematu bazy danych i jej tabel 47
 - Istniejąca baza kodu: pliki konfiguracyjne usługi Web 48
 - Badanie modeli i widoków w istniejącej bazie kodu 55
- Podsumowanie 62

4 Zarządzanie zasobami niezarządzanymi! 63

- 4.1 FlightController: badanie punktu końcowego GET /flight 65
 - Punkt końcowy GET /flight i jego działanie 65
 - Sygnatura metody: Znaczenie słów kluczowych ResponseType oraz typeof 67
 - Gromadzenie informacji o lotach za pomocą kolekcji 69
 - Łańcuchy połączenia, czyli jak doprowadzić inżyniera zabezpieczeń do zawału 70
 - Używanie IDisposable do zwalniania niezarządzanych zasobów 71
 - Odpytywanie bazy danych za pomocą SqlCommand 73
- 4.2 FlightController: Poznajemy GET /flight/{flightNumber} 77
- 4.3 FlightController: POST /flight 80
- 4.4 FlightController: DELETE /flight/{flightNumber} 85
- Ćwiczenia 86
- Podsumowanie 87

CZĘŚĆ 3 WARSTWA DOSTĘPU DO BAZY DANYCH..... 89

5 Konfigurowanie projektu i bazy danych za pomocą Entity Framework Core 91

- 5.1 Tworzenie rozwiązania i projektu .NET 5 92
- 5.2 Tworzenie i konfigurowanie usługi Web 96
 - Konfigurowanie usługi Web .NET 5 97

Tworzenie i używanie HostBuilder	99
Implementowanie klasy Startup	102
Używanie wzorca repozytorium/usługa w architekturze naszej usługi Web	105

5.3 Implementowanie warstwy dostępu do bazy danych	107
Entity Framework Core i inżynieria odwrotna	108
DbSet i przepływ pracy Entity Framework Core	110
Metody konfiguracji i zmienne środowiskowe	112
Ustawianie zmiennej środowiskowej w Windows	113
Ustawianie zmiennej środowiskowej w macOS	114
Odczytywanie zmiennych środowiskowych w czasie działania programu	115
Ćwiczenia	117
Podsumowanie	118

CZĘŚĆ 4 WARSTWA REPOZYTORIUM121

6 Wytwarzanie sterowane testami i wstrzykiwanie zależności 123

6.1 Wytwarzanie sterowane testami	125
Ćwiczenia	129
6.2 Metoda CreateCustomer	130
Dlaczego należy walidować argumenty wejściowe	131
Używanie wzorca AAA w pisaniu testów jednostkowych	132
Walidacja pod kątem nieprawidłowych znaków	133
Włamywanie danych testowych za pomocą atrybutu [DataRow]	136
Inicjalizatory obiektów i automatycznie generowany kod	137
Konstruktory, refleksje i programowanie asynchroniczne	139
Blokady, muteksy i semaforey	141
Wykonywanie synchroniczne do asynchronicznego... ciąg dalszy	143
Testowanie Entity Framework Core	144
Kontrolowanie zależności przy użyciu wstrzykiwania zależności	146
Ćwiczenia	152
Podsumowanie	153

7 Porównywanie obiektów 155

7.1 Metoda GetCustomerByName	156
Znaki zapytania: typu nullable i ich zastosowania	159
Niestandardowe wyjątki, LINQ i metody rozszerzające	159
7.2 Kongruencja: od średniowiecza do C#	164

Tworzenie klasy „porównującej” przy użyciu <code>EqualityComparer<T></code>	166
Testowanie równości poprzez nadpisanie metody <code>Equals</code>	169
Przeciążanie operatora równości	170
Ćwiczenia	173
Podsumowanie	175

8 Atrapy, typy ogólne i sprzężenie 177

8.1 Implementowanie repozytorium <code>Booking</code>	178
8.2 Walidacja wejścia, rozdzielanie zagadnień i sprzężenie	181
Ćwiczenia	186
8.3 Używanie inicjalizatorów obiektów	187
8.4 Testy jednostkowe z użyciem atrap	190
8.5 Programowanie przy użyciu typów ogólnych	195
8.6 Dostarczanie domyślnych wartości argumentów przy użyciu parametrów opcjonalnych	197
8.7 Wyrażenia warunkowe, typ <code>Func</code> i przełączniki	199
Trójargumentowy operator warunkowy	200
Rozgałęzianie wykonania przy użyciu tablicy funkcji	200
Instrukcje i wyrażenia <code>switch</code>	201
Odpytywanie o oczekujące zmiany w <code>Entity Framework Core</code>	203
Ćwiczenia	206
Podsumowanie	207

9 Metody rozszerzające, strumienie i klasy abstrakcyjne 209

9.1 Implementowanie repozytorium <code>Airport</code>	211
9.2 Pobieranie lotniska z bazy danych na podstawie przekazanego identyfikatora	212
9.3 Walidacja parametru wejściowego <code>AirportID</code>	214
9.4 Strumienie wyjściowe i zapewnienie odpowiedniej abstrakcji	216
9.5 Odpytywanie bazy danych o obiekt <code>Airport</code>	221
9.6 Implementowanie repozytorium <code>Flight</code>	230
Metoda rozszerzająca <code>IsPositive</code> oraz „magiczne liczby”	232
Pobieranie obiektu lotu z bazy danych	238
Ćwiczenia	241
Podsumowanie	242

CZĘŚĆ 5 WARSTWA USŁUGI. 243

10 Refleksja i imitacje 245

10.1 Powrót do wzorca repozytorium/usługa	246
---	-----

Jakie jest zastosowanie klasy usługi? 247

Ćwiczenia 248

10.2 Implementowanie klasy CustomerService 249

Konfigurowanie przypadku sukcesu: tworzenie klas szkieletowych 249

Jak usunąć swój własny kod 251

Ćwiczenia 253

10.3 Implementowanie BookingService 254

Testy jednostkowe przekraczające granice pomiędzy warstwami architektury 258

Różnica pomiędzy stubem a mockiem 260

Imitowanie klasy za pomocą biblioteki Moq 261

Wywoływanie repozytorium z poziomu usługi 268

Ćwiczenia 271

Podsumowanie 272

11 Sprawdzanie typów w czasie działania i obsługa błędów – spojrzenie drugie 275

11.1 Walidacja parametrów wejściowych metody warstwy usługi 276

*Sprawdzanie typów w czasie działania przy użyciu operatorów is
i as* 280

Sprawdzanie typu za pomocą operatora is 281

Sprawdzanie typów przy użyciu operatora as 282

Co zrobiliśmy w punkcie 11.1? 283

11.2 Sprzątanie klasy BookingServiceTests 284

11.3 Ograniczenia klucza obcego w klasach usługowych 286

Wywoływanie repozytorium Flight z klasy usługowej 287

Ćwiczenia 300

Podsumowanie 301

12 Stosowanie IEnumerable<T> oraz yield return 303

12.1 Czy potrzebujemy klasy AirportService? 304

12.2 Implementowanie klasy FlightService 306

Uzyskiwanie informacji o lotach z FlightRepository 306

Łączenie dwóch strumieni danych w widok 311

Używanie słów kluczowych yield return w blokach kodu try-catch 319

Implementing GetFlightByFlightNumber 324

Ćwiczenia 330

Podsumowanie 332

CZĘŚĆ 6 WARSTWA KONTROLERA 335**13 Oprogramowanie pośrednie, trasy HTTP i odpowiedzi HTTP 337**

- 13.1 Klasy kontrolerów w ramach wzorca repozytorium/usługa 338
- 13.2 Ustalanie, które kontrolery trzeba zaimplementować 341
- 13.3 Implementowanie klasy FlightController 342
 - Zwracanie odpowiedzi HTTP przy użyciu interfejsu *IActionResult* (*GetFlights*) 343
 - Wstrzykiwanie zależności do kontrolera za pomocą *middleware* 347
 - Implementowanie punktu końcowego *GET /Flight/{FlightNumber}* 356
- 13.4 Kierowanie żądań HTTP do kontrolerów i metod 361
 - Ćwiczenia 366
 - Podsumowanie 367

14 Serializacja i deserializacja JSON oraz niestandardowe wiązanie modelu 369

- 14.1 Implementowanie klasy BookingController 370
 - Wprowadzenie do deserializacji danych 372
 - Używanie atrybutu *[FromBody]* do deserializacji przychodzących danych HTTP 376
 - Użycie niestandardowego wiązania modelu i atrybutu metody dla wiązania 378
 - Implementowanie logiki punktu końcowego w metodzie *Create Booking* 381
- 14.2 Testy akceptacyjne i *middleware* Swagger 387
 - Ręczne wykonywanie testów akceptacyjnych na podstawie specyfikacji OpenAPI 388
 - Generowanie specyfikacji OpenAPI w czasie działania programu 392
- 14.3 Koniec podróży 399
 - Podsumowanie 399

- Dodatek A. Odpowiedzi do ćwiczeń 401
- Dodatek B. Lista kontrolna czystego kodu 411
- Dodatek C. Wskazówki instalacyjne 413
- Dodatek D. OpenAPI FlyTomorrow 417
- Dodatek E. Lista lektur 421