Angular

Profesjonalne techniki programowania

Wydanie IV

—

Adam Freeman

Helion

Apress

Tytuł oryginału: Pro Angular 9: Build Powerful and Dynamic Web Apps, 4th Edition

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-7543-7

First published in English under the title Pro Angular 9: Build Powerful and Dynamic Web Apps by Adam Freeman, edition: 4

Helion S.A.
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:
https://ftp.helion.pl/przyklady/angup4.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/angup4
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 4.

Krótkie wprowadzenie do HTML i CSS

Do świata programowania aplikacji internetowych programiści docierają wieloma różnymi ścieżkami i dlatego nie zawsze mają opanowane podstawowe technologie, na których oparte są te aplikacje. W tym rozdziale przedstawię krótkie wprowadzenie do języka HTML i frameworka Bootstrap CSS, którego użyłem do nadania stylu przykładom przedstawionym w książce. Z kolei w rozdziałach 5. i 6. znajdziesz wprowadzenie do języków JavaScript i TypeScript oraz informacje niezbędne do zrozumienia przykładów zaprezentowanych w pozostałej części książki. Jeżeli jesteś doświadczonym programistą, możesz pominąć te trzy rozdziały i od razu przejść do rozdziału 7., w którym wykorzystam framework Angular do utworzenia znacznie bardziej skomplikowanej i rzeczywistej aplikacji internetowej.

Utworzenie przykładowego projektu

Materiał przedstawiony w tym rozdziale wymaga tylko prostego projektu. Pracę rozpocznij od utworzenia katalogu o nazwie *HtmlCssPrimer*, a następnie umieść w nim plik *package.json* wraz z kodem przedstawionym na listingu 4.1.

Listing 4.1. *Zawartość pliku package.json w katalogu HtmlCssPrimer*

```
{
    "dependencies": {
      "bootstrap": "4.4.1"
}
```

Aby pobrać i zainstalować pakiety wymienione w pliku *package.json*, z poziomu katalogu *HtmlCssPrimer* wydaj poniższe polecenie:

```
$ npm install
```

Następnie w katalogu *HtmlCssPrimer* utwórz nowy plik o nazwie *index.html* i umieść w nim kod przedstawiony na listingu 4.2.

Listing 4.2. *Zawartość pliku index.html w katalogu HtmlCssPrimer*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
</head>
<body class="m-1">
    <h3 class="bg-primary text-white p-3">Lista Adama</h3>

    <div class="my-1">
        <input class="form-control" />
        <button class="btn btn-primary mt-1">Dodaj</button>
    </div>

    <table class="table table-striped table-bordered">
        <thead>
            <tr>
                <th>Zadanie</th>
                <th>Wykonane</th>
            </tr>
        </thead>
        <tbody>
            <tr><td>Kupić kwiaty</td><td>Nie</td></tr>
            <tr><td>Kupić buty</td><td>Nie</td></tr>
            <tr><td>Odebrać bilety</td><td>Tak</td></tr>
            <tr><td>Zadzwonić do Janka</td><td>Nie</td></tr>
        </tbody>
    </table>
</body>
</html>
```

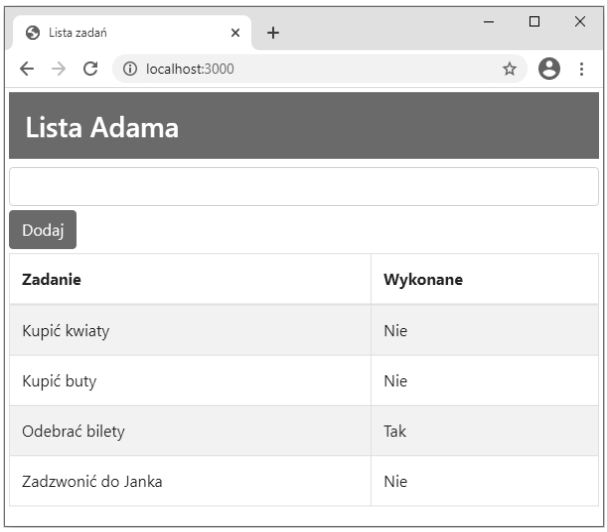
To jest kod HTML zawierający prostą listę rzeczy do zrobienia, podobną do utworzonej za pomocą frameworka Angular w rozdziale 2. Aby pobrać i uruchomić programistyczny serwer HTTP, z poziomu katalogu *HtmlCssPrimer* wydaj poniższe polecenie:

```
$ npx lite-server@2.5.4
```

Pakiet `lite-server` zawiera programistyczny serwer HTTP, który po wykryciu zmiany w plikach będzie automatycznie odświeżał stronę w przeglądarce WWW. Po pobraniu pakietu nastąpi uruchomienie serwera, a na ekranie zostanie wyświetlone nowe okno lub karta przeglądarki WWW wraz z treścią pokazaną na rysunku 4.1. (Jeżeli konfiguracja systemu uniemożliwia automatyczne otwieranie okien, możesz ręcznie wpisać adres *http://localhost:3000* w pasku adresu przeglądarki WWW).

Język HTML

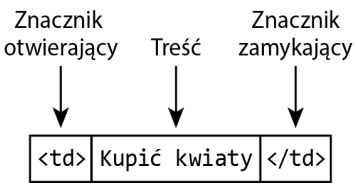
Sercem języka HTML jest **element**, który wskazuje przeglądarce WWW rodzaj treści umieszczonej w danym miejscu dokumentu HTML. Poniżej przedstawiłem element HTML pochodzący z dokumentu zaprezentowanego na listingu 4.2.



Rysunek 4.1. *Efekt uruchomienia przykładowej aplikacji*

```
...
<td>Kupić kwiaty</td>
...
```

Jak pokazałem na rysunku 4.2, element ten składa się z trzech części — znaczników: otwierającego i zamykającego oraz z treści.



Rysunek 4.2. *Struktura prostego elementu HTML*

Nazwą tego elementu (określaną również mianem **nazwy znacznika** lub po prostu **znacznikiem**) jest `<td>`. Ten znacznik wskazuje przeglądarce WWW, że umieszczona w nim treść powinna być traktowana jako komórka tabeli. Znacznik otwierający zawiera się w nawiasie ostrym (znaki < i >), natomiast znacznik zamykający posiada dodatkowo ukośnik, /, umieszczony tuż po znaku otwierającego nawiasu ostrego (<). Wszystko to, co znajduje się między znacznikami, jest treścią elementu — może nią być tekst, jak w omawianym przykładzie `Kupić kwiaty`, bądź też inne elementy HTML.

Element samozamykający się

W specyfikacji HTML zdefiniowano również elementy, które mogą nie zawierać treści. Tego rodzaju elementy są określane mianem **samozamykających się** (lub też *void*) i nie zawierają znacznika zamykającego, jak pokazałem poniżej.

```
...
<input />
...
```

Element samozamykający się jest zdefiniowany w jednym znaczniku, należy dodać ukośnik / przed ostatnim nawiasem ostrym (znak >). Element `<input>` jest najczęściej używanym elementem samozamykającym się i służy do tego, by umożliwić użytkownikowi podanie danych wejściowych za pomocą pola tekstowego, przycisku opcji lub pola wyboru. W dalszych rozdziałach wielokrotnie spotkasz się z wykorzystaniem tego elementu.

Atrybut

Istnieje możliwość przekazania przeglądarce WWW informacji dodatkowych poprzez dodanie do elementu tak zwanego **atrybutu**. Poniżej przedstawiłem przykład atrybutu w elemencie pochodzącym z dokumentu zdefiniowanego na listingu 4.2.

```
...
<link href="node_modules/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />
...
```

To jest znacznik `<link>` odpowiedzialny za wczytanie treści do dokumentu HTML. Mamy tutaj dwa atrybuty zaznaczone pogrubioną czcionką, aby można było je łatwiej dostrzec. Atrybut zawsze jest definiowany jako część znacznika początkowego, a ponadto posiada **nazwę** i **wartość**.

W omawianym przykładzie istnieją dwa atrybuty o nazwach `href` i `rel`. W przypadku znacznika `<link>` atrybut `href` określa treść przeznaczoną do zaimportowania, natomiast atrybut `rel` wskazuje przeglądarce WWW rodzaj importowanej treści. Dlatego też przedstawione tutaj atrybuty elementu `<link>` nakazują przeglądarce WWW import pliku *bootstrap.min.css* i potraktowanie go jako arkusza stylów, ponieważ wskazany plik zawiera style CSS.

Stosowanie atrybutu bez wartości

Nie każdy atrybut wymaga wartości. W takim przypadku zdefiniowanie atrybutu stanowi dla przeglądarki WWW sygnał, że dla danego elementu oczekiwany jest pewien rodzaj zachowania. Poniżej przedstawiłem przykład elementu wraz z atrybutem niewymagającym wartości (ten element nie pochodzi z listingu 4.2).

```
...
<input class="form-control" required />
...
```

Powyższy znacznik `<input>` zawiera dwa atrybuty. Pierwszy, `class`, powoduje przypisanie wartości, podobnie jak w poprzednim przykładzie. Natomiast drugi atrybut to po prostu słowo `required`. To jest przykład atrybutu niewymagającego wartości.

Cytowanie literałów w atrybucie

Spora część funkcjonalności frameworka Angular opiera swoje działanie na atrybutach elementów HTML. W większości przypadków wartości atrybutów są podawane w postaci wyrażeń JavaScript, podobnie jak pokazałem poniżej w elemencie pochodzącym z kodu przedstawionego w rozdziale 2.

```
...
<td [ngSwitch]="item.complete">
...
```

Atrybut zastosowany dla znacznika `<td>` nakazuje Angularowi odczyt wartości właściwości o nazwie `complete` w obiekcie przypisanym zmiennej `item`. Będą się zdarzać sytuacje, gdy konieczne będzie przekazanie konkretnej wartości zamiast oczekiwania od Angulara odczytania wartości z danych modelu. W takim przypadku wymagane jest zastosowanie dodatkowego cytowania wskazującego frameworkowi, że ma do czynienia z literałem. Spójrz na poniższy fragment kodu.

```
...
<td [ngSwitch]="'jabłka'">
...
```

Wartość atrybutu zawiera ciąg tekstowy `jabłka` ujęty zarówno w apostrofy, jak i w cudzysłów. Podczas analizy wartości atrybutu Angular odkrywa użycie apostrofów i przetwarza tę wartość jako literał ciągu tekstowego.

Treść elementu

Element może zawierać tekst oraz inne elementy, jak pokazałem poniżej.

```
...
<thead>
    <tr>
        <th>Zadanie</th>
        <th>Zrobione</th>
    </tr>
</thead>
...
```

Elementy w dokumencie HTML tworzą naturalną hierarchię. Dlatego też element `<html>` zawiera `<body>`, a ten z kolei zawiera elementy definiujące treść strony, które dalej mogą zawierać kolejne elementy itd. W przypadku kodu przedstawionego na listingu 4.2 element `<thead>`

zawiera elementy `<tr>`, które z kolei zawierają elementy `<th>`. Zagnieżdżanie elementów w ten sposób jest kluczową koncepcją w HTML, ponieważ znaczenie zewnętrznych elementów nadaje elementom znajdującym się wewnątrz.

Struktura dokumentu

Istnieją pewne kluczowe elementy definiujące podstawową strukturę dokumentu HTML: `<DOCTYPE>`, `<html>`, `<head>` i `<body>`. Poniżej przedstawiłem powiązania między wymienionymi elementami, przy czym usunąłem pozostałą treść z kodu strony internetowej.

```
<!DOCTYPE html>
<html>
<head>
    …treść nagłówka…
</head>
<body>
```

```
    …treść główna dokumentu…
</body>
</html>
```

Każdy z wymienionych elementów ma do odegrania konkretną rolę w dokumencie HTML. Element `<DOCTYPE>` informuje przeglądarkę WWW, że dany dokument to HTML, a dokładnie dokument HTML5. Wcześniejsze wersje specyfikacji HTML wymagały podania informacji dodatkowych. Na przykład poniżej przedstawiłem element `<DOCTYPE>` dla dokumentu HTML4:

```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
...
```

Element `<html>` wskazuje obszar dokumentu zawierający treść HTML. Ten element zawsze zawiera dwa inne elementy o kluczowym znaczeniu dla struktury: `<head>` i `<body>`. Nie zamierzam tutaj omawiać poszczególnych elementów specyfikacji HTML — jest ich po prostu zbyt wiele. Dokładne omówienie specyfikacji HTML5 wymagało napisania ponad 850 stron w innej mojej książce zatytułowanej *HTML5. Przewodnik encyklopedyczny* również wydanej przez Helion. Dlatego w tabeli 4.1 zamieściłem jedynie krótkie omówienie elementów wykorzystanych do zdefiniowania dokumentu HTML, którego kod przedstawiłem na listingu 4.2. To powinno Ci pomóc w zrozumieniu, jak element HTML wskazuje przeglądarce WWW rodzaj znajdującej się w nim treści.

Tabela 4.1. *Elementy HTML użyte w przykładowym dokumencie*

Element	Opis
`<DOCTYPE>`	Określa typ treści w dokumencie HTML.
`<body>`	Wskazuje obszar dokumentu, w którym znajdują się elementy dotyczące treści.
`<button>`	Przycisk — bardzo często używany w celu wysłania formularza sieciowego do serwera.
`<div>`	Ogólny element, bardzo często wykorzystywany do nadawania struktury dokumentu w celach prezentacyjnych.
`<h3>`	Nagłówek trzeciego poziomu.
`<head>`	Wskazuje obszar dokumentu, w którym znajdują się elementy dotyczące metadanych.
`<html>`	Wskazuje obszar dokumentu, w którym znajduje się kod HTML (to najczęściej cały dokument).
`<input>`	Ten element definiuje pole używane do pobrania pojedynczych danych od użytkownika.
`<link>`	Za pomocą tego znacznika można zaimportować do dokumentu HTML wskazaną treść.
`<meta>`	To jest znacznik zawierający opisowe dane dotyczące dokumentu, na przykład zastosowane kodowanie znaków.
`<table>`	Tabela używana do umieszczenia treści w wierszach i kolumnach.
`<tbody>`	Właściwe dane tabeli (w przeciwieństwie do nagłówka i stopki).
`<td>`	Komórka wyświetlająca treść w wierszu tabeli.

Tabela 4.1. Elementy HTML użyte w przykładowym dokumencie (ciąg dalszy)

Element	Opis
`<th>`	Komórka nagłówka w wierszu tabeli.
`<thead>`	Nagłówek tabeli.
`<title>`	Tytuł dokumentu, używany przez przeglądarkę WWW w celu ustawienia tytułu okna lub karty.
`<tr>`	Wiersz tabeli.

Poznajemy obiektowy model dokumentu (DOM)

Gdy przeglądarka WWW wczytuje i przetwarza dokument HTML, wtedy tworzy tak zwany **obiektowy model dokumentu** (ang. *document object model*, **DOM**). We wspomnianym modelu DOM obiekty JavaScript są używane do przedstawienia każdego elementu dokumentu. Dzięki temu model DOM stanowi mechanizm, za pomocą którego można w programowy sposób operować na zawartości dokumentu HTML.

W aplikacji zbudowanej na podstawie frameworka Angular bardzo rzadko zachodzi potrzeba bezpośredniej pracy z modelem DOM. Mimo wszystko bardzo ważne jest poznanie, w jaki sposób przeglądarka WWW utrzymuje „żywy" model dokumentu HTML reprezentowany przez obiekty JavaScript. Podczas modyfikacji tych obiektów przez Angular przeglądarka WWW uaktualnia wyświetlaną treść i tym samym odzwierciedla wprowadzone zmiany. To jest jedna z kluczowych koncepcji, na których opierają się aplikacje internetowe. Gdyby nie było możliwości modyfikacji modelu DOM, to niemożliwe by było utworzenie aplikacji internetowych działających po stronie klienta.

Framework Bootstrap

Elementy HTML wskazują przeglądarce WWW rodzaj znajdującej się w nich treści, ale nie zawierają żadnych informacji o sposobie, w jaki ta treść ma być wyświetlona. Informacje o sposobie wyświetlania treści są dostarczane przez kaskadowe arkusze stylów (ang. *cascading style sheets*, **CSS**). Style CSS składają się z obszernego zbioru **właściwości**, za pomocą których można skonfigurować praktycznie każdy aspekt wyglądu elementu. Ponadto zawierają zbiór **selektorów** pozwalających na stosowanie wspomnianych właściwości.

Jednym z największych problemów związanych ze stylami CSS jest to, że niektóre przeglądarki WWW nieco odmiennie interpretują właściwości. To może prowadzić do różnic w sposobie wyświetlania treści elementów HTML w poszczególnych urządzeniach. Monitorowanie i niwelowanie wspomnianych różnic okazało się trudnym zadaniem, stąd pojawienie się frameworków CSS, które mają pomagać programistom aplikacji internetowych w nadawaniu w prosty i spójny sposób stylu tworzonej przez nich treści HTML.

Jednym z frameworków, który zyskał dużą popularność, jest Bootstrap. Składa się on ze zbioru klas CSS, które można stosować w elementach i tym samym nadawać im spójny styl. Klasy można stosować także dla kodu JavaScript, co pozwala na przeprowadzenie dodatkowych usprawnień tego kodu. W książce zdecydowałem się na wykorzystanie stylów Bootstrap CSS, ponieważ pozwalają one na nadanie przykładom odpowiedniego wyglądu bez konieczności samodzielnego definiowania stylów CSS w każdym rozdziale. Natomiast nie użyłem żadnego kodu Bootstrap JavaScript, ponieważ interaktywność przykładów jest zapewniona przez framework Angular.

Nie będę się zbyt zagłębiać w szczegóły dotyczące frameworka Bootstrap, ponieważ nie stanowi on tematu książki. Jednak znajdziesz tutaj wystarczającą ilość informacji, aby zauważyć, które części przykładów są funkcjami Angulara, a które stylami Bootstrap. Framework Bootstrap oferuje znacznie więcej funkcji, niż tu przedstawiam. Więcej informacji na jego temat znajdziesz w witrynie *http://getbootstrap.com/*.

Stosowanie podstawowych klas Bootstrap

Style Bootstrap są stosowane za pomocą atrybutu `class` używanego do połączenia powiązanych ze sobą elementów. Atrybut `class` jest używany nie tylko do stosowania stylów CSS — choć jest to najczęściej spotykany sposób jego wykorzystania — ale stanowi podstawę pozwalającą na działanie frameworka Bootstrap i podobnych. Poniżej przedstawiłem przykład elementu HTML wraz z atrybutem `class`:

```
...
<button class="btn btn-primary mt-1">Dodaj</button>
...
```

Atrybut `class` przypisuje znacznikowi `<button>` trzy klasy, których nazwy są rozdzielone spacjami: `btn`, `btn-primary` i `mt-1`. Odpowiadają one kolekcjom stylów zdefiniowanych przez framework Bootstrap, jak to przedstawiłem w tabeli 4.2.

Tabela 4.2. *Trzy klasy CSS użyte w znaczniku <button>*

Klasa	Opis
`btn`	Ta klasa powoduje zastosowanie podstawowego stylu dla przycisku. Może być stosowana dla znaczników `<button>` i `<a>`, aby zapewnić ich spójny wygląd.
`btn-primary`	Ta klasa powoduje zastosowanie stylu kontekstu dostarczającego wskazówkę o przeznaczeniu przycisku. Więcej informacji na ten temat znajdziesz w sekcji „Użycie klas kontekstu”.
`mt-1`	Ta klasa zapewnia przestrzeń między górną krawędzią elementu i otaczającą go treścią. Więcej informacji na ten temat znajdziesz w sekcji „Użycie marginesu i dopełnienia”.

Użycie klas kontekstu

Jedną z podstawowych zalet użycia frameworka CSS takiego jak Bootstrap jest uproszczenie procesu tworzenia spójnego motywu w aplikacji. Bootstrap definiuje zbiór tak zwanych **stylów kontekstu** używanych do nadania spójnego stylu powiązanym ze sobą elementom. Wspomniane style kontekstu, wymienione w tabeli 4.3, są używane w nazwach klas powodujących nadanie stylów Bootstrap elementom.

Bootstrap oferuje klasy pozwalające na stosowanie stylów kontekstu w różnego rodzaju elementach. Poniżej pokazałem przykład nadania stylu kontekstu `primary` elementowi `<h3>`. Ten element pochodzi z pliku *index.html* utworzonego na początku rozdziału.

```
...
<h3 class="bg-primary text-white p-3">Lista Adama</h3>
...
```

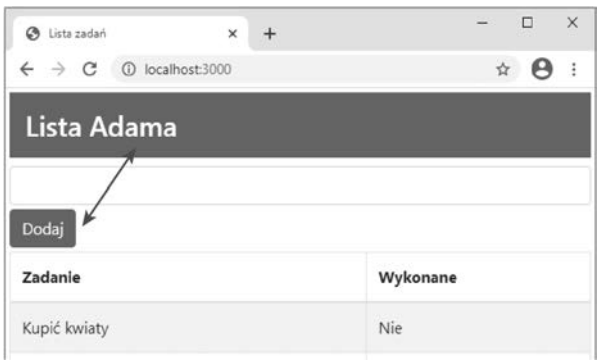
Tabela 4.3. *Style kontekstu we frameworku Bootstrap*

Klasa	Opis
`primary`	Ten kontekst służy do wskazania głównej akcji lub obszaru treści.
`secondary`	Ten kontekst służy do wskazania dodatkowego obszaru treści.
`success`	Ten kontekst służy do wskazania danych wyjściowych oznaczających sukces.
`info`	Ten kontekst służy do przedstawienia informacji dodatkowych.
`warning`	Ten kontekst służy do wyświetlania ostrzeżeń.
`danger`	Ten kontekst służy do wyświetlania poważnych ostrzeżeń.
`muted`	Ten kontekst służy do zmniejszenia znaczenia treści.
`dark`	Ten kontekst służy do zwiększenia kontrastu przez użycie ciemnego koloru.
`white`	Ten kontekst służy do zwiększenia kontrastu przez użycie koloru białego.

Jedną z klas przypisanych elementowi jest `bg-primary`. Powoduje ona zdefiniowanie koloru tła elementu na podstawie stylu koloru kontekstu. Poniżej przedstawiłem ten sam styl kontekstu zastosowany dla elementu `<button>`.

```
...
<button class="btn btn-primary mt-1">Dodaj</button>
...
```

Klasa stylu `btn-primary` powoduje nadanie elementowi `<button>` lub `<a>` stylu na podstawie kolorów użytych w stylu kontekstu. Wykorzystanie tego samego kontekstu do nadania stylu różnym elementom gwarantuje zachowanie spójności i wzajemne uzupełnianie się stylu treści, jak pokazałem na rysunku 4.3. Czerwoną strzałką wskazałem te elementy, dla których zostały zastosowane style kontekstu.



Rysunek 4.3. *Użycie stylów kontekstu pozwala na zachowanie spójności*

Użycie marginesu i dopełnienia

Bootstrap oferuje zestaw klas pomocniczych przeznaczonych do zdefiniowania dopełnienia, czyli przestrzeni między krawędzią elementu i jego treścią, oraz marginesu, czyli przestrzeni między krawędzią elementu i otaczających go elementów. Zaletą zastosowania tego rodzaju klas jest stała wielkość przestrzeni w aplikacji.

Nazwy tych klas mają doskonale zdefiniowany wzorzec. Poniżej przedstawiłem znacznik `<body>` pochodzący z pliku *index.html* utworzonego na początku rozdziału. W tym znaczniku została zastosowana klasa definiująca margines.

```
...
<body class="m-1">
...
```

Klasy definiujące margines i dopełnienie w elementach stosują doskonale opracowany schemat nazewnictwa. Najpierw mamy literę `m` oznaczającą margines (ang. *margin*) lub `p` oznaczającą dopełnienie (ang. *padding*). Następnie znajduje się myślnik. Dalej znajduje się myślnik. Na końcu mamy liczbę określającą ilość wolnej przestrzeni — `0` w przypadku braku przestrzeni bądź też `1`, `2` lub `3` dla coraz większej ilości przestrzeni. Istnieje możliwość dodania litery wskazującej na dodanie wolnej przestrzeni tylko dla określonej krawędzi: `t` oznacza górną (ang. *top*) krawędź, `b` oznacza dolną (ang. *bottom*), `l` oznacza lewą (ang. *left*), `r` oznacza prawą (ang. *right*), `x` oznacza lewą i prawą, `y` zaś górną i dolną.

Aby pomóc w zrozumieniu tego schematu nazewnictwa, w tabeli 4.4 wymieniłem klasy użyte w pliku *index.html*.

Tabela 4.4. *Przykładowe klasy Bootstrap definiujące margines i dopełnienie*

Klasa	Opis
`p-1`	Ta klasa powoduje zastosowanie dopełnienia dla wszystkich krawędzi elementu.
`m-1`	Ta klasa powoduje zastosowanie marginesu dla wszystkich krawędzi elementu.
`mt-1`	Ta klasa powoduje zastosowanie marginesu dla górnej krawędzi elementu.
`mb-1`	Ta klasa powoduje zastosowanie marginesu dla dolnej krawędzi elementu.

Zmiana wielkości elementu

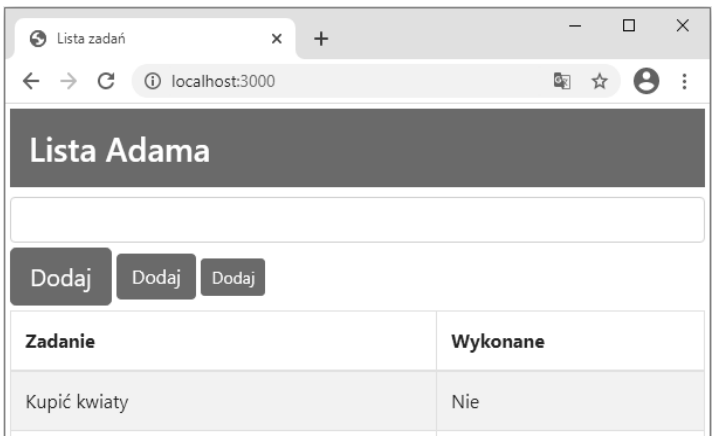
Sposób nadawania stylu pewnym elementom można zmienić przez użycie klasy wskazującej na wielkość elementu. Tego rodzaju klasa powstaje przez połączenie nazwy klasy podstawowej, myślnika i ciągu tekstowego `lg` lub `sm`. Na listingu 4.3 przedstawiłem przykład dodania do pliku *index.html* elementów `<button>` wraz z dostarczanymi przez framework Bootstrap klasami zmieniającymi wielkość elementu.

Listing 4.3. *Użycie w pliku index.html klasy zmieniającej wielkość elementu*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
</head>
<body class="m-1">
    <h3 class="bg-primary text-white p-3">Lista Adama</h3>

    <div class="my-1">
        <input class="form-control" />
```

```
        <button class="btn btn-lg btn-primary mt-1">Dodaj</button>
        <button class="btn btn-primary mt-1">Dodaj</button>
        <button class="btn btn-sm btn-primary mt-1">Dodaj</button>
    </div>

    <table class="table table-striped table-bordered">
        <thead>
            <tr>
                <th>Zadanie</th>
                <th>Wykonane</th>
            </tr>
        </thead>
        <tbody>
            <tr><td>Kupić kwiaty</td><td>Nie</td></tr>
            <tr><td>Kupić buty</td><td>Nie</td></tr>
            <tr><td>Odebrać bilety</td><td>Tak</td></tr>
            <tr><td>Zadzwonić do Janka</td><td>Nie</td></tr>
        </tbody>
    </table>
</body>
</html>
```

Klasa `btn-lg` powoduje utworzenie dużego przycisku, natomiast klasa `btn-sm` tworzy mały przycisk. Pominięcie klasy wielkości oznacza użycie domyślnej wielkości elementu. Zwróć uwagę na połączenie klas kontekstu i wielkości. Klasy modyfikujące frameworka Bootstrap współdziałają ze sobą, aby zapewnić pełną kontrolę nad sposobem nadawania stylów elementom. Efekt wprowadzonej zmiany w pliku *index.html* pokazałem na rysunku 4.4.



Rysunek 4.4. *Zmiana wielkości elementu*

Użycie frameworka Bootstrap do nadawania stylu tabeli

Framework Bootstrap zawiera także style przeznaczone do użycia w tabelach. Z tej możliwości korzystam w książce. W tabeli 4.5 wymieniłem klasy CSS frameworka Bootstrap przeznaczone do nadawania stylu tabelom.

Tabela 4.5. *Style Bootstrap przeznaczone do nadawania stylu tabeli*

Klasa	Opis
`table`	Tak klasa powoduje zastosowanie ogólnych stylów dla elementów tabeli (`<table>`) i ich treści.
`table-striped`	Tak klasa powoduje zastosowanie stylu naprzemiennego w wierszach w tabeli.
`table-bordered`	Tak klasa powoduje zdefiniowanie obramowania dla wszystkich wierszy i kolumn.
`table-hover`	Tak klasa powoduje wyświetlenie innego stylu, gdy kursor myszy zostanie umieszczony nad wierszem tabeli.
`table-sm`	Tak klasa powoduje zmniejszenie dopełnienia w tabeli, aby tym samym utworzyć zwięźlejszy układ.

Wszystkie wymienione klasy są stosowane bezpośrednio w elemencie `<table>`, jak pokazałem na listingu 4.4, na którym pogrubioną czcionką przedstawiłem style Bootstrap zastosowane w tabeli zdefiniowanej w pliku *index.html*.

Listing 4.4. *Użycie klas Bootstrap do nadawania stylu tabeli*

```
...
<table class="table table-striped table-bordered">
    <thead>
        <tr>
            <th>Zadanie</th>
            <th>Wykonane</th>
        </tr>
    </thead>
    <tbody>
        <tr><td>Kupić kwiaty</td><td>Nie</td></tr>
        <tr><td>Kupić buty</td><td>Nie</td></tr>
        <tr><td>Odebrać bilety</td><td>Tak</td></tr>
        <tr><td>Zadzwonić do Janka</td><td>Nie</td></tr>
    </tbody>
</table>
...
```

■ **Wskazówka** Podczas definiowania tabeli na listingu 4.4 zwróć uwagę na użycie elementu `<thead>`. Przeglądarka WWW automatycznie wygeneruje w `<tbody>` wszelkie niezbędne elementy `<tr>` będące bezpośrednimi elementami potomnymi `<table>`, o ile żaden nie został użyty. Jeśli będziesz polegał na tym zachowaniu podczas pracy z frameworkiem Bootstrap, otrzymasz dziwne wyniki, ponieważ większość klas CSS zastosowanych dla elementu `<table>` powoduje dodanie stylów do elementów potomnych elementu `<tbody>`.

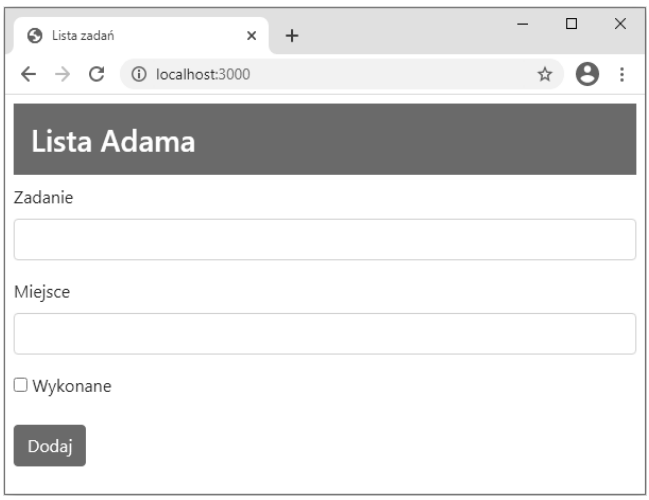
Użycie frameworka Bootstrap do tworzenia formularzy HTML

Framework Bootstrap zawiera style przeznaczone do nadawania elementom formularza HTML wyglądu spójnego wraz z pozostałymi elementami w aplikacji. Na listingu 4.5 w pliku *index.html* rozbudowałem elementy formularza i tymczasowo usunąłem tabelę.

Listing 4.5. *Przykład zdefiniowania w pliku index.html dodatkowych elementów formularza*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
</head>
<body class="m-2">
    <h3 class="bg-primary text-white p-3">Lista Adama</h3>
    <form>
        <div class="form-group">
            <label>Zadanie</label>
            <input class="form-control" />
        </div>

        <div class="form-group">
            <label>Miejsce</label>
            <input class="form-control" />
        </div>
        <div class="form-group">
            <input type="checkbox" />
            <label>Wykonane</label>
        </div>
        <button class="btn btn-primary">Dodaj</button>
    </form>
</body>
</html>
```

Nadanie podstawowych stylów formularza odbywa się poprzez zastosowanie klasy `form-group` w elemencie `<div>` zawierającym znaczniki `<label>` i `<input>`, gdzie znacznik `<input>` ma przypisaną klasę `form-control`. Style frameworka Bootstrap powodują wyświetlenie znacznika `<label>` nad elementem `<input>`, który zabiera 100% dostępnej przestrzeni poziomej, jak pokazałem na rysunku 4.5.



Rysunek 4.5. *Nadanie stylów elementom formularza*

Użycie frameworka Bootstrap do utworzenia układu opartego na siatce

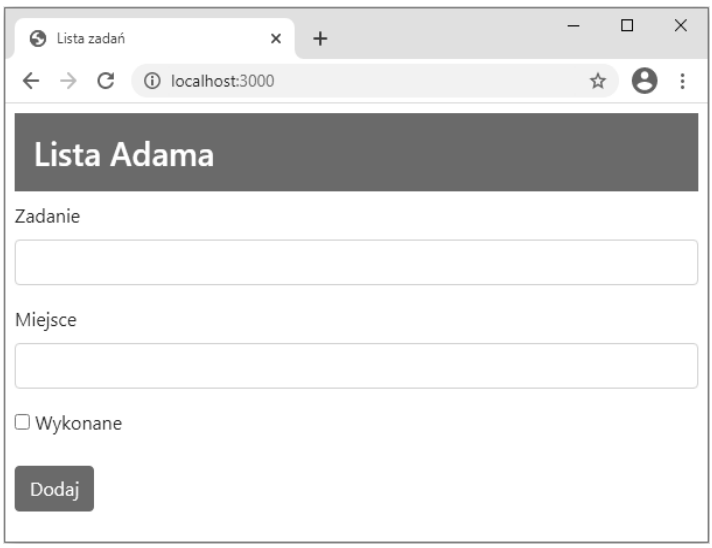
Framework Bootstrap zawiera klasy stylów przeznaczone do tworzenia różnego rodzaju układów opartych na siatce. Wspomniane układy mogą składać się z 1 – 12 kolumn i zapewniać obsługę układu responsywnego. (W tym przypadku „responsywny" oznacza, że układ siatki zmienia się na podstawie szerokości ekranu). Na listingu 4.6 pokazałem zmodyfikowaną wersję pliku *index.html* zawierającą układ oparty na siatce.

Listing 4.6. *Użycie w pliku index.html układu opartego na siatce*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
    <style>
        .row > div { border: 1px solid lightgrey; padding: 10px;
            background-color: aliceblue; margin: 5px 0; }
    </style>
</head>
<body class="m-2">
    <h3>Układ oparty na siatce</h3>
    <div class="container">
        <div class="row">
            <div class="col-1">1</div>
            <div class="col-1">1</div>
            <div class="col-2">2</div>
            <div class="col-2">2</div>
            <div class="col-6">6</div>
        </div>

        <div class="row">
            <div class="col-3">3</div>
            <div class="col-4">4</div>
            <div class="col-5">5</div>
        </div>

        <div class="row">
            <div class="col-6">6</div>
            <div class="col-6">6</div>
        </div>

        <div class="row">
            <div class="col-11">11</div>
            <div class="col-1">1</div>
        </div>

        <div class="row">
           <div class="col-12">12</div>
        </div>
    </div>
</body>
</html>
```

Oferowany przez framework Bootstrap układ oparty na siatce jest łatwy w użyciu. Najwyższego poziomu element `<div>` ma przypisaną klasę `container` (lub `container-fluid`, jeśli ma zabrać całą dostępną przestrzeń). Należy wskazać liczbę kolumn poprzez zastosowanie klasy `row` do elementu `<div>`. Ma to taki sam efekt jak przygotowanie układu opartego na siatce dla treści znajdującej się w danym elemencie `<div>`.

Każdy wiersz definiuje 12 kolumn. Liczbę kolumn zajmowanych przez każdy element potomny można określać przez przypisanie klasy, której nazwa to `col-`*`liczba_kolumn`*. Na przykład klasa `col-1` oznacza, że element zajmuje tylko jedną kolumnę. Z kolei `col-2` oznacza zabranie dwóch kolumn itd., aż do `col-12`. W przypadku klasy `col-12` element wypełnia cały wiersz. Na listingu 4.6 utworzyłem serię elementów `<div>` wraz z przypisanymi klasami `row`. Wspomniane elementy zawierały kolejne elementy `<div>`, w których również zastosowałem klasy `col-*`. Na rysunku 4.6 możesz zobaczyć wyświetlony w przeglądarce WWW układ oparty na siatce.



Rysunek 4.6. *Utworzony za pomocą frameworka Bootstrap układ oparty na siatce*

■ **Wskazówka** Framework Bootstrap nie stosuje żadnych stylów dla elementów znajdujących się w wierszu. Dlatego też na listingu 4.6 użyłem elementu `<style>` w celu zdefiniowania własnego stylu CSS określającego kolor tła, odległości między wierszami i obramowanie.

Utworzenie responsywnego układu opartego na siatce

Responsywna siatka powoduje dopasowanie układu do wielkości okna przeglądarki WWW. Podstawowym powodem tworzenia tego rodzaju układów jest umożliwienie urządzeniom mobilnym i tradycyjnym komputerom wyświetlanie tej samej treści i wykorzystanie zalet, jakie niesie ze sobą ewentualna dostępność dodatkowego miejsca na ekranie. Aby utworzyć responsywny układ oparty na siatce, klasy `col-*` w poszczególnych komórkach siatki zastąp klasami wymienionymi w tabeli 4.6.

Tabela 4.6. *Klasy frameworka Bootstrap przeznaczone do utworzenia elastycznego układu opartego na siatce*

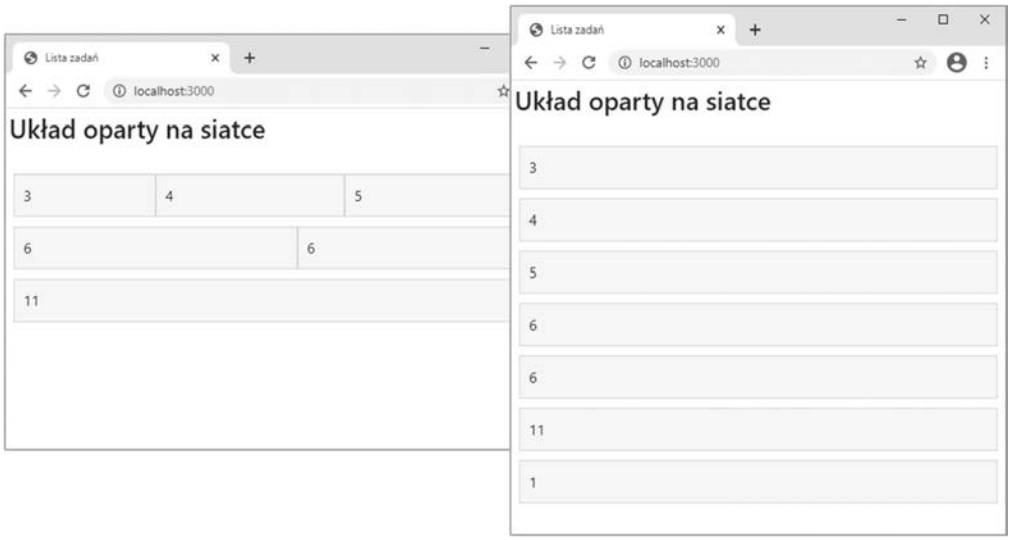
Klasa	Opis
`col-sm-*`	Komórki siatki są wyświetlane poziomo, gdy szerokość ekranu jest większa niż 576 pikseli.
`col-md-*`	Komórki siatki są wyświetlane poziomo, gdy szerokość ekranu jest większa niż 768 pikseli.
`col-lg-*`	Komórki siatki są wyświetlane poziomo, gdy szerokość ekranu jest większa niż 992 pikseli.
`col-xl-*`	Komórki siatki są wyświetlane poziomo, gdy szerokość ekranu jest większa niż 1200 pikseli.

Gdy szerokość ekranu jest mniejsza niż obsługiwana przez klasę, to komórki w wierszu siatki będą układane pionowo zamiast poziomo. Na listingu 4.7 pokazałem przykład responsywnego układu opartego na siatce zdefiniowanego w pliku *index.html*.

Listing 4.7. *Utworzenie responsywnego układu opartego na siatce*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
    <style>
        #gridContainer {padding: 20px;}
        .row > div { border: 1px solid lightgrey; padding: 10px;
            background-color: aliceblue; margin: 5px 0; }
    </style>
</head>
<body class="m-1">
    <h3>Układ oparty na siatce</h3>
    <div id="container">
        <div class="row">
            <div class="col-sm-3">3</div>
            <div class="col-sm-4">4</div>
            <div class="col-sm-5">5</div>
        </div>

        <div class="row">
            <div class="col-sm-6">6</div>
            <div class="col-sm-6">6</div>
        </div>

        <div class="row">
            <div class="col-sm-11">11</div>
            <div class="col-sm-1">1</div>
        </div>
    </div>
</body>
</html>
```

Usunąłem pewne wiersze siatki z poprzedniego przykładu i zastąpiłem klasy `col-*` klasami `col-sm-*`. W efekcie, jeśli okno przeglądarki WWW ma szerokość większą niż 576 pikseli, to wiersze są ułożone poziomo. W przypadku mniejszej szerokości okna wiersze są układane pionowo. Efekt możesz zobaczyć na rysunku 4.7.



Rysunek 4.7. *Przykład responsywnego układu opartego na siatce*

Utworzenie uproszczonego układu opartego na siatce

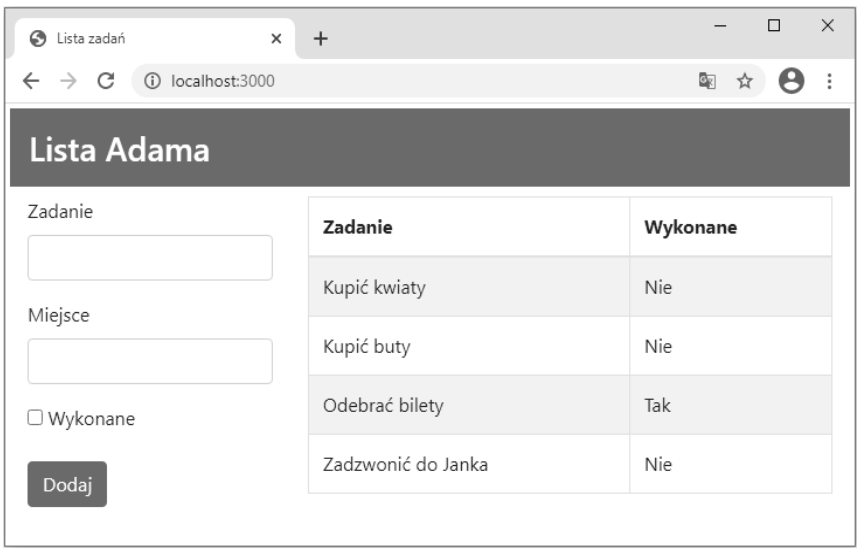
W większości przykładów w książce opierających się na siatce tworzonej za pomocą frameworka Bootstrap zdecydowałem się na zastosowanie uproszczonego podejścia. Polega ono na wyświetleniu treści w pojedynczym wierszu i podaniu jedynie liczby kolumn, jak pokazałem na listingu 4.8.

Listing 4.8. *Przykład użycia w pliku index.html uproszczonego układu opartego na siatce*

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista zadań</title>
    <meta charset="utf-8" />
    <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
          rel="stylesheet" />
</head>
<body class="m-1">
    <h3 class="bg-primary text-white p-3">Lista Adama</h3>
    <div class="container-fluid">
        <div class="row">
            <div class="col-4">
                <form>
                    <div class="form-group">
                        <label>Zadanie</label>
                        <input class="form-control" />
                    </div>
                    <div class="form-group">
                        <label>Miejsce</label>
                        <input class="form-control" />
                    </div>
                    <div class="form-group">
                        <input type="checkbox" />
                        <label>Wykonane</label>
                    </div>
```

```
                <button class="btn btn-primary">Dodaj</button>
            </form>
        </div>
        <div class="col-8">
            <table class="table table-striped table-bordered">
                <thead>
                    <tr>
                        <th>Zadanie</th>
                        <th>Opis</th>
                    </tr>
                </thead>
                <tbody>
                    <tr><td>Kupić kwiaty</td><td>Nie</td></tr>
                    <tr><td>Kupić buty</td><td>Nie</td></tr>
                    <tr><td>Odebrać bilety</td><td>Tak</td></tr>
                    <tr><td>Zadzwonić do Janka</td><td>Nie</td></tr>
                </tbody>
            </table>
        </div>
    </div>
</div>
</body>
</html>
```

W kodzie przedstawionym na listingu 4.8 zastosowałem klasy `col-4` i `col-8` w celu wyświetlenia obok siebie elementów `<div>`. To pozwala na wyświetlenie w poziomie formularza oraz tabeli wraz z listą rzeczy do zrobienia, jak pokazałem na rysunku 4.8.



Rysunek 4.8. *Przykład uproszczonego układu opartego na siatce*

Podsumowanie

W tym rozdziale zawarłem krótkie wprowadzenie do języka HTML i frameworka CSS o nazwie Bootstrap. Jeżeli chcesz osiągać zadowalające efekty w tworzeniu aplikacji internetowych, powinieneś dobrze poznać technologie HTML i CSS. Najlepszym sposobem nauki jest praktyka. Przykłady i opisy przedstawione w tym rozdziale powinny wystarczyć na początek i zapewnić Ci solidne podstawy pomagające w zrozumieniu przykładów zaprezentowanych w dalszej części książki. W kolejnym rozdziale będę kontynuować wprowadzenie, ale tym razem zajmę się podstawowymi funkcjami języka JavaScript, używanego w tej książce.

Dobre rozwiązanie dla aplikacji klienta? Z Angularem się uda!

Angular jest frameworkiem napisanym w języku TypeScript. Służy do tworzenia złożonych, nowoczesnych i wydajnych aplikacji działających po stronie klienta. Jest rozwijany przez Google i cieszy się dużym uznaniem projektantów aplikacji: ułatwia tworzenie poprawnego, czystego kodu, umożliwia bezproblemowe testowanie, a praca programisty z tym frameworkiem jest przyjemna, efektywna i satysfakcjonująca. Aby w pełni wykorzystać możliwości Angulara, koniecznie trzeba się zapoznać z architekturą jego aplikacji, a także z modułami, komponentami i powiązaniami danych.

To kolejne wydanie popularnego przewodnika, będącego starannie przygotowanym połączeniem opisu architekturalnego, specyfikacji technicznej i praktycznych wskazówek. Pozwala na niemal natychmiastowe rozpoczęcie pracy w Angularze. Na początek czeka Cię zapoznanie się ze wzorcem MVC, z jego zaletami i zastosowaniem. Następnie zaczniesz pisać funkcjonalne aplikacje — stopniowo będziesz przechodzić do coraz bardziej zaawansowanych i złożonych rozwiązań. Poszczególne przykłady zostały omówione w sposób jasny, zrozumiały i spójny. Dodatkowo w książce przedstawiono mnóstwo wskazówek i szczegółów, pozwalających na efektywną pracę i uzyskiwanie najlepszych efektów. Nie zabrakło także prezentacji najczęściej spotykanych problemów i metod ich rozwiązywania.

W książce:

- dokładne omówienie wzorca MVC
- tworzenie złożonych i dynamicznych aplikacji w Angularze
- używanie narzędzi ng podczas tworzenia projektów w Angularze
- rozszerzanie i dostosowywanie frameworka Angular
- prowadzenie testów jednostkowych dla projektów w Angularze

Adam Freeman — jest doświadczonym programistą i autorem świetnie przyjętych książek o programowaniu w Javie. Tworzył również duże systemy rozproszone (platformy e-commerce). Zajmował stanowiska kierownicze w wielu firmach, między innymi w Netscape, Sun Microsystems, NASDAQ i bankach. Obecnie jest na zasłużonej emeryturze. Swój czas przeznacza na pisanie i bieganie na długich dystansach.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-7543-7

INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 149,00 zł
Apress