Technologia i rozwiązania

Projektowanie nowoczesnych aplikacji sieciowych z użyciem AngularJS i Bootstrapa

Poznaj najpopularniejsze frameworki!

Stephen Radford

Helion

[PACKT] open source
community experience distilled
PUBLISHING

Tytuł oryginału: Learning Web Development with Bootstrap and AngularJS

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-1847-2

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/pnasab.zip

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

Printed in Poland.

Spis treści

ROZDZIAŁ 2

Projektowanie za pomocą AngularJS i Bootstrapa

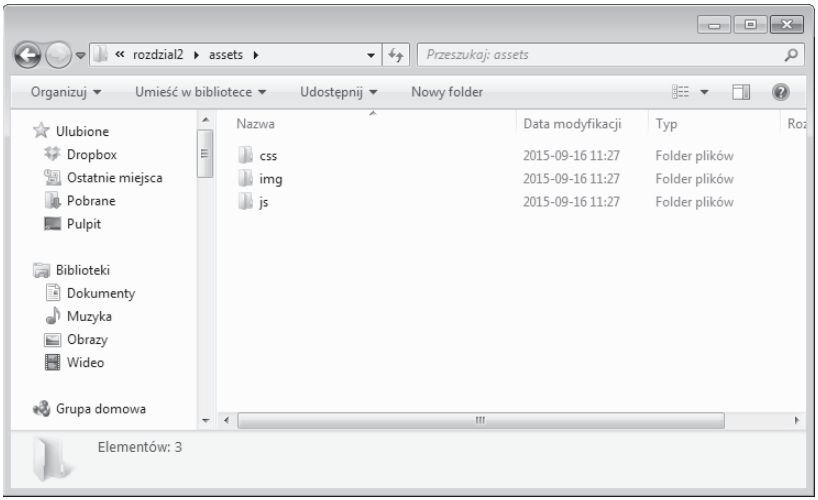
Teraz, gdy masz już za sobą opracowanie pierwszej aplikacji z użyciem AngularJS i Bootstrapa, pora podnieść poprzeczkę. W dalszej części książki będziemy używać obu frameworków do zbudowania aplikacji typu „menedżer kontaktów", wyposażonej w wyszukiwarkę tekstową oraz funkcje umożliwiające tworzenie, edytowanie i usuwanie rekordów. Zastanowimy się, jak utworzyć łatwy w utrzymaniu kod, a zarazem przyjrzymy się potencjałowi obydwu platform. A zatem do dzieła!

Przygotowania

Przygotujmy nowy katalog dla naszej aplikacji i nadajmy mu strukturę podobną jak dla programu „Witaj, świecie", opisanego w rozdziale 1. „Witaj, {{imię}}.

Struktura taka jak poniższa będzie w sam raz (patrz rysunek na następnej stronie).

Zauważ, że umieściłem znane Ci już foldery w nadrzędnym katalogu *assets*, aby uniknąć bałaganu. Skopiuj pliki Angulara i Bootstrapa z folderu rozdziału 1. „Witaj, {{imię}}" do odpowiednich folderów w nowym katalogu, natomiast w katalogu głównym utwórz plik *index.html*, który będzie podstawą naszej aplikacji do zarządzania kontaktami. Poniższy fragment kodu to bazowy dokument HTML zawierający odwołania do Bootstrapa i Angulara. Jak widać, zainicjalizowałem już Angulara przy użyciu atrybutu `ng-app`, umieszczonego w znaczniku `<html>`. Na tym etapie zawartość strony powinna być taka:




```
<!DOCTYPE html>
<html lang="pl" ng-app>
<head>
  <meta charset="utf-8">
  <title>Menedżer kontaktów</title>
  <link rel="stylesheet" href="assets/css/bootstrap.min.css">
  <script type="text/javascript" src="assets/js/angular.min.js"></script>
</head>
<body>

</body>
</html>
```

Struktura

Dysponujemy podstawową strukturą folderów oraz plikiem bazowym, możemy więc zacząć budować makietę aplikacji za pomocą Bootstrapa. Oprócz zbioru komponentów, takich jak elementy nawigacji i przyciski, których użyjemy do zbudowania struktury menedżera kontaktów, Bootstrap jest też wyposażony w elastyczny, responsywny system siatkowy, którego możliwości także wykorzystamy w naszej aplikacji.

Nawigacja

Do przełączania między widokami użyjemy komponentu `navbar`. Będzie on, rzecz jasna, znajdował się w górnej części ekranu.

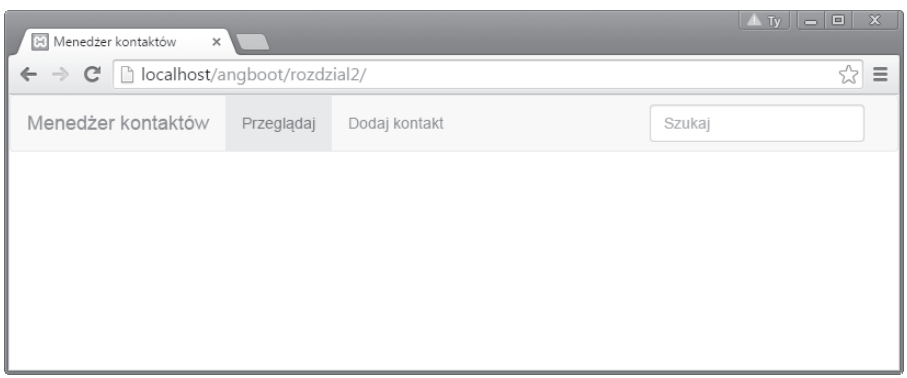
Zanim rozłożymy nawigację na składowe, przyjrzyjmy się, jak będzie wyglądała w całości:

```
<nav class="navbar navbar-default"role="navigation">
  <div class="navbar-header">
    <button type="button" class="navbar-toggle" data-toggle="collapse"
    data-target="#nav-toggle">
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="/">Menedżer kontaktów</a>
  </div>
  <div class="collapse navbar-collapse" id="nav-toggle">
    <ul class="nav navbar-nav">
      <li class="active"><a href="/">Przeglądaj</a></li>
      <li><a href="/add">Dodaj kontakt</a></li>
    </ul>
    <form class="navbar-form navbar-right" role="search">
      <input type="text" class="form-control" placeholder="Szukaj">
    </form>
  </div>
</nav>
```

Jak na coś, co ma być bardzo prostym komponentem naszej strony, ten kod wygląda na dość skomplikowany. Ale jeśli przeanalizujemy go krok po kroku, okaże się, że zawiera tylko najpotrzebniejsze składniki.

Znacznik `<nav>` obejmuje wszystkie elementy naszego paska nawigacji. Wewnątrz niego nawigacja jest podzielona na dwie sekcje: `navbar-header` i `navbar-collapse`. Te elementy odnoszą się tylko do wersji mobilnej i decydują o tym, co jest widoczne, a co ukryte po użyciu przełącznika nawigacji.

Atrybut `data-target` dla przycisku odpowiada atrybutowi `id` elementu `navbar-collapse`, dzięki czemu Bootstrap „wie", czym powinien sterować ten przycisk. Poniższy zrzut ekranu prezentuje wygląd elementów nawigacji na urządzeniach większych od tabletu.



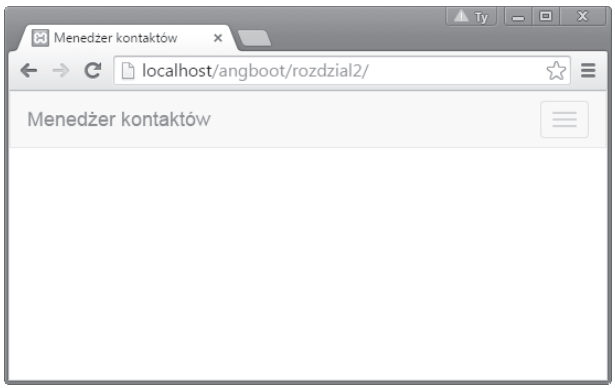
Nawigację umieścimy w znaczniku `<body>`, dzięki czemu jej elementy będą rozciągały się na całą szerokość okna przeglądarki.

Jeśli zwęzisz okno przeglądarki do szerokości poniżej 768 pikseli (szerokość ekranu iPada w ułożeniu pionowym), Bootstrap przełączy nawigację na wariant dostosowany do urządzeń mobilnych, z przełącznikiem w formie przycisku. Ale jeśli klikniesz ów przycisk, nic się nie stanie. To dlatego, że na razie nie dołączyliśmy jeszcze biblioteki ze skryptami JavaScript dla Bootstrapa, znajdującej się w pobranym wcześniej archiwum.

Skopiuj plik z tą biblioteką do katalogu *js* i dołącz ją do dokumentu w pliku *index.html*. Oprócz tego do aplikacji trzeba dołączyć bibliotekę jQuery, ponieważ wymagają jej skrypty JS dla Bootstrapa. Najnowszą wersję jQuery możesz pobrać ze strony *http://jquery.com*. Tak jak poprzednie pliki, należy umieścić ją w katalogu *js* i dołączyć w kodzie przed plikiem *bootstrap.js*. Upewnij się, że kolejność dołączania plików JavaScript w kodzie jest następująca:

```
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/angular.min.js"></script>
```

Jeśli odświeżysz stronę w przeglądarce, przycisk powinien teraz uaktywniać nawigację w wersji mobilnej.



Kilka słów o systemie siatkowym Bootstrapa

Dwunastokolumnowy system siatkowy Bootstrapa jest niezwykle elastyczny i umożliwia utworzenie makiety aplikacji responsywnej przy użyciu zaledwie kilku elementów — z wykorzystaniem modułowego charakteru CSS. Siatka Bootstrapa składa się z wierszy i kolumn, które za pomocą dostępnych klas można w różny sposób przystosowywać do własnych potrzeb. Zanim zaczniemy się nią posługiwać, powinniśmy utworzyć dodatkowy kontener na wiersze siatki, bo w przeciwnym razie cała struktura nie będzie reagować tak, jak powinna. Funkcję tego kontenera będzie pełnił zwykły znacznik `<div>`, który umieścimy pod paskiem nawigacji:

```
<div class="container"></div>
```

W rezultacie siatka zostanie wyśrodkowana, a dodanie właściwości `max-width` sprawi, że całość będzie wyglądała elegancko.

Istnieją cztery prefiksy nazw klas, decydujące o zachowaniu kolumn siatki. W większości wypadków będziemy się posługiwać prefiksem `col-sm-`. Prefiks ten sprawia, że po zwężeniu kontenera do szerokości poniżej 750 pikseli kolumny są ustawiane jedna nad drugą.

Pozostałe klasy odnoszą się do innych rozmiarów ekranu, ale ich zachowanie jest podobne. Poniższa tabela (zaczerpnięta z oficjalnej strony Bootstrapa: *http://getbootstrap.com*) ilustruje różnice między wszystkimi czterema klasami:

	Telefony (< 768 px)	Tablety (≥ 768 px)	Komputery (≥ 992 px)	Komputery (≥ 1200 px)
Zachowanie siatki	Pozioma przez cały czas	Początkowo zwinięta, pozioma po przekroczeniu określonej szerokości		
Maksymalna szerokość kontenera	Brak (automatyczna)	750 px	970 px	1170 px
Prefiks klasy	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
Maksymalna szerokość kolumny	Automatyczna	~62 px	~81 px	~97 px
Przesunięcie (offset)	Tak			
Kolejność kolumn	Tak			

Utwórzmy zatem prosty układ dwukolumnowy z obszarem na główną treść i paskiem bocznym. Ponieważ siatka składa się z 12 kolumn, musimy zadbać o dopasowanie do niej obszaru na treść, aby nie pozostawić na ekranie pustego miejsca.

Moim zdaniem osiem kolumn na treść i cztery na pasek boczny to doskonałe rozwiązanie, tylko jak zrealizować ten zamysł w praktyce?

Najpierw wewnątrz kontenera musimy utworzyć nowy znacznik `<div>` klasy `row`. Będzie to wiersz siatki, których to wierszy możemy utworzyć dowolną liczbę; każdy z nich rozciąga się na wszystkie 12 kolumn.

```
<div class="container">
   <div class="row">

   </div>
</div>
```

Ponieważ chcemy, aby na urządzeniach mobilnych kolumny automatycznie zmieniały układ na pionowy, użyjemy prefiksu `col-sm-`. Tworzenie kolumn jest bardzo proste: wystarczy wziąć żądany prefiks i dodać do niego liczbę kolumn siatki Bootstrapa, jaką ma ona obejmować. Zobaczmy, jak będzie wyglądał nasz prosty układ dwukolumnowy:

```
<div class="container">
  <div class="row">
    <div class="col-sm-8">
      To jest obszar na treść.
    </div>
    <div class="col-sm-4">
      To jest pasek boczny.
    </div>
  </div>
</div>
```

Po wyświetleniu układu na ekranie większym niż ekran urządzenia mobilnego Bootstrap automatycznie doda 30 pikseli odstępu między kolumnami (po 15 pikseli z każdej strony). W niektórych wypadkach będzie nam jednak zależało na zwiększeniu tego odstępu albo przeciwnie — na nieznacznym ścieśnieniu kolumn. Bootstrap umożliwia takie modyfikacje za pośrednictwem kolejnych klas, które należy dodać do kolumny.

Tak jak przy określaniu szerokości, trzeba w tym celu użyć odpowiedniego prefiksu — tym razem ze słowem `offset`:

```
<div class="col-sm-4 col-sm-offset-1"></div>
```

W tej sytuacji liczba na końcu oznacza liczbę kolumn siatki, względem których chcesz dokonać przesunięcia kolumny projektu. W rezultacie zwiększa się margines po jej lewej stronie.

> Pamiętaj, że przesunięcia kolumn są uwzględniane w łącznej liczbie 12 kolumn Bootstrapa w wierszu.

Wewnątrz istniejących kolumn możemy zagnieździć kolejne wiersze i kolumny, aby otrzymać bardziej skomplikowany szablon. Zobaczmy:

```
<div class="container">
  <div class="row">
    <div class="col-sm-8">
      <div class="row">
        <div class="col-sm-6">
          <p>Lorem ipsum dolor...<p>
        </div>
        <div class="col-sm-6">
          <p>Class aptent taciti...</p>
          </div>
        </div>
      </div>
   </div>
</div>
```

Ten kod tworzy dwie kolumny w przeznaczonym na główną treść kontenerze, który przygotowaliśmy wcześniej. Aby było to widoczne, wstawiłem w te kolumny fragmenty tekstu roboczego.

Jeśli otworzysz tak przygotowany dokument w przeglądarce, zapewne zauważysz, że strona została podzielona na trzy kolumny. Dzięki zagnieżdżeniu siatki zawsze możemy jednak utworzyć nowy wiersz z jedną kolumną, trzema kolumnami albo dowolną ich liczbą, jaka będzie potrzebna.

Klasy pomocnicze

Bootstrap jest wyposażony w kilka **klas pomocniczych**, których można użyć do dostosowania projektu. Są to klasy użytkowe, mające jedno określone zastosowanie. Przyjrzyjmy się kilku przykładom.

Elementy „pływające"

Zastosowanie elementów „pływających" jest niekiedy konieczne do uzyskania oczekiwanej struktury strony WWW. Bootstrap jest wyposażony w dwie klasy, umożliwiające przesuwanie elementów w lewą albo w prawą stronę:

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
```

Aby skutecznie się posługiwać elementami „pływającymi", trzeba je ująć w klasę `clearfix`. Dzięki temu zostaną one wyodrębnione ze struktury dokumentu i nie będą zaburzały jego układu:

```
<div class="clearfix">
  <div class="pull-left">...</div>
  <div class="pull-right">...</div>
</div>
```

Jeśli klasy „pływające" zostaną użyte bezpośrednio w odniesieniu do elementu klasy `row`, to nie trzeba dodatkowo troszczyć się o ręczne anulowanie „pływania" za pomocą klasy `clearfix`, ponieważ Bootstrap zrobi to automatycznie.

Wyśrodkowywanie elementów

Oprócz uaktywnienia „pływania" czasami zachodzi potrzeba wyśrodkowania elementów blokowych. Bootstrap umożliwia to za pośrednictwem klasy `center-block`:

```
<div class="center-block">...</div>
```

Powoduje ona ustawienie właściwości `margin-left` i `margin-right` na `auto`, co w konsekwencji wyśrodkowuje element.

Wyświetlanie i ukrywanie

Za pomocą CSS da się ukrywać i wyświetlać elementy, a Bootstrap jest wyposażony w dwie klasy, które to umożliwiają:

```
<div class="show">...</div>
<div class="hidden">...</div>
```

Warto zauważyć, że klasa `show` powoduje zmianę sposobu wyświetlania elementu na blokowy, należy jej więc używać tylko do elementów typu `block`, a nie wierszowych (`inline`) czy mieszanych (`inline-block`).

Bootstrap oferuje ponadto wiele klas umożliwiających ukrywanie i wyświetlanie elementów przy określonych rozmiarach ekranu. Klasy te bazują na tych samych predefiniowanych wielkościach ekranu, co siatka Bootstrapa.

Na przykład poniższy kod spowoduje ukrycie danego elementu przy określonej wielkości ekranu:

```
<div class="hidden-md"></div>
```

Następujący kod zaś spowoduje ukrycie elementu na urządzeniach średniej wielkości, ale element ten będzie nadal widoczny na telefonach, tabletach i zwykłych komputerach. Aby ukryć element na różnych urządzeniach, trzeba zastosować odpowiednio wiele klas:

```
<div class="hidden-md hidden-lg"></div>
```

Na tej samej zasadzie, tylko na odwrót, działają klasy uwidaczniające elementy przy określonych wielkościach ekranu. Jednak w odróżnieniu od klas typu `hidden` wymagają one określenia metody wyświetlania: `block`, `inline` albo `inline-block`:

```
<div class="visible-md-block"></div>
<div class="visible-md-inline"></div>
<div class="visible-md-inline-block"></div>
```

Oczywiście poszczególnych klas można używać w parach. Jeśli na przykład chcielibyśmy, aby na mniejszym ekranie dany element był wyświetlany jako blokowy, na większym zaś — w mieszanym trybie `inline-block`, moglibyśmy użyć następującego kodu:

```
<div class="visible-sm-block visible-md-inline-block"></div>
```

Jeśli zapomnisz nazw poszczególnych klas, po prostu ponownie zerknij do podpunktu „Kilka słów o systemie siatkowym Bootstrapa”.

Zastosowanie dyrektyw

Miałeś już okazję użyć **dyrektyw** Angulara, choć na razie niewiele wspominałem na ich temat. W istocie są to potężne funkcje, które można wywołać z poziomu atrybutu albo nawet własnego niestandardowego elementu — w Angularze jest mnóstwo takich funkcji. Dzięki nim operacje takie jak zapętlanie przetwarzania danych, obsługa kliknięć czy wysyłanie formularzy są szybkie i proste.

Po raz pierwszy użyłeś dyrektywy do zainicjalizowania Angulara na stronie; jak zapewne pamiętasz, była to dyrektywa `ng-app`. Wszystkich dyrektyw, z którymi będziesz miał do czynienia w tym rozdziale, używa się podobnie — poprzez dodanie atrybutu do elementu.

Zanim przyjrzymy się kolejnym wbudowanym dyrektywom, musimy napisać prosty kontroler. Utwórz nowy plik i nazwij go *controller.js*. Zapisz go w folderze *js* projektu i otwórz w edytorze tekstowym.

Zgodnie z tym, czego dowiedziałeś się w rozdziale 1., zatytułowanym „Witaj, {{imię}}", kontrolery to standardowe konstruktory JavaScript, do których możemy wstrzyknąć (ang. *inject*) usługi Angulara, takie jak `$scope`. Instancje tych konstruktorów są generowane w chwili wykrycia przez Angulara atrybutu `ng-controller`. To oznacza, że w ramach danej aplikacji możemy dysponować wieloma instancjami danego kontrolera, to zaś pozwala na wielokrotne użytkowanie kodu. W wypadku naszego kontrolera wystarczy prosta deklaracja funkcji:

```
function AppCtrl(){
}
```

Aby poinformować Angulara, że chcemy użyć tego kontrolera, powinniśmy umieścić go na stronie już po załadowaniu biblioteki AngularJS. Musimy też pamiętać o dodaniu dyrektywy `ng-controller` do otwierającego znacznika `<html>`:

```
<html ng-controller="AppCtrl">
…
<script type="text/javascript" src="assets/js/controller.js"></script>
```

Dyrektywy ng-click i ng-mouseover

Jedną z najprostszych rzeczy, jaką można zrobić w JavaScripcie, jest przechwycenie zdarzenia kliknięcia. Można tego dokonać za pomocą atrybutu `onclick` dla elementu, biblioteki jQuery albo detektora zdarzenia (ang. *listener*). W Angularze używa się do tego dyrektyw.

Aby zademonstrować ich działanie, utworzymy przycisk powodujący wyświetlenie okienka z komunikatem — prosta sprawa. Najpierw dodajmy przycisk do przygotowanego wcześniej obszaru treści:

```
<div class="col-sm-8">
  <button>Kliknij mnie</button>
</div>
```

Jeśli otworzysz tak zmodyfikowaną stronę w przeglądarce, zobaczysz na niej zwykły, standardowy przycisk HTML, co nie powinno stanowić zaskoczenia. Zanim dołączymy dyrektywę do tego elementu, musimy jeszcze umieścić odpowiedni uchwyt w kontrolerze. Będzie to funkcja kontrolera, powiązana z jego zakresem. To bardzo ważne, by funkcja była powiązana z zakresem, bo w przeciwnym razie nie będzie do niej dostępu z poziomu widoku:

```
function AppCtrl($scope){
  $scope.clickHandler = function(){
    window.alert('Kliknięty!');
  };
}
```

Jak już wiesz, na stronie może funkcjonować wiele zakresów, które są po prostu obiektami, do których Angular umożliwia dostęp widokom i kontrolerowi. Aby zapewnić ów dostęp kontrolerowi, wstrzyknęliśmy do niego usługę `$scope`. Usługa ta zapewnia dostęp do zakresu związanego z elementem, do którego dodaliśmy atrybut `ng-controller`.

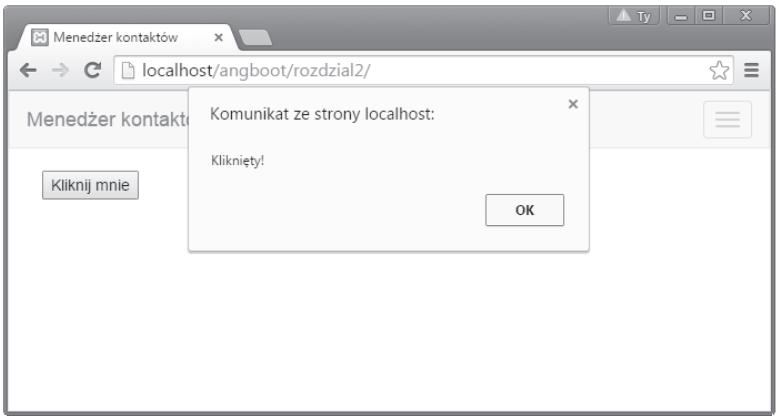
Funkcjonowanie Angulara w dużej mierze opiera się na **wstrzykiwaniu zależności** — to pojęcie, z którym być może nie miałeś dotychczas do czynienia. Mogłeś się już przekonać, że Angular jest podzielony na moduły i usługi. Każdy moduł i usługa są zależne od innych, a wstrzykiwanie zależności zapewnia przejrzystość relacji między nimi. Podczas testów jednostkowych można wstrzykiwać specjalnie spreparowane obiekty, umożliwiające zweryfikowanie poprawności działania danego elementu aplikacji. Wstrzykiwanie zależności umożliwia poinformowanie Angulara, od jakich usług zależy działanie naszego kontrolera. Framework dba o to, by usługi te były dostępne.

Kompleksowe omówienie wstrzykiwania zależności w AngularJS znajdziesz w oficjalnej dokumentacji pod adresem *https://docs.angularjs.org/guide/di*.

Uchwyt jest już przygotowany, teraz trzeba jeszcze dodać dyrektywę do przycisku. Tak jak poprzednio, należy to zrobić w postaci dodatkowego atrybutu. Tym razem przekażemy do niego nazwę funkcji, którą zamierzamy wykonać, czyli w tym wypadku `clickHandler`. Angular potraktuje całą zawartość dyrektywy jako wyrażenie, trzeba zatem pamiętać o dodaniu nawiasów wskazujących, że mamy do czynienia z funkcją:

```
<button ng-click="clickHandler()">Kliknij mnie</button>
```

Jeśli teraz otworzysz tę stronę w przeglądarce, to po kliknięciu przycisku pojawi się na ekranie okno komunikatu. Zauważ, że podczas wywoływania funkcji nie musimy się w widoku odwoływać do zmiennej `$scope`. Funkcje i zmienne, z których można skorzystać w danym widoku, funkcjonują w bieżącym zakresie lub dowolnym zakresie nadrzędnym.



Jeśli chcielibyśmy wyświetlić okno komunikatu w reakcji na wskazanie elementu kursorem myszy, a nie kliknięcie, to wystarczyłoby zmienić nazwę dyrektywy na `ng-mouseover`, gdyż działanie obydwu jest identyczne.

Dyrektywa ng-init

Dyrektywa `ng-init` służy do przetwarzania wyrażeń w bieżącym zakresie i może być używana samodzielnie lub w połączeniu z innymi dyrektywami. Ma ona wyższy priorytet niż inne dyrektywy, aby wyrażenie zostało przetworzone odpowiednio wcześnie.

Oto prosty przykład zastosowania dyrektywy `ng-init`:

```
<div ng-init="test = 'Witaj, świecie.'"></div>
{{test}}
```

Otwarcie strony z tym kodem w przeglądarce spowoduje wyświetlenie napisu „Witaj, świecie.”. Jak widać, przypisaliśmy wartość modelowi o nazwie `test`, a potem — za pomocą składni z podwójnymi nawiasami klamrowymi — wyświetliliśmy ją na ekranie.

Dyrektywy ng-show i ng-hide

W pewnych sytuacjach przydaje się możliwość sterowania wyświetlaniem jakiegoś elementu na drodze programowej. Dyrektywami `ng-show` i `ng-hide` można sterować za pośrednictwem wartości zwracanych przez funkcję albo model.

Spróbujmy rozbudować utworzoną wcześniej funkcję `clickHandler`, aby zademonstrować zastosowanie dyrektywy `ng-click` do przełączania widoczności elementu. Utworzymy w tym celu nowy model i będziemy sterować widocznością elementu poprzez przełączanie między logiczną prawdą a fałszem.

Zacznijmy od utworzenia elementu, który będziemy ukrywać i wyświetlać. W kodzie pod przyciskiem umieść następujący fragment:

```
<div ng-hide="isHidden">
  Kliknij przycisk powyżej, aby przełączyć widoczność.
</div>
```

Modelem jest tutaj wartość atrybutu `ng-hide`. Ponieważ funkcjonuje on w ramach naszego zakresu, możemy łatwo zmodyfikować go za pośrednictwem kontrolera:

```
$scope.clickHandler = function(){
  $scope.isHidden = !$scope.isHidden;
};
```

Ten kod po prostu zamienia wartość modelu na przeciwną, co w rezultacie przełącza widoczność elementu `<div>`.

Jeśli wypróbujesz ten przykład w przeglądarce, okaże się, że element nie jest domyślnie ukryty. Istnieje kilka sposobów na zmianę tego stanu rzeczy. Pierwszy polega na ustawieniu wartości `$scope.isHidden` na `true` w kontrolerze. Można też zmienić wartość tej zmiennej na `true` za pomocą dyrektywy `ng-init`. Ewentualnie moglibyśmy użyć dyrektywy `ng-show`, która działa odwrotnie niż `ng-hide` i wyświetla element wtedy, gdy wartość modelu wynosi `true`.

> Pamiętaj o załadowaniu Angulara w nagłówku strony, bo w przeciwnym razie dyrektywy `ng-hide` i `ng-show` nie zadziałają. To dlatego, że Angular korzysta z własnych klas do ukrywania elementów i muszą one zostać załadowane przed wyświetleniem strony.

Dyrektywa ng-if

Angular jest ponadto wyposażony w dyrektywę `ng-if`, której działanie jest podobne do `ng-show` i `ng-hide`. Z tym że `ng-if` fizycznie usuwa element z drzewa DOM, podczas gdy `ng-show` i `ng-hide` tylko przełączają jego widoczność.

Przyjrzyjmy się, jak można użyć dyrektywy `ng-if` w odniesieniu do poprzedniego kodu:

```
<div ng-if="isHidden">
  Kliknij przycisk powyżej, aby przełączyć widoczność.
</div>
```

Jeśli chcielibyśmy odwrócić działanie instrukcji, wystarczyłoby poprzedzić wyrażenie wykrzyknikiem:

```
<div ng-if="!isHidden">
  Kliknij przycisk powyżej, aby przełączyć widoczność.
</div>
```

Dyrektywa ng-repeat

Podczas projektowania aplikacji internetowej zapewne bardzo szybko staniesz przed koniecznością wygenerowania tablicy elementów. Na przykład w naszym menedżerze kontaktów będzie to lista kontaktów, ale może to być cokolwiek innego. Angular umożliwia realizowanie tego typu zadań za pomocą dyrektywy `ng-repeat`.

Oto przykładowe dane, z którymi moglibyśmy mieć do czynienia. Jak widać, jest to tablica obiektów z wieloma właściwościami. Aby wyświetlić dane, trzeba będzie odwołać się do każdej z tych właściwości. Na szczęście dyrektywa `ng-repeat` poradzi sobie z tym doskonale.

Oto kontroler z tablicą obiektów z danymi, przypisaną do modelu `contacts`:

```
function AppCtrl($scope){

    $scope.contacts = [
        {
            name: 'Janko Walski',
            phone: '01234567890',
            email: 'janko@walski.com'
        },
        {
            name: 'Karen Nerka',
            phone: '09876543210',
            email: 'karenne@email.com'
        }
    ];
}
```

Mamy tutaj do czynienia z zaledwie dwiema osobami, ale nietrudno sobie wyobrazić, że tablica mogłaby zawierać setki kontaktów pozyskanych za pośrednictwem API, a z taką ilością informacji trudno byłoby pracować bez dyrektywy `ng-repeat`.

Zacznij od umieszczenia w kontrolerze tablicy kontaktów i przypisania jej do właściwości `$scope.contacts`. Następnie otwórz plik *index.html* i utwórz znacznik `<ul>`. Ponieważ będziemy wielokrotnie powtarzać cały punkt nienumerowanej listy, dyrektywę `ng-repeat` trzeba dodać do elementu `<li>`:

```
<ul>
  <li ng-repeat="contact in contacts"></li>
</ul>
```

Jeśli wiesz, jak funkcjonują pętle w PHP albo w Ruby, to poczujesz się jak w domu. Utworzymy zmienną, do której będziemy mogli się odwołać w ramach powtarzanego elementu. Zmienna po słowie kluczowym `in` odwołuje się do modelu powiązanego z obiektem `$scope` w naszym kontrolerze. W rezultacie możemy odwołać się do dowolnej właściwości tego obiektu, zyskując nowy zakres przy każdej iteracji albo nowej pozycji. Właściwości te możemy wyświetlić na stronie za pomocą składni z podwójnym nawiasem klamrowym Angulara, zgodnie z informacjami podanymi w rozdziale 1., zatytułowanym „Witaj, {{imię}}”:

```
<ul>
  <li ng-repeat="contact in contacts">
    {{contact.name}}
  </li>
</ul>
```

Powyższy kod, zgodnie z oczekiwaniami, spowoduje wyświetlenie imion i nazwisk z naszej listy. W analogiczny sposób możemy łatwo odwołać się do dowolnej właściwości obiektu `contacts`, korzystając ze standardowej składni z kropką.

Dyrektywa ng-class

Często zachodzi potrzeba zmodyfikowania klasy albo dodania jej do elementu w sposób programowy. W tym celu możemy użyć dyrektywy `ng-class`. Umożliwia ona zdefiniowanie klasy do dodania lub usunięcia, w zależności od wartości modelu.

Dyrektywą `ng-class` można się posłużyć na dwa sposoby. W najprostszej postaci Angular po prostu użyje wartości modelu jako klasy CSS elementu:

```
<div ng-class="exampleClass"></div>
```

Jeśli model, do którego się odwołujemy, będzie niewłaściwy lub zwróci fałsz logiczny, klasa nie zostanie zastosowana. Takie rozwiązanie dobrze sprawdza się przy pojedynczych klasach, ale jeśli chciałbyś mieć trochę większą kontrolę nad całym procesem albo zastosować wiele klas do jednego elementu, wypróbuj poniższą składnię:

```
<div ng-class="{className: model, class2: model2}"></div>
```

To podejście jest trochę inne. Mamy zbiór nazw klas połączonych z modelami, których wartość chcemy sprawdzić. Jeśli model zwróci wartość `true`, klasa zostanie dodana do elementu.

Przyjrzyjmy się temu w praktyce. Użyjemy pól wyboru i atrybutu `ng-model`, poznanego przez Ciebie w rozdziale 1. „Witaj, {{imię}}”, do przypisania kilku klas do akapitu:

```
<p ng-class="{'text-center': center, 'text-danger': error}">
  Lorem ipsum dolor sit amet
</p>
```

Wykorzystałem tutaj dwie klasy Bootstrapa: `text-center` oraz `text-danger`. Ich zastosowanie jest uzależnione od dwóch modeli, na które możemy wpływać przy użyciu pól wyboru:

```
<label><input type="checkbox" ng-model="center">tekst wyśrodkowany</label>
<label><input type="checkbox" ng-model="error">tekst ostrzegawczy</label>
```

> Ujmowanie nazw klas w pojedyncze cudzysłowy jest potrzebne, tylko jeśli zawierają one myślniki; w przeciwnym razie bowiem Angular wygeneruje błąd.

Zaznaczenie pól spowoduje zastosowanie odpowiednich klas względem podanego wcześniej elementu.

Dyrektywa ng-style

Ta dyrektywa służy do dynamicznej zmiany stylu elementu za pośrednictwem Angulara, na podobnej zasadzie co `ng-class`. Aby to zademonstrować, utworzymy trzecie pole wyboru, które doda nowe style do poprzedniego elementu — akapitu.

Dyrektywa `ng-style` bazuje na standardowym obiekcie JavaScript, gdzie klucze obiektu odpowiadają stylom przeznaczonym do zmiany (takim jak kolor albo tło). Można je zastosować z poziomu modelu lub za pośrednictwem wartości zwróconej przez funkcję.

Zobaczmy, jak powiązać tę dyrektywę z funkcją sprawdzającą stan modelu. Dzięki temu będziemy mogli włączać i wyłączać style przy użyciu pola wyboru.

Najpierw otwórz plik *controller.js* i utwórz nową funkcję powiązaną z zakresem. Swoją funkcję nazwałem `styleDemo`:

```
$scope.styleDemo = function(){
    if(!$scope.styler){
        return;
    }
    return {
        background: 'red',
        fontWeight: 'bold'
    };
};
```

Wewnątrz funkcji musimy sprawdzić wartość modelu, który w tym wypadku nosi nazwę `styler`. Jeśli będzie to wartość `false`, nie musimy niczego zwracać — w przeciwnym razie zwracamy obiekt z właściwościami CSS. Zauważ, że w zwróconym obiekcie użyliśmy nazwy właściwości w formie `fontWeight` zamiast `font-weight`. Obie nazwy zadziałają, a Angular automatycznie przełączy zapis w stylu CamelCase na odpowiednią nazwę właściwości CSS. Trzeba jedynie pamiętać, że jeśli klucze obiektu JavaScript zawierają myślniki, klucze te trzeba ująć w cudzysłowy.

Ten model zostanie powiązany z polem wyboru, tak jak zrobiliśmy to w wypadku dyrektywy `ng-class`:

```
<label><input type="checkbox" ng-model="styler">ng-style</label>
```

Ostatnia rzecz polega na dodaniu dyrektywy `ng-style` do elementu akapitu:

```
<p .. ng-style="styleDemo()">
  Lorem ipsum dolor sit amet
</p>
```

Angular jest na tyle „sprytny”, że wywoła tę funkcję ponownie przy każdej zmianie zakresu. To oznacza, że gdy tylko wartość modelu zmieni się z `false` na `true`, nasz styl zostanie włączony, a w sytuacji odwrotnej — wyłączony.

Dyrektywa ng-cloak

Na koniec przyjrzymy się dyrektywie `ng-cloak`. Jeśli w kodzie strony HTML zostaną zastosowane szablony Angulara, to podczas wyświetlania strony w przeglądarce — tuż przed wczytaniem AngularJS i skompilowaniem kodu — na ekranie przez chwilę będą widoczne podwójne

nawiasy klamrowe. Aby tego uniknąć, można tymczasowo ukryć szablon aż do chwili jego pełnego przetworzenia.

W Angularze służy do tego dyrektywa `ng-cloak`. Definiuje ona dla wczytywanego elementu dodatkową regułę w postaci `display: none !important;`.

> Aby mieć pewność, że podczas wczytywania strony nic nie będzie migać, należy pamiętać o załadowaniu Angulara w nagłówku (<head>) strony HTML.

Pytania sprawdzające

1. Co dodaliśmy na górze strony, aby umożliwić przełączanie między widokami?
2. Z ilu kolumn składa się system siatkowy Bootstrapa?
3. Co to są dyrektywy i jak się najczęściej ich używa?
4. Jaką dyrektywę zastosowałbyś do zapętlenia przetwarzania danych?

Podsumowanie

W tym rozdziale omówiliśmy wiele zagadnień, zanim więc przejdziemy do kolejnego, podsumujmy zdobyte wiadomości.

Bootstrap pozwolił nam błyskawicznie opracować responsywną nawigację. Aby umożliwić korzystanie z nawigacji dostosowanej do urządzeń mobilnych, musieliśmy dołączyć do projektu plik JavaScript znajdujący się w archiwum z Bootstrapem.

Przyjrzeliśmy się też responsywnemu mechanizmowi siatkowemu o ogromnych możliwościach, na którym opiera się Bootstrap, i utworzyliśmy prosty, dwukolumnowy układ treści. Przy okazji zapoznaliśmy się z czterema prefiksami dla różnych klas kolumn oraz z zagnieżdżaniem elementów siatki. Aby przystosować układ strony do naszych potrzeb, wspomnieliśmy ponadto o kilku dostępnych w Bootstrapie klasach pomocniczych, odpowiedzialnych za „pływanie", wyśrodkowywanie i ukrywanie elementów.

W tym rozdziale zapoznaliśmy się też z wbudowanymi dyrektywami Angulara: funkcjami, które Angular pozwala wywoływać z poziomu widoku. Zanim się nimi zajęliśmy, musieliśmy utworzyć tak zwany kontroler, czyli funkcję, do której możemy przekazywać usługi Angulara za pośrednictwem procesu zwanego wstrzykiwaniem zależności.

Dyrektywy omówione w tym rozdziale będą odgrywały bardzo ważną rolę w programowaniu menedżera kontaktów w dalszej części książki. Dyrektywy takie jak `ng-click` i `ng-mouseover`

są po prostu nowymi sposobami obsługi zdarzeń, które dotychczas z pewnością obsługiwałeś w inny sposób, na przykład za pośrednictwem jQuery albo czystego JavaScriptu. Z kolei dyrektywy w rodzaju `ng-repeat` zapewne będą dla Ciebie nowością. „Przemycają" one do widoku pewne przydatne mechanizmy, na przykład pętle umożliwiające cykliczne przetwarzanie danych i wyświetlanie ich na stronie.

Przyjrzeliśmy się też dyrektywom, które śledzą modele w danym zakresie i zależnie od ich wartości wykonują różne działania. Z kolei dyrektywy takie jak `ng-show` i `ng-hide` umożliwiają wyświetlenie elementu na podstawie wartości modelu. Kolejnym przykładem była dyrektywa `ng-class`, która — na bazie wartości modeli — umożliwiła nam dodanie do elementów pewnych klas CSS.

Skorowidz

G

H

I

J

K

Ł

M

N

O

P

R

S

T

U

W

Z

Projektowanie nowoczesnych aplikacji sieciowych z użyciem AngularJS i Bootstrapa

Osoby, które projektują nowoczesne aplikacje internetowe, z pewnością natknęły się na frameworki Bootstrap i AngularJS. Są to świetne projekty open source, dzięki którym praca przy tworzeniu aplikacji internetowych o różnej skali staje się wydajna i przyjemna, a kod powstający w JavaScripcie i CSS jest wyjątkowo poprawny strukturalnie. Jeśli doszedłeś do wniosku, że warto rozwinąć swoje umiejętności i wyjść poza HTML oraz CSS przy pisaniu nowoczesnych aplikacji sieciowych, to Bootstrap i AngularJS stanowią doskonałą propozycję.

Książka, którą trzymasz w dłoniach, została pomyślana jako przystępny, bogato ilustrowany przewodnik po Bootstrapie i AngularJS. Jeśli masz podstawowe umiejętności w zakresie HTML, CSS i JavaScriptu, to dzięki niej zdobędziesz, a następnie rozwiniesz praktyczną wiedzę dotyczącą obu frameworków. Niejako przy okazji poznasz kilka istotnych koncepcji i ciekawych narzędzi, które okażą się zaskakująco przydatne przy programowaniu. Autor wyczerpująco, w interesujący sposób pokazuje, jak się pisze i rozwija aplikacje sieciowe.

Odkryj AngularJS, zaprzyjaźnij się z Bootstrapem — i pisz świetne projekty!

Dzięki tej książce:

- zapoznasz się z frameworkami AngularJS i Bootstrap
- nauczysz się korzystać z filtrów, routingu i widoków, a także zgłębisz tajniki filozofii CRUD
- krok po kroku napiszesz w pełni funkcjonalną aplikację sieciową
- poznasz ciekawe projekty związane z AngularJS i Bootstrapem

Stephen Radford — wszechstronny programista aplikacji sieciowych. Pracował w wielu firmach i zrealizował wiele ciekawych projektów, takich jak FTPloy — narzędzie typu SaaS, ułatwiające ciągłą aktualizację zmian w projektach. Obecnie prowadzi firmę Cocoon, specjalizującą się w aplikacjach sieciowych. Współpracuje ze startupami i przedsiębiorstwami w zakresie rozwijania nowych idei i tworzenia na ich podstawie aplikacji internetowych.

[PACKT] open source PUBLISHING community experience distilled

Helion

40418 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

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

ISBN 978-83-283-1847-2

cena: 39,90 zł