Capitolo 1

Hello, {{name}}

Il modo migliore per imparare a programmare è scrivere codice, ed è proprio questo ciò che faremo. Per comprendere quanto è facile essere subito operativi con Bootstrap e AngularJS, realizzeremo un'applicazione molto semplice che ci consentirà di digitare un nome e visualizzarlo sulla pagina in tempo reale. Scoprirete così l'efficacia del binding dei dati bidirezionale e il linguaggio per template integrato di Angular. Utilizzeremo Bootstrap per attribuire uno stile e una struttura all'app.
Prima di installare i framework, creeremo la struttura delle cartelle e il file `index.html` che sarà la base dell'app.

In questo capitolo

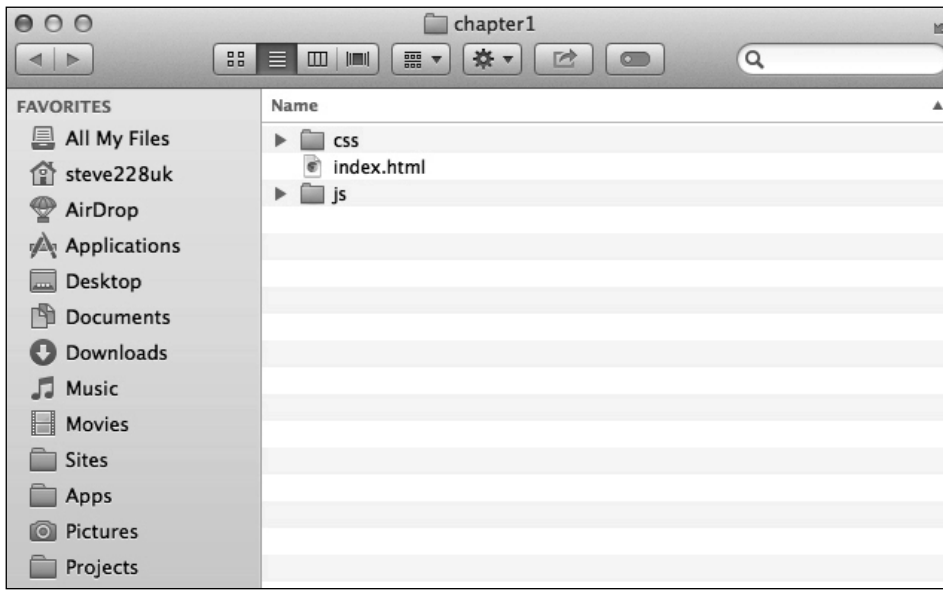
Impostazione

Per realizzare l'app con Angular e Bootstrap, procediamo all'impostazione, che consiste nel creare una pagina HTML e includere alcuni file. Innanzitutto create una nuova directory, *chapter1*, e apritela nel vostro editor. Create al suo interno un nuovo file, `index.html`, e immettete questo codice boilerplate:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title></title>
</head>
<body>
```

```
</body>
</html>
```

Si tratta di una pagina HTML standard con la quale opereremo dopo aver integrato Angular e Bootstrap.
Create due cartelle all'interno della cartella *chapter1*: *css* e *js*. La struttura completa delle cartelle dovrebbe essere simile alla seguente.



Installazione di AngularJS e Bootstrap

Installare questi framework è semplice come includere file CSS o JavaScript nella pagina. Lo potremmo fare con un *content delivery network* (CDN) come Google Code o MaxCDN, ma per ora recupereremo i file manualmente. Esaminiamo i passi che dovreste conoscere quando integrerete AngularJS e Bootstrap nel progetto.

Installazione di Bootstrap

Andate all'indirizzo `http://getbootstrap.com` e fate clic sul pulsante *Download Bootstrap*. Otterrete un file ZIP con l'ultima versione di Bootstrap che comprende CSS, font e file JavaScript. Le versioni precedenti includevano una directory delle immagini, che nella Versione 3 diventa *icon fonts*.
Per quest'app, ci interessa per ora soltanto un file: `bootstrap.min.css` presente nella directory *css*. Il foglio di stile fornisce tutta la struttura e quegli elementi graziosi, tra cui pulsanti e messaggi di avviso, per cui Bootstrap è conosciuto. Copiatelo nella directory *css* del progetto e aprite il file `index.html` nell'editor di testo.

Integrare Bootstrap è facile come collegare il file CSS che abbiamo appena copiato. Aggiungete quanto segue all'interno del tag <head>. Inserite questo tag nell'elemento <head> della pagina:

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

Installazione di AngularJS

Dopo aver integrato Bootstrap nella web app, procedete a installare Angular. Visitate il sito https://angularjs.org/ e fate clic sul pulsante *Download*. Vedrete alcune opzioni; a voi serve la versione stabile minificata.
Copiate il file che avete scaricato nella directory *js* del progetto e aprite il file index.html. Angular può essere integrato nell'app come qualsiasi altro file JavaScript. È preferibile includerlo nel tag <head> della pagina, altrimenti alcune funzioni a cui ricorrerete nel libro non saranno attive. Anche se non è necessario, dovrete compiere altri passi per integrare ancora di più Angular nel file HTML.
Inserite questo tag <script> nell' <head> della pagina.

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

Tutto fatto? Quasi. Dobbiamo dire ad Angular che intendiamo utilizzarlo nell'app. Angular richiede questo bootstrap, e il framework semplifica moltissimo questa procedura. Dovete semplicemente includere un altro attributo nel tag <html> di apertura:

```
<html lang="en" ng-app>
```

Ecco fatto! Ora Angular sa che vogliamo utilizzarlo.

> **NOTA**
>
> Angular ci consente anche di far precedere questi attributi da data- (per esempio data-ng-app) nel caso volessimo scrivere HTML5 valido.

Utilizzare AngularJS

Abbiamo visto molta teoria alla base di Angular; è giunto il momento di metterla in pratica. Dopo aver creato un'app funzionante, vedremo come abbellirla con Bootstrap.
Riaprite il file index.html, ma questa volta apritelo anche nel browser così da vedere ciò su cui state lavorando. Ecco a che punto siamo arrivati:

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

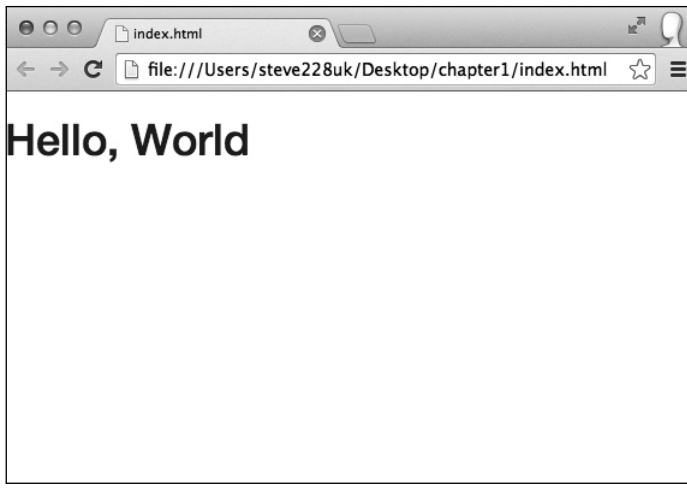
```
<body>

</body>
</html>
```

Abbiamo impostato Bootstrap e Angular e abbiamo inizializzato l'app con l'attributo `ng-app` nel tag di apertura `<html>`; ora passiamo rapidamente all'azione.
Realizzeremo un'app Hello, World un po' diversa. Invece di far comparire questa scritta, avremo un campo di testo che effettuerà il binding dei dati e li ripeterà automaticamente nella vista; tutto questo senza scrivere una sola riga di JavaScript. Iniziamo ad aggiungere il tag `<h1>` nel tag `<body>`:

```
<h1>Hello, World</h1>
```

Nel browser vedrete che Bootstrap ha aggiornato la visualizzazione predefinita. Al posto del font Times New Roman compare l'Helvetica e i margini di troppo attorno al bordo sono stati eliminati.



Ora dobbiamo includere l'input di testo e specificare anche il modello che intendiamo utilizzare. Il modello può essere di qualsiasi tipo, ma in questo caso sarà una stringa che l'input restituirà:

```
<input type="text" ng-model="name">
```

L'attributo `ng-model` dichiara il binding del modello su questo elemento, e tutto ciò che digiteremo nel campo di testo sarà associato a esso da Angular. Ovviamente non verrà visualizzato come per magia sulla pagina; dovremo dire al framework dove ripeterlo. Per visualizzare il modello sulla pagina, è sufficiente racchiudere il nome tra doppie parentesi graffe:

```
{{name}}
```

Inseritelo al posto di `World` nel tag `<h1>` e aggiornate la pagina nel browser. Se digitate il vostro nome nel campo di testo, vedrete che viene visualizzato automaticamente nell'header in tempo reale. Angular lo fa al posto vostro senza dover scrivere una sola riga di JavaScript.

Un risultato ottimo, ma sarebbe opportuno avere un'impostazione predefinita che eviti di far sembrare che non funziona ancora prima che un utente digiti il suo nome. Fortunatamente tutto ciò che è compreso tra le parentesi graffe viene analizzato come un'espressione AngularJS, e così è possibile verificare se il modello ha un valore; in caso contrario, può ripetere `World`. In Angular questa è un'espressione, ed è sufficiente aggiungere il doppio pipe come in JS:

```
{{name || 'World'}}
```

> **NOTA**
>
> Angular descrive in questo modo un'espressione: "Frammenti di codice simili a JavaScript che di solito sono posti in binding come `{{ espressione }}`."

È opportuno ricordare che si tratta di JavaScript, ed ecco perché dobbiamo inserire le virgolette per segnalare che si tratta di una stringa e non del nome di un modello. Se provaste a cancellarle, vedreste che Angular non visualizza di nuovo nulla. Ciò accade perché i modelli `name` e `World` non sono definiti.
Questi modelli possono essere definiti direttamente all'interno dell'HTML utilizzando, come abbiamo visto, un attributo, ma è anche possibile assegnare a essi un valore da un controller. A questo scopo create un nuovo file JS, `controller.js`, e includetelo nell'app:

```
<script type="text/javascript" src="js/controller.js"></script>
```

Inseritelo dopo aver incluso Angular nella pagina per evitare errori.

I controller sono semplicemente funzioni che Angular può utilizzare; esaminiamone uno:

```
function AppCtrl($scope){
}
```

Qui abbiamo dichiarato il controller (in sostanza una semplice funzione del costruttore JavaScript) e in esso abbiamo inserito lo scope. Lo *scope* è ciò a cui possiamo accedere all'interno della vista. Su un'unica pagina possono esistere molteplici controller e molteplici scope. Si tratta in sostanza di un oggetto JavaScript dei nostri modelli e delle nostre funzioni, sui quali Angular opera la sua magia; per esempio, lo scope della nostra applicazione per il momento appare così:

```
{
 name: "Stephen"
}
```

Lo scope cambia a seconda di ciò che si digita nel campo di testo. A esso si può accedere sia dalla vista sia dal controller.
Dopo aver creato il controller, dobbiamo dire ad Angular che intendiamo utilizzarlo. Per la nostra app ci serve un solo controller; aggiungiamo un secondo attributo al tag `<html>`:

```
ng-controller="AppCtrl"
```

Questo attributo dice ad Angular che vogliamo utilizzare la funzione `AppCtrl` che abbiamo appena creato come controller per la pagina. Potremmo ovviamente aggiungerlo a qualsiasi elemento sulla pagina compreso, se volessimo, il body.
Per verificare che tutto funzioni, specificheremo un valore iniziale per il modello. È facile come impostare una proprietà su qualsiasi oggetto:

```
function AppCtrl($scope) {
  $scope.name = "World";
}
```

Se aggiornate l'app nel browser, vedrete che `World` è precompilato come valore del modello. Questo è un ottimo esempio dell'efficace *binding dei dati bidirezionale* di Angular. Ci consente di utilizzare dati predefiniti di una API o di un database e poi modificarli direttamente nella vista prima di riottenerli nel controller.

> **NOTA**
>
> Angular descrive il binding dei dati come "la sincronizzazione dei dati tra i componenti del modello e della vista". Così, se modifichiamo il valore di un modello nella vista o nel controller JavaScript, tutto si aggiorna di conseguenza.

Bootstrap

Dopo aver creato l'applicazione Hello World e aver verificato che tutto funzioni come previsto, è il momento di utilizzare Bootstrap per aggiungere stile e struttura

alla nostra app. Per ora l'app è mal allineata sulla sinistra e tutto sembra troppo fitto; rimediamo con un po' di *scaffolding*. Bootstrap offre un ottimo sistema di griglie responsive *mobile first* che possiamo utilizzare aggiungendo alcuni div e alcune classi.
Prima, però, racchiudiamo il contenuto in un contenitore per far subito un po' di ordine.

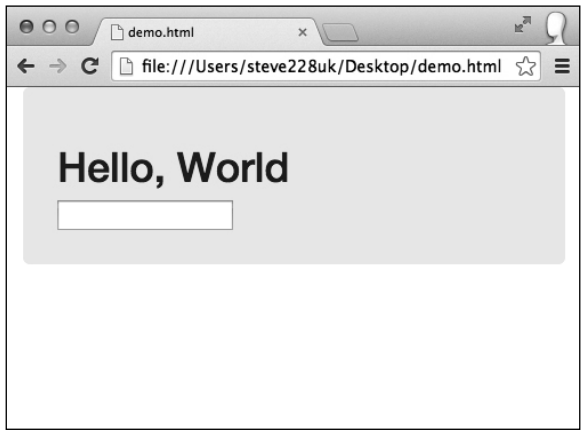
> **NOTA**
>
> Il concetto di *mobile first* consiste nel progettare/sviluppare innanzitutto per gli schermi più piccoli e aggiungere elementi al design invece di eliminarli.

```
<div class="container">
  <h1>Hello, {{name || 'World'}}</h1>
  <input type="text" ng-model="name">
</div>
```

Se ridimensionate la finestra del browser, dovreste iniziare a osservare la capacità di adattamento del framework e vedere la finestra comprimersi.

Può essere una buona idea racchiuderlo in quello che nella terminologia di Bootstrap è un *Jumbotron* (nelle versioni precedenti si chiamava *Hero Unit*). Farà risaltare molto di più il titolo. Possiamo ottenere questo risultato racchiudendo i tag `<h1>` e `<input>` in un nuovo div associato alla classe `jumbotron`:

```
<div class="container">
  <div class="jumbotron">
    <h1>Hello, {{name || 'World'}}</h1>
    <input type="text" ng-model="name">
</div>
</div>
```



Ha un aspetto decisamente migliore, ma il contenuto è ancora troppo vicino alla parte superiore della finestra del browser. Possiamo introdurre un ulteriore miglioramento con un header di pagina, anche se il campo di testo mi sembra ancora fuori posto. Sistemiamo innanzitutto l'header di pagina:

```
<div class="container">
  <div class="page-header">
    <h2>Chapter 1 <small>Hello, World</small></h2>
  </div>
  <div class="jumbotron">
    <h1>Hello, {{name || 'World'}}</h1>
    <input type="text" ng-model="name">
</div>
</div>
```

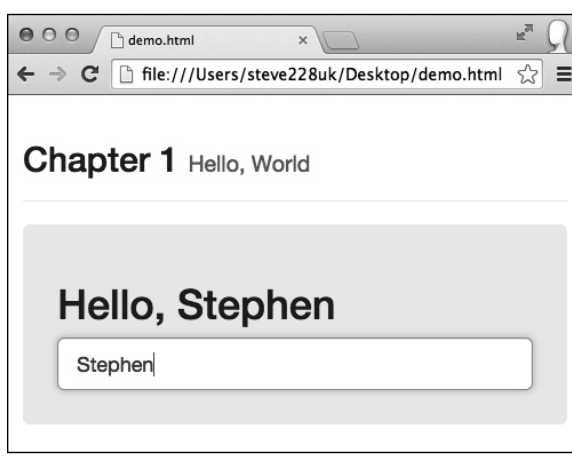
Ho inserito il numero e il titolo del capitolo. Il tag `<small>` all'interno del tag `<h2>` permette di distinguere efficacemente il numero dal titolo del capitolo.

La classe `page-header` aggiunge altro margine e padding, oltre a un sottile bordo inferiore. L'ultimo elemento che potremmo migliorare è la casella di testo. Bootstrap offre alcuni ottimi stili di testo e vale la pena utilizzarli. Per prima cosa dobbiamo aggiungere la classe `form-control` all'input di testo.
In questo modo la larghezza verrà impostata al 100% e alcuni aspetti stilistici miglioreranno, come i bordi arrotondati e un bagliore nel momento in cui l'elemento ottiene il focus:

```
<input type="text" ng-model="name" class="form-control">
```

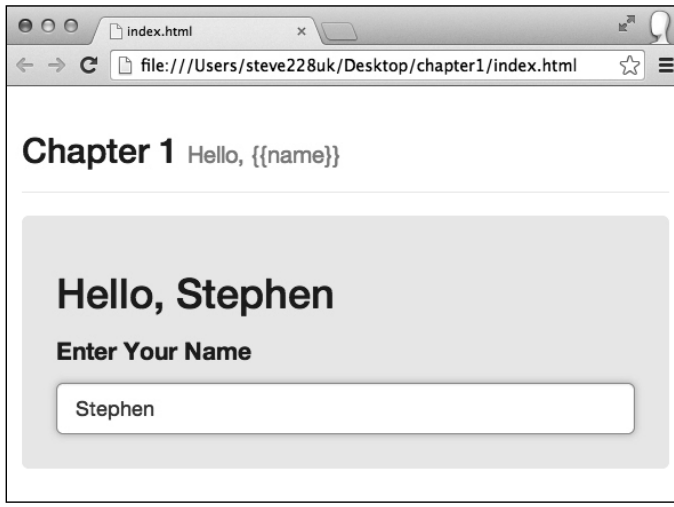
Va molto meglio, ma sembra ancora un po' piccolo rispetto all'header. Bootstrap offre altre due classi che rimpiccioliscono o ingrandiscono l'elemento, rispettivamente `input-lg` e `input-sm`. In questo caso, scegliamo la classe `input-lg` e la aggiungiamo all'input:

```
<input type="text" ng-model="name" class="form-control input-lg">
```



Dobbiamo ancora risolvere la questione della spaziatura perché è troppo a ridosso del tag `<h1>`. Forse è una buona idea aggiungere un'etichetta, così l'utente capirà che deve digitare nella casella. Bootstrap ci consente di prendere due piccioni con una fava perché include un margine nell'etichetta:

```
<label for="name">Enter Your Name</label>
<input type="text" ng-model="name" class="form-control input-lg" id="name">
```



Quiz

1. Come viene inizializzato Angular sulla pagina?
2. Che cosa si usa per visualizzare sulla pagina un valore del modello?
3. A che cosa corrisponde l'acronimo MVC?
4. Come creiamo un controller e come diciamo ad Angular che intendiamo utilizzarlo?
5. In Bootstrap 3 qual è il nuovo nome di Hero Unit?

Riepilogo

La nostra app è bella e funziona proprio come dovrebbe; riepiloghiamo ciò che abbiamo imparato in questo primo capitolo.
Innanzitutto abbiamo visto come è facile installare AngularJS e Bootstrap includendo un solo file JavaScript e un unico foglio di stile. Abbiamo anche osservato come viene inizializzata un'applicazione Angular e abbiamo iniziato a realizzare la nostra prima app.

L'app Hello, World che abbiamo creato, seppur molto semplice, illustra alcune caratteristiche fondamentali di Angular:

- espressioni;
- scope;
- modelli;
- binding dei dati bidirezionale.

Tutto questo è stato possibile senza scrivere una sola riga di JavaScript; infatti il controller che abbiamo creato serviva solo a illustrare il binding bidirezionale e non era un componente obbligatorio dell'app.
Con Bootstrap, abbiamo utilizzato alcuni dei numerosi componenti disponibili, tra cui le classi `jumbotron` e `page-header` per attribuire all'app un po' di stile e sostanza. Inoltre abbiamo visto in azione il nuovo design responsive *mobile first* senza dover affollare il markup con classi o elementi non necessari.
Nel Capitolo 2 esamineremo più nel dettaglio alcune caratteristiche fondamentali di AngularJS e Bootstrap e presenteremo il progetto che implementeremo nel corso del libro.