Web 开发经典丛书

Angular 高级编程

(第 3 版)

[美] 亚当 • 弗雷曼(Adam Freeman) 著
睢 丹 译

清华大学出版社
北 京

Pro Angular 6, Third Edition

By Adam Freeman
EISBN：978-1-4842-3648-2

图书在版编目(CIP)数据

Angular 高级编程(第 3 版) / (美)亚当・弗雷曼(Adam Freeman) 著；睢丹译. —北京：清华大学出版社，2019
(Web 开发经典丛书)
书名原文：Pro Angular 6, Third Edition
ISBN 978-7-302-52917-0

Ⅰ. ①A… Ⅱ. ①亚… ②睢… Ⅲ. ①超文本标记语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2019)第 083540 号

责任编辑：王　军　韩宏志
装帧设计：孔祥峰
责任校对：成凤进
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张**：34　　**字　　数**：1251 千字
版　　次：2019 年 6 月第 1 版　　**印　　次**：2019 年 6 月第 1 次印刷
定　　价：138.00 元

产品编号：082587-01

译 者 序

在 Web 开发领域，一般使用 HTML 作为前端页面元素的声明式语言，使用 CSS 技术作为展示样式的描述语言，JavaScript 作为业务处理交互的命令式语言。而构建非常复杂的 Web 应用时，纯粹而有限的 HTML 就显得非常不足，JavaScript 本身也会随着项目代码量的膨胀而变得难以维护和管理，研发工期和成本也会随之难以控制。此时，一般使用类库(如 jQuery、Dojo 等)或框架(如 Backbone、Ember 和 ExtJS 等)来提升开发效率，缩短项目的工期和降低成本，方便后续的维护和管理。

而 AngularJS 不仅是一个理念先进的前端开发框架，更是一种端对端的解决方案，已经被用于 Google 的多款产品中。AngularJS 遵从架构设计中的 MVC 模式，提倡展现、数据和逻辑处理组件的松耦合(类似 Flex 和 WPF)。AngularJS 通过指令技术对传统 HTML 实现了自然扩展，通过编译技术实现了数据模型与展现视图的双向自动同步，消除了前端开发中繁杂的 DOM 操作。最后通过模块化设计解决了 JS 代码管理维护和按需加载的问题，提高了广大前端程序员以及后端程序员的生产效率。而且这种解耦本身，也对前端的自动化测试技术提供了良好的支持。

AngularJS 诞生于 2009 年，由 Misko Hevery 和 Adam Abrons 创建，后为 Google 所收购。AngularJS 有许多特性，最为核心的是：MVVM、模块化、自动化双向数据绑定、语义化标签、依赖注入等。AngularJS 是专门为应用程序设计的 HTML。

Angular 2.0 是 2014 年 10 月 22 日在 ng-Europe 会议上发布的，该版本完全重写了 AngularJS。为避免开发人员的困惑，开发团队宣布每个框架都应该使用单独的术语，其中“AngularJS”指的是版本 1。x 版本和没有“JS”的“Angular”指的是版本 2 及更高版本。

Angular 6 于 2018 年 5 月 4 日发布。这是一个重要版本，它对底层框架的关注较少，考虑更多的是工具链，以及如何使 Angular 更易于在未来提高效率，如 ng update、ng add、Angular Elements、Angular Material + CDK Components、Angular Material Starter Components、CLI Workspaces、Library Support、Tree Shakable Providers、Animations Performance Improvements 和 RxJS v6。

Angular 7 让 Angular 变得更出色，为顺应现代趋势增加了比特币挖掘、虚拟滚动、拖放等功能。

本书分为三个部分，每部分涵盖一组相关的主题。本书第Ⅰ部分回顾了一些关键技术的基本知识，包括 HTML、CSS 和 TypeScript，并展示如何构建第一个 Angular 应用程序。第Ⅱ部分介绍 Angular 为构建应用程序提供的构造块，依次描述所有这些构造块。深入描述 Angular 包含的很多内置功能，以及丰富的 Angular 自定义选项。第Ⅲ部分介绍如何使用高级功能来创建更复杂、可扩展的应用程序。演示如何在 Angular 应用程序中进行异步 HTTP 请求，如何使用 URL 路由在应用程序中导航，以及在应用程序的状态发生变化时如何使 HTML 元素具有动画效果。

本书堪称 Angular 领域的里程碑式著作，涵盖了关于 Angular 的几乎所有内容。本书平实、通俗的讲解，丰富、翔实的示例，递进、严密的组织，可以让新手毫无压力地登堂入室，迅速吸取新一代 Web 应用开发的精髓。对于具有相关经验的用户，本书对 Angular 概念和技术细节的全面剖析，以及引人入胜、切中肯綮的讲解，将帮助读者彻底掌握这个框架，在自己职业技术修炼之路上更进一步。

这里要感谢清华大学出版社的编辑们，他们为本书的翻译投入了巨大热情并付出了很多心血。没有他们的帮助和鼓励，本书不可能顺利付梓。

对于这本经典之作，译者本着“诚惶诚恐”的态度，在翻译过程中力求“信、达、雅”，但是鉴于译者水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正。

译者

作 者 简 介

Adam Freeman 是一位经验丰富的 IT 专业人士，曾在多家公司担任过高级职位，曾经担任一家全球银行的首席技术官和首席运营官。退休之后，他热衷于写作和长跑。

技术评审员简介

Fabio Claudio Ferracchiati 是微软技术领域的高级顾问、高级分析师和高级开发人员。他在 BluArancio 公司(www.bluarancio.com)工作。他拥有.NET MCSD(Microsoft Certified Solution Developer，微软认证解决方案开发人员)、.NET MCAD(Microsoft Certified Application Developer，微软认证应用程序开发人员)、MCP(Microsoft Certified Professional，微软认证专家)等多项认证，而且是一位多产的作家和技术评审员。十多年来，他为多家国际杂志撰写了大量文章，与他人合著了十多本关于各类计算机主题的书籍。

目　录

第Ⅰ部分　Angular 基础知识

第 I 部分

Angular 基础知识

第 1 章

准 备 工 作

Angular 将服务器端开发领域的一些最佳实践用于增强浏览器中的 HTML，为更加简便地构建富应用程序(rich application)打下良好基础。Angular 应用程序围绕模型-视图-控制器(Model-View-Controller，MVC)设计模式构建，该模式的重点在于创建具有如下特点的应用程序：

- **可扩展**：一旦理解 Angular 的基本原理，即便是复杂的 Angular 应用程序，也很容易弄明白其运行方式，而这意味着可以轻易地改进应用程序，为用户创建新的有用功能。
- **可维护**：Angular 应用程序易于调试和修复，这意味着长期维护工作得以简化。
- **可测试**：Angular 对单元测试和端到端测试的支持都非常好，这意味着可以先于用户发现并修复缺陷。
- **兼容标准**：Angular 建立在 Web 浏览器的固有功能上，但是它能实现的功能并未受到这些固有功能的限制，而是能够创建可兼容标准的 Web 应用程序，能够利用最新功能(如 HTML5 API)和流行的工具与框架。

Angular 是由 Google 赞助和维护的开源 JavaScript 库。它已用于一些最大和最复杂的 Web 应用程序。本书将展示在自己的项目中充分利用 Angular 所需要知道的一切。

1.1 需要了解什么

在阅读本书之前，读者应该熟悉 Web 开发的基础知识，了解 HTML 和 CSS 的工作原理，最好熟悉 JavaScript。如果对这些细节不是非常清楚，那么本书将在第 4~6 章中温习一下如何使用 HTML、CSS 和 JavaScript。但是本书不会给出 HTML 元素和 CSS 属性的全面参考。一本关于 Angular 的书不可能涵盖 HTML 的方方面面。

1.2 本书结构

本书分为三个部分，每部分涵盖一组相关的主题。

1.2.1 第Ⅰ部分：Angular 基础知识

本书的第Ⅰ部分为阅读本书其余部分提供了准备信息。它包括本章内容，回顾了一些关键技术的基本知识，包括 HTML、CSS 和 TypeScript(JavaScript 的一个超集，用于 Angular 开发)。这部分还将展示如何构建第一个 Angular 应用程序，并引导完成构建更真实的应用程序(名为 SportsStore)的过程。

1.2.2 第Ⅱ部分：Angular 详解

本书的第Ⅱ部分介绍 Angular 为构建应用程序提供的构造块，依次描述所有这些构造块。Angular 包含很多内置的功能，这部分将深入描述这些功能，此外 Angular 还提供丰富的自定义选项，而这部分也将展示所有这些功能。

1.2.3 第Ⅲ部分：Angular 高级功能

本书的第Ⅲ部分介绍如何使用高级功能来创建更复杂、可扩展的应用程序。这部分演示如何在 Angular 应用程序中进行异步 HTTP 请求，如何使用 URL 路由在应用程序中导航，以及在应用程序的状态发生变化时如何使 HTML 元素具有动画效果。

1.3　大量示例

本书包含大量示例。学习 Angular 的最好方法就是通过示例，本书将尽可能多的示例打包进来。为使本书中的示例尽可能多，本书采用一个简单的约定来避免反复地列出文件的内容。当在某一章中首次使用一个文件时，将列出该文件的完整内容，如代码清单 1-1 所示。代码清单的标题中将包含文件的名称，以及应该在哪个文件夹中创建该文件。更改代码时，将以粗体显示修改过的语句。

代码清单 1-1　完整的示例文档

```
import { NgModule } from "@angular/core";
import { BrowserModule } from "@angular/platform-browser";
import { ProductComponent } from "./component";
import { FormsModule, ReactiveFormsModule  } from "@angular/forms";
import { PaAttrDirective } from "./attr.directive";

@NgModule({
   imports: [BrowserModule, FormsModule, ReactiveFormsModule],
   declarations: [ProductComponent, PaAttrDirective],
   bootstrap: [ProductComponent]
})
export class AppModule { }
```

代码清单 1-1 摘自第 15 章。这里暂不关注它的具体作用，只需要注意，这是一个完整清单，显示了文件的所有内容。对同一个文件进行一系列修改，或对大文件进行较小修改时，只展示发生改变的元素，以创建部分代码清单。部分代码清单的特征就是以英文省略号(...)开始和结尾，如代码清单 1-2 所示。

代码清单 1-2　部分代码清单

```
. . .
<table class="table table-sm table-bordered table-striped">
    <tr><th></th><th>Name</th><th>Category</th><th>Price</th></tr>
    <tr *ngFor="let item of getProducts(); let i = index" pa-attr>
        <td>{{i + 1}}</td>
        <td>{{item.name}}</td>
        <td pa-attr pa-attr-class="bg-warning">{{item.category}}</td>
        <td pa-attr pa-attr-class="bg-info">{{item.price}}</td>
    </tr>
</table>
. . .
```

代码清单 1-2 也是摘自第 15 章的后续代码清单。可以看到，这里只显示了 table 元素及其内容，并且突出显示了一些语句。希望借此引起读者的注意，关注示例的这一部分，以说明所描述的功能或技术。某些情况下，需要对同一个文件的不同部分进行更改，此时为简洁起见，需要省略一些元素或语句，如代码清单 1-3 所示。

代码清单 1-3　为简洁起见而省略部分语句

```
import { ApplicationRef, Component } from "@angular/core";
import { Model } from "./repository.model";
import { Product } from "./product.model";
import { ProductFormGroup } from "./form.model";

@Component({
    selector: "app",
    templateUrl: "app/template.html"
})
export class ProductComponent {
    model: Model = new Model();
    form: ProductFormGroup = new ProductFormGroup();

    // ...other members omitted for brevity...

    showTable: boolean = true;
}
```

利用这个约定可在本书中放入更多示例，但这也意味着很难定位某项技术。为此，在第 II 部分和第III部分中描述 Angular 特性的所有章都从一个内容摘要表格开始，描述该章包含的技术以及演示如何使用它们的代码清单。

1.4 获取示例代码

可以从 https://github.com/Apress/pro-angular-6/tree/master/Update%20for%20Angular%207 下载本书所有章的示例项目。这些示例项目都是针对 Angular 7 更新的代码，可以免费下载，其中包含用来重新创建示例需要的所有支持资源，这样就不必辛苦敲入这些代码。虽然不是一定要下载这些代码，但这是对示例进行实验的最简单方法。

请读者注意，本书对应的英文书籍是 *Pro Angular 6*。在翻译期间，Angular 升级到 Angular 7，因此本书的一部分代码也随之更新。由于 Angular 7 丢弃了 Angular 6 的个别功能，因此中文书籍特别更新了代码清单 14-26、14-27、14-28、15-1、15-2、15-4、15-6、15-21、16-15、16-17、17-14、17-15、17-22、17-24、17-25、19-14、19-27、20-26、20-27、20-30。删除了英文书中的代码清单 15-19。

如果读者想要查看 Angular 6 对应的代码，可访问 http://github.com/Apress/pro-angular-6。

读者也可扫描本书封底的二维码获取代码。但请注意，由于代码时常更新，要获取完整详明的代码，请访问上述网址下载。

1.5 如何搭建开发环境

第 2 章通过创建一个简单的应用程序来介绍 Angular，在介绍过程中，将说明如何建立使用 Angular 的开发环境。

1.6 联系作者

如果在运行本章中示例代码的过程中遇到问题，或者发现书中存在问题，请发电子邮件到 adam@adam-freeman.com，作者将竭尽所能提供帮助。

1.7 本章小结

本章概述了本书的内容和结构。学习 Angular 开发的最佳方法就是通过示例，因此，下一章将直奔主题，展示如何设置开发环境，并使用该环境创建第一个 Angular 应用程序。

第 2 章

第一个 Angular 应用程序

开始使用 Angular 的最佳方式就是动手创建一个 Web 应用程序。本章将展示如何搭建开发环境，并解释创建基本 Web 应用程序的过程：从静态的功能模拟开始，然后使用 Angular 功能创建一个简单的动态 Web 应用程序。在第 7~10 章中，将展示如何创建一个更复杂、更真实的 Angular 应用程序，但现在只需要一个简单的例子就足以演示 Angular 应用程序的主要组成部分，并为本书这部分的其他章搭建好开发环境。

如果未能理解本章的所有内容，也不必担心。Angular 的学习曲线比较陡峭，因此本章的目的只是介绍 Angular 开发的基本流程，以了解各个部分之间的关系。虽然现在不会立即明白这些方面，但是当读完这本书时，就会明白本章讲解的每个步骤以及其他方面。

2.1 准备开发环境

要进行 Angular 开发，就需要做一些准备工作。下面的几节将介绍如何设置并准备好创建第一个项目。很多流行的开发工具都对 Angular 提供了很好的支持，因此可以选择自己最喜欢的一款开发工具。

2.1.1 安装 Node.js

许多用于 Angular 开发的工具都依赖 Node.js(也称为 Node)，Node.js 创建于 2009 年，为采用 JavaScript 编写服务器端应用程序提供了一个简单而高效的运行库。Node.js 基于 Chrome 浏览器中使用的 JavaScript 引擎，提供了一个在浏览器环境之外执行 JavaScript 代码的 API。

虽然作为一款应用程序服务器，Node.js 已经取得了成功，但是本书之所以提到 Node.js，是因为它为新一代跨平台开发和构建工具提供了基础。由于 Node.js 团队做出的一些精妙的设计决策以及 Chrome JavaScript 运行库提供的跨平台支持，人们发现它可用来编写开发工具。简而言之，Node.js 已经成为 Web 应用程序开发的必备工具。

务必确保下载的 Node.js 版本与本书中使用的相同。尽管 Node.js 相对稳定，但是 API 仍然会不时地发生重大变更，这样可能会导致本书中的示例无法正常运行。

本书使用的版本是 8.11.3，这是本书写作时的长期支持(Long Term Support，LTS)版本。当阅读本书时，Node.js 可能会有更新的版本，但是为了正常运行本书中的例子，应该坚持使用 8.11.3 版本。可从 https://nodejs.org/dist/v8.11.3 获取 8.11.3 版本的完整系列，包括针对 Windows 和 macOS 的安装程序以及针对其他平台的二进制软件包。

安装 Node.js 时，务必选择正确的选项，将 Node.js 可执行文件添加到路径(Path 环境变量)中。安装完成后，运行以下命令：

```
node -v
```

如果安装过程按照预期进行，就会显示以下版本号：

```
V8.11.3
```

Node.js 安装程序包含了 Node 包管理器(Node Package Manager，NPM)，用于管理项目中的包。运行以下命令以确保 NPM 正常工作：

```
npm -v
```

如果一切正常工作，就会看到以下版本号：

```
5.6.0
```

2.1.2　安装 angular-cli 包

angular-cli 包已经成为开发过程中创建和管理 Angular 项目的标准方法。本书的第 1 版演示了如何从头开始搭建 Angular 项目，这是一个冗长且容易出错的过程，而 angular-cli 简化了这个过程。要安装 angular-cli 包，请打开一个新的命令提示符并运行以下命令：

```
npm install --global @angular/cli@7.0.2
```

注意，global 参数前有两个连字符。如果使用 Linux 或 macOS，则需要使用 sudo，如下所示：

```
sudo npm install --global @angular/cli@7.0.2
```

2.1.3　安装 Git

需要 Git 版本控制工具来管理 Angular 开发所需的一些软件包。如果正在使用 Windows 或 macOS，请从 https://git-scm.com/downloads 下载并运行安装程序。在 macOS 上，可能需要更改安全设置，才能打开尚未由开发者签署的安装程序。

大多数 Linux 发行版已经安装了 Git。如果要安装最新版本，请访问 https://git-scm.com/download/linux，了解发行版的安装说明。例如，对于 Ubuntu(这是我使用的 Linux 发行版)，使用以下命令：

```
sudo apt-get install git
```

完成安装后，打开一个新的命令提示符，运行以下命令，来检查 Git 是否已安装并可用：

```
git --version
```

此命令输出已安装的 Git 软件包的版本。在撰写本书时，Git for Windows 和 Git for Linux 的最新版本为 2.17，Git for macOS 的最新版本为 2.16.3。

2.1.4　安装编辑器

由于程序员使用的任何编辑器都可以用于从事 Angular 开发，因此可供选择的编辑器非常多。有些编辑器对 Angular 提供增强的支持，包括关键字高亮显示和良好的工具集成。如果还没有确定 Web 应用程序开发的首选编辑器，那么表 2-1 介绍了一些常见选项，以供参考。本书并不依赖任何具体的编辑器，你应该使用自己喜欢的编辑器。

表 2-1　支持 Angular 开发的常见编辑器

名称	描述
Sublime Text	Sublime Text 是一款商业跨平台编辑器，借助软件包可支持大多数编程语言、框架和平台。详见 www.sublimetext.com
Atom	Atom 是一款免费、开源的跨平台编辑器，特别强调自定义和可扩展性。详见 atom.io
Brackets	Brackets 是由 Adobe 开发的免费开源编辑器。详见 bracket.io
WebStorm	WebStorm 是一款付费的跨平台编辑器，集成了许多工具，开发者不必在开发过程中使用命令行。详见 www.jetbrains.com/webstorm
Visual Studio Code	是微软提供的开源的、跨平台编辑器，重点强调可扩展性，详见 code.visualstudio.com
Visual Studio	Visual Studio 是微软的旗舰开发工具。有免费的和商业的版本可供使用，它还附带了许多与 Microsoft 生态系统集成的工具

在选择编辑器时，最重要的考虑之一是能够过滤项目的内容，以便专注于一部分文件。Angular 项目可能有很多文件，许多文件都有相似的名称，因此能够查找和编辑正确的文件至关重要。编辑器可以通过不同的方式实现这

种聚焦功能，具体方法有两种：显示文件列表，可以打开这些文件进行编辑；或者将具有特定扩展名的文件排除在外。

2.1.5 安装浏览器

最后一项选择是浏览器，在开发过程中需要使用浏览器检查代码是否按照预期运行。所有最新的浏览器都为开发者提供了良好的支持，并能很好地运行 Angular。本书一直使用谷歌 Chrome，这也是推荐使用的浏览器。

2.2 创建并准备项目

在安装Node.js、angular-cli、编辑器和浏览器之后，就具备了开始开发之旅的基础。

2.2.1 创建项目

要创建项目，请选择一个方便的位置，并使用命令提示符来运行以下命令，以创建一个名为 todo 的新项目：

```
ng new todo
```

可在以上命令的末尾加上--default 参数，以便使用默认配置创建项目。ng 命令由 angular-cli 包提供，ng new 启动一个新项目。在安装过程中将创建一个名为 todo 的文件夹，其中包含启动 Angular 开发所需的所有配置文件，启动开发的一些占位符文件以及开发、运行和部署 Angular 应用程序所需的 NPM 软件包。NPM 软件包的数量非常多，这意味着项目的创建可能需要一段时间。

2.2.2 添加 Bootstrap CSS 包

ng new 命令创建的项目几乎包含了本章所需的所有内容。但没有包含 Bootstrap CSS 包，本书使用它来样式化 HTML 内容。运行以下命令，导航到 ng new 命令创建的 todo 文件夹，并将 Bootstrap 包添加到项目中：

```
cd todo
npm install bootstrap@4.1.1
```

要配置 Angular 开发工具来使用 Bootstrap CSS 文件，请将代码清单 2-1 所示的条目添加到 angular.json 文件的 styles 部分，该文件在创建项目时由 ng new 命令添加到 todo 文件夹。

代码清单 2-1 在 todo 文件夹的 angular.json 文件中配置 CSS

```
...
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "todo": {
      "root": "",
      "sourceRoot": "src",
          "projectType": "application",
          "prefix": "app",
          "schematics": {},
          "architect": {
            "build": {
              "builder": "@angular-devkit/build-angular:browser",
              "options": {
                "outputPath": "dist/todo",
                "index": "src/index.html",
                "main": "src/main.ts",
                "polyfills": "src/polyfills.ts",
                "tsConfig": "src/tsconfig.app.json",
                "assets": [
                  "src/favicon.ico",
                  "src/assets"
                ],
                "styles": [
                  "src/styles.css",
```

```
                "node_modules/bootstrap/dist/css/bootstrap.min.css"
              ],
              "scripts": []
            },
...
```

如第 11 章所述，angular.json 文件用于配置项目工具，代码清单中显示的语句将 Bootstrap CSS 文件合并到项目中，以便将其包含在发送到浏览器的内容中。

2.2.3　启动开发工具

一切就绪，现在是测试 Angular 开发工具的时候了。在 todo 文件夹中运行以下命令：

```
ng serve --port 3000 --open
```

该命令启动 Angular 开发工具，这些工具会执行一个初始的构建过程，为开发会话准备应用程序。这个过程需要一些时间，输出如下：

```
** Angular Live Development Server is listening on localhost:3000, open your browser on
http://localhost:3000/ **

Hash: ebb64e6046efff317389
Time: 6767ms
chunk {main} main.js, main.js.map (main) 10.8 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.7 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.06 MB [initial] [rendered]
[wdm]: Compiled successfully.
```

如果输出略微不同，只要在准备工作完成后看到“compiled successfully”消息，就不必担心。几秒钟后，将启动一个新的浏览器窗口，如图 2-1 所示，其中显示了创建项目时添加到项目中的占位符内容。

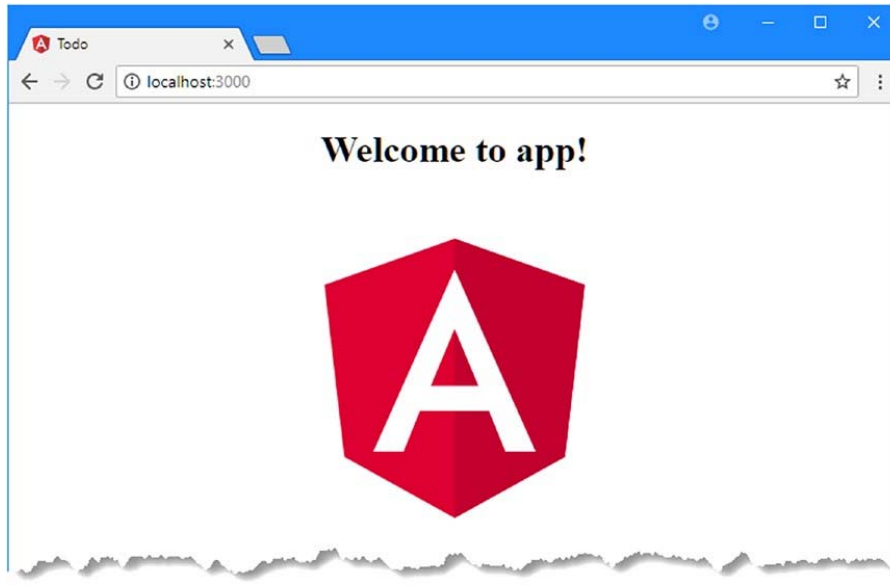

图 2-1　HTML 占位符内容

2.2.4　编辑 HTML 文件

下面首先删除项目创建时添加到项目中的占位符内容，这样就可以从包含静态内容的 HTML 文件开始，稍后使用 Angular 对其进行增强。编辑 todo/src 文件夹中的 index.html 文件，将内容替换为代码清单 2-2 所示的内容。

代码清单 2-2　src 文件夹中的 index.html 文件的内容

```
<!DOCTYPE html>
<html>
<head>
  <title>ToDo</title>
  <meta charset="utf-8" />
</head>
<body class="m-1 p-1">
  <h3 class="bg-primary text-white p-3">Adam's To Do List</h3>

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

  <table class="table table-striped table-bordered">
    <thead>
      <tr>
        <th>Description</th>
        <th>Done</th>
      </tr>
    </thead>
    <tbody>
      <tr><td>Buy Flowers</td><td>No</td></tr>
      <tr><td>Get Shoes</td><td>No</td></tr>
      <tr><td>Collect Tickets</td><td>Yes</td></tr>
      <tr><td>Call Joe</td><td>No</td></tr>
    </tbody>
  </table>
</body>
</html>
```

Angular 开发工具包含了一个功能，当项目发生变化时，会自动更新浏览器。保存 index.html 文件后，服务器将检测更改并更新应用程序，反映新的内容，如图 2-2 所示。

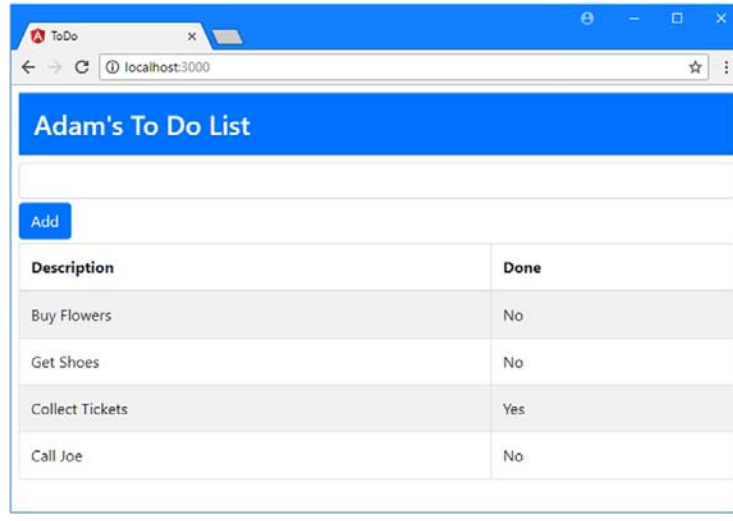

图 2-2　编辑 HTML 文件的内容

■ 提示：

更改一系列文件时，有时候浏览器可能无法加载和执行示例应用程序，特别是在运行后续章节中更复杂示例的情况下。大多数情况下，HTTP 开发服务器都能触发浏览器的重新加载，一切都会很正常，但是如果真的遇到问题，可能需要手动执行浏览器的重新加载。

index.html 文件中的 HTML 元素展示了本章中创建的简单 Angular 应用程序的外观。关键要素是显示用户名的横幅、input 元素、用来添加新的待办事项的 Add 按钮，以及包含所有待办事项并指示是否已完成的表格。

本书使用优秀的 Bootstrap CSS 框架来为 HTML 内容提供样式。Bootstrap 通过为元素指派 CSS 类来应用样式，

如下所示：

```
...
<h3 class="bg-primary text-white p-3">Adam's To Do List</h3>
...
```

这个 h3 元素被指派三个 CSS 类。bg-primary 类将元素的背景色设置为当前 Bootstrap 主题的主要颜色。还有其他主题颜色可用，包括 bg-secondary、bg-info 和 bg-danger。p-3 类为元素的所有边添加了一定量的填充，确保文本的周围有一些空白。text-white 类将文本颜色设置为白色，这将增加与背景色的对比度。本书应用 Bootstrap 时，HTML 元素将添加到这些类和其他类中。第 4 章将概述最常用的类。

在下一节中，将这段 HTML 内容从文件中移除，将其剪切成几个较小的部分，并使用它创建一个简单的 Angular 应用程序。

2.3　向项目中添加 Angular 功能

index.html 文件中的静态 HTML 内容充当基本应用程序的占位符。用户应该可以查看待办事项列表、清点已完成的项目并创建新项目。在接下来的几节中，将为项目添加一些基本的 Angular 功能，让待办事项应用程序变得鲜活起来。为了让应用程序尽可能简单，这里假设只有一位用户，而且不必保存应用程序的数据状态，这意味着如果浏览器窗口被关闭或重新加载，那么对待办事项列表的更改将会丢失。在稍后的例子中——包括第 7~10 章开发的 SportsStore 应用程序——将展示持久的数据存储。

2.3.1　准备 HTML 文件

向应用程序中添加 Angular 的第一步是准备 index.html 文件，如代码清单 2-3 所示。

代码清单 2-3　准备在 scr 文件夹的 index.html 文件中添加 Angular

```
<!DOCTYPE html>
<html>
<head>
    <title>ToDo</title>
    <meta charset="utf-8" />
</head>
<body class="m-1">
    <todo-app>Angular placeholder</todo-app>
</body>
</html>
```

代码清单 2-3 将 body 元素的内容替换为 todo-app 元素。HTML 规范中并没有 todo-app 元素，浏览器在解析 HTML 文件时会忽略它，但是这个元素是 Angular 世界的入口，并被应用程序内容替代。当保存 index.html 文件时，浏览器将重新加载文件并显示占位符消息，如图 2-3 所示。

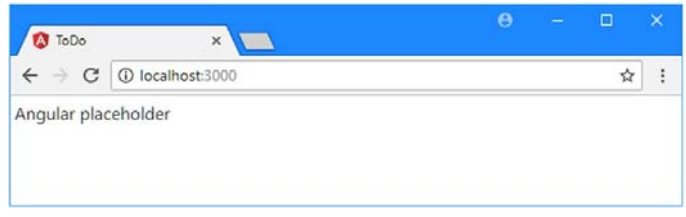

图 2-3　准备 HTML 文件

2.3.2　创建数据模型

当创建应用程序的静态模型时，数据分布在所有 HTML 元素中。用户名称包含在 header 元素中，如下所示：

```
...
<h3 class="bg-primary text-white p-3">Adam's To Do List</h3>
...
```

而待办事项的内容则包含在表格的 td 元素中，如下所示：

```
...
<tr><td>Buy Flowers</td><td>No</td></tr>
...
```

接下来的任务是将所有数据集中起来，创建一个数据模型。将数据与数据的呈现方式进行分离，这是 MVC 模式的关键思想之一，如第 3 章所述。

■ 提示：
这里对模型进行了简化。模型还可以包含创建、加载、存储和修改数据对象所需的逻辑。在 Angular 应用程序中，此逻辑通常位于服务器端，并通过 Web 服务访问。有关详细信息请参见第 24 章。

Angular 应用程序通常用 TypeScript 语言编写。第 6 章介绍 TypeScript，并解释它的工作原理及用途。TypeScript 是 JavaScript 的一个超集，但其主要优点之一是可以让开发者使用最新的 JavaScript 语言规范编写代码，其中一些新增功能并非在所有运行 Angular 应用程序的浏览器中都支持。在上一节中，angular-cli 添加到项目中的众多包中有一个名为 TypeScript 编译器的包，这个软件包有一个设置项，可在检测到 TypeScript 文件变更时自动生成浏览器友好的 JavaScript 文件。

为了向应用程序中添加数据模型，将一个名为 model.ts 的文件添加到 todo/src/app 文件夹(TypeScript 文件的扩展名为.ts)，并添加如代码清单 2-4 所示的代码。

代码清单 2-4　todo/src/app 文件夹中 model.ts 文件的内容

```
var model = {
    user: "Adam",
    items: [{ action: "Buy Flowers", done: false },
    { action: "Get Shoes", done: false },
    { action: "Collect Tickets", done: true },
    { action: "Call Joe", done: false }]
};
```

TypeScript 最重要的功能之一是可以直接编写“普通的”JavaScript 代码，就像直接面向浏览器编程一样。在代码清单 2-4 中，使用 JavaScript 对象字面量语法为一个名为 model 的全局变量赋值。数据模型对象有一个 user 属性(它提供了应用程序用户的名称)和一个 items 属性(该属性设置为一个对象数组，里面的每个对象都有 action 和 done 属性，表示待办事项列表中的一个任务)。

这是使用 TypeScript 时最重要的一个方面：不必使用 TypeScript 提供的特性，而只使用所有浏览器都支持的 JavaScript 功能来编写整个 Angular 应用程序，如代码清单 2-5 中的代码那样。

但是 TypeScript 的部分价值在于，它能够把使用 JavaScript 语言中最新功能的代码转换成可以在任何地方运行的代码，即使在不支持这些功能的浏览器中也是如此。代码清单 2-5 显示了使用 ECMAScript 6 标准(称为 ES6)中添加的 JavaScript 功能改写的数据模型。

代码清单 2-5　在 src/app 文件夹的 model.ts 文件中使用 ES6 功能

```
export class Model {
    user;
    items;

    constructor() {
        this.user = "Adam";
        this.items = [new TodoItem("Buy Flowers", false),
                      new TodoItem("Get Shoes", false),
                      new TodoItem("Collect Tickets", false),
                      new TodoItem("Call Joe", false)]
    }
}

export class TodoItem {
    action;
    done;

    constructor(action, done) {
        this.action = action;
```

```
            this.done = done;
        }
    }
```

这依然是标准的 JavaScript 代码，但 class 关键字是在语言的更高版本中引入的，大多数 Web 应用程序开发人员并不熟悉，这是因为旧版浏览器不支持它。class 关键字用于定义类型，可以使用 new 关键字实例化这些类型，以创建具有明确定义的数据和行为的对象。

在 JavaScript 语言的最新版本中添加的许多功能都是语法糖，帮助程序员避免一些最常见的 JavaScript 陷阱，例如不常见的类型系统。class 关键字不会改变 JavaScript 处理类型的方式，它只是让具有其他语言(如 C#或 Java)使用经验的程序员更熟悉和更容易使用它。我喜欢 JavaScript 的类型系统，它是动态的，表达能力也不错，但使用类更可预测，更不容易出错，并且简化了 Angular 的使用，这是因为 Angular 是针对最新的 JavaScript 功能而设计的。

■ 提示：

如果不熟悉在 JavaScript 规范的最新版本中添加的功能，也不要担心。第 5 章和第 6 章提供了一些入门知识，介绍如何使用一些能够让 Angular 更易用的 JavaScript 功能来编写代码，第 6 章还介绍了一些有用的 TypeScript 特有功能。

---

export 关键字与 JavaScript 模块有关。在使用模块时，每个 TypeScript 或 JavaScript 文件被认为是一个独立的功能单元，使用 export 关键字来标识要在应用程序其他位置使用的数据或类型。JavaScript 模块用于管理项目中不同文件之间产生的依赖关系，并避免在 HTML 文件中手动管理一组复杂的 script 元素。有关模块如何工作的详细信息，请参见第 7 章。

2.3.3　创建模板

应用程序需要一种途径向用户显示模型中的数据值。在 Angular 中，这项工作由模板完成，这里的模板是指包含由 Angular 执行的指令的 HTML 片段。该项目的 angular-cli 设置程序在 src/app 文件夹中创建一个名为 app.component.html 的模板文件。这个文件经过编辑，添加了如代码清单 2-6 所示的标记来替换占位符内容。该文件的名称遵循标准 Angular 命名约定，稍后将具体解释。

代码清单 2-6　src/app 文件夹中 app.component.html 文件的内容

```
<h3 class="bg-primary p-1 text-white">{{ getName() }}'s To Do List</h3>
```

稍后会向这个文件中添加更多元素，但刚开始时一个 h3 元素足矣。使用双括号{{和}}就可以在模板中包含数据值，而 Angular 会对双括号之间的内容求值，以获取要显示的值。

{{和}}字符是数据绑定的示例，这意味着它们在模板和数据值之间建立关联。数据绑定是一项重要的 Angular 功能，本章向示例应用程序添加功能时以及第 II 部分详细描述这些功能时，会给出更多示例。在这里，数据绑定会告诉 Angular 调用一个名为 getName 的函数，并使用返回结果作为 h3 元素的内容。目前在应用程序的任何位置都找不到 getName 函数，下一节将创建它。

2.3.4　准备组件

Angular 组件负责管理模板并为其提供所需的数据和逻辑。这似乎表明组件的功能比较宽泛，这是因为组件作为 Angular 应用程序的组成部分，它们完成大部分的重要工作。总之，它们可用于完成各种任务。

目前，项目中有一个数据模型，其中包含一个需要显示名称的 user 属性，还有一个模板可以通过调用 getName 属性来显示该名称。项目还需要一个组件，充当它们之间的桥梁。angular-cli 设置程序在 todo/src/app 文件夹中创建了一个名为 app.component.ts 的占位符组件文件，该文件经过编辑，将原始内容替换成代码清单 2-7 所示的代码。

代码清单 2-7　src/app 文件夹中 app.component.ts 文件的内容

```
import { Component } from "@angular/core";
import { Model } from "./model";

@Component({
```

```
    selector: "todo-app",
    templateUrl: "app.component.html"
})
export class AppComponent {
    model = new Model();

    getName() {
       return this.model.user;
    }
}
```

这仍然是 JavaScript 代码，但它依赖从前可能没有的一些陌生功能，这些功能是 Angular 开发的基础。该代码清单中的代码可以分为三个主要部分，如以下几节所述。

1. 理解导入语句

import 关键字与 export 关键字相对应，用于声明对 JavaScript 模块内容的依赖。代码清单 2-7 中两次用到 import 关键字，如下所示：

```
...
import { Component } from "@angular/core";
import { Model } from "./model";
...
```

该代码清单使用第 1 条 import 语句加载@angular/core 模块，其中包含关键的 Angular 功能，包括对组件的支持。在使用模块时，import 语句指定在大括号之间要导入的类型。在这里，import 语句用于从模块加载组件类型。@angular/core 模块包含许多已经打包在一起的类，以便浏览器可以将它们全部加载到单个 JavaScript 文件中。

第 2 条 import 语句用于从项目的文件中加载 Model 类。此类导入语句的目标均以./开头，表示该模块是相对于当前文件定义的。

请注意，所有 import 语句都不包含文件扩展名。这是因为 import 语句的目标与浏览器加载的文件之间的关系由模块加载器管理，稍后会在 2.3.5 节中配置模块加载器。

2. 理解装饰器

代码清单 2-7 中最奇怪的部分是：

```
...
@Component({
    selector: "todo-app",
    templateUrl: "app.component.html"
})
...
```

这是一个装饰器(decorator)，它提供关于类的元数据。这里是@Component 装饰器，顾名思义，它告诉 Angular 这是一个组件。装饰器通过其属性提供配置信息，对于@Component 装饰器，它包括名为 selector 和 templateUrl 的属性。

selector 属性指定一个 CSS 选择器，用于匹配该组件所要应用的 HTML 元素：这里指定了 todo-app 元素(添加到代码清单 2-3 的 index.html 文件中)。Angular 应用程序启动时，Angular 将扫描当前文档中的 HTML，并查找与组件对应的元素。Angular 会找到 todo-app 元素，并知道应该将其置于该组件的控制之下。

templateUrl 属性用于指定组件的模板，对于该组件而言，该模板为 app.component.html 文件。本书第 II 部分描述了可用于@Component 装饰器的其他属性以及 Angular 支持的其他装饰器。

3. 理解类

代码清单 2-7 的最后一部分定义了一个类，Angular 实例化该类以创建组件：

```
...
export class AppComponent {
    model = new Model();

    getName() {
       return this.model.user;
    }
}
...
```

这些语句定义了一个名为 AppComponent 的类，它拥有 model 属性和 getName 函数，它提供了支持代码清单 2-6 所示模板中数据绑定所需的功能。

当创建 AppComponent 类的新实例时，把 model 属性设置为代码清单 2-5 中定义的 Model 类的新实例。getName 函数返回由 Model 对象定义的 user 属性的值。

2.3.5　将应用程序组合起来

至此，构建一个简单的 Angular 应用程序所需的 3 个关键功能——模型、模板和组件都已完成。把更改保存到 app.component.ts 文件中时，已经实现了足够多的功能，可以将 3 个部分组合在一起，并显示如图 2-4 所示的输出。

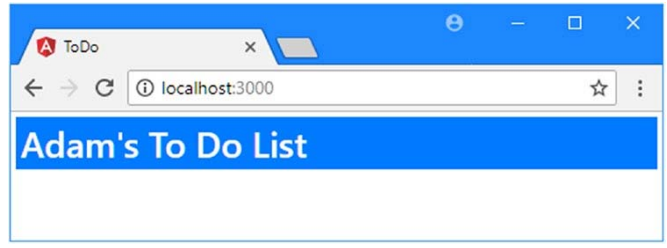

图 2-4　示例应用程序中的简单 Angular 功能

使用 angular-cli 创建项目的一个好处是不必担心创建 Angular 应用程序所需的基本文件，而缺点是跳过这些文件意味着会错过一些值得探索的重要细节。

Angular 应用程序需要模块(module)。由于命名选择的缘故，Angular 开发中使用了两种类型的模块。JavaScript 模块是包含 JavaScript 功能(通过 import 关键字使用该功能，参见第 6 章)的文件。另一种类型的模块是 Angular 模块，它用于描述应用程序或一组相关功能。每个应用程序都有一个根模块(root module)，它为 Angular 提供启动应用程序所需的信息。

当 angular-cli 设置项目时，它在 todo/src/app 文件夹中创建了一个名为 app.module.ts 的文件(这是根模块的常用文件名)，并添加了如代码清单 2-8 所示的代码。

代码清单 2-8　src/app 文件夹中 app.module.ts 文件的默认内容

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

@NgModule({
    declarations: [AppComponent],
    imports: [BrowserModule],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular 模块的目的是通过@NgModule 装饰器定义的属性来提供配置信息。第 21 章将详细解释模块的工作原理，但目前只需要知道，装饰器的 imports 属性告诉 Angular，该应用程序将在浏览器中运行，需要导入所需的相关功能，declarations 和 bootstrap 属性告诉 Angular 关于应用程序中的所有组件，以及应该使用哪个组件来启动应用程序(在这个简单示例应用程序中只有一个组件，因此它是这两个属性的唯一值)。

要创建待办事项应用程序，需要使用Angular的特性来处理表单元素，这些元素是在名为@angular/forms的Angular模块中定义的。为启用这些功能，需要修改app.module.ts文件，如代码清单2-9所示。

代码清单 2-9　在 src/app 文件的 app.module.ts 文件中启用表单支持

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
```

```
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular 应用程序还需要一个引导文件，其中包含启动应用程序所需的代码。引导文件名为 main.ts，它在 todo/src 文件夹中创建，其代码如代码清单 2-10 所示。本章的 main.ts 文件不需要修改。

代码清单 2-10　src 文件夹中 main.ts 文件的内容

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

虽然本书着重于在 Web 浏览器中运行的应用程序，但 Angular 旨在运行于各种环境中。引导文件中的代码选择要使用的平台，并加载根模块，这是应用程序的入口。

■ **提示：**

调用 platformBrowserDynamic( ).bootstrapModule 方法适合基于浏览器的应用程序，浏览器平台是本书的重点。如果正在使用不同的平台(例如 Ionic 移动开发框架)，就必须使用所用平台特有的不同引导方法。支持 Angular 的每个平台的开发人员都提供了平台特有的引导方法的详细信息。

浏览器执行引导文件中的代码，这会调用 Angular，后者又处理 HTML 文档并发现 todo-app 元素。用于定义组件的 selector 属性匹配到 todo-app 元素，这让 Angular 删除占位符内容，并将其替换为组件模板(从 app.component.html 文件自动加载)。模板经过解析，发现了{{和}}数据绑定，并对其中包含的表达式求值，调用 getName 方法并显示图中所示的结果。虽然这个结果可能不会给人留下深刻印象，但这是一个良好开端，它为添加更多功能提供了基础。

■ **提示：**

在任何 Angular 项目中，都必须耗费一段时间来定义应用程序的主要部分，并将它们有机地组织起来。在这段时间里，可能会感觉到自己在做很多工作，但是很少有回报。但可以肯定的是，这种初期投入最终会得到应有的回报。当开始构建更加复杂的实际 Angular 应用程序(如第 7 章中的更大示例)时，可以看出：虽然需要大量的初始设置和配置，但是随后就可以快速实现所需的功能。

2.4　向示例应用程序中添加功能

现在应用程序的基本结构已经到位，接下来可以添加在本章开头用静态 HTML 内容模拟的其余功能。在接下来的几节中，将添加包含待办事项列表的表格以及用于创建新项的 input 元素和按钮。

2.4.1　添加待办事项表格

Angular 模板的功能可不仅仅是显示简单的数据值。第 II 部分将描述模板的各种功能，但是对于这里的示例应用程序，将使用一项功能：针对数组中的每个对象，把一组 HTML 元素添加到 DOM 中。在这里，数组就是数据模型中的待办事项集合。首先，代码清单 2-11 向组件添加了一个方法，该方法为模板提供了待办事项数组。

代码清单 2-11　向 src/app 文件夹的 app.component.ts 文件添加一个方法

```
import { Component } from "@angular/core";
import { Model } from "./model";

@Component({
    selector: "todo-app",
    templateUrl: "app.component.html"
})
export class AppComponent {
    model = new Model();

    getName() {
       return this.model.user;
    }

    getTodoItems() {
       return this.model.items;
    }
}
```

getTodoItems 方法返回 model 对象的 items 属性值。代码清单 2-12 更新组件的模板以利用这个新方法。

代码清单 2-12　在 src/app 文件夹的 app.component.html 文件中显示待办事项

```
<h3 class="bg-primary p-1 text-white">{{ getName() }}'s To Do List</h3>

<table class="table table-striped table-bordered">
    <thead>
        <tr><th></th><th>Description</th><th>Done</th></tr>
    </thead>
    <tbody>
        <tr *ngFor="let item of getTodoItems(); let i = index">
            <td>{{ i + 1 }}</td>
            <td>{{ item.action }}</td>
            <td [ngSwitch]="item.done">
                <span *ngSwitchCase="true">Yes</span>
                <span *ngSwitchDefault>No</span>
            </td>
        </tr>
    </tbody>
</table>
```

模板的新增部分依赖几个不同的 Angular 功能。第一个特性是*ngFor 表达式，用于为数组中的每个项重复指定区域的内容。这是指令的一个示例，第 12~16 章将描述该功能(指令是 Angular 开发的重要组成部分)。*ngFor 表达式应用于元素的属性，如下所示：

```
...
<tr *ngFor="let item of getTodoItems(); let i = index">
...
```

此表达式告诉Angular将其宿主元素(即tr元素)作为模板处理，针对组件的getTodoItems方法返回的每个对象应该重复套用该模板。表达式的“let item”部分规定应该将每个对象赋给一个变量item，以便在模板中引用它。

*ngFor 表达式还跟踪正在处理的当前对象在数组中的索引，并将其赋给第二个变量 i：

```
...
<tr *ngFor="let item of getTodoItems(); let i = index">
...
```

其结果就是，对于 getTodoItems 方法返回的每个对象，tr 元素及其内容都将被复制并插入 HTML 文档中；对于每次迭代，都可以通过变量 item 访问当前的待办事项对象，可以通过变量 i 访问对象在数组中的位置。

■ 提示：
在使用*ngFor 表达式时切勿忘记字符*。第 16 章会解释它的含义。

在 tr 模板中共有两处数据绑定(可以通过字符{{和}}来识别)，如下所示：

```
...
<td>{{ i + 1 }}</td>
```

```
<td>{{ item.action }}</td>
...
```

这些绑定引用由*ngFor 表达式创建的变量。绑定不仅用于引用属性和方法名称，它们也可以用于执行简单的 JavaScript 操作。在第一个绑定中可以看到一个这样的例子，把 i 变量和 1 相加。

■ 提示：

对于简单的转换，可以将 JavaScript 表达式直接嵌入到绑定中，就像上面的做法一样。但对于更复杂的操作，Angular 有一个名为管道(pipe)的功能，见第 18 章的描述。

tr 模板中的其他模板表达式演示了如何有选择性地生成内容：

```
...
<td [ngSwitch]="item.done">
    <span *ngSwitchCase="true">Yes</span>
    <span *ngSwitchDefault>No</span>
</td>
...
```

[ngSwitch]表达式是一个条件语句，用于根据指定的值(这种情况下为 item.done 属性)将不同的一组元素插入文档中。嵌套在 td 元素中的是两个 span 元素，它们分别包含注解*ngSwitchCase 和*ngSwitchDefault，分别对应于普通 JavaScript switch 代码块中的 case 和 default 关键字。第 13 章将详细描述 ngSwitch(第 12 章描述方括号的含义)，但结果是当 item.done 属性的值为 true 时，第一个 span 元素被添加到文档中，而第二个 span 元素则在 item.done 为 false 时被添加到文档中。结果是 item.done 属性的 true/false 值被转换成包含 Yes 或 No 的 span 元素。将更改保存到模板中时，浏览器会重新加载，待办事项表格将显示出来，如图 2-5 所示。

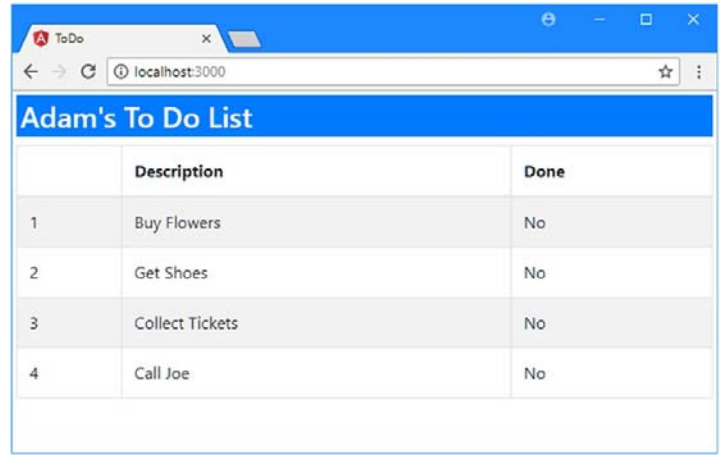

图 2-5 显示待办事项表格

如果使用浏览器的 F12 开发工具，将可以看到模板生成的 HTML 内容。通过查看页面源代码的方式并不能看到这些内容，这种方式只显示服务器发送的 HTML，而不显示 Angular 使用 DOM API 进行的更改。

可以看到模型中的每个待办事项对象在表格中是否生成了一行，并用局部变量 item 和 i 的值填充该行各列的内容，以及如何使用 Yes 或 No 来表示该任务是否已完成。

```
...
<tr>
    <td>2</td>
    <td>Get Shoes</td>
    <td><span>No</span></td>
</tr>
...
```

2.4.2 创建双向数据绑定

目前，模板仅包含单向数据绑定，这意味着它们用于显示数据值，但不能做任何改变。Angular 还支持双向数

据绑定，可用于显示数据值并进行更新。双向绑定是与 HTML 表单元素一起使用的，代码清单 2-13 向模板添加了一个复选框(checkbox)类型的 input 元素，该元素将允许用户将待办事项标记为已完成。

代码清单 2-13　在 src/app 文件夹的 app.component.html 文件中添加双向绑定

```
<h3 class="bg-primary p-1 text-white">{{getName()}}'s To Do List</h3>

<table class="table table-striped table-bordered">
    <thead>
        <tr><th></th><th>Description</th><th>Done</th></tr>
    </thead>
    <tbody>
        <tr *ngFor="let item of getTodoItems(); let i = index">
            <td>{{i + 1}}</td>
            <td>{{item.action}}</td>
            <td><input type="checkbox" [(ngModel)]="item.done" /></td>
            <td [ngSwitch]="item.done">
                <span *ngSwitchCase="true">Yes</span>
                <span *ngSwitchDefault>No</span>
            </td>
        </tr>
    </tbody>
</table>
```

ngModel 模板表达式在数据值(这里是 item.done 属性)和表单元素之间建立双向绑定。把更改保存到模板时，表格中会出现一个包含复选框的新列。复选框的初始值由 item.done 属性设置，就像常规单向绑定一样，但是当用户切换复选框的状态(选中或未选中)时，Angular 将进行响应，即更新指定的模型属性。

为演示这是如何工作的，这里保留了包含 Yes/No 显示的列(在模板中使用 ngSwitch 表达式的 done 属性值来生成)。当切换复选框的状态(选中或未选中)时，相应的 Yes/No 值也会发生变化，如图 2-6 所示。

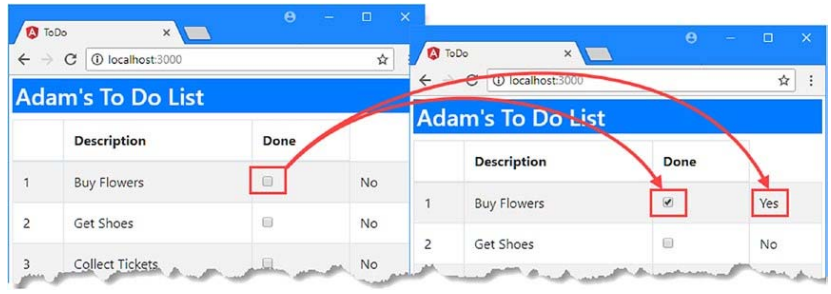

图 2-6　使用双向数据绑定修改模型值

这揭示了一个重要的 Angular 特性，即数据模型是实时的。这意味着在数据模型更改时数据绑定(甚至单向数据绑定)将得到更新。这简化了 Web 应用程序开发工作，因为这意味着开发者不必确保显示结果在应用程序状态变更时自动更新。

过滤待办事项

复选框可让数据模型得到更新，下一步是一旦待办事项被标记为已完成，就将其删除。代码清单 2-14 更改组件的 getTodoItems 方法，以便过滤掉已完成的任何事项。

代码清单 2-14　在 src/app 文件夹的 app.component.ts 文件中过滤待办事项

```
import { Component } from "@angular/core";
import { Model } from "./model";

@Component({
    selector: "todo-app",
    templateUrl: "app.component.html"
})
export class AppComponent {
    model = new Model();

    getName() {
        return this.model.user;
```

```
    }

    getTodoItems() {
        return this.model.items.filter(item => !item.done);
    }
}
```

这是一个 lambda 函数的示例，也称为胖箭头函数，它是表达标准 JavaScript 函数的一种更简洁的方法。lambda 表达式中的箭头可读作“转到”，例如“item 转到非 item.done”。lambda 表达式是 JavaScript 语言规范最近新增的特性，它们为使用函数作为实参的常规方法提供了替代方法：

```
...
return this.model.items.filter(function (item) { return !item.done });
...
```

无论选择哪种方式定义传递给 filter 方法的表达式，结果都是只显示未完成的待办事项。由于数据模型是实时的，因此更改会立即反映到数据绑定中，勾选待办事项的复选框会将其从视图中删除，如图 2-7 所示。

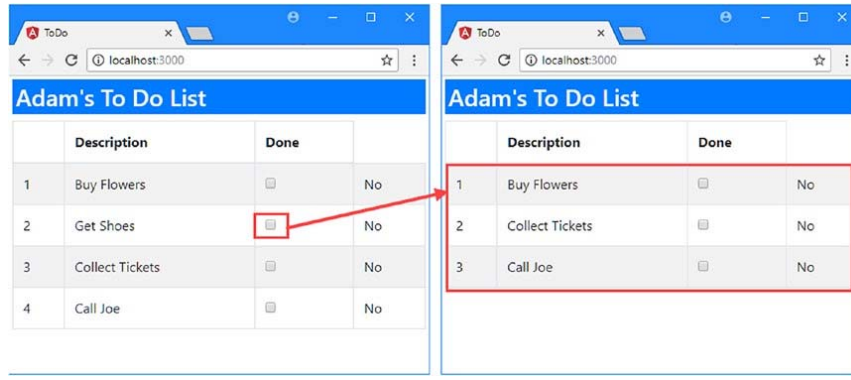

图 2-7　过滤待办事项

2.4.3　添加待办事项

下一步是在基本功能的基础上进行构建，允许用户创建新的待办事项，并将其存储到数据模型中。代码清单 2-15 向组件的模板添加了新的元素。

代码清单 2-15　在 src/app 文件夹的 app.component.html 文件中添加元素

```
<h3 class="bg-primary p-1 text-white">{{getName()}}'s To Do List</h3>
<div class="my-1">
    <input class="form-control" #todoText />
    <button class="btn btn-primary mt-1" (click)="addItem(todoText.value)">
        Add
    </button>
</div>
<table class="table table-striped table-bordered">
    <thead>
        <tr><th></th><th>Description</th><th>Done</th></tr>
    </thead>
    <tbody>
        <tr *ngFor="let item of getTodoItems(); let i = index">
            <td>{{i + 1}}</td>
            <td>{{item.action}}</td>
            <td><input type="checkbox" [(ngModel)]="item.done" /></td>
            <td [ngSwitch]="item.done">
                <span *ngSwitchCase="true">Yes</span>
                <span *ngSwitchDefault>No</span>
            </td>
        </tr>
    </tbody>
</table>
```

这个 input 元素具有一个名称以字符#开头的属性，用于定义一个变量，以引用模板的数据绑定中的元素。该变量名为 todoText，button 元素中的绑定将用到这个变量。

```
...
<button class="btn btn-primary mt-1" (click)="addItem(todoText.value)">
...
```

这是一个事件绑定的示例，它告诉 Angular 调用一个名为 addItem 的组件方法，它使用 input 元素的 value 属性作为方法的实参。代码清单 2-16 在组件中实现了 addItem 方法。

■ 提示：

现在不要关心如何区分不同的绑定。第 II 部分将解释 Angular 支持的不同类型的绑定，以及每种绑定所需的不同类型的方括号或圆括号的含义。它们并不像初看起来那样复杂，尤其是在明白它们如何与 Angular 框架的其余部分融为一体之后。

代码清单 2-16　在 src/app 文件夹的 app.component.js 文件中添加方法

```
import { Component } from "@angular/core";
import { Model, TodoItem } from "./model";

@Component({
    selector: "todo-app",
    templateUrl: "app.component.html"
})
export class AppComponent {
    model = new Model();

    getName() {
        return this.model.user;
    }

    getTodoItems() {
        return this.model.items.filter(item => !item.done);
    }

    addItem(newItem) {
        if (newItem != "") {
            this.model.items.push(new TodoItem(newItem, false));
        }
    }
}
```

import 关键字可用于从模块中导入多个类，代码清单中的一条 import 语句已更新，以便在组件中使用 TodoItem 类。在组件类中，addItem 方法接收由模板中的事件绑定发送过来的文本，使用它创建一个新的 TodoItem 对象并将其添加到数据模型中。这些更改的结果是，可通过在 input 元素中输入文本，并单击 Add 按钮，来创建新的待办事项，如图 2-8 所示。

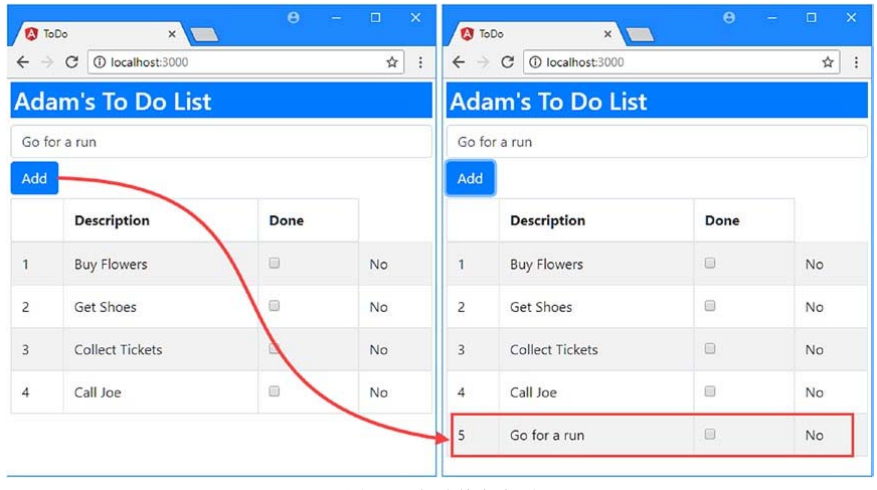

图 2-8　创建待办事项

2.5 本章小结

本章展示了如何创建第一个简单的 Angular 应用程序，从应用程序的 HTML 静态模拟内容转变为动态应用，让用户创建新的待办事项，并将现有事项标记为已完成。

如果现在还不能理解本章的所有内容，不必担心。在这个阶段要了解的重要内容是 Angular 应用程序的大体形态，即它是围绕数据模型、组件和模板而构建的。如果能记住这三个关键构造块，就为理解接下来的内容提供了背景知识。下一章将 Angular 放在这个背景中讲述。

第 3 章

将 Angular 放在上下文中

本章将 Angular 放到 Web 应用程序开发的环境中，为接下来的章节打基础。Angular 的目标是将仅用于服务器端开发的工具和功能引入 Web 客户端，以便开发、测试和维护丰富且复杂的 Web 应用程序。

Angular 允许扩展 HTML 来工作，在习惯它之前，这看起来是个奇怪的想法。Angular 应用程序通过定制元素来表达功能，复杂的应用程序可以生成一个混合了标准标记和定制标记的 HTML 文档。

Angular 支持的开发风格是通过使用“模型-视图-控制器”(MVC)模式派生出来的，尽管有时也称为“模型-视图-之类的”，因为在使用 Angular 时，可以遵循这种模式的变体数不胜数。本书重点介绍标准 MVC 模式，因为它是最成熟和使用最广泛的。下面各节将解释 Angular 能够带来显著好处(以及存在更好选项)的项目特征，描述 MVC 模式，并描述一些常见的缺陷。

本书和 Angular 的发布计划

谷歌为 Angular 采用激进的发布计划。这意味着每六个月就发布一系列次版本和一个主版本。次版本不应该破坏任何现有的特性，而应该主要包含 bug 修复。主要版本可能包含大量更改，但可能不提供向后兼容性。

要求读者每 6 个月买一本新书既不公平也不合理，特别是 Angular 的大部分特性即使在主版本中也不太可能改变。相反，在主版本发布之后，我将向 GitHub 知识库(https://github.com/Apress/pro-angular-6)发布本书的更新。

这对我和 Apress 来说是一个试验，目前还不知道这些更新以何种形式出现。重要的是，我不知道 Angular 的主版本会包含哪些内容，但我们的目标是通过补充它包含的示例来延长本书的寿命。

我没有承诺更新会是什么样的，它们将采取什么样的形式，或者将这些更新放在本书的新版本中之前，它们需要多长时间才能生成。当新的 Angular 版本发布时，请保持开放的心态，并检查本书的存储库。如果对如何在实验过程中改进更新有任何想法，请发送电子邮件到 adam@adam-freeman.com 告诉我。

3.1 理解 Angular 的强项

Angular 并不是解决所有问题的方案，知道什么时候应该使用 Angular，什么时候应该寻找替代方案是很重要的。Angular 提供的功能以前只对服务器端开发人员可用，但目前在浏览器中完全可用。这意味着每次加载应用了 Angular 的 HTML 文档时，Angular 都有很多工作要做：必须编译 HTML 元素，必须评估数据绑定，需要执行组件和其他构建块等，为第 2 章和本书描述的特性构建支持。

这些工作需要时间来完成，时间的长短取决于 HTML 文档的复杂性、相关的 JavaScript 代码，最重要的是浏览器的质量和设备的处理能力。在功能强大的台式机上使用最新的浏览器时，不会注意到任何延迟，但在功能不足的智能手机上使用旧浏览器确实会减慢 Angular 应用程序的初始设置速度。

因此，我们的目标是尽可能少地执行这个设置过程，并在应用程序执行时尽可能多地将其交付给用户。这意味着要仔细考虑所构建的 Web 应用程序的类型。总的来说，有两种 Web 应用程序：往返式和单页式。

3.1.1 往返式应用程序和单页式应用程序

很长时间以来，Web 应用程序都是按照往返模式开发的。浏览器向服务器请求一个初始 HTML 文档。用户交互操作——例如单击链接或提交表单——的结果是，浏览器请求和接收一个全新的 HTML 文档。在这种应用程序中，