

Chandermani Arora, Kevin Hennessy 著

Angular 2实例

(影印版)

Angular 2 by Example

 东南大学出版社
SOUTHEAST UNIVERSITY PRESS



Packt>

Angular 2 实例(影印版)

Angular 2 by Example

Chandermani Arora, Kevin Hennessy 著

南京 东南大学出版社

图书在版编目(CIP)数据

Angular 2 实例:英文/(印)钱德玛尼·阿罗拉
(Chandermani Arora), (美)凯文·汉尼斯(Kevin Hen-
nessy)著. —影印本. —南京:东南大学出版社, 2017.10

书名原文: Angular 2 by Example

ISBN 978-7-5641-7359-3

I. ①A… II. ①钱… ②凯… III. ①超文本标记
语言—程序设计—英文 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 193635 号

图字:10-2017-117 号

© 2016 by PACKT Publishing Ltd

Reprint of the English Edition, jointly published by PACKT Publishing Ltd and Southeast University Press, 2017.
Authorized reprint of the original English edition, 2017 PACKT Publishing Ltd, the owner of all rights to
publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 PACKT Publishing Ltd 出版 2016。

英文影印版由东南大学出版社出版 2017。此影印版的出版和销售得到出版权和销售权的所有者
——PACKT Publishing Ltd 的许可。

版权所有,未得书面许可,本书的任何部分和全部不得以任何形式重制。

Angular 2 实例(影印版)

出版发行:东南大学出版社

地 址:南京四牌楼 2 号 邮编:210096

出 版 人:江建中

网 址: <http://www.seupress.com>

电子邮件: press@seupress.com

印 刷:常州市武进第三印刷有限公司

开 本:787 毫米×980 毫米 16 开本

印 张:32

字 数:627 千字

版 次:2017 年 10 月第 1 版

印 次:2017 年 10 月第 1 次印刷

书 号:ISBN 978-7-5641-7359-3

定 价:96.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Credits

Authors

Chandermani Arora
Kevin Hennessy

Copy Editor

Safis Editing

Reviewer

Josh Kurz

Project Coordinator

Devanshi Doshi

Commissioning Editor

Sarah Crofton

Proofreader

Safis Editing

Acquisition Editor

Kirk D'Costa

Indexer

Mariamamm Chettiyar

Content Development Editor

Samantha Gonsalves

Graphics

Kirk D'Penha

Technical Editor

Madhunikita Sunil Chindarkar

Production Coordinator

Shantanu N. Zagade

About the Authors

Chandermani Arora is a software craftsman, with a passion for technology and expertise on the web stack.

With more than a decade of experience under his belt, he has architected, designed, and developed solutions of all shapes and sizes on the Microsoft platform.

He has been building apps on Angular 1 from its early days. Such is his love for the framework that every engagement that he is a part of has an Angular footprint.

Being an early adopter of the Angular 2 framework, he tries to support the platform in every possible way – be it writing blog posts on various Angular topics or helping his fellow developers on StackOverflow, where he is often seen answering questions on the Angular2 channel.

An ex-MSFT, he now works for Technovert where he leads a bunch of awesome developers who build cloud-scale web applications using Angular and other new age frameworks.

He is also the author for the first edition of this book, *AngularJS by Example*.

Writing this book has just been a surreal experience, and I would like to thank my Technovert family who have supported me in all possible ways, be it helping me with the sample apps, reviewing the content, or offloading some of my professional commitment to make sure I get enough time for book writing. And finally I want to express my gratitude towards my family. I know your blessings are always there with me.

Kevin Hennessy is a Senior Software Engineer with Applied Information Sciences. He has 18 years of experience as a developer, team lead, and solutions architect, working on web-based projects, primarily using the Microsoft technology stack. Over the last several years, he has presented and written about single-page applications and JavaScript frameworks, including Knockout, Meteor, and Angular 2. Most recently, he spoke about Angular 2 at the All Things Open Conference. His corporate blog is <http://blog.appliedis.com/?s=Kevin+Hennessy>.

I would like to acknowledge my wife, Mary Gene Hennessy. Her unstinting love and support (and editorial suggestions) through the period of late nights and weekends I spent writing this book, have made me ever more aware and appreciative of how truly amazing it is to be married to her.

About the Reviewer

Josh Kurz is a Technical Architect at Turner Broadcasting System. He has written a book on AngularJS, called *Mastering AngularJS Directives*, and he has contributed to many open source projects.

I would like to thank my baby girl Evelyn for being the sweetest girl in the world.

www.PacktPub.com

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www.packtpub.com/mapt>

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Table of Contents

| | |
|--|----|
| Preface | 1 |
| Chapter 1: Getting Started | 9 |
| Angular basics | 10 |
| The component pattern | 10 |
| Using the component pattern in web applications | 11 |
| Why weren't components used before in Angular? | 11 |
| What's new that enables Angular to use this pattern? | 12 |
| Web Components | 12 |
| Angular and Web Components | 13 |
| Language support in Angular | 13 |
| ES2015 | 14 |
| TypeScript | 15 |
| Putting it all together | 16 |
| Angular modules | 16 |
| The basic steps to building Angular applications | 17 |
| The customary Hello Angular app – Guess the Number! | 17 |
| Setting up a development server | 18 |
| Building Guess the Number! | 19 |
| Designing our first component | 19 |
| The host file | 20 |
| An HTML page | 20 |
| Script tags | 21 |
| Custom elements | 22 |
| The component file | 22 |
| The import statement | 22 |
| Decorators | 23 |
| Defining the class | 24 |
| The module file | 26 |
| Bootstrapping | 27 |
| We're up-and-running! | 28 |
| Digging deeper | 28 |
| Interpolation | 29 |
| Tracking changes in the number of tries | 30 |
| Expressions | 30 |
| The safe navigation operator | 31 |
| Data binding | 32 |
| Property binding | 32 |

| | |
|---|----|
| Event binding | 32 |
| Structural directives | 33 |
| Revisiting our app | 34 |
| Looking at how our code handles updates | 35 |
| Maintaining the state | 36 |
| Component as the container for the state | 36 |
| Change detection | 37 |
| Initializing the app | 39 |
| Loading the modules needed by our application | 39 |
| Bootstrapping our app | 42 |
| Tools | 43 |
| Resources | 44 |
| Summary | 45 |
| Chapter 2: Building Our First App - 7 Minute Workout | 47 |
| What is 7 Minute Workout? | 48 |
| Downloading the code base | 49 |
| Setting up the build | 50 |
| The build internals | 52 |
| Code transpiling | 52 |
| Organizing code | 54 |
| The 7 Minute Workout model | 55 |
| App bootstrapping | 58 |
| App loading with SystemJS | 59 |
| Our first component – WorkoutRunnerComponent | 60 |
| Component life cycle hooks | 65 |
| Building the 7 Minute Workout view | 69 |
| The Angular 2 binding infrastructure | 72 |
| Interpolations | 73 |
| Property binding | 73 |
| Property versus attribute | 74 |
| Property binding continued... | 75 |
| Quick expression evaluation | 77 |
| Side-effect-free binding expressions | 77 |
| Angular directives | 78 |
| Target selection for binding | 79 |
| Attribute binding | 80 |
| Style and class binding | 81 |
| Attribute directives | 82 |
| Styling HTML with ngClass and ngStyle | 82 |
| Exploring Angular modules | 84 |

| | |
|--|-----|
| Comprehending Angular modules | 84 |
| Adding a new module to 7 Minute Workout | 86 |
| Learning more about an exercise | 88 |
| Adding descriptions and video panels | 88 |
| Providing component inputs | 89 |
| Structural directives | 93 |
| The ever-so-useful NgFor | 94 |
| NgFor performance | 95 |
| Angular 2 security | 96 |
| Trusting safe content | 98 |
| Formatting exercise steps with innerHTML binding | 99 |
| Displaying the remaining workout duration using pipes | 100 |
| Angular pipes | 100 |
| Implementing a custom pipe – SecondsToTimePipe | 102 |
| Adding the next exercise indicator using ngIf | 105 |
| Pausing an exercise | 107 |
| The Angular event binding infrastructure | 110 |
| Event bubbling | 111 |
| Event binding an \$event object | 111 |
| Two-way binding with ngModel | 112 |
| Summary | 113 |
| Chapter 3: More Angular 2 – SPA, Routing, and Data Flows in Depth | 115 |
| Exploring Single Page Application capabilities | 116 |
| The Angular SPA infrastructure | 117 |
| Angular routing | 117 |
| Angular router | 119 |
| Routing setup | 120 |
| Pushstate API and server-side url-rewrites | 121 |
| Adding start and finish pages | 122 |
| Route configuration | 123 |
| Rendering component views with router-outlet | 124 |
| Route navigation | 125 |
| Link parameter array | 127 |
| Using the router service for component navigation | 127 |
| Using the ActivatedRoute service to access route params | 129 |
| Angular dependency injection | 130 |
| Dependency injection 101 | 130 |
| Exploring dependency injection in Angular | 132 |
| Tracking workout history | 133 |
| Building the WorkoutHistoryTracker service | 134 |
| Integrating with WorkoutRunnerComponent | 136 |
| Registering dependencies | 136 |

| | |
|---|-----|
| Angular providers | 137 |
| Value providers | 137 |
| Factory providers | 138 |
| Injecting dependencies | 139 |
| Constructor injection | 139 |
| Explicit injection using injector | 140 |
| Dependency tokens | 140 |
| String token | 141 |
| Integrating with WorkoutRunnerComponent – continued | 142 |
| Adding the workout history page | 143 |
| Sorting and filtering history data using pipes | 145 |
| The orderBy pipe | 145 |
| The search pipe | 147 |
| Pipe gotcha with arrays | 148 |
| Angular change detection overview | 150 |
| Hierarchical injectors | 152 |
| Registering component level dependencies | 152 |
| Angular DI dependency walk | 155 |
| Dependency injection with @Injectable | 157 |
| Tracking route changes using the router service | 159 |
| Fixing the video playback experience | 160 |
| Using thumbnails for video | 161 |
| Using the angular2-modal dialog library | 161 |
| Creating custom dialogs with angular2-modal | 163 |
| Cross-component communication using Angular events | 165 |
| Tracking exercise progress with audio | 165 |
| Building Angular directives to wrap HTML audio | 166 |
| Creating WorkoutAudioComponent for audio support | 168 |
| Understanding template reference variables | 172 |
| Template variable assignment | 173 |
| Using the @ViewChild decorator | 173 |
| The @ViewChildren decorator | 174 |
| Integrating WorkoutAudioComponent | 175 |
| Exposing WorkoutRunnerComponent events | 176 |
| The @Output decorator | 177 |
| Eventing with EventEmitter | 178 |
| Raising events from WorkoutRunnerComponent | 179 |
| Component communication patterns | 180 |
| Injecting a parent component into a child component | 181 |
| Using component lifecycle events | 183 |
| Sibling component interaction using events and template variables | 184 |
| Summary | 187 |
| Chapter 4: Personal Trainer | 189 |

| | |
|---|-----|
| The Personal Trainer app – the problem scope | 190 |
| Personal Trainer requirements | 191 |
| The Personal Trainer model | 191 |
| Sharing the workout model | 192 |
| The model as a service | 193 |
| The Personal Trainer layout | 193 |
| Personal Trainer navigation with routes | 194 |
| Getting started | 195 |
| Introducing child routes to Workout Builder | 198 |
| Adding the child routing component | 199 |
| Updating the WorkoutBuilder component | 201 |
| Updating the Workout Builder module | 202 |
| Updating app.routes | 203 |
| Putting it all together | 203 |
| Lazy loading of routes | 205 |
| Integrating sub- and side-level navigation | 211 |
| Sub-level navigation | 211 |
| Side navigation | 212 |
| Implementing workout and exercise lists | 214 |
| WorkoutService as a workout and exercise repository | 214 |
| Workout and exercise list components | 217 |
| Workout and exercise list views | 218 |
| Workouts list views | 218 |
| Exercises list views | 221 |
| Building a workout | 222 |
| Finishing left nav | 223 |
| Adding WorkoutBuilderService | 224 |
| Adding exercises using ExerciseNav | 226 |
| Implementing the Workout component | 227 |
| Route parameters | 227 |
| Route guards | 228 |
| Implementing the CanActivate route guard | 229 |
| Implementing the Workout component continued... | 231 |
| Implementing the Workout template | 232 |
| Angular forms | 233 |
| Template-driven and model-driven forms | 234 |
| Template-driven forms | 234 |
| Getting started | 234 |
| Using NgForm | 235 |
| ngModel | 236 |
| Using ngModel with input and textarea | 237 |

| | |
|--|-----|
| Using ngModel with select | 239 |
| Angular validation | 240 |
| ngModel | 240 |
| The Angular model state | 241 |
| Angular CSS classes | 241 |
| Workout validation | 243 |
| Displaying appropriate validation messages | 243 |
| Adding more validation | 244 |
| Managing multiple validation messages | 245 |
| Custom validation messages for an exercise | 246 |
| Saving the workout | 247 |
| More on NgForm | 249 |
| Fixing the saving of forms and validation messages | 250 |
| Model-driven forms | 252 |
| Getting started with model-driven forms | 253 |
| Using the FormBuilder API | 255 |
| Adding the form model to our HTML view | 257 |
| Adding form controls to our form inputs | 257 |
| Adding validation | 258 |
| Adding dynamic form controls | 259 |
| Saving the form | 260 |
| Custom validators | 261 |
| Integrating a custom validator into our forms | 262 |
| Summary | 263 |

Chapter 5: Supporting Server Data Persistence 265

| | |
|--|-----|
| Angular and server interactions | 266 |
| Setting up the persistence store | 266 |
| Seeding the database | 268 |
| The basics of the HTTP module | 269 |
| Personal Trainer and server integration | 270 |
| Loading exercise and workout data | 270 |
| Loading exercise and workout lists from a server | 271 |
| Adding the HTTP module and RxJS to our project | 272 |
| Updating workout-service to use the HTTP module and RxJS | 272 |
| Modifying getWorkouts() to use the HTTP module | 274 |
| Updating the workout/exercise list pages | 275 |
| Mapping server data to application models | 276 |
| Loading exercise and workout data from the server | 279 |
| Fixing the builder services | 281 |
| Fixing the Workout and Exercise components | 282 |
| Updating the router guards | 283 |
| Performing CRUD on exercises/workouts | 284 |
| Creating a new workout | 285 |

| | |
|---|-----|
| Updating a workout | 286 |
| Deleting a workout | 287 |
| Fixing the upstream code | 287 |
| Using promises for HTTP requests | 289 |
| The async pipe | 291 |
| Cross-domain access and Angular | 292 |
| Using JSONP to make cross-domain requests | 292 |
| Cross-origin resource sharing | 296 |
| Handling workouts not found | 297 |
| Fixing the 7 Minute Workout app | 299 |
| Summary | 300 |
| Chapter 6: Angular 2 Directives in Depth | 301 |
| Classifying directives | 302 |
| Components | 302 |
| Attribute directives | 302 |
| Structural directives | 303 |
| Building a remote validator directive | 303 |
| Validating workout names using async validator | 305 |
| Building a busy indicator directive | 310 |
| Injecting optional dependencies with the @Optional decorator | 312 |
| Implementation 1 – using renderer | 313 |
| Angular renderer, the translation layer | 316 |
| Host binding in directives | 317 |
| Property binding using @HostBinding | 317 |
| Attribute binding | 318 |
| Event binding | 319 |
| Implementation 2 – BusyIndicatorDirective with host bindings | 319 |
| Directive injection | 321 |
| Injecting directives defined on the same element | 322 |
| Injecting directive dependency from the parent | 322 |
| Injecting a child directive (or directives) | 323 |
| Injecting descendant directive(s) | 324 |
| Building an Ajax button component | 324 |
| Transcluding external components/elements into a component | 328 |
| Content children and view children | 328 |
| Injecting view children using @ViewChild and @ViewChildren | 331 |
| Tracking injected dependencies with QueryList | 332 |
| Injecting content children using @ContentChild and @ContentChildren | 333 |
| Dependency injection using viewProvider | 334 |
| Understanding structural directives | 338 |

| | |
|--|-----|
| TemplateRef | 339 |
| ViewContainerRef | 340 |
| Component styling and view encapsulation | 341 |
| Overview of Shadow DOM | 342 |
| Shadow DOM and Angular components | 344 |
| Summary | 348 |
| Chapter 7: Testing Personal Trainer | 349 |
| The need for automation | 350 |
| Testing in Angular | 350 |
| Types of testing | 351 |
| Testing – who does it and when? | 351 |
| The Angular testing ecosystem | 352 |
| Getting started with unit testing | 353 |
| Setting up Karma for unit testing | 354 |
| The Karma configuration files | 355 |
| The Karma test shim file | 357 |
| Organization and naming of our test files | 359 |
| Unit-testing Angular applications | 360 |
| Unit-testing pipes | 360 |
| Running our test files | 362 |
| Unit-testing components | 364 |
| Angular testing utilities | 364 |
| Managing dependencies in our tests | 365 |
| Unit-testing WorkoutRunnerComponent | 365 |
| Setting up component dependencies | 366 |
| Mocking dependencies – workout history tracker | 366 |
| Mocking dependencies – workout service | 367 |
| Mocking dependencies – router | 368 |
| Configuring our test using TestBed | 368 |
| Starting unit testing | 371 |
| Debugging unit tests in Karma | 371 |
| Unit-testing WorkoutRunner continued... | 373 |
| Using Jasmine spies to verify method invocations | 374 |
| Using Jasmine spies to verify dependencies | 375 |
| Testing event emitters | 376 |
| Testing interval and timeout implementations | 377 |
| Testing workout pause and resume | 378 |
| Unit-testing services | 379 |
| Mocking HTTP request/response with MockBackend | 379 |
| Unit-testing directives | 383 |
| The TestBed class | 384 |
| Testing remote validator | 384 |

| | |
|--|-----|
| Getting started with E2E testing | 387 |
| Introducing Protractor | 388 |
| Setting up Protractor for E2E testing | 390 |
| TypeScript configuration | 391 |
| Writing E2E tests for the app | 392 |
| Executing our E2E tests | 393 |
| Setting up backend data for E2E testing | 395 |
| More E2E tests | 395 |
| Testing WorkoutRunner | 397 |
| Using page objects to manage E2E testing | 397 |
| Summary | 400 |
| Chapter 8: Some Practical Scenarios | 401 |
| Building a new app | 402 |
| Seed projects | 402 |
| Seed and scaffolding tools | 403 |
| Yeoman | 403 |
| angular-cli | 404 |
| Angular 2 performance | 405 |
| Byte size | 405 |
| Initial load time and memory utilization | 406 |
| The Angular rendering engine | 407 |
| Server-side rendering | 408 |
| Offloading work to a web worker | 408 |
| Performant mobile experience | 410 |
| Change detection improvements | 411 |
| Change detection | 411 |
| Change detection setup | 412 |
| When does change detection kick in? | 413 |
| How does change detection work? | 416 |
| Change detection performance | 420 |
| Using immutable data structures | 421 |
| Using Observables | 423 |
| Manual change detection | 424 |
| Handling authentication and authorization | 425 |
| Cookie-based authentication | 426 |
| Token-based authentication | 429 |
| Handling authorization | 436 |
| Adding authorization support | 436 |
| Sharing user authentication context | 437 |
| Restricting routes | 437 |
| Conditionally rendering content based on roles | 438 |

| | |
|---|-----|
| Migrating Angular 1 apps | 439 |
| Should I migrate? | 439 |
| Advantages of Angular 2 | 440 |
| Developing Angular 1 apps today for easy migration | 441 |
| One component per file | 441 |
| Avoiding inline anonymous functions | 441 |
| Avoiding \$scope! | 442 |
| Using controller as (controller aliasing) syntax everywhere | 443 |
| Avoiding ng-controller | 444 |
| Building using the Angular 1.5+ component API | 445 |
| What to migrate? | 446 |
| Preparing for Angular 2 migration | 447 |
| Identifying third-party dependencies | 447 |
| jQuery libraries | 447 |
| Angular 1 libraries | 447 |
| Choice of language | 448 |
| Migrating Angular 1's Personal Trainer | 449 |
| Setting up Angular 1's Personal Trainer locally | 449 |
| Identifying dependencies | 450 |
| Setting up the module loader | 451 |
| Enabling TypeScript | 454 |
| Adding Angular 2 | 456 |
| Bootstrapping the hybrid app | 458 |
| Injecting Angular 2 components into Angular 1 views | 460 |
| Migrating our first view to Angular 2 component | 460 |
| Injecting Angular 1 dependencies into Angular 2 | 462 |
| Registering Angular 2 components as directives | 463 |
| Rules of engagement | 464 |
| Angular 1 directives and Angular 2 components | 465 |
| Resource sharing and dependency injection | 466 |
| Sharing an Angular 1 service | 466 |
| Sharing an Angular 2 service | 467 |
| Change detection | 468 |
| Migrating the start and finish pages | 468 |
| Angular 1 directive upgrade | 470 |
| Replacing angular-translate with ng2-translate | 471 |
| Using a bootstrap-ready callback for initialization | 472 |
| Integrating the start and finish pages | 474 |
| Getting rid of angular-translate | 475 |
| Replacing the ui-bootstrap library | 477 |
| Learnings | 480 |
| Summary | 481 |
| Index | 483 |