AngularJS 인 액션

AngularJS In Action
by Brian Ford and Lukas Ruebbelke

AngularJS 인 액션

초판 1쇄 발행 2015년 12월 22일

지은이 루카스 루벨키, 브라이언 포드
옮긴이 장현희
펴낸이 장성두
펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 문발로 141 뮤즈빌딩 403호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **이메일** jeipub@gmail.com

편집부 이민숙, 이 슬, 이주원 / **소통·기획팀** 민지환, 현지환
본문디자인 성은경
용지 신승지류유통 / **인쇄** 해외정판사 / **제본** 광우제책사

ISBN 979-11-85890-39-5 (93000)
값 22,000원

AngularJS 인 액션
AngularJS in Action

루카스 루벨키, 브라이언 포드 지음 / 장현희 옮김

제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저/역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 기재한 회사명 및 제품명은 각 회사의 상표 및 등록명입니다.
- 이 책에서는 ©, ®, ™ 등의 기호를 생략하고 있습니다.
- 이 책에서 사용하고 있는 실제 제품 버전은 독자의 학습 시점에 따라 책의 버전과 다를 수 있습니다.
- 이 책의 예제 소스 파일들은 다음 사이트에서 다운로드할 수 있습니다.
  – https://github.com/angularjs-in-action
- 책의 내용과 관련된 문의사항은 옮긴이나 출판사로 연락주시기 바랍니다.
  – 옮긴이: aspnetmvp@gmail.com
  – 출판사: jeipub@gmail.com

차례

추천사 x
옮긴이 머리말 xi
머리말 xii
감사의 말 xiii
이 책에 대하여 xiv
저자 소개 xviii
책 표지에 대하여 xix
베타리더 후기 xx

PART I AngularJS와 친해지기 1

CHAPTER 1 안녕하세요, AngularJS 3

1.1 AngularJS의 장점 4
1.2 AngularJS의 큰 그림 살펴보기 7
1.3 첫 번째 AngularJS 애플리케이션 8
1.3.1 모듈 12 / **1.3.2** 뷰와 컨트롤러 13 / **1.3.3** 서비스 18 / **1.3.4** 디렉티브 21
1.4 요약 23

CHAPTER 2 AngularJS 애플리케이션 구성하기 25

2.1 안젤로 살펴보기 25
2.2 AngularJS 애플리케이션 구조 29
2.3 안젤로의 기반 살펴보기 31
2.4 라우트의 구성 및 탐색 33
2.5 몇 가지 모범 사례 37
2.6 요약 39

PART II AngularJS 제대로 활용하기 41

CHAPTER **3** 뷰와 컨트롤러 43

3.1 큰 그림 44
3.2 AngularJS에서의 뷰의 정의 46
3.3 AngularJS에서의 컨트롤러의 정의 48
3.3.1 다이제스트 주기 48 / **3.3.2** controller-as 문법 50 / **3.3.3** AngularJS의 이벤트 51
3.4 속성과 표현식 52
3.4.1 ngRepeat를 이용해서 스토리 표시하기 52 / **3.4.2** 필터 56 / **3.4.3** 표현식 58
3.5 모범 사례 및 테스트 65
3.6 요약 68

CHAPTER **4** 모델과 서비스 69

4.1 모델 및 서비스의 개념 70
4.1.1 서비스 이해하기 71 / **4.1.2** 서비스의 생명주기 72 / **4.1.3** 서비스의 종류 73
4.2 $http를 이용한 모델 76
4.2.1 $http 서비스란? 77 / **4.2.2** 첫 모델 구현하기 77 / **4.2.3** $http 서비스의 편의 메서드들 79
4.3 프라미스 82
4.3.1 프로미스란 무엇일까? 82 / **4.3.2** 프로미스 활용하기 82
4.3.3 $http.success와 $http.error 메서드 85
4.3.4 프로미스를 이용해 코드를 순차적으로 실행하기 86
4.4 $http 인터셉터 87
4.4.1 인터셉터가 필요한 이유 87 / **4.4.2** 인터셉터의 활용 87
4.5 서비스 데코레이터 89
4.5.1 왜 데코레이터가 필요할까? 89 / **4.5.2** 로그 개선하기 89
4.6 테스트에 대한 고려 91
4.6.1 서비스 테스트하기 91 / **4.6.2** $httpBackend 서비스를 이용해서 원격 서버를 모의 호출하기 93
4.6.3 모범 사례 95
4.7 요약 96

CHAPTER 5 디렉티브 97

5.1 디렉티브 소개 97

5.1.1 디렉티브란 무엇인가? 97 / **5.1.2** 디렉티브가 필요한 이유 98
5.1.3 왜 우리는 디렉티브를 원하는가? 98

5.2 디렉티브 기초 다지기 99

5.2.1 사용자 스토리 디렉티브 99

5.3 고급 기능들 107

5.3.1 드래그 앤 드롭 기능 107 / **5.3.2** drag-container 디렉티브 적용하기 110
5.3.3 컨트롤러 구현하기 110 / **5.3.4** drop-container 디렉티브 구현하기 112
5.3.5 drop-container 디렉티브 활용하기 115 / **5.3.6** 컨트롤러 구현하기 115
5.3.7 drop-target 디렉티브 구현하기 123 / **5.3.8** drop-target 디렉티브의 활용 125
5.3.9 컨트롤러 구현하기 126 / **5.3.10** $dragging 서비스 구현하기 127
5.3.11 StoryboardCtrl 컨트롤러 수정하기 128

5.4 서드파티 라이브러리 다시 활용하기 130

5.4.1 Flot 설치하기 131 / **5.4.2** 디렉티브 구현하기 132 / **5.4.3** 디렉티브 활용하기 132
5.4.4 데이터 처리하기 133 / **5.4.5** 이제는 '격리된 스코프'에 대해 살펴볼 시간 134
5.4.6 최종 마무리: Flot 통합하기 137

5.5 디렉티브 테스트하기 138

5.6 모범 사례 141

5.7 요약 141

CHAPTER 6 애니메이션 143

6.1 애니메이션 소개 143

6.1.1 AngularJS가 애니메이션을 처리하는 방식 144 / **6.1.2** 애니메이션의 이름 규칙 145
6.1.3 애니메이션 활성화하기 145

6.2 CSS 트랜지션 146

6.2.1 기본 트랜지션 구현하기 147 / **6.2.2** ng-enter 트랜지션 정의하기 147
6.2.3 ng-leave 트랜지션 정의하기 147 / **6.2.4** 이동 애니메이션 구현하기 148

6.3 CSS 애니메이션 150

6.3.1 기본 애니메이션 클래스 정의하기 150 / **6.3.2** 애니메이션 키프레임 정의하기 151
6.3.3 요소에 이동 애니메이션 적용하기 154

6.4 자바스크립트 애니메이션 155

6.4.1 자바스크립트 애니메이션 정의하기 156 / **6.4.2** 자바스크립트 애니메이션 이벤트 156
6.4.3 자바스크립트 애니메이션 클래스 157 / **6.4.4** TweenMax 라이브러리 158
6.4.5 애니메이션을 실제로 적용하기 159

6.5 테스트 160

6.6 모범 사례 160

6.7 요약 161

CHAPTER 7 라우트로 웹사이트 구성하기 163

7.1 AngularJS 라우팅 구성 요소 164

7.2 AngularJS에서 라우트 생성하기 165

7.2.1 ngRoute와 ngView를 이용해서 첫 번째 라우트 정의하기 165 / **7.2.2** ngView 추가하기 165
7.2.3 $routeProvider를 이용해서 라우트 설정하기 166 / **7.2.4** 라우트 탐색 설정하기 167
7.2.5 리뷰 168

7.3 라우트에 매개변수 사용하기 168

7.3.1 리뷰 171

7.4 라우트에서 의존성 해석하기 171

7.4.1 리뷰 173

7.5 라우트 이벤트 173

7.5.1 리뷰 174

7.6 테스트 174

7.7 모범 사례 176

7.8 요약 176

CHAPTER 8 폼과 유효성 검사 179

8.1 AngularJS의 폼 유효성 검사 180

8.1.1 HTML 폼 요소 확장하기 180 / **8.1.2** 유효성 검사 추가하기 182
8.1.3 유효성 검사와 CSS 187 / **8.1.4** 폼 유효성 검사, $setPrisitine, 그리고 $setUntouched 188

8.2 테스트 188

8.3 모범 사례 192

8.4 요약 192

APPENDIX A 카르마 설치하기 193

Node.js와 노드 패키지 매니저(NPM) 설치하기 193 / Windows 환경에서 Node.js 설치하기 194
Mac OS X 환경에서 Node.js 설치하기 199 / 리눅스에서 Node.js 설치하기 204

Git 설치하기 204

Windows용 Git 설치하기 204 / Mac OS X에서 Git 설치하기 205 / 리눅스에서 Git 설치하기 206
카르마 패키지 설치하기 206 / 카르마 초기화하기 207 / 카르마 활용하기 211

APPENDIX B Node.js 서버 설정하기 212

안젤로의 Node.js 백엔드 소스 코드 다운로드하기 212

몽고DB 설치하기 213

Windows에서 몽고DB 설치하기 213 / 맥 OS X에서 몽고DB 설치하기 217
Node.js 백엔드와 함께 안젤로 실행하기 218

APPENDIX C Firebase 서버 설정하기 219

Firebase 계정 생성하기 219 / 앱 관리하기 220 / 안젤로와 Firebase 통합하기 221

APPENDIX D 안젤로 앱 실행하기 222

안젤로 소스 코드 다운로드하기 222 / 웹 서버 실행하기 223

찾아보기 225

추천사

나는 ng-europe 콘퍼런스에서 처음 루카스(Lukas)를 만나자마자 그가 얼마나 멋진 친구인지를 바로 알 수 있었다. 우리는 콘퍼런스가 끝난 후에도 이야기를 계속했고, 그가 주최한 해커톤(Hackathon)에도 참가하는 등 지속적인 관계를 맺어왔다. 우리가 공유하고 함께 끌어냈던 열정은 사람들이 뭔가 멋진 것을 만들 수 있도록 돕는 일이었다. 그리고 그가 이 책을 쓰기로 한 이유 역시 같다. 본인 자신을 위한 것이 아니라 바로 독자 여러분을 위해서!

AngularJS는 매우 이른 시간에 대중적인 프론트엔드 프레임워크로 자리를 잡았다. 또한, 가장 빠르게 변화하고 있는 프레임워크이기도 하다. 이 책을 통해 독자 여러분은 컨트롤러(Controller), 템플릿(Template), 디렉티브(Directive), 서비스(Service), 팩토리(Factory) 그리고 프로바이더(Provider) 등을 공부하면서 AngularJS를 이용해 멋진 애플리케이션을 구현하는 방법을 처음부터 끝까지 살펴볼 수 있다. 이 책은 AngularJS의 기능들을 아주 상세히 들여다보지는 않지만, AngularJS가 어떤 기능들을 제공하며 각 기능을 어떻게 활용할 것인지를 직접 실습해볼 수 있도록 안내한다. 게다가 애플리케이션을 구현하는 방법뿐만 아니라 코드를 테스트하는 방법도 학습하게 된다.

안젤로(Angello)는 매우 흥미로운 애플리케이션이다. 이 애플리케이션은 AngularJS를 위해 만들어진 라이브러리 중 가장 많은 관심을 받는 라이브러리를 이용해 구현하는 Trelllo와 아주 유사한 프로젝트 관리 도구다. 이 책을 통해 안젤로의 여러 기능을 구현한 방법을 살펴보면서 ui-router와 Auth0, Firebase 및 기타 라이브러리들을 사용하는 방법을 학습하게 될 것이다.

이 책의 진정한 가치를 곧 발견하게 될 것이라 확신한다. 즐겁게 읽어주기를 바란다!

마틴 곤토브니카스(Martin Gontovnikas)

_Auth0의 개발 전도사

옮긴이 머리말

이 책의 번역을 맡게 된 시점에 역자는 한창 AngularJS를 기반으로 웹 애플리케이션을 개발하던 중이었다. AngularJS에 경험이 풍부한 동료와 함께 프로젝트를 맡았는데, 역자는 백엔드 시스템 개발에 주력하고 있던 터라 프론트 엔드 영역을 개발할 때는 필요할 때마다 관련 자료나 도움말을 찾아보면서 업무를 진행했었다. 그때 마침 이 책의 번역 의뢰를 받고서는 매일 출퇴근하는 버스 안에서 조금씩 번역을 해 나가면서 AngularJS의 기초를 다질 수 있었고, 덕분에 프로젝트 역시 무난히 진행할 수 있었다.

이 책은 단일 페이지 애플리케이션(SPA, Single Page Application) 개발에 최적화된 AngularJS 프레임워크를 처음 접하는 개발자에게 적합한 서적이다. 중간중간 저자도 언급했지만, AngularJS의 기본 모듈을 대체하여 더 자주 사용되는 서드파티 모듈들에 대해서는 고려하지 않는다. AngularJS 자체에만 집중하고 있어서 전반적으로 고급 기법을 소개하기보다는 기본에 충실한 서적이라고 할 수 있다. 그런데도 완전하게 동작하는 하나의 애플리케이션을 처음부터 끝까지 구현해 보는 접근법은 전통적이지만 확실히 도움된다. 특히, AngularJS의 구성 요소들의 특성상 단편적인 설명이나 간단한 해설만으로는 명확하게 개념을 잡기가 다소 어려울 수도 있기에 이 방법이 더 유용한 것 같다.

역자의 경험을 토대로 고백하자면, AngularJS 프레임워크는 다른 프레임워크에 비해 학습 곡선이 높은 편이다. 하지만 일단 모든 개념을 이해하고 나면 이만큼 강력한 프레임워크가 드문 것도 사실이다. 모쪼록 이 책이 AngularJS를 본격적으로 활용하고자 하는 독자 여러분에게 좋은 길라잡이가 되어주기를 바라 마지않는다.

늘 좋은 책을 만들기 위해 불철주야 노력하시는 장성두 실장님과 제이펍 식구들, 그리고 사랑하는 가족들에게 지면을 빌어 감사의 말씀을 전하며 머리말을 마친다.

옮긴이 **장현희**

머리말

평소 필자의 소망 중 가장 원대했던 꿈은 책을 집필하는 것이었다. 그리고 마침내 그 꿈을 이루게 되었다. 필자는 아직도 워드프레스 블로그에 AngularJS에 대한 첫 번째 포스트를 '발행'했을 때의 부담감을 생생히 기억하고 있다. 그것은 마치 미사일을 발사하는 버튼을 누르는 것과도 같은 느낌이었고, 이후 이 포스트로 인해 어떤 일이 벌어질 줄은 전혀 몰랐다. 다만, 이 미사일들을 발사했다는 사실이 가장 난해하면서도 예기치 못한 방법으로 내 삶을 바꿔놓으리라는 것만을 어렴풋이 느꼈을 뿐이다.

완벽한 개발자가 되는 것보다 누군가에게 더 많은 도움을 주기로 한 순간은 필자의 인생에서 가장 중요한 전환점이었다. 별 볼 일 없는 블로그 포스트를 작성하기 시작한 후로 다양한 사람의 피드백을 살펴볼 기회를 얻으면서 필자는 점차 탄탄한 블로그 포스트 쓰는 법을 터득해갔다. 그러면서 AngularJS 코어팀과도 친분을 맺게 되는 계기가 되기도 했다. 당시의 AngularJS는 아직 상대적으로 대중화되지 못했었기 때문에 그들의 레이더에 필자가 포착될 수 있었을 것이다. 한마디로 운이 좋았다.

이 책의 집필 의뢰 메일을 받은 그 날 오후는 아마 절대 잊지 못할 것이다. 필자를 보는 사람이 아무도 없다는 것을 확인하고는 온종일 이 작은 행복을 누리며 춤을 춰댔다. 고등학교 시절의 은사님이 지금 필자의 모습을 보았더라면 좋았을 텐데!

그 순간 이후로 필자가 대형 애플리케이션을 개발하고 AngularJS 코드에 기여하면서 깨우친 모든 것을 독자 여러분과 공유할 수 있다는 사실은 엄청난 특권이라고 생각했다. 필자의 인생에 있어 가장 멋진 여정이었으며, 이 책이 출간되기까지 도움을 주신 모든 분에게 감사의 인사를 전한다.

루카스 루벨키(Lukas Ruebbelke)

감사의 말

AngularJS 커뮤니티의 전폭적인 지원이 없었다면 이 책은 출간되지 못했을 것이다. 브래드 그린(Brad Green), 이고르 미나르(Igor Minar), 미스코 헤베리(Misko Hevery), 브라이언 포드(Brian Ford) 그리고 마티어스 니에메라(Matias Niemela)가 보여준 우정과, 뭔가 멋들어진 것을 만드는 것처럼 보이는 훌륭한 예제에 도움을 준 것에 대해 감사한다. 그리고 많은 피드백을 남겨준 제프 웰프레이(Jeff Whelpley), 브랜든 틸레이(Brandon Tilley), 오마르 곤잘레즈(Omar Gonzalez), 마틴 곤토브니카스(Martin Gontovnikas), 조 임스(Joe Eames) 외 많은 분께 감사한다. 이 모든 분의 도움 덕에 안젤로와 이 책을 만들어낼 수 있었다. 그리고 끝내주는 드래그 앤 드롭 예제에 도움을 준 조프 굿맨(Geoff Goodman)에게도 고마움을 표하고 싶다. 그리고 이 책을 끝까지 마칠 수 있도록 많은 도움을 아끼지 않은 조나단 가비(Jonathan Garvey)에게는 정말 큰 신세를 졌다. 여러분이 지금 이 책을 볼 수 있게 된 것도 모두 조나단의 도움과 열정 덕분이다. 추천사를 써준 마틴과 이 프로젝트의 시작에 기여해준 브라이언에게는 특별히 고마움을 전한다.

이 책의 편집자인 신시아 캐인(Cynthia Kane)에게도 감사한다. 그녀는 무한한 인내심으로 내가 더 나은 필자가 될 수 있도록 도와주었으며, 때로는 내가 그다지 좋아하지 않는(예를 들면, 글쓰기 같은) 일을 계속할 수 있도록 옆에서 자극을 주기도 했다. 또한, 이 책을 위해 함께 일해준 매닝(Manning) 출판사의 모든 관계자 여러분에게도 감사한다.

매닝의 미리 보기 프로그램(MEAP, Manning Early Access Program) 독자 및 이 책을 리뷰해준 아메드 카타브(Ahmed Khatta), 브라이언 쿡시(Brian Cooksey), 채드 데이비스(Chad Davis), 대니얼 브레토이(Daniel Bretoi), 페르난도 몬테이로 코바야시(Fernando Monteiro Kobayashi), 그레고리 쥬로우스키(Gregor Zurowski), 지라니 샤익(Jeelani Shaik), 제프 콘달(Jeff Condal), 제프 큐닝험(Jeff Cunningham), 리차드 스캇-로빈슨(Rechard Scott-Robinson), 로버트 카스토(Robert Casto), 로베르토 로야스(Roberto Rojas) 그리고 윌리엄 E. 윌러(William E. Wheller)를 비롯해 많은 독자가 이 책의 초벌 원고를 읽어보고 수많은 오탈자 교정과 피드백을 제공해주었다. 모두에게 감사하며, 특히 기술자문을 맡아 이 책의 코드를 모두 검사하고 책이 출간되기 직전의 짧은 시간에도 이 책의 초벌 원고를 읽고 검토해준 리차드 스캇-로빈슨에게 특히 감사의 인사를 전한다.

이 책에 대하여

이 책의 목표는 독자들이 실무에서 활용할 수 있는 실용적인 기법을 익히게 하며, 의미 있는 애플리케이션을 개발하기 위해 필요한 도움을 첫 단계에서부터 주는 동시에 가장 실용적인 조언을 제공하는 것에 있다. 이 책에서 제공하는 예제 애플리케이션인 안젤로(Angello)는 Firebase 및 Node.js를 기반으로 한 백엔드 코드가 포함되어 있으며, Auth0를 이용한 소셜 로그인과 같은 몇 가지 추가 기능을 제공한다. 물론, 모든 기능은 완벽하게 동작한다.

우리는 이 책을 통해 어떤 범위의 주제를 다룰 것인지, 그리고 더 중요한 것은 어떤 주제를 다루지 않을 것인지를 결정하기 위해 진지하게 고민했다. 물론, 토끼굴을 파헤치듯이 AngularJS의 내부 구조를 샅샅이 소개하기가 조금 더 쉬웠겠지만, 그렇게 얻어진 지식만으로는 실제 애플리케이션을 구현하기는 어렵다. 아마도 이 책을 통해 AngularJS의 모든 것을 다루지 않는다는 것을 인정하는 필자는 우리가 처음이 아닐까 싶다. AngularJS의 모든 것을 다루려고 했다면 이 책은 지금보다 세 배쯤 더 두꺼워졌을 것이다.

우리는 이 책을 읽는 독자들에 대해 몇 가지를 가정하고 그 내용은 이 책에서 다루지 않는다. 독자들이 HTML, CSS 그리고 자바스크립트에 대한 기본적인 지식을 갖춘 것으로 간주한다. 그래서 AngularJS를 통해 달성하고자 하는 일과 밀접하게 관련이 있지 않는 한, CSS나 HTML에 대해서 언급하지 않을 것이다.

로드맵

우리는 이 책을 크게 두 섹션으로 구분한다. 첫 번째 섹션은 AngularJS에 대해 친절하게 소개하는 부분이며, 그 이후에는 안젤로의 구현을 시작하면서 AngularJS의 여러 부분에 대한 조언들을 제공하는 부분으로 이루어져 있다.

I부, 'AngularJS와 친해지기'에서는 AngularJS를 개략적으로 살펴보면서 각각의 개념이 어떻게 동작하는지, 그리고 이들을 어떻게 한데 뭉쳐 사용할 수 있는지를 살펴본다(제1장). 그리고 이

런 개념들을 재정립하기 위해 직접 AngularJS 애플리케이션을 개발해본다. 제2장에서 개발할 예제 애플리케이션은 우리가 최종적으로 구현할 애플리케이션을 축소한 것이다.

II부, 'AngularJS 제대로 활용하기'에서는 서버 측 커뮤니케이션, 디렉티브, 애니메이션, 라우팅 및 폼과 유효성 검사 등 고급 주제들을 다룬다. II부를 구성하는 각 장마다 개별적인 주제들의 근간이 되는 개념을 살펴보고, 실제 애플리케이션에서 이런 개념들을 어떻게 활용하는지를 살펴보게 된다. 그리고 각 장을 마무리할 때는 테스트와 실제 사례들을 논의할 것이다. 제3장에서는 사용자가 실제로 보게 될 화면을 구성하기 위해 뷰와 컨트롤러를 결합하는 방법을 살펴보고, 사용자의 동작을 가로채서 이벤트들을 처리하는 방법을 알아본다. 제4장에서는 서비스를 통해 컨트롤러를 확장하는 기법을 소개한 후에 $http 서비스를 이용해 원격 서버와 통신하는 방법을 알아본다. 제5장에서는 보다 복잡한 기능을 구현하기 위해 레이아웃을 디렉티브로 컴포넌트화하는 방법을 설명한다. 그리고 제6장에서는 애니메이션을 활용함으로써 레이아웃을 보다 미려하게 만드는 방법도 살펴본다. 제7장에서는 AngularJS에서 라우팅을 활용해 애플리케이션의 상태를 연결하는 방법과 resolve 컴포넌트를 이용해 필요한 데이터를 미리 로드하는 방법, 그리고 $routeParams 서비스를 통해 라우트 간 변수를 전달하는 방법을 알아보게 된다. 그리고 제8장에서는 사용자가 입력하는 데이터를 보호하기 위한 폼 유효성 검사를 통해 사용자 경험을 끌어올리는 방법을 마지막으로 살펴보게 된다.

그리고 부록에서는 카르마(Karma) 프레임워크와 Node.js 서버, Firebase 서버를 설정하고 예제 애플리케이션을 실행하는 방법에 대해 알아본다.

소스 코드에 적용된 규칙과 예제 다운로드

이 책에 사용된 예제 소스 코드는 주변의 글꼴과는 다른 고정폭 글꼴을 사용해 표시한다. 일부 예제에서는 핵심 개념을 설명하기 위해 코드에 주석을 덧붙이는 경우도 있으며, 코드에 대한 부가 설명을 위해 본문 중에 숫자를 표시하는 경우도 있다. 코드는 지면의 활용도를 높이기 위해 줄바꿈과 들여쓰기를 이용해 정렬해두었다.

이 책의 예제 코드는 Github(https://github.com/angularjs-in-action)에서 찾아볼 수 있다. 예제 애플리케이션의 전체 코드는 https://github.com/angularjs-in-action/angello 저장소에 보관 중이다. 또한, 이 예제의 축소 버전의 코드는 https://github.com/angularjs-in-action/angello-lite 저장소에 찾을 수 있다.

애플리케이션의 설치 및 실행에 대한 상세한 절차는 리드미(readme) 파일에 작성해두었다. 저장소는 계속해서 수정 및 버그 패치가 이루어질 예정이니 자주 방문해서 확인하기 바란다.

그리고 출판사의 웹사이트(http://www.manning.com/AngularJsSinAction)에서도 소스 코드를 다운로드할 수 있도록 제공하고 있다.

> **참고** 이 책을 쓰는 현재 Angular 버전 2의 알파 버전이 릴리즈되었지만, 아직 제대로 된 애플리케이션의 구현에 사용할 수 있는 수준은 아니라고 판단한다. 관련해서 최대한 빠른 시일 내에 Angular 2 버전을 기반으로 안젤로를 업그레이드할 계획이다.

소프트웨어 요구사항

예제 애플리케이션을 실행하기 위해서는 Node.js를 설치해야 한다. Node.js의 설치 방법은 https://nodejs.org의 문서를 참고하기 바란다. 또한, 테스트 코드의 실행을 위해 카르마 프레임워크의 설치 역시 필요하다. 관련 내용은 http://karma-runner.github.io/0.12/index.html 페이지를 참고하기 바란다. 웹 애플리케이션을 브라우저에 표시하기 위해 가벼운 웹 서버인 serve 모듈을 npm을 통해 설치하기를 권장한다. 관련 패키지는 https://www.npmjs.com/package/serve에서 다운로드할 수 있다.

참고 자료

- 가장 기본적인 참고 자료는 https://github.com/angularjs-in-action 저장소다.
- 실제로 동작하는 안젤로 애플리케이션은 http://www.angelloinaction.com에서 살펴볼 수 있다.
- 그 외 여러 자료를 제공하는 '아직 배고픈 개발자(One Hungry Mind, http://onehungrymind.com)' 블로그 또한 참고해볼 만하다. 안젤로와 관련된 독자들의 피드백을 바탕으로 추가적인 내용들을 계속해서 포스팅할 예정이다.

저자와의 온라인 교류

이 책을 구매하면 매닝 출판사가 운영하는 사설 웹 포럼에 자유롭게 접근할 수 있다. 이 포럼에서는 책에 대한 의견을 공유할 수도 있고, 기술적인 질문을 올릴 수도 있으며, 저자 및 다른 사용자들로부터 도움을 얻을 수도 있다. 포럼을 이용하려면 www.manning.com/

AngularJSinAction 링크를 방문해보기 바란다. 저자와의 온라인 교류(Author Online) 페이지는 일단 회원 가입 후 포럼에 접근하는 방법과 이를 통해 받을 수 있는 지원, 그리고 포럼을 이용하기 위한 규칙 등을 설명하고 있다.

매닝 출판사가 이 책의 독자들에게 공약한 것은 독자들끼리, 그리고 독자와 저자가 의미 있는 대화를 나눌 수 있는 공간이다. 이 포럼은 저자들의 자발적인 (그리고 무보수) 참여로 운영되지만, 의무적으로 참여해야 하는 것은 아니다. 따라서 저자들의 관심을 끌 만한 도전적인 질문들을 올려주기를 권한다.

이 포럼과 이 포럼에 쌓여온 갖가지 논의들은 이 책이 계속 인쇄되는 동안 출판사의 웹사이트를 통해 지속적으로 접근할 수 있다.

저자 소개

루카스 루벨키(Lukas Ruebbelke)

루카스는 2001년에 플래시(Flash)와 함께 프로그래밍을 시작해 프로토타입 언어라고 말할 수 있는 액션스크립트 1.0으로 프로그래밍을 익혔다. 15년이 지난 지금은 자바스크립트로 완전히 전향했다.

미국 애리조나주의 피닉스에 거주하며, 이 지역에서 가장 큰 밋업(Meetup)을 운영하면서 커뮤니티에 기여하고 있다. 또한, 자신의 블로그(http://onehungrymind.com)를 열정적으로 운영하고 있고, ng-conf, ng-europe, ng-vegas 등 여러 콘퍼런스에서 강연도 하고 있다. 프로그래밍을 통해 삶을 변화시킬 수 있으리라는 확신을 가지고 있으며, 이 책 역시 그의 열정이 표출된 또 다른 결과물이다.

브라이언 포드(Brian Ford)

브라이언은 구글의 Angular 코어팀에서 근무하는 개발자다. 종종 자신을 '팀에서 가장 꼰대'라고 표현하기도 하는 그는 미시건 주립 대학에서 컴퓨터 공학을 전공하던 중 Angular 코어 개발에 참여하면서 커뮤니티에 기여하기 시작했다.

책 표지에 대하여

《AngularJS 인 액션》의 표지 그림은 '크로아티아 부코바에서 온 남자'다. 이 그림은 2003년 크로아티아의 Ethnographic Museum in Split에서 발행한 니콜라 아르세노빅(Nikola Arsenovic)의 19세기 중반 크로아티아 전통 의상 앨범을 재구성한 것이다. 이 그림은 중세 로마 시대의 중심가에 위치한 박물관의 사서에게서 얻은 것으로, 기원전 304년 경 디오클레티아누스(Diocletian) 황제가 은퇴해 머물렀던 궁전의 유적에서 발견되었다.

부코바는 크로아티아 동부의 중규모 도시다. 부카(Vuka) 강과 다뉴브(Danube) 강의 합류 지점에 있는, 크로아티아에서 가장 큰 운하가 있는 도시다. 부코바는 지리적 이점으로 인해 자연환경이 잘 보존되어 있으며, 수백 년간 오스트리아와 서방 국가를 이어주고 있다. 표지의 그림은 그가 일요일에 즐겨 입던 옷이다. 파란색의 울 바지와 검은색의 울 조끼를 흰색 리넨 셔츠 위에 착용하고 있으며, 헐렁한 망토를 두르고 있는 모습은 이 지역의 부유층이 주로 복잡하고 화려한 자수로 치장했다는 것을 보여준다.

옷을 입는 습관과 생활양식은 200년의 세월과 함께 바뀌었고 지역에 따라 다르기는 하지만, 그 시절의 부유층은 거의 사라졌다. 이제는 겨우 몇 마일 떨어진 다른 마을이나 도시는 고사하고, 서로 다른 대륙에 거주한다고 해서 떨어져 있다고 말하기는 어려운 시대가 됐다. 분명 우리는 문화적 다양성을 더욱 다양한 개인의 삶(확실히 더욱 다양하며 빠르게 변하는 기술 의존적 삶)을 영위하는 데 사용하고 있다.

매닝 출판사는 2세기 이전 지역 주민들의 다양한 삶의 방식을 투영한 표지를 독창적이고 진취적인 컴퓨터 비즈니스 도서에 사용함으로써 이런 작품과 고서로부터 얻은 일러스트에 다시 생명을 불어넣을 수 있게 된 것을 영광으로 생각한다.

베타리더 후기

고재도(KT)

'In Action'시리즈는 늘 그렇듯이 이번에도 저의 기대를 저버리지 않았습니다. '안젤로'라는 애플리케이션을 처음부터 끝까지 만들어가면서 AngularJS의 기본적인 개념부터 실제 활용법까지 다루고 있습니다. 한 가지 아쉬운 점이 있다면, 현재의 1.4 버전과 앞으로 나올 1.5 버전에 대한 내용이 없다는 점입니다. 하지만 본 도서를 통하여 AngularJS를 본 프로젝트에 바로 사용해보고 싶은 사람들의 욕구는 충분히 충족시켜 줄 것입니다.

김승환(네이미)

실용적인 예제 위주로 잘 설명되어 있어 AngularJS를 처음 접하는 분들이 쉽게 접근할 수 있을 것 같습니다. 'In Action' 시리즈가 입문서이긴 하지만, 조금 더 깊이 다뤘더라면 하는 아쉬움이 살짝 들었습니다.

김예리(링크잇)

'In Action'시리즈의 베타리더가 되어 개인적으로 영광이었고, 또한 베타리더가 되어 책을 읽으니 평소보다 더 꼼꼼히 읽게 되는 것 같아 좋았습니다. 초반에는 AngularJS의 큰 그림을 잡는데 도움이 되었고, 예제를 통해서는 AngularJS의 거의 모든 부분을 다루어 자세하고 깊게 공부할 수 있었습니다. 그동안 정리되지 않았던 부분이 한 번에 정리가 되는 느낌입니다. 예제가 정말 좋은데, 여러분도 다운로드하여 하나라도 놓치지 말고 분석해보세요. AngularJS의 고수가 되실 겁니다! 공부하는 여러분을 응원합니다!

김용균(이상한모임)

AngularJS를 사용해서 한 프로젝트를 작게 먼저 만들고 다시 크게 만드는 과정으로 진행하는데, 전체 코드를 제공하고 그중 중요한 부분을 짚어주는 방식으로 설명하고 있습니다. 코드를

따라가며 읽지 않으면 다소 복잡하게 느껴질 수 있지만, AngularJS에서 코드 간 관계를 어떻게 분리하고 계층을 나눠 개발하는지에 대한 전체적인 관점을 학습하는 데에는 큰 도움이 되었습니다.

김종욱(KAIST)

이 책은 하나의 서비스(안젤로라는 애플리케이션)를 개발하면서 AngularJS가 어떻게 구동되는지, 그리고 어떻게 사용해야 하는지를 잘 설명한 책입니다. 각 장을 단계별로 학습한다면 자신의 손으로 직접 만든 서비스(안젤로)를 만들 수 있습니다. 다만, 아쉬운 점이 있다면 일부 번역 투의 문장이 보였는데 출간 전에 바로잡아줄 것으로 기대합니다. 그리고 도입 부분에 간략하게나마 AngularJS에 대한 기본 형식과 관련 내용을 소개했다면, 초보자들이 큰 그림을 그려가며 학습하는 데 도움이 되었을 것 같습니다. 이에 반해, 독자 스스로가 단계별로 자신의 손으로 직접 서비스를 개발한다는 것이 굉장히 마음에 들었고, 다양한 기법과 흥미로운 예시들을 많이 제공하고 있어서 좋았습니다.

김지헌(honeymon.io)

이 책은 AngularJS의 핵심 컴포넌트를 큰 그림을 보여주며, 모듈별로 기능을 설명하고 있습니다. 그리고 '주의할 점', '테스트'와 '요약'과 같은 구성을 통해 친절히 설명해주고 있습니다. 'AngluarJS를 언젠가 한번 써봐야지' 하는 다짐만 하곤 했는데, 이 책을 살펴보면서 토이 프로젝트에 바로 적용해봐야겠습니다. 학습 비용이 좀 큰 편이기는 하지만, 여러 유무형의 비용을 고려하면 jQuery만으로 코딩했을 때보다 AngularJS를 통해 프레임워크가 강요하는 방식으로 개발하는 것이 적절하다는 생각을 해봅니다.

이철민(카카오)

밤하늘의 별만큼 늘어가고 있는 프론트엔드 기술과 프레임워크 중에서도 인기 있는 AngularJS에 관한 책입니다. 단지 인기가 있다는 이유로 AngularJS를 무작정 도입하여 사내 툴을 제작한 적이 있었는데, 이 책이 진작 나왔더라면 좀 더 깔끔한 구조로 만들 수 있지 않았을까 하는 생각을 해봅니다. 책에서 예제로 든 '안젤로'를 마스터한다면 기본적인 SPA는 충분히 제작할 수 있을 것입니다. 자바스크립트에 관한 기초 지식이 있고 앵귤러로 삽질해본 경험이 있는 분들에게 좋은 책일 것 같습니다.

이후평(승광)

자바스크립트 프레임워크라고는 jQuery밖에 사용해보지 않은 저에게 AngularJS는 기초적인 개념도 어려웠고, 웹 문서를 뒤져가며 필요한 기능을 붙여나가면서도 해결하기 어려운 문제에 계속 봉착했습니다. 때마침 읽게 된 《AngularJS 인 액션》은 체계적인 구조와 제대로 된 사용법을 통해 당장 잘못된 부분을 고칠 수 있게 도와주었고, 앞으로 구현해야 할 기능에 대한 가이드를 제공해주었습니다.

최아연

책 전체가 이해하기 쉬운 흐름으로 구성되어 있어서 막힘없이 베타리딩을 진행할 수 있었습니다. AngularJS를 빠르게 경험하며 배우려는 분들께 좋을 것 같아요!

PART

I

AngularJS와 친해지기

AngularJS의 세계에 동참하게 된 것을 환영한다. I부에서는 AngularJS에 대해 개략적으로 살펴보고 간단한(하지만 나름 유용한) 웹 애플리케이션을 개발해보면서 AngularJS를 소개하고자 한다.

제1장에서는 AngularJS의 핵심 개념들에 대해 설명하고 이 개념들이 정확히 어떤 역할을 수행하는지, 그리고 서로 어떻게 연결되어 동작하는지를 알아보고자 한다. 또한, 이 책의 예제 애플리케이션을 간략하게 재구성한 애플리케이션을 처음부터 구현해본다. 제2장에서는 알려진 모범 사례들을 바탕으로 유지보수 및 확장이 용이한 AngularJS 애플리케이션을 구성하는 방법을 알아본다.

I부를 마치면 AngularJS의 주요 부분에 대해 자세히 이해하게 될 것이며, 이것들의 구동 방식에 익숙해질 것이다. 예제 애플리케이션 개발을 잘 따라 해보면 AngularJS 세계로의 여행을 시작하기에 충분한 배경적 지식을 습득할 수 있다.

AngularJS는 굉장히 동적이면서도 빠르게 변화하는 프레임워크다. 그러므로 https://github.com/angularjs-in-action 저장소를 자주 참고해서 최신의 코드 예제들을 확인하기를 권한다. 또한, 첫 번째 프로젝트의 코드는 https://github.com/angularjs-in-action/angello-lite 저장소에서 살펴보기 바란다.

PART I

Get acquainted with AngularJS

CHAPTER

1

안녕하세요, AngularJS

이 장에서 학습할 내용

- AngularJS를 사용해야 하는 이유
- AngularJS가 가져다주는 편리함
- AngularJS를 개략적으로 이해하기
- 첫 번째 AngularJS 애플리케이션 개발하기

수년 전까지만 해도 웹 페이지에는 서버에 데이터를 전송하고 서버가 이를 처리한 후 완전히 새로운 웹 페이지를 렌더링하게 하는 식의 로직들이 존재했었다. 이른바 '호출 후 새로고침(call and refresh)' 방식으로 인해 사용자 경험은 단절되기 일쑤였고, 네트워크의 대기 시간이 길어질수록 좋지 않은 사용자 경험은 더욱 악화되었다.

그러나 XMLHttpRequest 객체의 등장과 페이지를 실제로 새로고침하지 않고도 서버에 비동기적으로 처리를 요청할 수 있는 기법이 소개되면서 상황은 반전되었다. 이 기법을 이용하면 사용자는 원격 호출에 의존해야 하는 작업을 처리하면서도 그 응답이 도착할 때까지 계속해서 애플리케이션을 사용할 수 있어서 보다 일률적인 사용자 경험을 제공할 수 있다. 바로 이 시점이 자바스크립트 프레임워크의 첫 번째 물결이 일어난 시점이다. 자바스크립트 프레임워크들의 등장은 많은 부분에서 개발자들의 수고를 덜어주었으며, 자바스크립트가 더 안정적으로 동작할 수 있는 기반이 되었다.

수많은 프레임워크들 중에서도 jQuery가 압도적인 지지를 받는다는 점에 대해서는 많은 개발자가 동의할 것이다. jQuery가 수많은 브라우저의 각기 다른 부분을 모조리 추상화하여 개발자들이 브라우저에 관계없이 웹사이트에 단일 API를 사용할 수 있게 해주었다는 사실 때문이다. 그 후로는 애플리케이션과 유사하게 동작하는 웹사이트를 개발할 수 있도록 구현된 프레임워크들이 등장했다. 이들은 완전히 새로운 방식을 도입했다. 예를 들어, jQuery는 DOM을 조작하는 데 필요한 탁월한 도구들을 제공하지만, 애플리케이션 구조에 맞게 코드를 정리하는 것에 대한 실질적인 가이드라인을 제공하지는 않았다. 'jQuery 애플리케이션'을 구현한 코드가 오히려 유지보수가 어렵고 확장성이 떨어지는 괴상한 코드로 변하고 말았다는 안타까운 소식을 듣게 되는 것은 바로 이 점 때문이다.

이후 유지보수가 쉬운 대규모 자바스크립트 애플리케이션 개발에 대한 수요가 증가하면서 자바스크립트 프레임워크는 전성시대를 맞이한다. 최근 2~3년 사이에는 수많은 프레임워크가 등장하고 또 아무도 모르는 사이에 사라져 갔다. 그러나 몇몇 프레임워크는 유지보수가 쉽고 확장 및 테스트가 수월한 대규모 웹 애플리케이션을 개발하기 위한 필수 옵션으로 자리매김했다. 그중에서도 첫손가락에 꼽히지는 않지만 매우 대중적으로 알려진 프레임워크가 바로 구글이 개발한 AngularJS다.

AngularJS는 오픈 소스 웹 애플리케이션 프레임워크로, 개발자들에게 안정적인 기반 코드와 활발한 커뮤니티, 그리고 풍부한 생태계를 제공한다. 자세한 기술적 세부사항을 들여다보기에 앞서 AngularJS의 장점들을 먼저 살펴보기로 하자.

1.1 AngularJS의 장점

이번 절에서는 AngularJS의 여러 장점에 대해 간략하게 훑어본다.

코드를 조직적으로 관리할 수 있는 직관적인 프레임워크

앞에서 설명했듯이 유지보수와 협업, 가독성, 그리고 확장성을 고려해서 코드를 조직화하는 방법에 관한 수요가 엄청나게 존재한다. AngularJS는 코드를 직관적인 장소에 배치하고, 필요한 경우 코드를 리팩토링할 수 있는 명확한 방법을 제시할 수 있도록 구성되어 있다. 사용자 인터페이스의 외관과 행동을 정의하기 위해 해당 정보를 제공하는 코드를 작성해야 할 필요가 있다면, AngularJS는 이미 그런 코드들을 보관할 위치를 정해두고 있다. 애플리케이션이 사용해야 하는 도메인 모델을 서버로부터 가져와야 한다면, 물론 이런 코드들을 위한 위치도 정해

져 있다. 코드를 이용해서 DOM을 조작해야 하는 경우를 대비해 이런 코드를 위한 위치 역시도 명확하게 정의되어 있다!

개발자의 숙면을 보장하는 테스트 가능한 코드

테스트 가능한 코드는 프레임워크가 제공하는 여러 흥미진진한 기능들에 비하면 그다지 중요하게 인식되지 못하는 부분일 수도 있다. 그러나 이는 성숙기에 접어든 모든 프레임워크의 숨은 공신이다. AngularJS는 처음부터 테스트 가능성을 염두에 두고 개발되었으며, 디자인에 관한 의사 결정 역시 테스트를 고려하고 있으므로 테스트는 AngularJS 내에서 어마어마한 영향을 발휘한다. 우리가 개발한 애플리케이션이 실제로 동작하리라는 것을 알 수 있을까? 이 질문에 대해 애플리케이션이 아직 오동작을 한 적이 없다고 답한다면 그것은 깊이 있는 답변이라 할 수 없다. 이슈가 발견되는 것은 시간 문제이기 때문이다.

버그를 완전히 없애지는 못하겠지만, 엄격한 테스트를 통해 버그의 발생 가능성을 상당히 배제할 수는 있다. 테스트가 가능한 코드의 작성을 고려한 프레임워크라면 그 자체가 바로 테스트 코드를 작성하기 위한 프레임워크다. 또한, 테스트 코드를 작성하면 어딘가에 문제가 발생했을 때 그 원인을 상대적으로 빨리 규명할 수 있게 된다. 밤이 되면 혹시나 새벽 2시쯤에 문제가 생겼으니 지금 당장 고쳐내라는 데브옵스(DevOps)팀의 전화를 받게 되지는 않을까 하는 걱정은 접어두고 마음 편히 잠자리에 들 수 있다.

엄청난 코드를 절약할 수 있는 양방향 데이터 바인딩

양방향 데이터 바인딩은 AngularJS가 제공하는 여러 기능 중에서도 최고의 기능이다. 백만 년 전, 우리가 jQuery 애플리케이션을 작성하던 시절에는 jQuery를 이용해서 DOM에서 특정 요소를 찾아내 이벤트를 리스닝하고, DOM 요소의 값을 파싱해서 이 값을 가지고 필요한 작업을 수행하곤 했다. AngularJS를 이용하면 단순히 자바스크립트 속성을 정의하고 이것을 HTML에 바인딩하면 된다. 그것으로 끝이다. 당연히 이 과정에서 고려해야 할 다양한 시나리오가 있겠지만, jQuery 애플리케이션을 새로 작성했더니 전체 코드의 양이 확연히 줄어들었다는 소식을 들어본 경우는 거의 없었다.

앞서 언급했던 HTML과 자바스크립트의 동기화를 위한 코드들을 모조리 없앨 수 있다면, 더 짧은 시간 내에 더 적은 노력으로 훨씬 많은 일을 할 수 있다. 내가 즐겁게 할 수 있는 일을 할 시간이 더 많아진다는 의미다.

HTML 코드를 대신하는 템플릿

HTML은 본질적으로 간편한 레이아웃과 구조를 위해 디자인된 제한적인 언어일 뿐 복잡한 상호작용에 적합한 언어는 아니다. 다시 말하면, 요즘 우리가 알고 있는 현대적인 웹 애플리케이션의 세상을 위해 만들어진 언어는 아니라는 뜻이다. 일부 프레임워크들은 HTML을 문자열로 완전히 추상화하거나 또는 전처리기 등을 도입하여 이 한계를 극복하려는 시도를 해 왔다. 그러나 문제는 선언적 메커니즘으로서의 HTML은 상당히 괜찮은 언어이며 바로 이 점이 HTML의 짜증나는 부분이라는 점이다. 아마도 대부분의 사용자가 느끼고 있을 것이다.

대규모 팀에 소속되어 있다면 HTML 템플릿을 전담해서 만들어주는 UI/UX 담당자가 있을 것이다. 이 경우 그들이 이미 익숙한 기술과 절차를 계속해서 사용할 수 있도록 하는 것이 중요하며, AngularJS 역시 이 부분에 대해서도 잘 배려하고 있다. 왜냐하면 AngularJs는 HTML을 그대로 끌어안으면서도 필요하다면 HTML의 한계를 극복할 수 있는 능력을 개발자에게 제공하고 있기 때문이다.

자바스크립트와의 손쉬운 통합을 위한 데이터 구조

반대로 생각해보면, POJO(Plain Old JavaScript Objects) 객체들을 다른 기술들과 통합하는 것은 말도 안 되게 쉽다. 프레임워크가 제공하는 적절한 메커니즘을 사용하여 별도의 변환 작업 없이 자바스크립트 객체를 사용하거나 내보낼 수 있다면, 다른 데이터 원본으로부터 데이터를 소비하는 과정 자체가 더 효율적이 될 수 있다.

즉, 서버에서 JSON 모델을 렌더링하고 AngularJS가 애플리케이션을 시작하는 시점에 해당 객체를 곧바로 사용할 수 있다. 또한, 지금 처리 중인 모델을 특별한 변환 과정을 거치지 않고 다른 기술(예를 들면, 애플리케이션 서버)에게 전달할 수도 있다.

AngularJS가 제공하는 기능 중에는 본질적으로 학술 분야에 가까운 기능이 일부 존재한다. 필자들은 실용적인 관점에서 AngularJS가 개발자에게 편의성을 제공하기 위해 구현된 몇 가지 부분에 대해 개략적으로나마 설명하려고 노력했다. 궁극적으로, 안정적인 코드를 빠르고 효과적으로 작성할 수 있어서 더 많은 시간과 에너지를 더욱 의미 있는 다른 일에 쏟을 수 있도록 도와주는 프레임워크가 바로 우리가 원하는 프레임워크다.

1.2 AngularJS의 큰 그림 살펴보기

지금부터 AngularJS를 아주 간략하게 살펴볼 예정이다. 그리고 이 책 전체에 걸쳐 필자들이 소개하고자 하는 내용에 대한 기본적인 개념들을 설명하고자 한다(표 1.1 참조). 이 책을 끝까지 읽고 나면 그림 1.1에 대해 완벽하게 이해할 수 있을 것이며, AngularJS의 각 개념이 어떻게 상호간에 연동되는지를 이해하게 될 것이다. 나아가, 각각의 개념에 내포된 의미를 정확하게 깨닫는다면 문제를 해결하기 위한 명확하고 빠른 길을 발견하게 될 것이다.

표 1.1 AngularJS의 핵심 컴포넌트들

컴포넌트	목표
모듈 (Module)	모듈은 AngularJS 애플리케이션을 구성하는 코드를 정리하는 데 도움을 주는, 일종의 컨테이너(container)로서의 역할을 수행한다. 또한, 모듈은 하위 모듈(sub module)을 활용해서 필요한 기능을 쉽게 구성할 수 있다.
설정 (Config)	AngularJS 애플리케이션의 설정 코드는 애플리케이션이 실제로 실행되기 전에 적용될 설정 정보들을 관리하기 위한 영역이다. 라우트를 설정하거나 동적으로 구성되는 서비스 등을 여기에서 관리하면 편리하다.
라우트 (Routes)	라우트는 애플리케이션의 특정 상태로 이동하는 경로를 정의하기 위한 개념이다. 게다가 특정 라우트에 대해 어떤 템플릿과 컨트롤러를 사용할 것인지를 설정할 수도 있다.
뷰 (Views)	AngularJS에서 뷰는 AngularJS가 관련된 모든 자바스크립트와 함께 DOM을 컴파일하고 렌더링한 이후에 생성된다.
$scope	$scope 객체는 기본적으로 AngularJS 애플리케이션 내부에서 뷰와 컨트롤러를 결합하는 객체다. controller-as 문법이 등장하면서 $scope 객체를 명시적으로 사용하는 경우가 많이 줄어들었다.
컨트롤러 (Controller)	컨트롤러는 뷰가 바인딩하고 다룰 수 있는 속성과 메서드를 정의하기 위한 객체다. 통상 컨트롤러는 스스로가 제어하는 뷰에만 집중할 수 있도록 최대한 가볍게 작성하는 것이 좋다.
디렉티브 (Directive)	디렉티브는 AngularJS의 뷰에 대한 확장 기능이다. 디렉티브를 이용해 원하는 동작을 캡슐화한, 재사용 가능한 사용자 정의 요소들을 구현할 수 있다. 즉, HTML을 위한 컴포넌트 혹은 데코레이터(decorator)라고 생각하면 된다. 디렉티브는 뷰를 확장하며 그 기능을 여러 곳에서 사용하는 경우에 활용하면 좋다.
서비스 (Service)	서비스는 AngularJS 애플리케이션의 공통 기능을 구현하기 위한 컴포넌트다. 예를 들어, 여러 컨트롤러가 공유해야 하는 데이터가 필요하다면 이 데이터를 서비스 객체로 구현하고 컨트롤러들은 이 서비스 객체를 통해 데이터에 액세스할 수 있게 구현하면 된다. 서비스는 컨트롤러를 확장하며 전역적으로 액세스가 가능하다.

AngularJS가 제공하는 컴포넌트들은 앞으로 다른 장을 통해 차례대로 더 깊이 살펴보겠지만, 그에 앞서 애플리케이션을 개발할 수 있는 기초를 다지기 위한 개념들을 먼저 살펴보기로 하자.

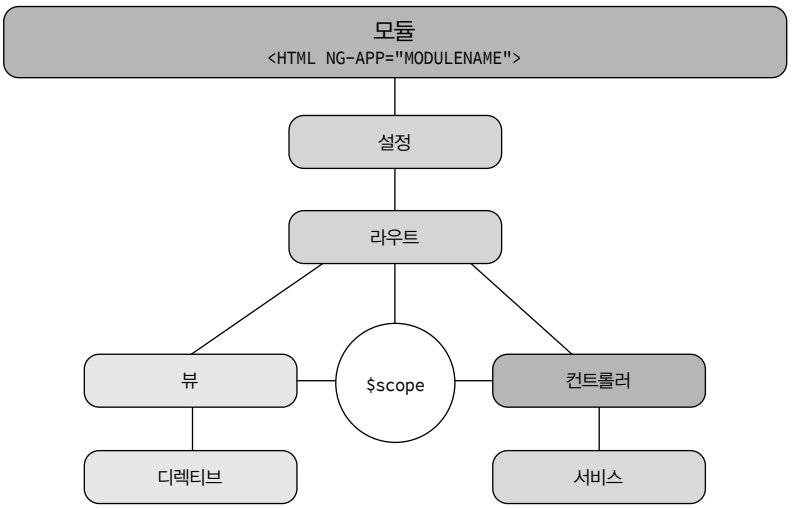

그림 1.1 **AngularJS의 큰 그림**

1.3 첫 번째 AngularJS 애플리케이션

앞에서 우리는 AngularJS 애플리케이션을 구성하는 각 컴포넌트를 살펴보았다. 하지만 이 컴포넌트들을 언제 어떻게 활용해야 하는 걸까? 이해를 돕기 위해 이번에는 AngularJS를 이용해서 직접 애플리케이션을 만들어보도록 하자. 이 책의 예제 애플리케이션의 축소판을 구현해보면 그 과정에서 AngularJS의 각 컴포넌트를 어떻게 결합하여 사용하는지를 파악할 수 있게 된다. AngularJS가 실제로 동작하는 모습을 직접 보면서 학습하게 될 것이고, 이런 예제들을 한데 모아 보다 큰 규모의, 그리고 완벽하게 동작하는 애플리케이션을 구현하게 될 것이다.

이 책의 예제 애플리케이션은 안젤로(Angello)라는 애플리케이션으로, 사용자 스토리(user story)를 관리하는 트렐로(Trello)라는 애플리케이션의 클론(clone)이다. 트렐로의 클론이라는 것이 무슨 의미일까? 일부 독자들은 이미 알고 있겠지만, 트렐로는 웹 기반의 프로젝트 관리 도구이며 일본의 자동차 회사인 토요타(Toyota)가 1980년대에 발표했던 기술을 근간으로 하고 있다. 프로젝트에서 작업의 단위(unit of work)는 아이템(또는 스토리)이라는 단위로 이루어지며, 각 스토리는 진행 상태에 따라 보드 상에 각기 다른 위치에 나열하게 된다. 따라서 보드 자체가 프로젝트를 표현하게 되는 셈이다. 안젤로에 대해서는 다음 장에서 조금 더 자세히 살펴보겠지만, 안젤로의 메인 화면은 그림 1.2에서, 그리고 축소 버전의 화면은 그림 1.3에서 확인할 수 있다. 안젤로 축소 버전의 전체 소스 코드는 https://github.com/angularjs-in-action/angello-lite

에서 다운로드할 수 있다. 최신 버전의 코드를 독자 여러분의 로컬 머신에 다운로드하기 위한 과정은 이 페이지의 README.md 파일에 작성해두었으니 참고하기 바란다.

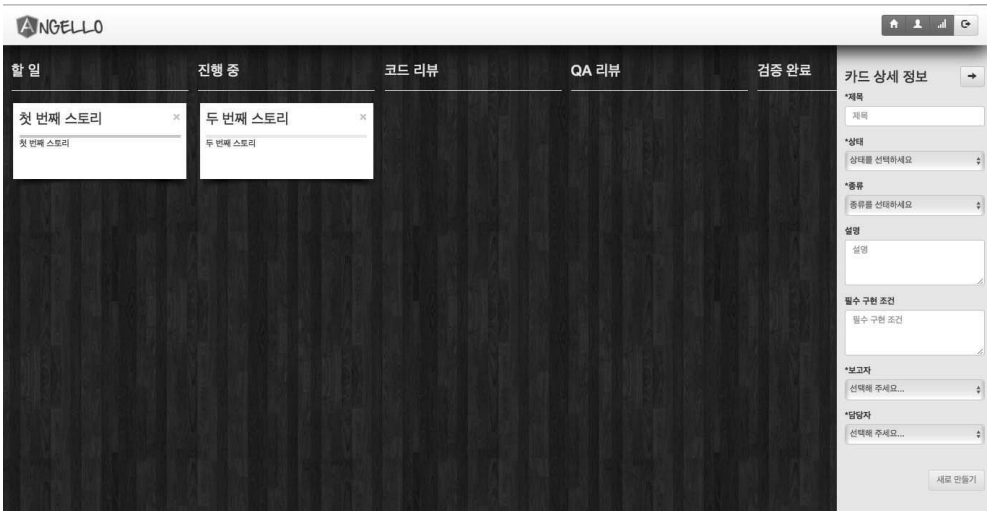

그림 1.2 **안젤로가 동작하는 모습**

이 책을 읽어나가는 동안 우리는 그림 1.2에서 살펴본 안젤로라는 애플리케이션을 개발하게 된다. 화면의 왼쪽에는 해야 할 작업을 표현하는 흰색 카드에 각각 첫 번째 스토리, 두 번째 스토리라고 표시되어 있다. 그리고 작업의 진척 상황은 화면의 왼쪽부터 해야 할 일, 진행 중, 코드 리뷰, QA 리뷰, 그리고 검증 완료 등의 칼럼으로 나누어져 있다. 작업을 진행하는 동안 각 카드는 드래그 앤 드롭(drag-and-drop)으로 해당 작업의 진행 상태를 표시하는 칼럼으로 이동하게 된다. 각 작업 아이템과 스토리의 상세 내용은 화면의 오른쪽에 나타난다. 이미 짐작했겠지만, 안젤로의 각 스토리는 프로젝트를 시작한 후 종료할 때까지 구현해야 할 컴퓨터 소프트웨어의 단위를 표현한다. 우선은 그림 1.3과 같은 축소 버전의 개발을 시작해보자.

그림 1.3 **안젤로 Lite가 동작하는 모습**

안젤로 Lite 이 애플리케이션은 필요한 파일들을 CDN[1]에서 다운로드하기 때문에 반드시 웹 서버를 통해 실행되어야 한다. 코드를 웹 서버를 통해 실행하는 방법은 몇 가지가 있지만, 그중에서도 가장 손쉬운 방법은 npm 패키지인 serve를 사용하는 방법이다.

안젤로 Lite 버전을 설치하는 순서는 다음과 같다.

- Node.js를 설치한다. 설치와 관련된 자세한 내용은 http://nodejs.org를 참고하기 바란다.
- 명령 줄에서 npm install -g serve 명령을 실행해서 serve 패키지를 설치한다.[2]
- 앞에서 소개한 Github URL에서 안젤로 Lite 버전을 다운로드한 후, angello-lite라는 디렉터리에 저장한다.
- angello-lite 디렉터리로 이동해서 serve 명령을 실행한다.
- 웹 브라우저를 실행하고 http://localhost:3000을 방문하면 애플리케이션이 실행된 모습을 볼 수 있다.

안젤로 Lite는 제2장부터 개발할 안젤로를 간소화한 버전이다. 이 애플리케이션에 저장하는 모

1 역주 Content Delivery Network의 약자로, 원거리의 클라이언트에게 콘텐츠를 빠르게 전송하기 위한 기술이다.

2 역주 OS X 또는 리눅스 사용자들은 sudo npm install -g serve와 같이 슈퍼유저 권한으로 해당 명령을 실행해야 한다.

든 데이터는 영구히 저장되는 것이 아니라 메모리에만 저장된다. 따라서 페이지를 새로 고치면 모든 데이터를 잃게 된다. 스토리의 상세 정보를 보려면 화면 왼쪽에서 제목과 설명이 표시된 카드를 클릭하면 된다. 그러면 상세 정보가 화면 오른쪽에 나타난다. 텍스트 상자와 드롭다운 목록을 이용해서 스토리를 변경하고 저장하면, 브라우저에서 페이지를 새로고침하기 전까지는 데이터가 유지된다. 새로운 스토리를 생성하려면 왼쪽의 덧셈 기호를 클릭하면 된다. 그러면 새로운 제목과 설명을 입력하는 대화 상자가 나타난다. 다른 데이터들과 함께 카드의 제목과 설명을 입력하면 그 내용이 실시간으로 요약 상자에 표시된다.

그림 1.4는 앞서 살펴봤던 큰 그림과 관련해서 이번에 우리가 구현하게 될 부분이 어느 컴포넌트에 속하는지를 보여준다. 우선은 모듈을 먼저 구성한 다음, 뷰와 컨트롤러를 각각 index.html 파일과 MainCtrl 컨트롤러에 구현할 예정이다. 그 다음에는 AngelloModel 서비스와 story 디렉티브를 구현한다.

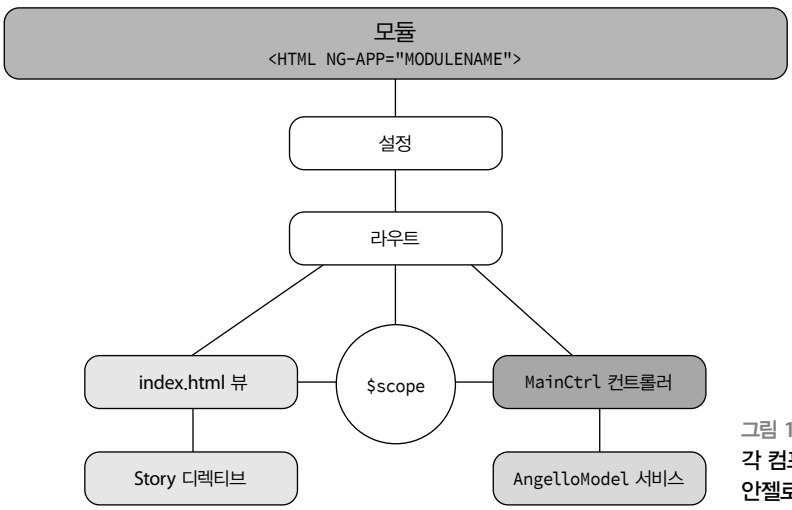

그림 1.4 AngularJS의 각 컴포넌트에 대응하는 안젤로 모듈들

이번 장에서 안젤로 Lite 애플리케이션의 소스 코드를 줄 단위로 설명하지는 않겠지만, 중요한 부분들은 모두 설명할 것이기 때문에 무엇이 어떻게 동작하는지에 대해서는 확실하게 이해하게 될 것이다. 이번 장을 마칠 즈음이면 AngularJS 저녁 모임에 나가서도 아는 척 좀 할 수 있는 수준이 될 수 있다.

안젤로 Lite 애플리케이션을 구현할 때 유념할 점은 이 애플리케이션이 마스터-디테일 인터페이스(master-detail interface)를 채택하고 있다는 점이다. 그리고 이는 거의 모든 단일 페이지 웹

애플리케이션(single page web application)이 한두 개의 페이지로 구성된다는 점과 일맥상통한다. 마스터-디테일 인터페이스를 한 페이지에서 구현하는 방법을 이해하는 것은 웹 페이지를 구현하는 일반적인 방법을 배우기 위한 좋은 토대가 된다.

1.3.1 모듈

AngularJS에서 **모듈**(module)의 역할은 애플리케이션을 논리적인 단위로 나누어 조직화하는 것이다. AngularJS는 모듈을 통해서 애플리케이션이 어떻게 설정되어 있으며, 어떻게 동작해야 할 것인지를 파악한다. 전체 큰 그림에서 모듈의 위치는 그림 1.5에서 확인할 수 있다.

코드를 살펴보면 Angello라는 이름의 모듈의 인스턴스를 생성해 myModule이라는 이름의 변수에 대입하는 것을 볼 수 있다.

```
// app.js
var myModule = angular.module('Angello', []);
```

두 번째 매개변수는 필요한 경우 추가적인 기능을 제공하기 위한 서브 모듈들을 전달하기 위한 배열이다. 일반적으로 구현해야 할 기능들을 서브 모듈에 나누어 구현한 후, 애플리케이션의 주 모듈에 삽입(inject)하는 것이 적절한 구현 방법으로 받아들여지고 있다. 그렇게 함으로써 모듈을 이동하는 것은 물론 테스트하기가 쉬워지는 장점이 있다.

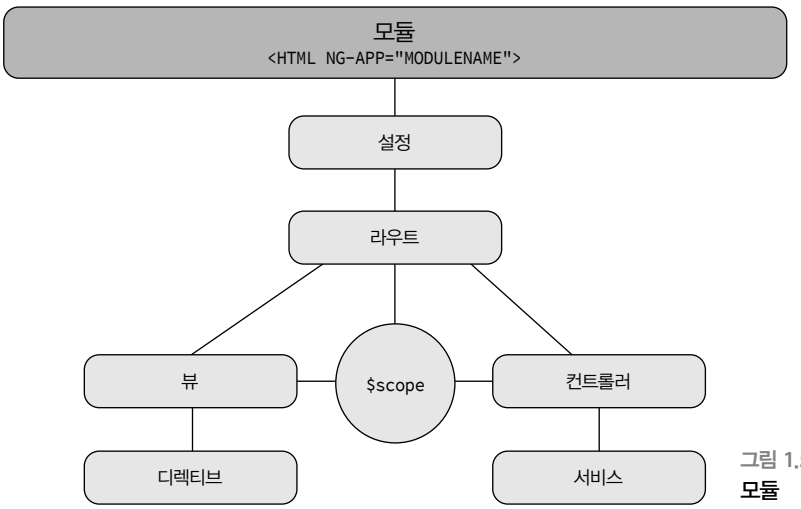

그림 1.5 **AngularJS의 모듈**

이제 안젤로 Lite 애플리케이션의 myModule 속성에 필요한 컴포넌트들을 정의할 수 있다. 예를 들어, 예제에서는 AngelloHelper와 AngelloModel이라는 두 가지 서비스를 정의한다. 그리고 MainCtrl이라는 이름의 컨트롤러와 story라는 이름의 디렉티브도 구현한다.

```
// app.js
var myModule = angular.module('Angello', []);
myModule.factory('AngelloHelper', function() { });
myModule.service('AngelloModel', function() { });
myModule.controller('MainCtrl', function() { });
myModule.directive('story', function() { });
```

Angello 모듈을 정의하고 필요한 컴포넌트들의 인스턴스를 모두 구성했으면, 이제는 Angello 모듈을 AngularJS 애플리케이션의 진입점으로 지정하여 애플리케이션을 작동시킬 차례다. AngularJS 애플리케이션을 시작하는 가장 쉬운 방법은 AngularJS 애플리케이션을 동작하게 할 HTML 요소에 ng-app 특성을 추가하는 방법이다. 예제에서는 애플리케이션이 전체 페이지를 관리하기 때문에 html 태그에 ng-app="Angello"라는 특성을 추가한다. 이렇게 해서 Angello 모듈을 통해 AngularJS 애플리케이션이 시작하게 된다.

```
<!-- index.html -->
<html ng-app="Angello">
```

그러면 지금부터 간략한 설명과 함께 나머지 컴포넌트들에 살을 붙여 나가자.

1.3.2 뷰와 컨트롤러

AngularJS를 배울 때 가장 이해하기 어려운 부분 중 하나는 DOM과 상태(state)를 분리하는 것이다. AngularJS는 공식적으로는 모델-뷰-아무거나(MVW, Model-View-Whatever) 프레임워크다. 여기서 아무거나(whatever)의 의미는 가장 생산적으로 코드를 구현할 수 있는 패턴이라면 무엇이든 적용해도 된다는 의미다. 편의를 위해 AngularJS가 그림 1.7과 같이 모델-뷰-뷰모델(MVVM, Model-View-ViewModel) 디자인 패턴을 따른다고 생각하자.

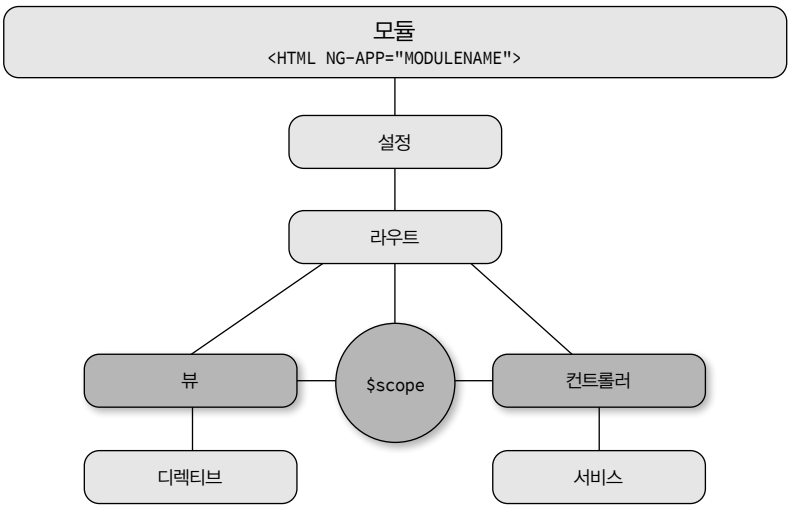

그림 1.6 **뷰와 컨트롤러**

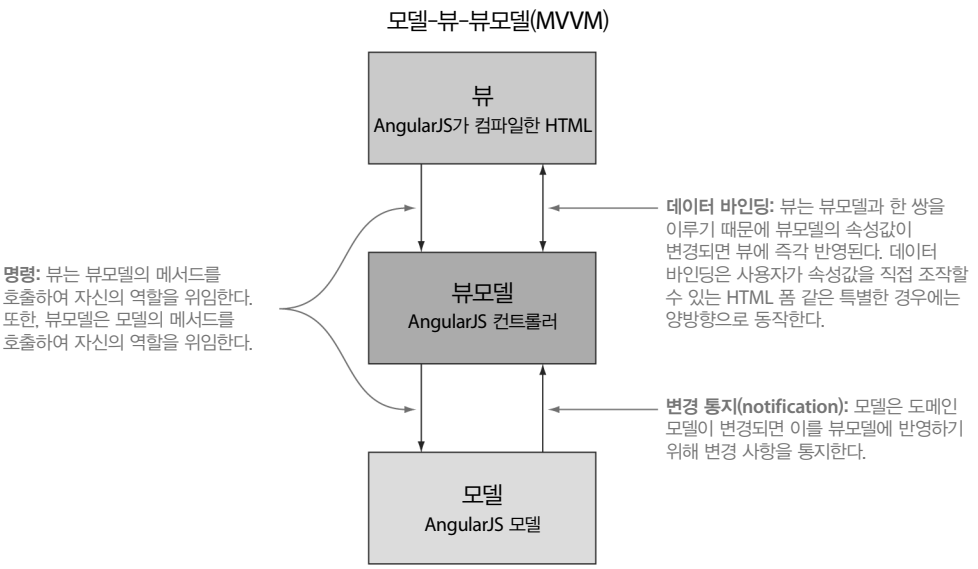

그림 1.7 **모델-뷰-뷰모델**

모델에 대한 내용은 서비스 관련 절에서 살펴보고 지금은 뷰와 뷰모델에 조금 더 집중해보자. 그림 1.7에서 보듯이, MVVM 패턴에서 뷰는 (본질적으로) AngularJS에서의 뷰이며 컨트롤러는 뷰모델의 역할을 담당한다.

컨트롤러는 뷰에 바인딩될 상태를 제공하며, 뷰에서 발생한 명령을 컨트롤러에 전달하여 필요

한 작업을 수행할 수 있게 한다. 이로 인해 뷰가 자신의 상태를 유지할 필요가 없으며(컨트롤러가 제공하는 상태를 표시하는 역할만을 담당하기 때문이다), 뷰가 다른 동작을 수행할 필요도 없게 된다(필요한 동작은 컨트롤러에 위임하기 때문이다).

이 과정이 어떻게 이루어지는지 살펴보기 위해 먼저 DOM에 ng-controller 디렉티브를 추가해서 MainCtrl 컨트롤러의 인스턴스를 생성하자. 예제에서는 MainCtrl as main과 같이 controller-as 구문을 사용한다. 이렇게 하면 HTML 파일 내부에서 MainCtrl이라는 이름의 컨트롤러를 main이라는 이름으로 사용할 수 있게 된다.

```
<!--index.html-->
<div ng-controller="MainCtrl as main">
</div>
```

뷰 내부에서 속성을 바인딩하려면 필요한 속성을 그저 컨트롤러에 선언만 하면 된다. 예를 들어, MainCtrl 컨트롤러에 this.title과 같이 속성을 선언하면 뷰 내에서는 <h1>{{main.title}}</h1>과 같이 중복된 중괄호를 이용해서 해당 속성을 바인딩할 수 있다. 그러면 title 속성의 값이 변경될 때마다 변경 사항이 DOM에 즉각 반영된다. 단순한 문자열 속성을 바인딩하는 것은 정말 쉬우므로 이보다는 조금 더 어려운 것을 해보기 위해 실제 컬렉션에 바인딩하는 과정을 살펴보자. 우선, 여러 개의 story 객체를 저장한 배열을 만든 후 이 배열을 MainCtrl 컨트롤러의 stories 속성으로 정의한다.

```
// app.js
myModule.controller('MainCtrl', function() {
    var main = this;

    //...
    main.stories = [
        {
            title: '첫 번째 스토리',
            description: '첫 번째 사용자 스토리',
            criteria: '요구사항 정리 중...',
            status: '해야할 일',
            type: '기능',
            reporter: '웹지니',
            assignee: '웹지니'
        },
        {
            title: '두 번째 스토리',
            description: '두 번째 사용자 스토리',
            criteria: '요구사항 정리 중...',
```

```
            status: '백로그',
            type: '기능',
            reporter: '웹지니',
            assignee: '웹지니'
        },
        {
            title: '세 번째 스토리',
            description: '세 번째 사용자 스토리',
            criteria: '요구사항 정리 중...',
            status: '코드 리뷰',
            type: '개선',
            reporter: '웹지니',
            assignee: '웹지니'
        }
    ];
    // ...
});
```

THIS 필자들은 보통 나중에 다시 참조할 필요가 있는 객체는 최상위 객체인 this 객체에 저장하는 것을 선호한다. this 객체는 함수 수준의 범위를 토대로 문맥이 변화하는 특징이 있다. 그리고 this 객체에 추가하는 참조 속성의 이름은 우리가 뷰에서 controller-as 문법을 사용해서 정의한 이름(예제의 경우는 MainCtrl as main 구문을 통해 main이라고 정의한 이름)을 사용하는 것을 선호한다. 이렇게 하면 코드 읽기가 쉽고 HTML과 자바스크립트 양쪽에서 모두 같은 이름을 사용할 수 있다.

우리는 main.stories 컬렉션을 마스터-디테일 뷰에서 마스터에 해당하는 영역에 목록 형태로 표시할 것이다. 그러기 위해서는 우선 main.stories 배열에 저장된 아이템들을 표시할 개별적인 요소들을 생성해야 한다. 이런 경우, ng-repeat 디렉티브를 이용하면 main.stories 배열의 각 아이템에 순서대로 접근하여 각 아이템을 토대로 ng-repeat 디렉티브를 지정한 HTML 요소와 그 자식 요소들의 복사본을 만들어낼 수 있다. callout이라는 클래스가 지정된 div 태그에 ng-repeat="story in main.stories"와 같이 특성을 지정하면, AngularJS가 main.stories 배열을 대상으로 루프를 실행해서 배열 내의 각 아이템을 story라는 이름으로 참조할 수 있게 해준다. 이를 이용해 다음과 같이 해당 객체를 자식 요소들에 바인딩할 수 있다.

```
<!-- index.html -->
<div ng-controller="MainCtrl as main">
    <div class="col-md-4">
        <h2>Stories</h2>
        <div class="callout"
             ng-repeat="story in main.stories"
```

```
            ng-click="main.setCurrentStory(story)">
            <h4>{{story.title}}</h4>
            <p>{{story.description}}</p>
        </div>
    </div>
</div>
```

각각의 story 객체는 title과 description 속성을 가지고 있으며, 이 속성들은 각각 {{story.title}} 및 {{story.description}}과 같은 표현식을 이용해 바인딩할 수 있다. AngularJS는 각 템플릿 인스턴스에 문맥을 제공하는 능력이 탁월하기 때문에 story 인스턴스가 잘못된 객체에 대한 참조를 가지지는 않을지 염려하지 않아도 된다. 이런 능력은 특히 ng-click="main.setCurrentStory(story)"와 같이 함수를 호출하면서 객체를 매개변수로 전달할 때 올바른 객체가 전달되도록 보장하기 때문에 매우 중요하다.

지금까지 속성을 바인딩하는 방법은 물론 표현식을 이용해 바인딩을 수행하는 방법을 알아보았다. 또한, 컨트롤러에 메서드를 정의하고 뷰에서 이 메서드를 호출하는 방법도 알게 되었다. 예를 들어, main.stories 배열에 새로운 story 객체를 추가하는 main.createStory라는 메서드를 컨트롤러에 다음과 같이 선언할 수 있다.

```
// app.js
myModule.controller('MainCtrl', function() {
    var main = this;

    //...
    main.createStory = function() {
        main.stories.push({
            title: '새 사용자 스토리',
            description: '설명을 입력하세요.',
            criteria: '요구사항 정리 중...',
            status: '백로그',
            type: '기능',
            reporter: '미정',
            assignee: '미정'
        });
   };
   //...
});
```

createStory 메서드를 MainCtrl 컨트롤러에 선언했으므로 이제 뷰에서 호출이 가능하다. 뷰의 앵커(anchor) 태그에 다음과 같이 ng-click 디렉티브를 지정해서 main.createStory 메서드를 호출할 수 있다.

```
<!-- index.html -->
<div ng-controller="MainCtrl as main">
    <div class="col-md-4">
        <h2>스토리 목록</h2>
        <div class="callout"
            ng-repeat="story in main.stories"
            ng-click="main.setCurrentStory(story)">
        </div>
        <div>
          <a class="btn btn-primary" ng-click="main.createStory()">
              <span class="glyphicon glyphicon-plus"></span>
          </a>
        </div>
    </div>
</div>
```

뷰모델을 이용하는 방법은 jQuery로 구현된 전통적인 애플리케이션에 존재하던 흐름을 완전히 뒤집는다. jQuery를 이용하는 경우라면 DOM을 먼저 조회한 후 이벤트 리스너를 바인딩했어야 했다. 그리고 그 이벤트가 발생하면 이벤트를 해석하고 현재 상태를 알아내기 위해 DOM을 파싱한 후에라야 정작 필요한 작업을 수행할 수 있었다. 이로 인해 HTML과 자바스크립트가 서로 너무 강하게 연결된다. 그러나 뷰모델을 사용함으로써 이 둘 사이의 관계를 분리할 수가 있다. 이제는 컨트롤러가 뷰의 변경 사항을 감지해내는 것이 아니라 뷰가 컨트롤러에게 특정 작업을 수행하도록 명령을 내릴 수 있게 된 것이다.

MVVM MVVM 패턴에 대한 심도 있는 논의는 이 책의 범위를 벗어난다. 그러나 필자들은 위키피디아의 문서(http://en.wikipedia.org/wiki/Model_View_ViewModel)를 한 번쯤 읽어볼 것을 권한다. 선언적인 마크업과 주요 로직을 분리함으로써 더 안정적이면서도 테스트가 쉬운 코드를 작성할 수 있게 된다.

1.3.3 서비스

컨트롤러는 가볍게 유지되어야 하고 특정 뷰만을 처리해야 하는데, 두 개의 컨트롤러가 동일한 정보를 공유해야 한다면 이 문제를 어떻게 해결할 수 있을까? 당연히 두 컨트롤러는 서로의 존재에 대해 인지하고 있어서는 안 된다. 그렇다면 어느 한 컨트롤러가 다루던 정보가 사실은 다른 컨트롤러도 필요로 하는 정보였다면 어떻게 이 정보를 공유할 수 있을까? 이에 대한 답변은 AngularJS 서비스(service)다. 컨트롤러로부터 공통의 데이터를 추출하고 이를 서비스를 통해 전체 애플리케이션에 노출할 수 있다. 그림 1.8은 MVVM 패턴에서 모델에 해당하는 AngularJS의 컴포넌트를 보여준다.

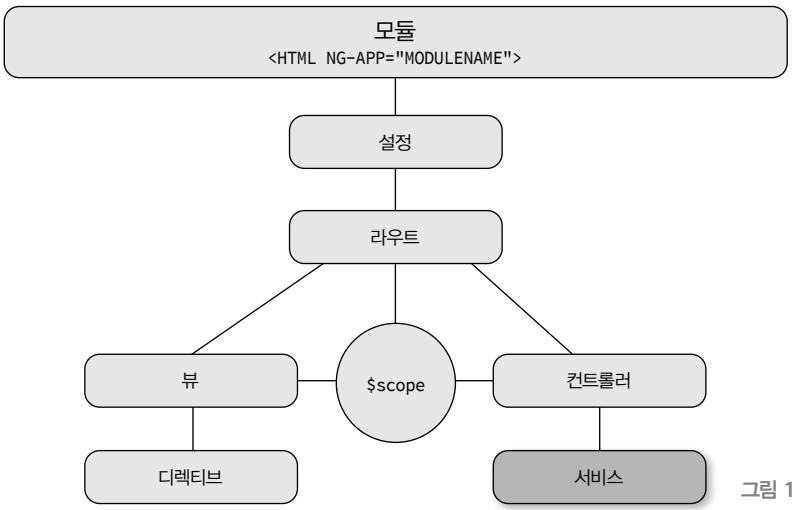

그림 1.8 **서비스**

이전 절에서 우리는 stories 컬렉션을 MainCtrl 컨트롤러에서 직접 조작했었다. 이번에는 이 컬렉션을 AngelloModel 서비스를 통해 노출하고 MainCtrl 컨트롤러가 이 정보를 활용하도록 구현해보자. 이를 위해 다음과 같이 AngelloModel 객체에 stories 속성을 선언하고 MainCtrl 컨트롤러에서 사용했던 것과 동일한 컬렉션을 구성하자.

```
// app.js
myModule.service('AngelloModel', function() {
    var service = this,
        stories = [
            {
                title: '첫 번째 스토리',
                description: '첫 번째 사용자 스토리',
                criteria: '요구사항 정리 중...',
                status: '해야할 일',
                type: '기능',
                reporter: '웹지니',
                assignee: '웹지니'
            },
            // ...
        ];

    service.getStories = function () {
        return stories;
    };
});
```

이번 예제에서는 AngelloModel 객체를 MainCtrl 컨트롤러의 생성자 함수에 매개변수로서 전달한다. AngularJS는 **의존성 주입**(DI, Dependency Injection) 기법을 이용해 각 객체가 요구하는 의존성 객체들을 제공한다. 의존성 주입은 이름만 보면 쉬워 보이지만 그 구현은 이름만큼 쉽지는 않다. AngularJS는 이 컨트롤러가 AngelloModel 객체의 인스턴스를 필요로 한다는 것을 탐지하고는 이 객체의 인스턴스를 생성한 후에 MainCtrl 컨트롤러의 생성자에 주입하여 컨트롤러가 요구하는 의존성을 맞춰주게 된다.

```
// app.js
myModule.controller('MainCtrl', function(AngelloModel) {
    var main = this;

    // ...
    main.stories = AngelloModel.getStories();
    // ...
});
```

이제 main.stories 속성에 AngelloModel.getStories() 함수의 리턴값을 대입하면 된다. 이 방법의 장점은 스토리 데이터를 얻어오는 방법이나 장소에 관해 MainCtrl가 전혀 알고 있을 필요가 없다는 것이다. 이 책의 나머지 내용을 통해 계속해서 자세히 살펴보겠지만, 원격 서버를 호출하여 얻어온 데이터 역시 마찬가지 방법으로 손쉽게 다룰 수 있다.

한 가지 예제를 더 살펴본 후 디렉티브에 대한 이야기로 넘어가도록 하자. AngularJS의 서비스는 단순히 공통의 상태를 저장하는 것뿐만이 아니라 유틸리티 함수 같은 공통 기능들도 공유할 수 있는 객체다. 예를 들어, 배열을 매개변수로 전달받아 속성 매개변수를 바탕으로 인덱스를 생성하는 buildIndex처럼 아주 일반적으로 사용되는 메서드가 필요한 경우가 있다. 이 메서드를 통해 배열에 매번 루프를 실행할 필요 없이 배열 내에서 필요한 아이템을 찾을 수 있다. 이런 공통 함수들은 여러 곳에서 사용될 수 있으므로 다음과 같이 AngelloHelper 서비스에 구현하는 것이 바람직하다.

```
// app.js
myModule.factory('AngelloHelper', function() {
    var buildIndex = function(source, property) {
        var tempArray = [];

        for(var i = 0, len = source.length; i < len; ++i) {
            tempArray[source[i][property]] = source[i];
        }
```

```
        return tempArray;
    };

    return {
        buildIndex: buildIndex
    };
});
```

이처럼 코드를 세분화하면 코드 자체가 다른 코드와 격리되어 다른 런타임 컨텍스트(Runtime context)에 더 이상 의존하지 않기 때문에 유지보수 및 테스트가 수월해진다.

1.3.4 디렉티브

디렉티브(directive)는 AngularJS가 제공하는 컴포넌트 중 가장 강력하면서도 재미난 컴포넌트다. 사실, 우리는 이미 지난 절에서 실제로 동작하는 디렉티브들을 만나보았다. 예를 들면, ng-click 특성을 어떤 요소에 지정했던 경우는 AngularJS가 제공하는 내장 디렉티브를 이용함으로써 특정 요소의 동작을 확장한 것이다. 마찬가지로, 페이지에 ng-app이나 ng-controller 같은 특성을 지정한 경우 역시 AngularJS의 디렉티브를 이용하여 정적인 페이지에 새로운 동작을 추가한 것이라고 할 수 있다. 그림 1.9에서는 디렉티브가 AngularJS의 큰 그림 중 어느 부분에 해당하는지를 보여준다.

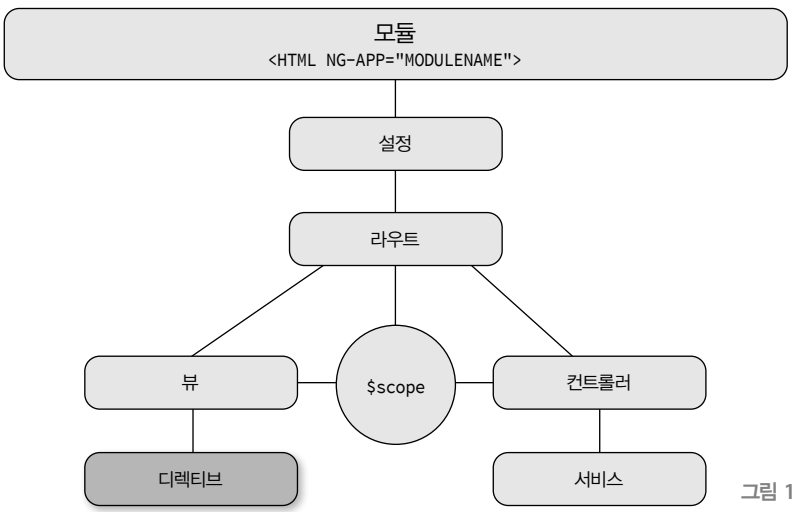

그림 1.9 **디렉티브**

그러면 안젤로 Lite에 간단한 디렉티브를 추가해보자. 이번에 추가할 디렉티브는 페이지 내에서 각 스토리 아이템을 표현할 story 디렉티브다. 디렉티브를 선언하는 방법은 컨트롤러나 서비스를 선언하는 경우와 마찬가지로 이름과 실제 동작을 구현할 함수를 조합하여 선언한다.

```
// app.js
myModule.directive('story', function(){
    return {
        scope: true,
        replace: true,
        template:'<div><h4>{{story.title}}</h4><p>{{story.description}}</p></div>'
    };
});
```

이 함수는 디렉티브 정의 객체(DDO, Directive Definition Object)를 리턴한다. 이 객체는 디렉티브를 구성하기 위한 정보를 정의한 객체다. 예제에서는 이 디렉티브의 매 인스턴스마다 새로운 스코프(scope) 객체가 필요하다는 정보와 함께, 디렉티브가 선언된 요소와 교체될 템플릿을 정의하고 있다. 템플릿 마크업은 앞서 우리가 현재 선택된 스토리 아이템의 title 속성과 description 속성값을 표시하기 위해 사용했던 것과 동일한 코드를 사용하고 있으므로 이미 익숙한 코드일 것이다.

이제 디렉티브를 선언했으므로 이제는 div 태그가 아닌 story 태그를 사용하도록 페이지의 HTML을 수정하자. 어? story 태그라니, 그런 태그가 있었나? 이제는 그런 태그가 있다!

```
<div ng-controller="MainCtrl as main">
    <div class="col-md-4">
        <h2>Stories</h2>
        <story class="callout"
               ng-repeat="story in main.stories"
               ng-click="main.setCurrentStory(story)">
        </story>
        <!-- ... -->
    </div>
</div>
```

지금까지 디렉티브를 이용해 HTML이 뭔가 새로운 동작을 하도록 확장하는 방법을 살펴보았다. 비록 간단한 예제이기는 하지만, 구현하고 싶은 HTML 태그와 특성을 직접 구현할 수 있다면 어떤 애플리케이션이라도 만들 수 있는 계기가 될 것이다.

1.4 요약

이상으로 안젤로 Lite 버전에 대한 설명을 마치도록 한다. 지금까지의 내용을 통해 그림 1.1에서 살펴봤던 대부분의 주요 컴포넌트들이 실제로 동작하는 모습을 살펴볼 수 있었다. 이 책의 나머지 부분에서는 안젤로 애플리케이션을 개발해 가면서 각각의 개념을 더 자세히, 그리고 더 유용하게 활용하는 방법을 살펴볼 것이다.

다음 장으로 넘어가기 전에 이 장에서 학습했던 내용들을 간단히 요약해보자.

- AngularJS는 대규모 자바스크립트 애플리케이션을 손쉽게 작성하고 관리하기 위한 목적으로 만들어진 프레임워크다.
- AngularJS는 처음부터 테스트를 염두에 두고 만들어졌다. 그 결과, 깔끔하고 안정적이며 확장 가능한 코드를 손쉽게 작성할 수 있다.
- 데이터 바인딩을 이용하면 지루한 DOM 이벤트 관련 코드를 더 이상 작성할 필요가 없기 때문에 (수만 라인까지는 아니지만) 수천 라인의 코드를 절약할 수 있다.
- AngularJS의 템플릿은 단순한 HTML이기 때문에 AngularJS의 UI를 구현하는 데 있어 기존의 기술을 손쉽게 활용할 수 있다.
- 순수 자바스크립트 객체(POJO, Plain Old JavaScript Objects)를 이용하면 다른 시스템과의 통합이 더욱 손쉬워진다.
- AngularJS의 모듈은 애플리케이션을 구성하기 위한 컨테이너들이다.
- AngularJS의 뷰는 컨트롤러와 함께 컴파일되고 렌더링된다.
- 컨트롤러는 뷰를 위한 뷰모델로서 뷰가 필요로 하는 데이터와 메서드를 제공하는 역할을 담당한다.
- 서비스는 AngularJS 애플리케이션이 공통적으로 사용하는 데이터와 기능을 캡슐화한다.
- 디렉티브는 사용자가 직접 정의하는 컴포넌트 또는 특성으로 HTML에 새롭고 강력한 기능을 부여할 수 있다.