Hanbit eBook
Realtime 36

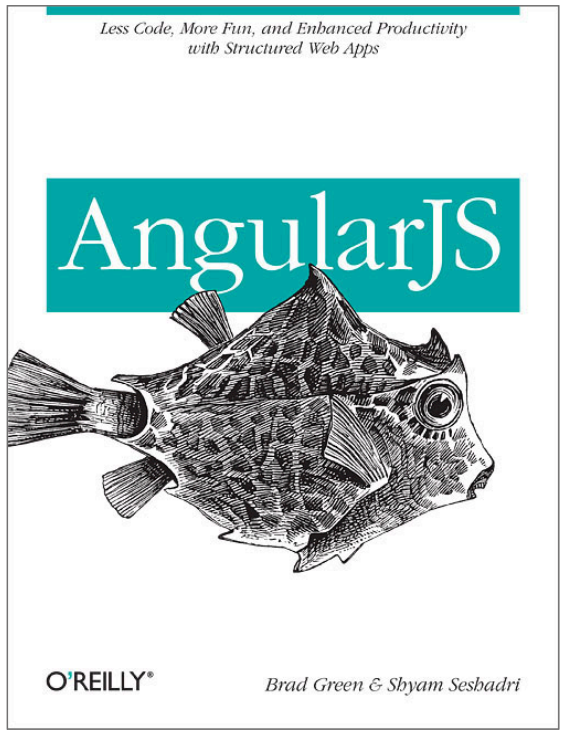
MVC 패턴을 구현하는 자바스크립트 프레임워크

AngularJS 기초편

AngularJS

브래드 그린, 샤이엄 세샤드리 지음 / 김지원 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는 O'REILLY의
AngularJS의
번역서입니다.

MVC 패턴을 구현하는 자바스크립트 프레임워크

AngularJS 기초편

MVC 패턴을 구현하는 자바스크립트 프레임워크 AngularJS 기초편

초판발행 2013년 7월 31일

지은이 브래드 그린, 샤이엄 세샤드리 / **옮긴이** 김지원 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-641-8 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 김병희 / **편집** 이세진
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
마케팅 박상용, 박주훈

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

저자 소개

지은이_ 브래드 그린Brad Green

구글 AngularJS 프로젝트 팀에서 엔지니어 관리자를 맡고 있으며, 접근성과 지원 공학을 총괄 감독한다. 구글에 입사하기 전에는 인터넷 기업을 만들어 팔던 AvantGo 사에서 초창기 모바일 웹 개발자로 근무하다가, 출장요식업에 뛰어들어 고단한 몇 년을 보냈다. 대학을 졸업하고 NeXT Computer 사에서 스티브 잡스 밑에서 데모 소프트웨어를 만들고 잡스의 슬라이드 프레젠테이션을 디자인했던 것이 브래드의 첫 직장 경험이다. 브래드는 아내와 두 자녀를 데리고 캘리포니아 주 마운틴 뷰에 살고 있다.

지은이_ 샤이엄 세샤드리Shyam Seshadri

Fundoo Solutions 사의 사주이자 CEO다. AngularJS에 관해 컨설팅하고 워크숍을 개최한다. 인도 시장을 겨냥한 혁신적 제품 개발에 주력하며, AngularJS를 주제로 한 워크숍을 운영하고 컨설팅한다. Fundoo Solutions 사를 창립하기 전에는 하이데라바드에 있는 명문 Indian School of Business에서 MBA 과정을 마쳤다. 샤이엄은 대학 졸업 후 첫 직업으로 구글에서 다수의 프로젝트를 진행했다. 그 중에는 AngularJS가 처음으로 사용된 구글 피드백Google Feedback 프로젝트도 있다. 그리고 다양한 내부 도구도 제작했다. 현재는 인도 나비뭄바이에서 회사를 운영하고 있다.

역자 소개

옮긴이_ **김지원**

웹 기술뿐 아니라 온갖 분야에 발을 뻗고 싶어하는 바람기를 지녔지만 역부족이다. 배워야 할 것이 갈수록 늘어나 시간의 결핍을 느낀다. 워드프레스, 프라이드치킨, 꿀, 분재, 컴퓨터 음악을 좋아한다. 기술 문서, 매뉴얼, 유비쿼터스 관련 논문을 번역한 바 있고 해외 논문 DB 구축 관련 작업에도 참여했다. 번역서로는 『한 권으로 끝내는 정규표현식』(한빛미디어, 2010), 『웹 표준 가이드: HTML5+CSS3』(한빛미디어, 2010), 『프로젝트로 배우는 HTML5+자바스크립트』(한빛미디어, 2012), 『리팩토링』(한빛미디어, 2012), 『엘리멘틀 디자인 패턴』(한빛미디어, 2013), 『HTML5+CSS3 디자인 패턴』(한빛미디어, 2013), 『HTML5 웹소켓 프로그래밍』(한빛미디어, 2013) 등이 있다.

저자 서문

Angular 프레임워크의 기원은 2009년의 구글 피드백Google Feedback이라는 프로젝트로 거슬러 올라간다. 필자는 테스트 가능한 코드를 작성하면서 수개월간 개발 속도와 기능의 문제로 난관을 겪었다. 6개월여간 작성한 프론트엔드front-end 코드가 대략 17,000줄이었다. 그런데 당시 프로젝트 팀원이던 미스코 헤브리Misko Hevery는 자신이 취미로 작성한 오픈소스 라이브러리를 사용하면 2주도 안 걸려서 우리가 작성한 코드 전체를 새로 작성할 수 있을 거라고 큰소리쳤다.

필자는 프로젝트가 2주쯤 지연돼도 별문제는 없을 것이고 설령 미스코가 장담한 대로 기한 내에 다시 작성하지 못 하더라도 기한에 쫓겨 허둥대는 미스코의 표정을 보는 재미라도 있겠다는 생각에 그렇게 해보라고 했다. 예상대로 미스코는 기한을 넘겨 3주만에 완료했다. 그래도 6개월이 걸렸던 개발을 3주라는 단시간에 재현했다는 점에 우리 팀 전원은 경악했는데, 더 놀랍게도 미스코가 새로 작성한 애플리케이션의 코드 분량이 원래의 17,000줄의 1/10도 안 되는 1,500줄에 불과했다. 미스코가 이뤄낸 성과는 추진해볼 가치가 있어 보였다.

미스코가 간단한 선언 문서로 창안했던 각종 개념을 중심으로 해서, 미스코와 나는 웹 개발자의 경험을 간소화할 팀을 만들기로 했다. 이 책의 공동 저자인 샤이엄 세샤드리Shyam Seshadri는 구글 피드백 팀에서 Angular의 첫 출시 애플리케이션 개발을 지휘했다.

그때부터 우린 구글의 여러 팀과 수백 명의 오픈소스 기부자의 도움을 받아 Angular를 개발했다. 많은 개발자가 일상적인 작업에 Angular 프레임워크를 이용하며 엄청난 Angular 지원망에 기여한다.

우리는 여러분이 나눠주는 지식을 얻게 되리란 생각에 감개무량하다.

감사의 글

Angular 프레임워크를 탄생시킨 미스코 헤브리에게 각별히 고마움을 전한다. 미스코 덕택에 웹 애플리케이션 작성법을 기존과 전혀 다른 방식으로 생각하고 실천할 수 있었다. 이고르 미나Igor Minar는 Angular 프로젝트의 안정화와 체계화에 기여했고 활성화된 지금의 오픈소스 커뮤니티의 모체를 만들었다. 보이타 지나Vojta Jina는 Angular의 많은 부분을 작성했으며 덕분에 우리는 테스트를 유례없이 신속하게 할 수 있었다. 나오미 블랙Naomi Black, 존 린퀴스트John Lindquist, 매사이어스 마셔스 니멜라Mathias Matias Niemela는 숙련된 솜씨로 편집을 도와주었다. 앞서 나열한 모든 분들과 더불어, 다방면에서 도움을 주고 실시간 애플리케이션 제작 과정에서 피드백을 통해 우리에게 Angular의 가치 있는 사용법을 알려준 Angular 커뮤니티 분들께 감사의 인사를 남긴다.

브래드 그린, 샤이엄 세샤드리

역자 서문

왜 AngularJS인가? 개발자가 구글의 AngularJS 플랫폼을 선택할 수밖에 없는 이유는 다음과 같다.

- 양방향 데이터 바인딩이 가능하다 - AngularJS로 개발한 애플리케이션은 클라이언트에서 서버로뿐만 아니라 서버에서 클라이언트로도 실시간 변경 감지가 이뤄진다. 감시, 리스너, 캡처 기능을 통해 개발한 코드가 실행되고 모델을 조작한 후 발생하는 변경사항을 감시한다.
- 모델, 뷰, 컨트롤러, 서비스 등 여러 구성요소로 분리된다 - 지시어, 필터, 모듈 등의 추상 객체를 이용해 균형을 맞출 수 있다. 이로써 복잡도의 감소와 관심사의 분리라는 두 마리 토끼를 얻을 수 있다.
- 편리하고 친숙한 패턴이 많다 - MVC나 종속물 주입 같은 유명한 패턴 외에도 종속물 관리 같은 다수의 패턴이 들어 있어서 체계적인 구성으로 개발할 수 있다.
- 테스트용 코드를 쉽게 작성할 수 있다 - AngularJS 공식 온라인 강좌 페이지에도 Jasmine 문법을 사용한 단위 테스트와 클라이언트-서버 테스트를 코드로 작성하는 방법이 예시돼 있다.

모든 프레임워크가 그렇듯 비록 AngularJS 역시 완벽할 순 없지만, 사소한 단점에 비해 얻을 수 있는 것이 많다. AngularJS에 관한 전반적인 내용이 이 책의 본문에 자세히 설명돼 있으니 자세한 얘기는 본문을 숙지하기 바란다.

AngularJS를 개발에 사용하면 길고 복잡한 코딩의 분량을 획기적으로 줄일 수 있다. 아주 간단하고 코딩 길이가 짧은 프로젝트라면 물론 큰 이득을 얻지 못 할 수도 있지만, 데이터 수정 시에 빠른 반응성이 요구되는 UI를 구상하고 있다면

AngularJS는 개발에 있어서 반드시 고려할 만하다. 플랫폼에 이미 내장돼 있는 지시어 말고도 개발자가 직접 정의한 지시어를 템플릿에 사용함으로써 모델, 컨트롤러와 바로 연결이 가능한 점은 강력한 기능이다. 종속물 주입 또한 상위의 기능에 필요한 종속물(종속 객체, 종속 함수, 종속 모듈 등)을 마치 혈관으로 연결된 링거에 주사기로 항생제를 주입하듯이 손쉽게 끼워 넣음으로써 매우 직관적이며 메서드 체인 형태로 호출이 가능하다.

이 모든 잡다한 서론을 뒤로 하고 지금 당장 본문의 첫 페이지로 가서 어떤 놀라운 장점이 있고 어떤 부분이 자신의 개발에 필요한지 살펴보자.

번역을 마무리하며,

김지원

도움을 주신 분들

베타테스터_ 원종필

탁월함의 갈망에 빠져있는 게임 클라이언트 프로그래머다. Unity를 사용해서 일을 하고 있으며, 업무의 질을 높이기 위해 고민 중이다. 최근에는 웹 개발에 관심을 두고 있다.

베타테스터_ 이기동

중3 때 ASP를 시작으로 VB, C#, FLEX, MFC를 거쳐, 현재는 JAVA와 JQM을 이용하여 전자지갑 모바일 웹을 개발하고 있는 책 읽기 좋아하는 프로그래머다. 튜닝이나 해킹에 관심이 많으며, 요즘은 먼가 만들 생각에 아두이노에 빠져있다. 제일 사랑하는 건 아내고, 좋아하는 건 아내가 해준 밥이며, 설거지를 무척이나 싫어한다.

베타테스터_ 이수한

자신에게 떨어지는 일거리를 코딩으로 승화하고 싶어하는 평범한 월급쟁이 직장인이다. 게임공학이 전공이만 운 좋게 게임 외에 다양한 프로젝트를 수행하였다. 평소 서버, 플랫폼 쪽 구현 기술에 관심이 많으며 언젠가는 나만의 플랫폼을 설계 및 구현해보겠다는 꿈을 꾸고 있는 중이다.

베타테스터_ 이아름

공대 아름이와 이름만 같은 코스모스 졸업을 앞둔 취업준비생이다. 정보통신공학 전공으로 현재 프로그래머가 되기 위해 밤낮없이 코딩에 매진하고 있다.

대상 독자 및 예제 파일

초급 **초중급** 중급 중고급 고급

이 도서는 AngularJS의 핵심을 빠르게 확인하고 싶은 독자를 대상으로 한다. 도서의 내용을 보다 잘 이해하려면, HTML과 자바스크립트를 어느 정도 알고 있어야 한다. 다음과 같은 독자들에 많은 도움이 될 것이다.

- 규모 있는 웹 애플리케이션 프로젝트의 실무 개발자
- 프레임워크 기반으로 자바스크립트에 익숙해지려는 웹 퍼블리셔
- jQuery 입문 이상으로 나아가려는 자바스크립트 개발자

이 도서의 예제 소스 코드는 다음 웹 사이트에서 다운받을 수 있다.

- https://github.com/shyamseshadri/angularjs-book (영문 버전)
- http://www.hanb.co.kr/exam/2641 (한글 버전)

영문 버전은 AngularJS 1.0.4 버전을 사용하였으며, 한글 버전은 AngularJS 1.0.7을 사용하였다. 이 도서에 있는 예제는 AngularJS 1.0.7을 사용하여 소스 코드를 테스트하였다.

소스 코드에는 구글 AngularJS 프레임워크 파일의 URL로 인클루딩되어 있다. 하지만 소스 코드의 간결성을 위해 일부 코드는 AngularJS 사용하였다. 이 도서에서 사용한 AngularJS 프레임워크 파일은 다음 웹 사이트에서 다운받을 수 있다.

- http://angularjs.org/

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | AngularJS 이해하기

뛰어난 웹 기반 애플리케이션을 제작하는 우리의 능력이 대단해진 만큼, 개발 과정도 많이 복잡해졌다. 필자는 Angular 팀에서 Ajax 애플리케이션을 개발할 때 부딪히는 난관을 줄이고자 했으며, 구글에 있을 때는 지메일, 구글맵, 캘린더 등 대형 웹 애플리케이션의 힘든 제작 과정을 처음부터 끝까지 해냈다. 그리고 이 경험을 활용해서 많은 사람에게 도움을 줄 수 있을 것이라 생각했다.

필자는 두세 줄의 코드를 작성했던 맨 처음과 비슷하게 웹 애플리케이션을 작성하고 싶었고, 그렇게 만든 결과물은 너무 형편없어 아연실색했다. 필자는 코딩 과정이 웹 브라우저의 특이한 내부 동작 원리에 맞춰나가는 게 아니라 창조한다는 느낌이길 바랬다.

그와 동시에, 처음에 어떤 설계가 애플리케이션을 쉽게 제작하고 이해할 수 있을지 판단할 수 있으면서 애플리케이션이 발전해가는 중간중간에 용이한 테스트/확장/유지보수를 위해 합리적 선택을 계속 해나갈 수 있는 환경을 원했다.

이를 위해서 필자는 Angular 프레임워크를 사용했는데, 얻어낸 성과를 생각하면 아주 기쁘다. 특히, Angular 관련 오픈소스 커뮤니티는 필자에게 많은 지원과 정보를 제공해주었다. 여러분도 커뮤니티에 가입해서 Angular를 훨씬 더 개선할 수 있는 방법을 공유했으면 좋겠다.

이 책의 예제 소스 코드는 GitHub[01]에 올려두었다. GitHub 페이지에서 소스 코드를 직접 볼 수도 있고 다운받아 실행해볼 수 있다.

---

01 https://github.com/shyamseshadri/angularjs-book

1.1 AngularJS를 이해하는 데 필요한 각종 개념

Angular 애플리케이션을 제작할 때 활용하는 핵심 개념이 몇 가지 있다. 필자가 그 개념들을 만든 것이 아니라 단지 다양한 개발 환경에서 효율성이 높은 문법을 비중 있게 차용해서 HTML, 웹 브라우저, 그 밖의 익숙한 각종 웹 표준 기술을 포괄하는 방식으로 구현했을 뿐이다.

1.1.1 클라이언트 측 템플릿

멀티 페이지multi-page 웹 애플리케이션은 서버에 있는 데이터를 HTML과 조립하고 연결한 후, 완성된 페이지를 웹 브라우저에 보내는 방법으로 페이지를 생성한다. Ajax 애플리케이션(싱글 페이지single-page 애플리케이션) 대부분도 이와 같은 방식으로 HTML을 생성한다. Angular 애플리케이션은 템플릿과 데이터가 웹 브라우저로 보내진 후, 웹 브라우저에서 조립된다는 점이 다르다. 결국 서버의 역할은 템플릿에 사용될 정적 리소스를 올려두고 템플릿에 필요한 데이터를 필요할 때마다 제공하는 것뿐이다.

데이터와 템플릿을 웹 브라우저에서 조립하는 작업을 Angular로 작성하면 소스 코드 1-1과 같다. 이번 예제는 프로그래밍의 입문으로 여겨지는 Hello World로, "Hello, World"를 하나의 문자열로 작성하는 것이 아니라 "Hello"라는 인사말을 나중에 수정할 수 있게끔 데이터로 구조화할 것이다.

hello.html 파일을 열고 소스 코드 1-1과 같이 템플릿을 작성하자.

소스 코드 1-1 Hello World 애플리케이션의 템플릿(hello.html)

```
<html ng-app>
<head>
    <script src="angular.js"></script>
    <script src="controllers.js"></script>
</head>
<body>
    <div ng-controller='HelloController'>
        <p>{{greeting.text}}, World</p>
    </div>
</body>
</html>
```

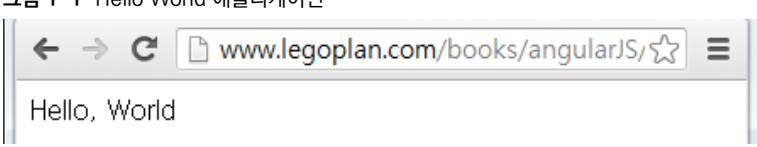
controllers.js 파일의 로직은 소스 코드 1-2와 같다.

소스 코드 1-2 Hello World 애플리케이션의 컨트롤러(controllers.js)

```
function HelloController($scope) {
    $scope.greeting = {
        text : 'Hello'
    };
}
```

선호하는 웹 브라우저에서 소스 코드 1-1의 hello.html을 로딩하면 그림 1-1과 같이 나타난다.

그림 1-1 Hello World 애플리케이션



위에서 사용한 HelloController 메서드는 흔히 사용하는 대부분의 메서드와 달리 다음과 같은 특징이 있다.

- HTML 파일 안에는 이벤트 리스너를 붙일 위치를 식별할 수 있는 클래스나 ID가 전혀 없다.
- HelloController 메서드가 greeting.text에 Hello를 할당한다면 이벤트 리스너를 등록하거나 콜백callback을 작성하지 않아도 된다.
- HelloController 메서드는 일반 자바스크립트 클래스며 Angular에 속한 어느 클래스도 상속받지 않는다.
- HelloController 메서드는 필요한 $scope 객체를 생성하지 않고 매개변수로 받는다.
- HelloController의 생성자를 호출하는 코드나 생성 시점을 알아내는 코드를 작성하지 않아도 된다.

나중에 더 많은 차이점을 알아보겠지만, 이 정도의 차이점만 보더라도 Angular 애플리케이션의 구조가 기존 애플리케이션과 다르다는 것만큼은 확실히 알 수 있다.

필자가 왜 Angular를 이용한 설계 방식을 택했고, Angular는 어떠한 원리로 돌아가는지 궁금할 것이다. 이제 Angular가 다른 곳에서 차용해온 유익한 몇 가지 개념을 살펴보자.

1.1.2 모델 뷰 컨트롤러(MVC)

MVCModel View Controller 애플리케이션 구조는 1970년대에 스몰토크Smalltalk의 구성 요소로서 도입됐다. 스몰토크에 속해 있던 초창기때부터 MVC는 사용자 인터페이스가 필수인 거의 모든 데스크톱 개발 환경에 사용됐다. C++, 자바, Objective-C 중 어느 것을 사용하더라도 MVC 구조를 일부 활용할 수 있다. 그러나 최근까지 MVC는 웹 개발에 거의 사용되지 않았다.

MVC는 개발자가 코드에서 데이터(모델model) 관리, 애플리케이션 로직(컨트롤러controller), 데이터 표현(뷰view)를 명확하게 서로 분리하는 것이 주된 목적이다.

뷰는 모델에서 데이터를 가져와 사용자에게 표시한다. 사용자가 클릭이나 타이핑 등으로 애플리케이션을 조작하면, 컨트롤러는 모델에 있는 데이터를 변경하는 식으로 응답한다. 모델은 변경이 발생했음을 뷰에 알려서, 뷰가 변경사항을 현재 표시 중인 내용에 적용하게 한다.

Angular 애플리케이션에서 뷰는 DOMDocument Object Model, 문서 객체 모델이고, 컨트롤러는 자바스크립트 클래스며, 모델 데이터는 객체 속성에 저장된다.

필자가 MVC를 깔끔하다고 생각하는 이유는 다음과 같다.

첫째, MVC에는 무엇을 어디에 넣어야 할지에 대한 멘탈 모델mental model이 있어서, 개발자가 매번 멘틀 모델을 생각하지 않아도 된다. 둘째, 자신이 작성한 코드가 MVC 구조에 따라 구성되어 있음을 한눈에 알 수 있어서 프로젝트 팀원들도 무슨 내용의 코드인지를 쉽게 이해할 수 있다. 셋째, MVC는 애플리케이션을 보다 쉽게 확장하고 유지하며 테스트할 수 있는 굉장한 장점들이 있다.

1.1.3 데이터 바인딩

Ajax 싱글 페이지 애플리케이션이 보편화되기 전에는, 전송된 데이터를 사용자에게 보여주기 전에 HTML의 문자열을 데이터와 합치는 방식으로 사용자 인터페이스를 손쉽게 작성할 수 있는 레일스Rails, PHP, JSP 같은 플랫폼을 이용했다.

jQuery 같은 라이브러리는 MVC 모델을 클라이언트로 확장해서 개발자가 비슷한 방식으로 인터페이스를 작성할 수 있으며, 페이지 전체가 아니라 DOM의 일부분만을 따로 업데이트할 수 있다. 이 책에서는 템플릿 HTML 문자열을 데이터와 합친 후, 플레이스홀더placeholder 요소에 있는 innerHtml 속성을 설정하여 합친 데이

터를 DOM 내부의 원하는 위치에 삽입하는 방식을 사용한다.

이 방식은 다른 건 다 좋은데, UI에 업데이트된 데이터를 삽입하거나 사용자 입력 내용에 따라 데이터를 변경할 때, UI와 자바스크립트 속성에 데이터를 할당하기 위해 추가해야 할 작업이 상당히 많다.

그런데 개발자가 코딩하지 않고도 이 모든 작업이 저절로 처리된다면 얼마나 좋을까? 개발자는 단지 UI의 어느 부분에 어떤 자바스크립트 속성을 할당할지 선언만 하고 동기화는 자동으로 이뤄지게 할 수 있다면 얼마나 좋을까? 이런 식의 프로그래밍을 '데이터 바인딩'이라고 한다. 필자는 Angular에 데이터 바인딩 기능을 포함시켰다. 데이터 바인딩을 MVC와 연동하면, 코딩 작업 없이 뷰와 모델을 작성할 수 있기 때문이다. 데이터를 한 곳에서 다른 곳으로 옮길 때 필요한 작업 대부분은 자동으로 이뤄진다.

이러한 기능을 눈으로 확인하기 위해 소스 코드 1-1의 템플릿을 동적으로 수정해 보자. 앞에서와 같이 HelloController 메서드는 greeting.text 값을 한 번 지정하고 이후에는 절대로 변경하지 않는다. 동적으로 만들기 위해 사용자의 입력 내용에 따라 greeting.text의 값이 변하는 텍스트 인풋을 추가하자. 수정한 템플릿은 소스 코드 1-3과 같다.

소스 코드 1-3 Hello World 애플리케이션의 템플릿 1차 수정(HelloDynamic.html)

```
<html ng-app>
<head>
    <script src="angular.js"></script>
    <script src="controllers.js"></script>
</head>
<body>
    <div ng-controller='HelloController'>
        <input ng-model='greeting.text'>
```

```
        <p>{{greeting.text}}, World</p>
    </div>
</body>
</html>
```

HelloController 컨트롤러는 그대로 두면 된다. 이 파일을 웹 브라우저에서 로딩하면 그림 1-2와 같은 화면이 나타난다.

그림 1-2 Hello World 애플리케이션의 기본 상태

인풋 필드input field 안에 있는 Hello를 '안녕'으로 수정하면 화면이 그림 1-3과 같이 변한다.

그림 1-3 Hello World 애플리케이션에서 입력 내용을 수정한 결과

인풋 필드에 'change 이벤트 리스너'를 붙이지 않고도 동적으로 업데이트되는 UI를 작성했다. 서버에서 변경된 내용이 UI에 동적으로 업데이트될 뿐만 아니라, UI에서 변경된 내용이 서버에 동적으로 저장되기도 한다. 앞서 작성한 컨트롤러에서 서버로 요청하고, 응답을 받으며, scope.greeting.text 속성에 반환되는 값을 지정할 수도 있다. Angular는 인풋의 내용과 중괄호 안의 텍스트를 반환되는 값으로 자동 업데이트한다.

1.1.4 종속물 주입

앞에서도 언급했지만 HelloController를 통해 이뤄지는 작업 대부분을 개발자가 작성할 필요가 없음은 수백 번 강조해도 지나치지 않다. 예컨대 데이터 바인딩을 수행하는 $scope 객체는 자동으로 넘어오기 때문에, 함수를 호출해서 객체를 생성하지 않아도 된다. 단지 HelloController 생성자 안에 $scope를 넣기만 하면 요청이 이뤄진다.

뒤에 나올 장들에서 살펴보겠지만, 요청할 수 있는 것은 $scope 객체만이 아니다. 예를 들어 사용자 웹 브라우저에 표시된 현재 URL에 데이터를 바인딩하려면, 소스 코드 1-4와 같이 HelloController 생성자 메서드에 $location 매개변수도 추가로 전달해서 이를 관리하는 객체를 요청할 수 있다.

소스 코드 1-4 Hello World 애플리케이션의 컨트롤러에 위치를 추가(controllers.js)

```
function HelloController($scope, $location) {
    $scope.greeting = {
        text : 'Hello'
    };
    // 여기에 $location을 사용해 유용한 기능을 구현
}
```

이 놀라운 결과를 가능케 한 것은 Angular의 '종속물 주입 시스템Dependency Injection System'이다. 종속물 주입을 이용하면 개발자는 종속물을 작성하지 않아도 되며 작성한 클래스가 직접 필요한 것을 요청하는 개발 방식을 따르면 된다.

이런 개발 방식은 '디미터의 법칙Law of Demeter'[02]이라는 디자인 패턴을 따른다. 디미

02 역자 주_ 디미터의 법칙 패턴이란 객체 간의 약결합을 요구하는 객체지향 프로그램을 개발하기 위한 소프트웨어 설계 패턴으로 다음과 같은 원칙을 지닌다.
- 각 객체는 자신에게 밀접히 관련된 객체들에 대한 정보만을 지니되, 한정된 정보만 지녀야 한다.
- 각 객체는 밀접히 관련된 객체끼리만 소통해야 하며, 관련 없는 객체와 소통해선 안 된다.
- 직접적으로 관련된 객체끼리만 소통해야 한다.

터의 법칙을 다른 말로 '최소 지식의 원칙the principle of least knowledge'이라고 한다. HelloController의 역할은 greeting 모델의 초기 상태를 설정하는 것으로, 디미터의 법칙 패턴에 따라 HelloController는 $scope 객체가 생성되는 원리나 $scope 객체가 있는 위치 같은 불필요한 정보에는 관여하지 말아야 한다.

이 특징은 Angular 프레임워크로 생성된 객체에만 적용되는 것이 아니다. 소스 코드 1-4의 미완성된 부분을 직접 작성해보자.

1.1.5 지시어

Angular의 가장 큰 장점 중 하나는 템플릿을 HTML 파일로 작성할 수 있다는 것이다. 이것이 가능한 이유는 프레임워크 코어에 강력한 DOM 변환 엔진을 넣어두었기 때문이다. DOM 변환 엔진을 통해 개발자는 HTML 문법을 확장할 수 있다.

앞에서 템플릿 예제를 통해 HTML 명세에는 나와 있지 않은 새로운 속성 몇 가지를 이미 살펴보았다. 예제는 데이터 바인딩을 위한 이중 중괄호 표기, 컨트롤러가 뷰의 어느 부분을 감독할지를 지정하는 ng-controller, 인풋을 해당 모델의 구성물에 바인딩하는 ng-model을 사용한다. 이러한 HTML 확장 문법을 '지시어directive'라고 한다.

Angular에는 애플리케이션의 뷰를 쉽게 정의할 수 있는 여러 지시어가 있다(나중에 더 많은 지시어를 살펴볼 것이다). 이런 지시어는 템플릿을 정의할 수 있다. 지시어를 선언해서 애플리케이션의 구동 원리를 지정할 수도 있고, 재사용 가능한 구성요소를 생성할 수도 있다.

그리고 Angular에 내장된 지시어를 사용할 수도 있지만, 자신만의 지시어를 작성해서 HTML 템플릿 기능을 자유롭게 확장할 수도 있다.

1.2 장바구니 예제

조금 복잡한 예제를 통해 Angular의 더 많은 기능을 알아보자. 쇼핑몰 애플리케이션을 제작한다고 생각해보자. 쇼핑몰 애플리케이션의 어딘가에는 사용자의 장바구니를 표시하고 사용자가 수정할 수 있게 해야 한다. 다음 장바구니 부분의 코드를 살펴보자.

소스 코드 1-5 장바구니 애플리케이션(order-form.html)

```
<!doctype html>
<html lang='ko' ng-app>
<head>
    <title>장바구니</title>
</head>
<body ng-controller="CartController">
    <h1>내 장바구니</h1>
    <div ng-repeat="item in items">
        <span>{{item.title}}</span>
        <input ng-model="item.quantity">
        <span>{{item.price | currency}}</span>
        <span>{{item.price * item.quantity | currency}}</span>
        <button ng-click="remove($index)">삭제</button>
    </div>
    <script src="angular.js"></script>
    <script>
        function CartController($scope) {
            $scope.items = [ {
                title : '페인트 그릇',
                quantity : 8,
                price : 3.95
            }, {
                title : '땡땡이 리본',
```

```
                quantity : 17,
                price : 12.95
            }, {
                title : '공깃돌',
                quantity : 5,
                price : 6.95
            } ];
            $scope.remove = function(index) {
                $scope.items.splice(index, 1);
            }
        }
    </script>
</body>
</html>
```

웹 브라우저에서 소스 코드 1-5를 로딩하면 그림 1-4와 같은 UI를 볼 수 있다.

그림 1-4 장바구니 UI

여기서는 소스 코드 1-5를 간단히 분석해보자. 더 자세한 내용은 이 책의 나머지 장을 참고하기 바란다.

맨 위의 코드부터 차근차근 살펴보자.

```
<html lang='ko' ng-app>
```

어떤 요소에 ng-app 지시어를 지정하면, 페이지에서 그 요소에 해당하는 부분을 Angular가 관리하게 된다. 앞의 코드에서 ng-app 지시어를 지정한 요소가 〈html〉이므로, Angular는 페이지 전체를 관리하게 된다. 이렇게 Angular가 페이지 전체를 관리하게 지정하는 것이 목적에 대부분 맞지만, 다른 메서드를 사용해 페이지를 관리하는 기존 애플리케이션에 Angular를 통합할 때는 애플리케이션의 〈div〉 요소에 지정하기도 한다.

```
<body ng-controller="CartController">
```

Angular에서 개발자는 '컨트롤러'라고 하는 자바스크립트 클래스로 구성된 페이지 영역을 관리한다. CartController가 〈body〉와 〈/body〉 사이의 모든 것을 관리하도록 선언하려면, 앞의 코드처럼 body 태그에 컨트롤러를 삽입하면 된다.

```
<div ng-repeat="item in items">
```

앞의 코드에서 ng-repeat 지시어는 지정된 〈div〉 요소 내부의 DOM을 items 배열의 모든 원소에 한 번씩 복사하라고 명령한다. 그리고 〈div〉의 모든 사본에서 현재 요소[03]에 item이라는 속성을 지정해서 템플릿 안에 사용할 수 있게 한다. 보다시피 이것은 제품명, 수량, 단가, 총액, 항목 삭제 버튼이 들어 있는 〈div〉를 3개 생성한다.

```
<span>{{item.title}}</span>
```

03 역자 주_ ng-repeat는 프로그램 루프 같은 개념이다. 이 지시어가 지정된 요소 안에 포함된 요소는 계속 반복된다. c 언어 같으면 i=0; i<5; i++ 라는 for문 조건절 하위에서 i가 가리키는 것이 현재가 된다.

Hello World 예제에서 보았다시피 이중 중괄호 {{ }}를 사용해 데이터 바인딩하면, 변수의 값을 페이지 안에 삽입하고 동기화를 유지할 수 있다. 표현식 {{item.title}}은 반복문 안에 현재 item을 가져와서, item의 title 속성에 할당돼 있는 내용을 DOM에 삽입한다.

```
<input ng-model="item.quantity">
```

앞의 코드처럼 ng-model 지시어를 지정하면 인풋 필드와 item.quantity의 값 사이에 데이터 바인딩이 이뤄진다.

〈span〉 요소로 감싸인 {{ }} 표기법은 "값을 여기에 삽입해라"는 일방적인 관계를 설정한다. 사용자 입장에서는 이 기능으로 충분하지만, 장바구니 애플리케이션에서는 사용자가 인풋의 수량을 변경하는 시점을 알아야 실시간으로 표시되는 총액을 변경할 수 있다.

ng-model 지시어를 지정했으므로 변경사항이 계속 모델에 반영된다. ng-model 지시어의 선언으로 인해 item.quantity의 값이 텍스트 필드에 삽입될 뿐만 아니라, 사용자가 새 값을 입력할 때마다 item.quantity를 자동으로 업데이트한다.

```
<span>{{item.price | currency}}</span>
<span>{{item.price * item.quantity | currency}}</span>
```

단가와 총액을 달러 단위로 표시하고자 한다. Angular에 있는 '필터' 기능을 이용하면 텍스트를 변환할 수 있다. 값을 화폐 단위로 형식화하는 currency라는 내장 필터가 있는데, 이 필터를 이용하면 값을 달러 단위의 형식으로 변환할 수 있다. 필터에 대한 상세한 내용은 다음 장을 참고하자.

```
<button ng-click="remove($index)">삭제</button>
```

앞의 코드로 인해, 제품 옆에는 장바구니 항목 삭제를 위해 사용자가 클릭할 수 있는 [삭제] 버튼이 생긴다. [삭제] 버튼을 클릭하면 remove() 함수가 호출된다. 삭제할 항목을 알려주기 위해 remove() 함수의 인자로 ng-repeat 반복 횟수가 저장되어 있는 $index를 전달했다.

```
function CartController($scope) {
```

CartController 메서드는 장바구니 로직을 관리한다. 컨트롤러에 $scope가 필요하다는 것을 Angular에 알리기 위해 $scope를 매개변수로 전달했다. $scope를 통해 UI를 구성하는 요소들의 데이터 바인딩이 이뤄진다.

```
$scope.items = [
   { title : '페인트 그릇', quantity : 8, price : 3.95 },
   { title : '땡땡이 리본', quantity : 17, price : 12.95 },
   { title : '공깃돌', quantity : 5, price : 6.95 }
];
```

$scope.items를 정의해서 사용자 장바구니에 든 항목들의 집합을 나타내는 더미 데이터 해시dummy data hash를 생성했다. UI와 데이터 바인딩이 가능해야 하므로, 사용자 장바구니에 든 항목을 $scope에 추가한다.

당연한 이야기지만, 장바구니가 메모리에서만 구동될 수는 없으므로 서버와 통신해서 데이터를 데이터베이스 같은 영구 저장소에 적절히 저장하게 해야 한다. 데이터의 영구 저장에 대해서는 뒤에 나오는 장들에서 다루겠다.

```
$scope.remove = function(index) {
    $scope.items.splice(index, 1);
}
```

UI에 바인딩하려면 remove() 함수가 필요하므로 $scope에도 remove() 함수를 추가했다. 메모리에서만 구동되는 장바구니 애플리케이션은 remove() 함수가 배열에 있는 항목을 삭제하는 것만 가능하다. ng-repeat 지시어로 생성된 〈div〉 목록은 데이터 바인딩에 의해 실시간 반영되므로, 항목을 삭제하면 해당 항목의 행이 사라지면서 자동으로 아래 항목이 그 행으로 올라온다. 이 remove() 함수는 사용자가 항목의 [삭제] 버튼을 클릭할 때마다 UI에서 호출됨을 명심하자.

1.3 나머지 장에서 다룰 내용

이 장에서는 Angular의 가장 기본적인 문법과 아주 간단한 예제만을 살펴보았다. 나머지 장에서는 Angular 프레임워크의 필수 기능에 대해서 알아볼 것이다.

2 | AngularJS 애플리케이션 해부

필요한 함수만 골라 사용하는 일반 라이브러리와 달리, Angular 프레임워크의 모든 구성요소는 협업 솔루션으로 사용하게끔 설계되어 있다. 이 장에서는 Angular의 기본 구성요소를 간결하게 설명하겠다. 이를 통해 각 요소의 연동 방법을 이해할 수 있을 것이다. 구성요소 대부분에 대한 상세한 설명은 나머지 장에서 다룬다.

2.1 Angular 호출

Angular를 시작하기 위해 모든 애플리케이션에 반드시 다음 두 가지 작업을 해야 한다.

1. angular.js 라이브러리 파일을 로딩해야 한다.
2. ng-app 지시어를 지정해서 DOM의 어느 부분을 관리할지 Angular에 전달해야 한다.

2.1.1 스크립트 로딩

angular.js 라이브러리 로딩은 간단하며 일반 자바스크립트 라이브러리를 로딩하는 것과 방법이 같다. 구글의 CDNContent Delivery Network, 콘텐츠 전송 네트워크에 올려져 있는 스크립트 파일을 로딩하는 방법은 다음과 같다.

```
<script
  src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js">
</script>
```

구글의 CDN을 이용하는 것은 여러모로 좋다. 구글 서버는 빠른데다가 어떤 애플

리케이션이든 캐시에 스크립트를 저장하는 기능이 있기 때문이다. 따라서 사용자는 Angular 플랫폼 기반의 애플리케이션을 여러 개 사용할 때 Angular 스크립트 파일을 한 번만 받으면 된다. 게다가 사용자가 구글 CDN의 Angular 링크를 이용한 웹 사이트를 방문한 적이 있으면, 다른 웹 사이트를 방문할 때 Angular 스크립트 파일을 다시 받지 않아도 된다.

Angular 스크립트 파일을 로컬이나 다른 위치에 두고 사용하고 싶다면, 구글 Angular 스크립트 파일을 받아서 자신이 원하는 위치에 두고 src 속성에 정확한 파일 경로를 지정하면 된다.

2.1.2 ng-app 지시어로 Angular의 경계 선언

ng-app 지시어를 이용하면 페이지의 어느 부분을 관리할지 Angular에 명령할 수 있다. 순수 Angular 애플리케이션을 제작할 때는 ng-app 지시어를 다음과 같이 〈html〉 태그에 넣는다.

```
<html ng-app>
    ...
</html>
```

앞의 코드는 페이지 안의 모든 DOM 요소를 관리하라고 Angular에 명령한다.

기존 애플리케이션이 있는데, 그 애플리케이션이 자바나 레일스 같은 타 기술을 이용해 DOM을 관리하게 돼 있다면, 다음과 같이 Angular가 관리하게 할 부분을 〈div〉 같은 요소로 묶은 후 ng-app 지시어를 그 요소에 넣으면 된다.

자바나 레일스 같은 기술을 이용해 DOM을 관리하는 애플리케이션이 있다면, 다음과 같이 Angular가 관리하게 할 부분을 〈div〉 같은 요소로 묶은 후 ng-app 지시어를 그 요소에 넣으면 된다.

```
<html>
    ...
    <div ng-app>
        ...
    </div>
    ...
</html>
```

2.2 모델 뷰 컨트롤러

Angular가 모델 뷰 컨트롤러 애플리케이션 설계 방식을 지원한다고 1장에서 설명했다. Angular 애플리케이션은 설계가 아주 유연하지만, 다음과 같은 기본적으로 지켜야 할 규칙이 있다.

- 모델에는 애플리케이션의 현 상태를 나타내는 데이터가 들어간다.
- 뷰는 현 상태를 나타내는 데이터를 표시한다.
- 컨트롤러는 모델과 뷰의 관계를 관리한다.

모델은 객체 속성을 사용하거나 데이터가 담긴 기본 타입만을 사용해 작성한다. 모델 변수는 특별할 것이 없다. 사용자에게 어떤 텍스트를 표시하려면 문자열을 다음과 같이 지정하면 된다.

```
var someText = '지금 여러분의 여정은 시작됐습니다.';
```

뷰는 HTML 페이지로 템플릿을 작성하고 거기에 모델에서 전달받은 데이터를 합쳐서 생성한다. 앞에서 보았다시피 DOM 안에 플레이스홀더placeholder를 삽입하고, 플레이스홀더의 텍스트를 다음과 같이 지정하면 된다.

```
<p>{{someText}}</p>
```

이중 중괄호 문법은 기존 템플릿에 새로운 내용을 삽입한다는 뜻에서 인터폴레이션interpolation이라고 한다.

컨트롤러는 클래스 또는 타입이며, 다음과 같이 모델을 구성할 객체나 기본 타입을 전달 받은 $scope 객체에 할당해서 Angular에 알리는 것이 목적이다.

```
function TextController($scope) {
    $scope.someText = someText;
}
```

이제까지 설명한 코드를 조합하면 다음과 같다.

소스 코드 2-1 1.dataBinding.html

```
<html ng-app>
<body ng-controller="TextController">
    <p>{{someText}}</p>

    <script
        src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/
        angular.min.js">
    </script>

    <script>
        function TextController($scope) {
            $scope.someText = '지금 여러분의 여정은 시작됐습니다.';
        }
    </script>
```

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

앞의 코드를 웹 브라우저에서 로딩하면 다음과 같이 출력되는 것을 볼 수 있다.

지금 여러분의 여정은 시작됐습니다.

이 기본 스타일 모델은 단순한 경우에나 사용할 수 있다. 애플리케이션 대부분은 데이터를 보관할 모델 객체를 작성해야 한다. 메시지 모델 객체를 작성하고 그 객체를 통해 someText를 저장해보자.

```
var someText = '지금 여러분의 여정은 시작됐습니다.';
```

그러기 위해서는 앞의 코드를 다음과 같이 수정한다.

```
var messages = {};
messages.someText = '지금 여러분의 여정은 시작됐습니다.';
function TextController($scope) {
    $scope.messages = messages;
}
```

그리고 이것을 템플릿 파일에서 다음과 같이 적용하자.

```
<p>{{messages.someText}}</p>
```

나중에 $scope 객체를 설명할 때 보겠지만, 이런 식으로 모델 객체를 작성하면 프로토타입 상속prototypal inheritance으로 인해 $scope 객체에서 발생할 수 있는 예기치 못한 동작을 막을 수 있다.

장기적으로 바람직한 습관을 이야기하면서도, 앞의 예제는 TextController를 전역 스코프에 작성했다. 예제니까 그래도 되지만, 원래 컨트롤러는 애플리케이션의 관련 부분들을 나타내는 네임스페이스를 사용하는 모듈 안에 정의하는 것이 올바른 방법이다.[01] 수정한 코드는 다음과 같다.

소스 코드 2-2 2.dataBinding.html

```
<html ng-app='myApp'>
<body ng-controller="TextController">
    <p>{{someText.message}}</p>

    <script
        rc="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/
        angular.min.js">
    </script>

    <script>
        var myAppModule = angular.module('myApp', []);

        myAppModule.controller('TextController',
            function TextController($scope) {
            var someText = {};
            someText.message = '지금 여러분의 여정은 시작됐습니다.';
            $scope.someText = someText;
        });
    </script>
</body>
</html>
```

01 역자 주_ 'myApp'은 애플리케이션 관련 부분을 나타내는 네임스페이스고 myAppModule은 네임스페이스 myApp을 사용하는 모듈이다.

수정한 코드를 살펴보자. 먼저 ng-app 지시어에 모듈명 myApp을 지정했다. 이후에 스크립트에서 Angular 객체를 호출해서 myApp이라는 모듈을 생성한 후, 생성한 모듈의 컨트롤러 함수를 호출하여 작성한 컨트롤러 함수를 매개변수로 전달했다.

뒤에서 모듈에 대해 보다 자세히 살펴볼 것이다. 지금은 전역 네임스페이스를 제쳐두고 생각하는 것이 좋은데, 그러기 위한 방법으로 모듈을 사용한다.

2.3 템플릿과 데이터 바인딩

Angular 애플리케이션에서 템플릿은 다른 정적 리소스처럼 서버에서 로딩되거나, <script> 태그 안에 정의하는 HTML 문서에 불과하다. 템플릿 안에 UI를 정의할 때는 표준 HTML과 Angular 지시어를 사용한다. Angular 지시어를 사용하려면 UI 컴포넌트가 필요하다.

Angular는 우선 웹 브라우저에서 템플릿을 데이터와 합쳐 완전한 애플리케이션을 만든다. 이렇게 템플릿과 데이터를 합치는 코드는 1장에서 장바구니에 든 품목들을 표시하는 예제를 통해 살펴보았다.

```
<div ng-repeat="item in items">
   <span>{{item.title}}</span>
   ...
</div>
```

앞의 코드는 바깥의 를 한 번 표시한 후, items 배열의 원소마다 안의 내용을 반복해서 표시한다.

그럼 이 데이터는 어디에서 왔을까? 장바구니 예제에서는 데이터를 코드의 배열 안에 정의했다. 이런 방식은 UI를 제작하면서 어떤 식으로 돌아가는지 테스트할

때 적합하다. 그러나 애플리케이션 대부분은 서버에 저장되어 있는 데이터를 사용한다. 웹 브라우저에서 띄운 애플리케이션이 서버에 접속해서 사용자가 로딩한 페이지에 필요한 데이터를 요청하면, Angular는 그 데이터를 템플릿에 결합시킨다.

기본적인 시작 단계는 다음과 같다.

1. 사용자가 애플리케이션의 첫 페이지를 요청한다.
2. 사용자의 웹 브라우저가 서버로 HTTP 접속을 하고 템플릿이 든 index.html 페이지를 로딩한다.
3. Angular가 index.html 페이지 안에 로딩되고, 페이지 로딩이 완료될 때까지 기다렸다가 ng-app 지시어를 찾아서 템플릿 경계를 인식한다.
4. Angular는 템플릿을 살피면서 지시어와 바인딩을 찾아 리스너 등록과 DOM 조작을 수행하고 서버에서 초기 데이터를 가져온다. 마침내 애플리케이션의 부트스트랩이 완료되고 템플릿은 DOM을 통해 뷰로 변환된다.
5. 애플리케이션은 서버에 접속해서 필요한 경우 사용자에게 보여줄 추가 데이터를 로딩한다.

1단계부터 3단계까지는 모든 Angular 애플리케이션에 공통적으로 적용된다. 개발자는 4단계와 5단계를 선택할 수 있다. 두 단계는 동시에 이뤄질 수도 있고 순차적으로 이뤄질 수도 있다. 맨 처음에 뜨는 뷰에는 요청 횟수를 줄여 성능을 높이기 위해 표시할 데이터를 HTML 템플릿과 함께 수신하는 방법도 있다.

Angular를 이용해 애플리케이션을 구조화하면, 애플리케이션의 템플릿과 그 안에 채워질 데이터를 따로 유지할 수 있다. 그래서 템플릿을 캐시에 저장할 수 있다. 최초 로딩 후에는 새 데이터만 웹 브라우저로 수신되면 된다. 자바스크립트, 이미지, CSS를 비롯한 리소스와 마찬가지로, 이렇게 독립된 템플릿을 캐시에 저장하면 애플리케이션의 성능이 훨씬 더 좋아진다.

2.3.1 텍스트 표시

ng-bind 지시어를 사용하면 UI 내부의 어느 위치에서든 텍스트를 표시하고 업데이트할 수 있다. ng-bind 지시어의 문법은 두 가지다. 다음과 같은 이중 중괄호 문법은 앞서 살펴보았다.

```
<p>{{greeting}}</p>
```

또 다른 문법은 다음과 같이 ng-bind 지시어를 속성으로 지정하는 것이다.

```
<p ng-bind="greeting"></p>
```

앞의 두 방법 중 어느 것을 사용하든 결과는 같다. 만약 greeting 모델 변수에 "Hi there" 값이 지정돼 있다면 Angular는 다음과 같은 HTML을 생성한다.

```
<p>Hi there</p>
```

이 HTML은 웹 브라우저에 "Hi there"이라는 문자열을 표시한다.

그런데 왜 한 문법이 다른 형태의 문법보다 더 많이 사용될까? 필자가 이중 중괄호를 사용하는 인터폴레이션 문법을 만든 이유는 보다 읽기 편하고 코딩을 위한 타이핑 분량을 줄이기 위해서다. 두 문법 모두 결과는 같지만, 이중 중괄호 문법을 사용하면 애플리케이션의 최초 페이지인 index.html을 로딩할 때 Angular가 중괄호 문법을 데이터로 치환하기 전에 사용자가 렌더링 중인 템플릿을 보게 될 가능성이 있다. 단, 로딩된 이후에는 뷰에 이런 문제가 생기지 않는다.

이렇게 되는 이유는 웹 브라우저가 HTML 페이지를 로딩한 후, 그 페이지를 렌더