Hanbit RealTime 136

Spring MVC 4 익히기

스프링 MVC 프레임워크를 이용한 스프링 웹 애플리케이션 개발

지오프로이 워렌 지음 **김지헌** 옮김

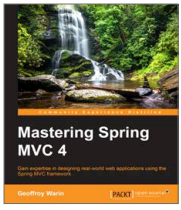
[PACKT] PUBLISHING 한빛미디어 Hanbit Media, Inc.

Hanbit RealTime 136

Spring MVC 4 익히기

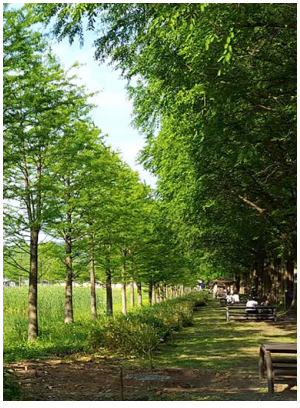
스프링 MVC 프레임워크를 이용한 스프링 웹 애플리케이션 개발

지오프로이 워렌 지음 **김지헌** 옮김



이 도서는
Mastering Spring MVC 4(PACKT publishing)의
번역서입니다

[PACKT] PUBLISHING
한빛미디어 Hanbit Media, Inc.



표지 사진 **김은진**

이 책의 표지는 김은진님이 보내 주신 풍경사진을 담았습니다.
리얼타임은 독자의 시선을 담은 풍경사진을 책 표지로 보여주고자 합니다.

사진 보내기 **ebookwriter@hanbit.co.kr**

Spring MVC 4 익히기 스프링 MVC 프레임워크를 이용한 스프링 웹 애플리케이션 개발

초판발행 2016년 8월 24일

지은이 지오프로이 워렌 / **옮긴이** 김지헌 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 9월 30일 제10-1779호
ISBN 978-89-6848-828-3 95000 / **정가** 17,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 정지연
디자인 표지 강은영, 내지 여동일, 조판 최송실
마케팅 박상용, 송경석, 변지영 / **영업** 김형진, 김진불, 조유미

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

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

지은이_ **지오프로이 워렌(Geoffroy Warin)**

소프트웨어 장인 운동Software Craftsmanship movement과 오픈소스 확산을 지지하고 이에 대한 확고한 신념을 가진 개발자로, 개발을 시작한 지 10년이 넘었다. 그는 주로 자바와 자바스크립트로 엔터프라이즈급 웹 애플리케이션을 구상하는 일을 해왔다.

지오프로이는 백엔드와 프론트엔드 양쪽에 모두 능통하며, 깔끔한 코드와 테스트 가능성에 많은 관심이 있다. 그는 개발자들이 그들의 고객에게 지속적인 가치를 제공하려면 읽을 수 있는 코드를 작성하기 위해 노력해야 한다고 믿는다.

그는 테스트 주도 개발Test-driven Development 접근법과 훌륭한 소프트웨어 설계를 활성화하기 위해 페어 프로그래밍과 멘토링을 활용한다. 현재 자바 웹스택 과정을 가르치고 있으며 그루비와 스프링의 광신자이기도 하다. 또한, Packt에서 출간한 『Learning Spring Boot』(2014)와 『Spring Boot Cookbook』(2015) 리뷰팀에서 스프링 생태계에 최근 추가된 주요 항목을 다뤘다.

그의 블로그 http://geowarin.github.io와 트위터 계정 https://twitter.com/geowarin을 살펴보면 스프링과 자바스크립트 프로그래밍에 관한 뜨끈뜨끈한 소식들을 살펴볼 수 있다.

늦은 시간까지 집필할 수 있도록 응원해준 내 삶의 반려자 로리Laure, 내 프로젝트를 아낌없이 지원해준 Bi-SAM의 동료들에게 감사를 표한다.

역자 소개

옮긴이_ **김지헌**

자바를 주력으로 하는 개발자로, 인터넷에서는 허니몬honeymon이라는 이름으로 활동하고 있다. 다양한 취미활동(스포츠 클라이밍, 로드 레이싱, 스쿠버다이빙 등)을 하며 유유자적한 생활을 즐긴다. Spring Boot 1.2 버전의 레퍼런스 문서를 번역한 것이 인연이 되어 이 책을 번역하게 됐다.

스프링을 기반으로 개발한 지 꽤 오래됐다. 하지만 여전히 스프링으로 개발할 때 애플리케이션에 필요한 기능들에 대한 의존성을 추가하고 그에 대해 구성하는 일은 녹록지 않다. 이런 어려움을 해소할 수 있는 좋은 도구인 스프링부트Spring Boot도 나온 지 꽤 됐다(2014년 4월에 버전 1.0.0이 모습을 드러냈다. https://www.infoq.com/news/2014/04/spring-boot-goes-ga). 필요한 기능에 대한 의존성을 자동구성을 통해 추가하는 것만으로 애플리케이션을 컨테이너를 내장한 단독 실행 가능한 JAR 형태로 손쉽게 배포할 수 있게 됐다.

저자가 이 책이 집필했을 때는 1.2.1 버전이었는데, 번역하는 사이에 1.2.5에서 1.4.0까지 나오며 빠른 속도로 버전업되고 있다(책 내용 중에 일부는 Deprecate되거나 변경된 것들이 있을 것이다). 스프링부트는 앞으로도 꾸준하게 버전을 업그레이드하며 다양한 기능을 지원할 것이다. 그 근간에는 `@AutoConfiguraton`과 `@Conditional` 애너테이션을 통해 빈이나 클래스의 유무로 자동구성되어 별다른 설정을 하지 않아도 되는 편이성이 있다. 그러면서 `~Context.xml`로 정의되던 XML 설정이 `~Configuration.java`의 자바 클래스로 설정할 수 있게 됐다. XML보다 프로그래밍적인 처리를 하면서 조건별로 구성할 수 있고, 정적 코드의 장점을 이용해 리팩토링도 가능한 장점이 있다.

스프링부트를 제대로 사용하려면 레퍼런스 문서, 사용하려는 기능과 관련된 `Configuration`과 `Properties`를 살펴보는 것이 좋다. `Properties` 클래스의 경우 애플리케이션의 `application.properties`(or `yml`)를 통해 그 값을 주입할 수 있으며, 여기에 사용되는 키는 애플리케이션 실행 시 인자로 전달해 오버라이드할 수 있다. 이 특징을 통해 필요에 따라 애플리케이션이 실행되는 환경에 맞춰 실행 인자를 변경할 수 있는 유연성도 제공한다.

이 책에서는 실제 DB와 연동해 데이터를 CRUD하지는 않고, `Repository` 클래스 내부의 맵 객체를 이용해 데이터를 제어한다. 나중에 Spring Data JPA에 대한 의존성을 추가하고 DB 설정만 하면 `JpaRepository`를 구현하여 손쉽게 DB 연동을 할 수 있다. 이처럼 스프링부트에서는 기존보다 기능을 추가하고 확장하는 것이 유연하고 정말 쉬워졌다.

스프링부트는 가벼운 형태의 웹 애플리케이션으로, 컨테이너에 담아 실행하는 데 적합하며 리눅스 시스템에서 서비스로 등록하고 실행하는 것도 가능하다. 이 가능성은 분명 강점으로 부각될 것이다. 스프링부트가 어떻게 동작하는지 이해하고 나서 많이 다뤄보고 능숙해지면 이만한 도구가 자바 쪽에서 있을까 하는 생각이 들 정도로 좋은 도구다. 사실 스프링부트를 제대로 사용하려면 스프링 프레임워크에 대한 이해가 필요하다. 스프링부트가 동작하는 근간에는 스프링 프레임워크가 깔려 있다. 내장형 컨테이너 위에 스프링부트 개발팀에 의해 기본 구성된 자동 설정들이 조율되면서 개발자 입맛에 맞는 기능들을 선별해 쉽게 웹 애플리케이션을 구성하고 구동할 수 있다.

이 책은 Spring MVC를 기준으로 웹 애플리케이션을 구성하고 아이디어를 추가하면서 기능을 하나하나 구현해 나간다. 그리고 그것을 클라우드 플랫폼에 배포하면서 웹 애플리케이션의 전체적인 개발과 배포 과정을 거치게 된다. 사실 나는 젠킨스를 통해 WAS에 배포해 운영하는 것까지 해본 게 고작이었다. 최근에서야 클라우드 환경에서 배포하고 운영하는 경험을 하고 있는데, 이것은 또 다른 세상이다.

나는 "아는 만큼 보인다."라는 말을 좋아한다. 내가 아는 것이 늘어날수록 볼 수 있는 것들이 늘어나고 그러면 다시 내가 알아야 하는 것들이 늘어난다. 너무나 부족한 실력으로 번역한 책이지만 번역하는 과정에서 새로운 것들을 많이 알 수 있었다(지금은 많이 잊어버렸지만 이 책이 나오면 필요할 때 꺼내 볼 수 있으니 그것은 그것대로 보람찬 일이다). 그

리고 내 부족함을 새삼 느낄 수 있는 좋은 계기였다. 이 책을 읽는 분들에게도 좋은 자극이 되길 바란다.

궁금한 점이나 잘못된 점은 ihoneymon@gmail.com으로 알려주시면 나중에 치맥 제공하겠습니다. +_+)b

P.S. 난 노는 게 좋다. -_-);;

웹 개발자로서 나는 새로운 것을 만들어내는 것을 좋아하며, 이것들을 온라인에 재빨리 배포하고 다음 생각으로 넘어간다. 세상에 있는 애플리케이션들은 서로가 연결되어 있으며, 필요하면 소셜미디어를 통해 제품과 복잡한 시스템을 홍보하며 사용을 유도해 사용자들에게 엄청난 가치를 제공할 수 있다.

최근까지 자바 개발자는 이런 세상과 동떨어져 있었지만, 스프링부트Spring Boot의 출현과 클라우드 플랫폼의 저변 확대에 힘입어 별도의 지출 없이 매우 놀라운 애플리케이션을 만들어내고 유사 이래 모든 것을 만들어낼 수 있게 됐다.

이 책에서는 아무것도 없는 상태에서 쓸 만한 웹 애플리케이션을 구축해간다. 이 애플리케이션은 국제화, 폼 검증, 분산 세션, 캐시, 소셜 로그인, 멀티스레드 프로그래밍과 좀 더 다양하고 멋진 기능이 있으며 거기에 더해, 애플리케이션에 대한 테스트도 완벽할 것이다.

이 책을 다 읽어갈 즈음이면 여러분은 웹에서 사용할 수 있는 출시 가능한 애플리케이션을 가지게 될 것이다. 더는 시간을 허비하지 말고 코드를 작성해 보자.

이 책에서 다루는 것들

1장 스프링 웹 애플리케이션 설정하기는 스프링부트를 이용해 간결하게 시작한다. 여기서 이용하는 STSSpring Tool Sutie와 깃Git은 생산성을 높여주고, 스프링부트 뒤에서 마법을 부리듯 애플리케이션을 빠르게 구축하는 데 도움을 준다.

2장 MVC 구조 익히기에서는 간단한 트위터Twitter 검색 엔진을 만드는 과정을 소개한다. 이 과정에서 Spring MVC의 기본과 웹 구조에 대한 이론을 다룬다.

3장 폼과 복잡한 URL 매핑 다루기에서는 사용자 프로필 폼을 어떻게 만드는지를 이해할 수 있게 돕는다. 또한, 클라이언트뿐만 아니라 서버에 전달된 데이터의 유효성을

검증하는 방법과 애플리케이션에서 다른 언어를 사용할 수 있게 만드는 방법 등을 다룬다.

4장 파일 업로드와 오류 다루기에서는 프로필 폼에서 파일을 업로드하는 과정을 안내한다. 발생한 오류를 Spring MVC에서 처리하는 방법과 정의한 오류 페이지를 사용자에게 표시하는 법을 보여준다.

5장 RESTful 애플리케이션 만들기에서는 RESTful 아키텍처 이론을 설명한다. 사용자 관리 API를 만들어 HTTP 호출로 접근하고 API를 설계하는 데 도움이 되는 도구를 살펴본다. 또한, 어떻게 하면 손쉽게 문서를 작성할지 의견을 나눈다.

6장 애플리케이션 보안에서는 애플리케이션 보안에 관해 설명한다. RESTful API에 대한 기본 HTTP 인증 보안과 웹 페이지에 앞서 로그인 페이지를 노출하는 방법을 다룬다. 그리고 어떻게 트위터를 통해 로그인하고 레디스Redis 서버에 세션을 저장해 애플리케이션을 확장하는지에 대해 설명한다.

7장 운에 맡기지 않기 - 단위 테스트와 인수 테스트에서는 애플리케이션 테스트에 대해 살펴본다. 테스트와 TDD에 대해 알아보고, 컨트롤러에 대한 단위 테스트를 어떻게 작성하는지, 라이브러리를 사용해 통합end-to-end 테스트를 어떻게 설계하는지 알아본다. 마지막으로 그루비로 어떻게 생산성을 높이고 테스트 가독성을 향상시킬지를 살펴본다.

8장 요구사항 최적화에서는 애플리케이션 최적화에 대해 살펴본다. 이 장에서는 캐시-컨트롤Cache-control과 Gzipping을 사용하는 방법을 다룬다. 또한, 트위터 검색결과를 레디스의 인-메모리In-memory에서 어떻게 사용하는지 살펴보고 멀티스레드 검색을 어떤 방법으로 하는지도 보여준다. 이에 더해 ETags를 구현하는 방법과 웹소켓WebSockets 사용법도 다룬다.

9장 웹 애플리케이션을 클라우드 환경에 배포하기에서는 애플리케이션을 배포하는 방법을 소개한다. 각기 다른 PaaS 솔루션을 비교하면서 차이점을 살펴보고, 클라우드 파운드리Cloud Foundry와 히로쿠Heroku에 어떻게 애플리케이션을 배포하는지 설명한다.

10장 스프링 웹을 넘어서에서는 전체적인 스프링 생태계의 모습과 현재 웹 애플리케이션이 어떻게 만들어지며 앞으로 어떻게 변화되어 갈지를 논의한다.

이 책을 읽는 데 필요한 것들

최첨단 웹 애플리케이션을 만들어가는 동안 설치해야 할 것은 많지 않다. 우선 애플리케이션을 구축하는 데 Java 8이 필요하다. 강제사항은 아니지만, 프로젝트의 변경이력 관리에는 깃을 사용하길 권한다. 이는 히로쿠에 애플리케이션을 배포할 경우 필요하다. 더 나아가 작업 내용을 쉽게 백업할 수 있고, 과거 이력과 차이점을 비교하면서 점점 발전하는 코드를 볼 수 있다. 이 두 가지 혜택은 첫 장에서 제공하는 깃을 사용하면 얻을 수 있다.

또한, 자신에게 맞는 통합개발환경IDE, Integration Development Environment을 사용하길 권한다. 여기서는 STS(무료)와 인텔리제이IntelliJ IDEA(한 달간 체험판 사용 가능)를 사용한다. OS X를 사용한다면 홈브류Homebrew(http://brew.sh/)를 사용해 보길 바란다. 이 패키지 매니저는 이 책에서 사용하는 도구들을 통합해 설치할 수 있다.[01]

01 역자주_데비안 계열 리눅스에서는 apt-get 패키지 관리자를 사용할 수 있고, 레드햇 계열 리눅스에서는 yum 패키지 관리자를 사용할 수 있다.

이 책의 대상 독자

이 책은 자신의 웹 개발 기술이 향상되길 바라는, 스프링 프로그래밍에 관한 기본적인 지식을 가진 개발자에게 적합하다. 또한, 스프링 프레임워크에 대한 사전지식이 필요하다.

참고

- NOTE 경고나 중요 내용을 보여준다.
- TIP 팁이나 요령을 보여준다.

예제코드

이 책의 예제코드는 다음에서 다운로드할 수 있다.

- **원서 예제 코드** https://github.com/Mastering-Spring-MVC-4/mastering-spring-mvc4 (단축 URL https://goo.gl/a1qDwF)
- **번역 예제 코드** https://github.com/ihoneymon/mastering-spring-mvc-4 (단축 URL https://goo.gl/zRDTNm)

차례

chapter 1

스프링 웹 애플리케이션 설정하기

이번 장에서는 바로 코드를 작성해 보고 웹 애플리케이션을 설정하며, 이 설정을 기반으로 이 책의 나머지 부분에 대한 작업을 진행한다. 이 책에서는 스프링부트Spring Boot의 자동구성Autoconfiguration 기능을 활용해 선언문이나 설정 파일 없이 애플리케이션을 빌드하는데, 1장에서는 스프링부트가 어떻게 동작하고 어떻게 설정하는지를 큰 그림을 그리며 설명하겠다.

스프링을 시작하는 방법에는 다음 4가지가 있다.

- Spring Tool Suite 이용해 스타터 코드 생성
- IntelliJ를 사용해 스프링부트 지원 기능 확인
- 스프링 웹사이트(http://start.spring.io)에서 설정 가능한 ZIP 파일 내려받기
- `curl` 명령어를 사용해 http://start.spring.io와 같은 결과 얻기

이 책을 보는 동안 그레이들Gradle과 Java 8을 사용하는 것을 두려워하지 마라. 여러분이 여전히 메이븐과 이전 버전의 자바로 작업하고 있더라도 사용하는 환경에서 쉽게 작업할 방법을 찾을 것이라 확신한다. 많은 공식 스프링 튜토리얼이 그레이들 빌드와 메이븐 빌드로 구성되어 있으며, 메이븐 예제도 쉽게 찾을 수 있다. 스프링 4는 Java 8에 완벽하게 최적화됐는데, 코드를 단순하게 만들 수 있는 람다Lambda의 장점을 활용하지 않는 것은 부끄러운 일이다.

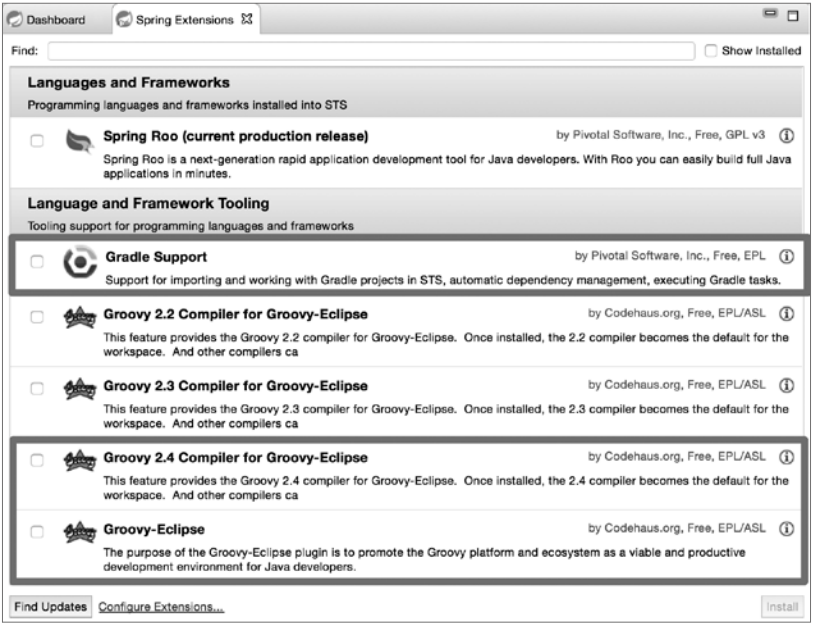
여러분에게 종종 깃 명령어를 보여줄 것이다. 안정 상태가 됐을 때 진행 상황과 커밋을 살펴보는 것이 좋은데, 이것은 이 책에서 제공하는 소스 코드와 작업한 내용을 비교하기 쉽게 한다. 9장 웹 애플리케이션을 클라우드 환경에 배포하기에서 히로쿠에 애플리케이션을 배포할 때 초기 시작 단계부터 깃으로 코드의 변경이력을 관리하기를 권장한다. 이번 장 끝부분에서는 깃을 어떻게 시작할지 소개하겠다.

1.1 STS로 시작하기

스프링을 시작하는 가장 좋은 방법 중 하나는 스프링 커뮤니티에서 제공하는 STS를 이용해 수많은 튜토리얼과 스타터 프로젝트를 살펴보는 것이다. STS는 다양한 스프링 프로젝트를 작업할 수 있도록 이클립스에 맞춤 설계했으며, 그루비Groovy와 그레이들도 잘 동작한다. 다른 IDE로 작업하고 있더라도 STS를 사용해 보길 바란다. STS에서 열리는 'Getting Started' 프로젝트들을 살펴보면 스프링의 방대한 생태계를 탐험하는 기회를 얻을 수 있다.

먼저 https://spring.io/tools/sts/all에 방문해 STS 최신 배포본을 다운로드한다. 첫 스프링부트 프로젝트를 생성하기에 앞서서 STS에서 사용할 그레이들 플러그인을 설치해야 한다. 대시보드에서 [Manage IDE Extension] 버튼을 찾으면 Language and framework Tooling 영역에서 Gradle Support 소프트웨어를 다운로드할 수 있다. 또한, 다음 그림처럼 Goovy-Eclipse와 Groovy 2.4 compiler를 설치하길 권장한다. 이는 뒤에서 다룰 Geb을 이용한 인수 테스트를 설정하는 데 필요하다.

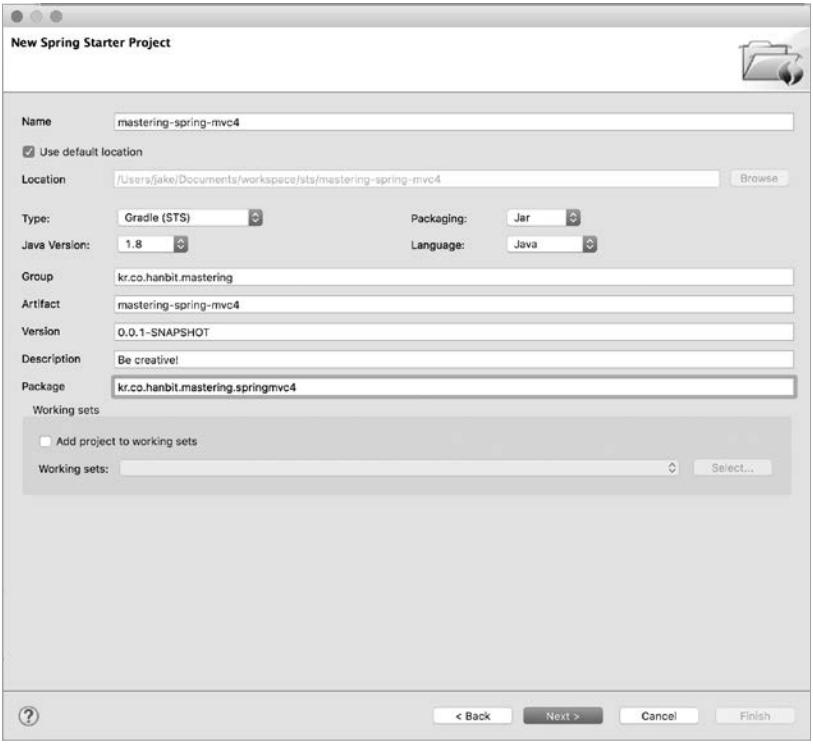
그림 1-1 그레이들 플러그인 설치



이제 시작하는 데 필요한 두 가지 중요한 선택을 해야 한다.

첫 번째로, 메뉴에서 [File → New → Spring Starter Project]로 이동하면 [그림 1-2]와 같은 화면을 볼 수 있는데, 이는 http://start.spring.io와 같은 기능이 IDE에 내장된 것이다.

그림 1-2 스프링 스타터 프로젝트 선택



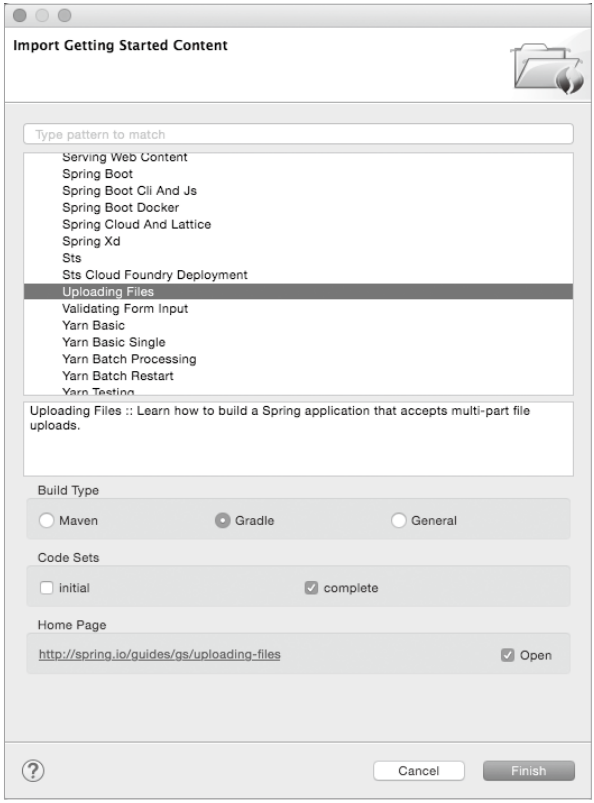
http://spring.io에서 접근할 수 있는 모든 튜토리얼을 살펴보려면 상단 메뉴에서 [File → New → Import Getting Started Content]를 선택하면 된다.

NOTE

튜토리얼을 따라하면서 스타터 코드를 확인하거나 바로 전체 코드를 얻을 수 있다.

Import Getting Started Content에서 다양하고 흥미로운 콘텐츠를 볼 수 있는데, 이를 탐구해 보는 것도 굉장히 도움이 될 수 있다. 다양한 기술을 스프링으로 통합하는 과정이 매우 흥미로울 것이다.

그림 1-3 접근 가능한 튜토리얼 보기



[그림 1-2]처럼 웹 프로젝트를 생성한다. 프로젝트는 그레이들 애플리케이션으로, JAR로 제작되고 Java 8을 사용한다. 이 책에서 사용할 설정은 다음과 같다.

표 1-1 이 책에서 사용할 설정

속성	값
이름(Name)	mastering-spring-mvc4
유형(Type)	Gradle project
압축 형태(Packaging)	JAR
자바 버전(Java version)	1.8
언어(Language)	Java

속성	값
그룹(Group)	kr.co.hanbit.mastering
아티팩트(Artifact)	mastering-spring-mvc4
버전(Version)	0.0.1-SNAPSHOT
설명(Description)	Be creative!
패키지(Package)	kr.co.hanbit.mastering.springmvc4

[그림 1-2]를 보면 자바 버전(Java version) 항목에서 사용할 스프링부트의 버전을 물어봤는데, 여기서 선택한 버전에 따라 프로젝트에 필요한 의존성들이 추가된다. 이 책을 집필하는 시점에서 스프링부트 최신버전은 1.2.5이나[01] 계속 업데이트가 되고 있으므로 수시로 최신 배포본을 점검하길 바란다(많은 변화 중에서 DevTools의 변화를 손꼽을 수 있다. 좀 더 자세한 정보는 https://spring.io/blog/2015/06/17/devtools-in-spring-boot-1-3을 참고하길 바란다).

설정 화면 하단을 보면 다양한 부트 스타터 라이브러리들을 선택하는 체크박스가 있다. 여기서 선택한 의존성들은 빌드 파일에 추가되어 다양한 스프링 프로젝트를 위한 자동구성을 제공한다. 현재는 Spring MVC에만 집중해야 하니 'Web'에만 체크하자.

TIP 웹 애플리케이션을 JAR로 배포한다?

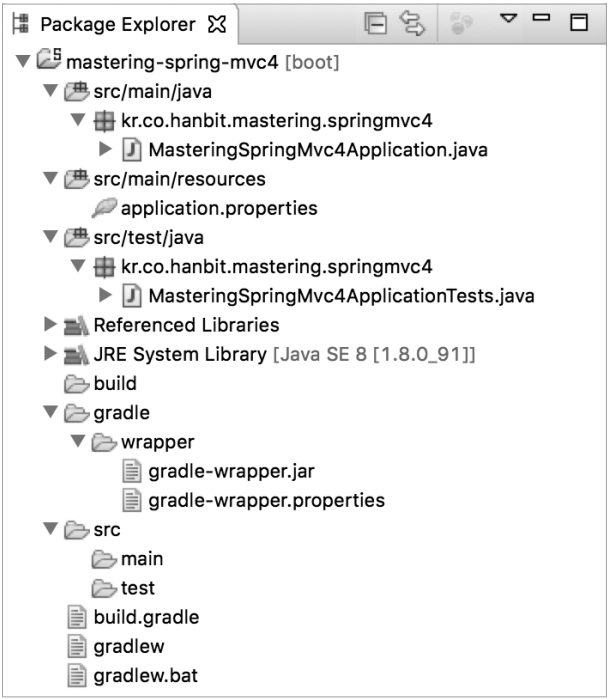
웹 애플리케이션을 JAR 파일로 엮는다(package)는 것에서 기묘함을 느낄 것이다. 일반적으로 패키징은 WAR 파일을 사용하는데, 항상 요구되는 상황은 아니다. 기본으로 스프링부트는 Fat JAR 파일을 생성하고 그 안에는 애플리케이션의 모든 의존성을 포함하고 있으며 `java -jar`를 사용해 웹 서버를 시작할 수 있는 편이성을 제공한다.

이 책에서도 애플리케이션은 JAR 파일로 패키징된다. WAR 파일을 만들고 싶다면 http://spring.io/guides/gs/convert-jar-to-war/ 페이지를 참고하기 바란다.

01 역자주_2016년 5월 17일 기준으로 스프링부트 최신 버전은 1.4.0이다.

아직 [Finish]를 클릭하지 않았다면 클릭하고서 만들어진 프로젝트 구조를 살펴보자.

그림 1-4 프로젝트 구조



우선 메인 클래스 MasteringSpringMvc4Application과 이를 테스트하는 MasteringSpringMvc4ApplicationTest를 볼 수 있다. 그리고 비어있는 두 개의 폴더 static과 template을 볼 수 있는데, 이곳에는 정적 웹(asset, 이미지와 스타일 시트, 그 밖의 것들)과 템플릿(jsp, freemarker, Thymeleaf 등)을 넣는다. 마지막으로 비어있는 application.properties 파일이 있다. 이 파일은 기본 스프링부트 설정 파일로, 이번 장에서 이 파일이 얼마나 다루기 쉬운지와 스프링부트에서 어떻게 사용하는지를 볼 수 있다. build.gradle은 빌드 파일로, 나중에 자세히 살펴보겠다.

진행할 준비가 됐다면 애플리케이션의 메인 메서드를 실행한다. 메인 메서드를 실행하면 웹 서버가 작동한다. 애플리케이션의 메인 메서드로 이동해 클래스 위에서 마우스 우클릭하면 뜨는 툴바에서 [Run as → Spring Application]을 클릭하거나 툴바에서 녹색 플레이 버튼을 클릭한다. http://localhost:8080으로 이동하면 오류가 발생하는데, 걱정하지 말고 계속 진행한다.

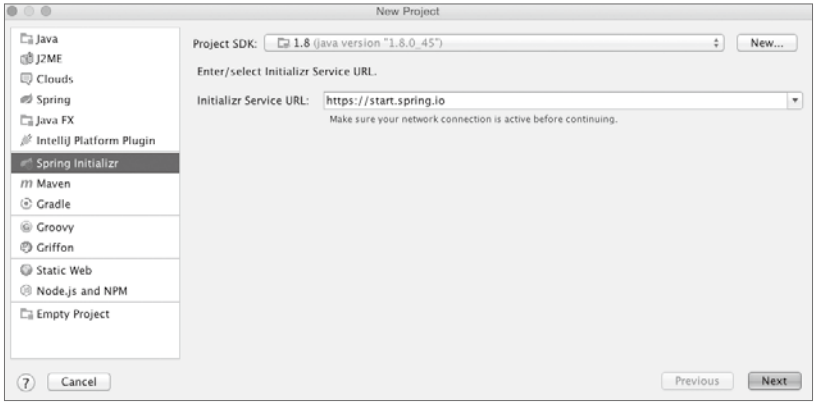
이제 STS 없이 동일한 프로젝트를 어떻게 생성하는지 살펴보자. 이 파일들은 STS에서 생성한 파일과 모두 동일함을 알 수 있을 것이다.

1.2 IntelliJ로 시작하기

인텔리제이는 자바 개발자 사이에서 매우 인기있는 도구로, 필자 역시 몇 년 전부터 젯브레인JetBrains의 라이선스를 구매해 사용하고 있다. 인텔리제이 역시 매우 빠르게 스프링부트 프로젝트를 만드는 방법을 제공한다.

먼저 'New project' 메뉴로 이동해 'Spring Initializer' 프로젝트 유형을 선택한다. STS와 동일한 선택항목들이 주어지므로 세부적인 설정은 앞서 설명한 부분을 참조한다.

그림 1-5 프로젝트 유형 선택



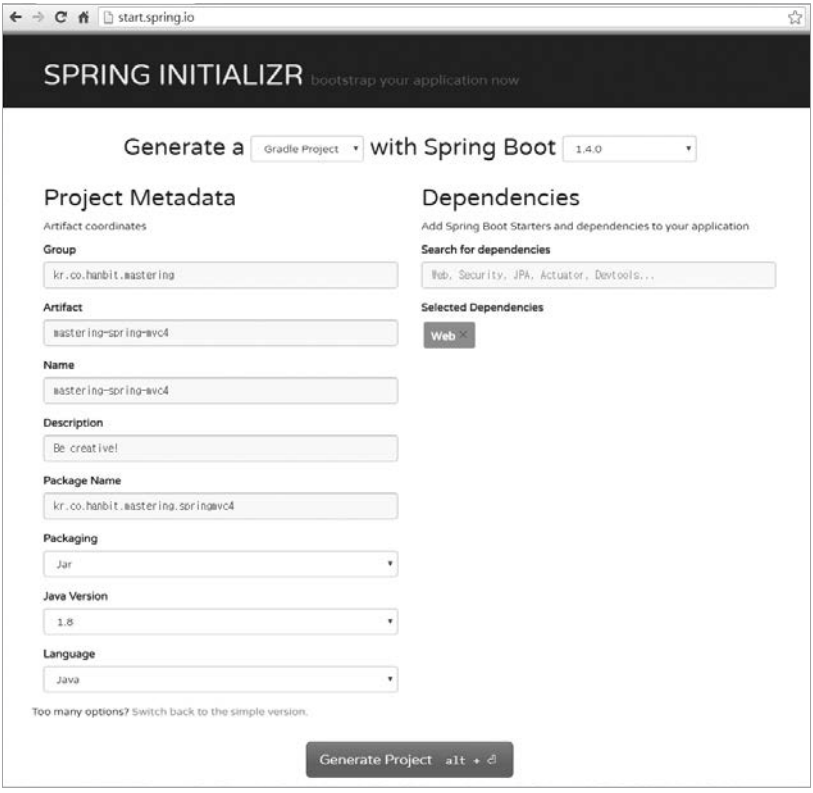
TIP

인텔리제이 안에서 '그레이들 프로젝트 불러오기'를 해야 할 때가 있다. 이럴 때는 그레이들 래퍼Gradle wrapper를 먼저 생성하는 것이 좋다(자세한 내용은 1.4.1 그레이들 빌드 참고). 필요하다면 build.gradle 파일을 열어 프로젝트를 다시 불러올 수 있다.

1.3 start.spring.io로 시작하기

http://start.spring.io로 이동해 Spring Initializer를 시작하자(이 놀라운 웹 사이트 뒤에 있는 부트스트랩Bootstrap같은 시스템을 잘 알고 있어야 한다). 앞서 언급한 링크로 이동하면 다음 화면을 볼 수 있다.

그림 1-6 프로젝트 설정



STS에서 본 것과 같은 항목들을 볼 수 있고, [Generate Project] 버튼을 클릭하면 스타터 프로젝트가 포함된 ZIP 파일을 다운로드한다.

1.3.1 커맨드라인으로 시작하기

콘솔 사용을 좋아한다면 curl http://start.spring.io을 사용할 수 있다. curl 요청을 어떻게 구성하는지 살펴보자. 예를 들어, 앞의 내용과 같은 프로젝트는 다음 명령어를 입력하면 쉽게 생성된다.

```
$ curl http://start.spring.io/starter.tgz \
-d name=mastering-spring-mvc4 \
-d dependencies=web \
-d language=java \
-d JavaVersion=1.8 \
-d type=gradle-project \
-d packageName=kr.co.hanbit.mastering.springmvc4 \
-d packaging=jar \
-d baseDir=mastering-spring-mvc4 | tar -xzvf -

% Total % Received % Xferd  Average  Speed  Time  Time  Time  Current
                           Dload    Upload Total Spent Left  Speed
20    53332    20        10784    100 1  84    6455  110   0:00:08
0:00:01 0:00:076453
mastering-spring-mvc4/gradlew
mastering-spring-mvc4/
mastering-spring-mvc4/gradle/
mastering-spring-mvc4/gradle/wrapper/
mastering-spring-mvc4/src/
mastering-spring-mvc4/src/main/
mastering-spring-mvc4/src/main/java/
mastering-spring-mvc4/src/main/java/kr/
mastering-spring-mvc4/src/main/java/kr/co/
mastering-spring-mvc4/src/main/java/kr/co/hanbit/
mastering-spring-mvc4/src/main/java/kr/co/hanbit/mastering/
mastering-spring-mvc4/src/main/java/kr/co/hanbit/mastering/springmvc4/
mastering-spring-mvc4/src/main/resources/
mastering-spring-mvc4/src/main/resources/static/
```

```
mastering-spring-mvc4/src/main/resources/templates/
mastering-spring-mvc4/src/test/
mastering-spring-mvc4/src/test/java/
mastering-spring-mvc4/src/test/java/kr/
mastering-spring-mvc4/src/test/java/kr/co/
mastering-spring-mvc4/src/test/java/kr/co/hanbit/
mastering-spring-mvc4/src/test/java/kr/co/hanbit/mastering/
mastering-spring-mvc4/src/test/java/kr/co/hanbit/mastering/springmvc4/
mastering-spring-mvc4/build.gradle
mastering-spring-mvc4/gradle/wrapper/gradle-wrapper.jar
mastering-spring-mvc4/gradle/wrapper/gradle-wrapper.properties
mastering-spring-mvc4/gradlew.bat
mastering-springmvc4/
src/main/java/kr/co/hanbit/mastering/springmvc4/MasteringSpringMvc4Ap
plication.java
mastering-spring-mvc4/src/main/resources/application.properties
mastering-springmvc4/
src/test/java/kr/co//hanbit/mastering/springmvc4/MasteringSpringMvc4Applicatio
nTests.java
```

이제 콘솔에서 벗어나지 않고도 스프링을 시작할 수 있게 됐다!

> TIP
>
> 앞의 명령어를 alias로 만들면 매우 빠르게 스프링 애플리케이션 프로토타입을 만들 수 있다.

1.4 시작하기

애플리케이션은 준비됐다. 어떻게 작성하는지 살펴보기 전에 앞에서 한 작업을 깃으로 저장하자. 깃에 대해 잘 모른다면 두 가지 튜토리얼을 따라해 보기를 권한다.

- https://try.github.io 대화식 튜토리얼로, 기본 깃 명령어를 단계별로 배울 수 있다.
- http://pcottle.github.io/learnGitBranching 깃을 트리와 비슷하게 구조화해 기초부터 고급까지 깃으로 가능한 것들을 보여주는 뛰어난 대화식 시각화 튜토리얼이다.

TIP 깃 설치

윈도우에서는 깃 배쉬Git bash를 설치해야 하는데, 이와 관련된 설치 파일은 https://git-for-windows.github.io/에서 다운로드할 수 있다. OS X에서는 홈브류를 사용하고 있다면 이미 깃이 설치되어 있지만, 그렇지 않다면 `brew install git` 명령을 내리면 된다. 미심쩍을 때는 https://git-scm.com/book/en/v2/Getting-Started-Installing-Git 문서를 살펴보길 바란다.

깃으로 변경이력을 관리하려면 콘솔에서 다음 명령어를 입력한다.

```
$ cd app
$ git init
```

인텔리제이를 사용 중이라면 .idea와 *.iml 생성 파일을 무시하고, 이클립스를 사용 중이라면 .classpath와 .settings 폴더를 커밋해야 한다. 어느 경우든지 .gradle과 build 폴더를 무시해야 한다. 그다음 .gitignore 파일을 생성하고 다음 문장을 입력한다.

```
# IntelliJ project files
.idea
*.iml

# gradle
.gradle
Build
```

이제 깃에 나머지 파일들을 모두 추가할 수 있다.

```
$ git add .
$ git commit -m "Generated with curl start.spring.io"
[master (root-commit) 1d0aeb3] Generated with curl start.spring.io
9 files changed, 337 insertions(+)
create mode 100644 .gitignore
create mode 100644 build.gradle
create mode 100644 gradle/wrapper/gradle-wrapper.jar
```

```
create mode 100644 gradle/wrapper/gradle-wrapper.properties
create mode 100755 gradlew
create mode 100644 gradlew.bat
create mode 100644
src/main/java/kr/co/hanbit/mastering/springmvc4/MasteringSpringMvc4Applica
tion.java
create mode 100644 src/main/resources/application.properties
create mode 100644
src/test/java/kr/co/hanbit/mastering/springmvc4/MasteringSpringMvc4Applica
tionTests.java
```

1.4.1 그레이들 빌드

그레이들에 익숙하지 않다면 메이븐의 후계자로 생각하자. 그레이들은 최신 빌드 도구로, 메이븐처럼 자바 애플리케이션 구조를 관례convention로 사용한다. 이 책의 프로젝트 소스를 살펴보면 src/main/java 또는 웹 앱에서는 src/main/webapp 등이 이에 해당한다.

메이븐을 좋아하지 않는다면 다양한 빌드 태스크를 제공하는 그레이들 플러그인으로 갈아탈 수 있다. 그레이들은 그루비 DSL도메인 정의 언어, Domain Specific Language로 빌드 태스크 추가 작성을 허용하는 매력적인 기능을 제공한다. 또한, 기본 라이브러리 파일들을 쉽게 조작하고 태스크 사이의 의존성을 정의하며 끊임없이 작업을 실행하게 할 수 있다.

> **TIP 그레이들 설치하기**
>
> OS X를 사용 중이라면 `brew install gradle` 명령으로 그레이들을 설치할 수 있다. 또는 다른 *Nix 계열 시스템이라면(Mac 포함) GVM(http://gvmtool.net)[02]으로 설치할 수 있다. 대안으로는 https://gradle.org/downloads에서 바이너리 배포본을 다운로드할 수 있다.

02 역자주_애플리케이션 설치 자동화 도구로, 현재는 SDKMAN(http://sdkman.io/)으로 이름이 변경됐다.

그레이들로 애플리케이션을 만들 때 가장 먼저 그레이들 래퍼를 생성한다. 그레이들 래퍼는 코드를 공유할 경우 애플리케이션을 동일한 버전의 그레이들로 빌드하게 지원하는 작은 스크립트로, 그레이들 래퍼를 생성하는 명령어는 `gradle wrapper`다.

```
$ gradle wrapper
:wrapper

BUILD SUCCESSFUL

Total time: 6.699 secs
```

새로 생성된 파일을 살펴보면 스크립트 두 개와 디렉터리 두 개를 볼 수 있다.

```
$ git status -s
 ?? .gradle/
 ?? gradle/
 ?? gradlew
 ?? gradlew.bat
```

.gradle 디렉터리에는 그레이들 바이너리들이 있는데, 변경이력 관리 항목에 이들을 커밋하고 싶지는 않을 것이다. 앞에서 살펴본 .gitignore 파일에서 build 디렉터리를 추가할 때 .gradle/ 디렉터리도 함께 추가해 무시하게 해두었으니 안심하고 추가하자.

```
$ git add .
$ git commit -m "Added Gradle wrapper"
```

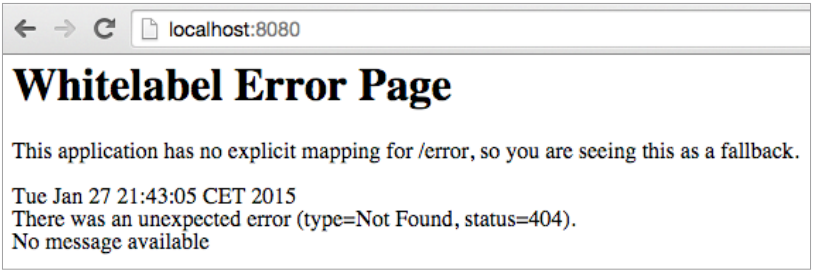
.gradle/ 디렉터리에는 바이너리들을 어떻게 얻을지에 대한 정보가 있으며, 다른 두 파일은 윈도우를 위한 배치 스크립트(gradlew.bat)와 다른 시스템을 위한 스크립트다. 또한, IDE에서 애플리케이션을 실행하는 대신 그레이들로 애플리케이션

을 실행할 수 있는데, 다음 명령어를 입력하면 애플리케이션에 있는 내장 톰캣이 실행된다.

```
$ ./gradlew bootrun
```

로그를 보면 8080 포트에서 서버가 실행된다는 것을 알 수 있다. 확인해 보자.

그림 1-7 톰캣 서버 실행 확인



실망하는 모습이 상상이 된다. 여러분이 만든 애플리케이션은 공개할 준비가 아직 안 됐다. 하지만 프로젝트가 구성되어 두 개의 파일에 의해 수행되는 것은 매우 인상적이다. 이 파일들을 한번 살펴보자. 먼저 그레이들 빌드 파일인 build.gradle을 보자.

```
buildscript {
    ext {
        springBootVersion = '1.3.4.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:
        ${springBootVersion}")
    }
}

apply plugin: 'java'
```

```
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'spring-boot'

jar {
    baseName = 'mastering-spring-mvc4'
    version = '0.0.1-SNAPSHOT'
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    compile('org.springframework.boot:spring-boot-starter-web')
    testCompile('org.springframework.boot:spring-boot-starter-test')
}

eclipse {
    classpath {
        containers.remove('org.eclipse.jdt.launching.JRE_CONTAINER')
        containers 'org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.
internal.debug.ui.launcher.StandardVMType/JavaSE-1.8'
    }
}

task wrapper(type: Wrapper) {
    gradleVersion = '2.13'
}
```

여기서 알 수 있는 것들은 무엇일까?

- 스프링부트 플러그인에 대한 의존성은 메이븐 중앙저장소에 있다.
- 자바 프로젝트이고, IDE 프로젝트 파일은 인텔리제이나 그레이들을 위해 그레이들이 생성한다.
- 애플리케이션은 JAR 파일을 생성한다.
- 프로젝트의 의존성은 메이븐 중앙저장소에서 호스팅된다.
- 클래스패스에는 출시를 위한 `spring-boot-starter-web`과 테스트를 위한 `spring-boot-startertest`가 포함되어 있다.

- 이클립스를 위한 추가 설정이 있다.
- 그레이들 래퍼의 버전은 2.13이다.

스프링부트 플러그인은 프로젝트를 실행하는 데 필요한 모든 의존성 라이브러리를 포함하는 Fat JAR를 생성할 것이다. 다음 명령어를 입력하고 빌드해 보자.

```
$ ./gradlew build
```

build/libs 디렉터리에서 JAR 파일을 찾을 수 있다. 이 디렉터리에는 파일이 두 개 있는데, Fat JAR라 불리는 mastering-spring-mvc4-0.0.1-SNAPSHOT.jar와 의존성이 포함되지 않은 mastering-spring-mvc4-0.0.1-SNAPSHOT.jar.original이다.

> **TIP 실행 가능한 JAR**
>
> 스프링부트의 강점 중 하나는 애플리케이션이 웹 서버를 포함한 모든 것을 내장하고 있어 손쉽게 JAR 파일을 재배포할 수 있다는 것이다. `java -jar mastering-spring-mvc4-0.0.1-SNAPSHOT.jar`를 실행하면 8080 포트로 톰캣이 실행되고, 이것만으로 배포가 끝난다. 이것은 클라우드나 출시를 위한 배포를 매우 쉽게 해준다.

주 의존성은 `spring-boot-starter-web`이다. 스프링부트는 일반적인 의존성과 스프링 설정을 자동으로 애플리케이션에 적용하는 상당히 많은 스타터를 제공한다.

`spring-boot-starter-web`은 `tomcat-embed`와 Spring MVC에 대한 의존성이 있다. 게다가 일반적으로 사용하는 Spring MVC 설정과 '/' 루트 패스에 대한 디스패처 리스닝Dispatcher Listening, 앞에서 보았던 404 페이지에 대한 오류 제어, 기본적인 뷰 리졸버ViewResolver 설정 등을 제공한다.

이와 관련된 자세한 사항은 뒤에서 살펴보고, 먼저 다음 내용을 보자.