퍼펙트 Perfect Ruby on Rails 루비 온 레일즈

Ruby on Rails 4 Applicaition Programming by Yoshihiro Yamada

퍼펙트 루비 온 레일즈
Perfect Ruby on Rails

초판 1쇄 발행 2015년 11월 30일

지은이 야마다 요시히로
옮긴이 윤인성
펴낸이 장성두
펴낸곳 제이펍

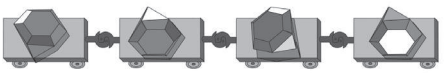
출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 문발로 141 뮤즈빌딩 403호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **이메일** jeipub@gmail.com

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

ISBN 979-11-85890-32-6 (93000)
값 34,000원

퍼펙트 루비 온 레일즈

Perfect Ruby on Rails



야마다 요시히로 지음 | 윤인성 옮김

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자, 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다. 본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 본문 중 일본 내의 실정에만 국한되어 있는 내용이나 그림은 일부를 삭제하거나 국내 실정에 맞도록 변경하였으니 참고 바랍니다.
- 최신 Rails 버전에서는 본문에 등장하는 Appname :: Application 대신에 Rails.application을 사용합니다. 설치된 Rails 버전과 소스 코드가 충돌하는 경우, 반드시 해당 내용을 확인하시기 바랍니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락해 주시기 바랍니다.
  - 옮긴이: rintiantta@naver.com
  - 출판사: jeipub@gmail.com

차례

옮긴이 머리말

일단, 이 책은 Ruby에 대한 기본 설명을 전혀 하지 않습니다. 따라서 Ruby와 관련된 내용을 알고 있어야만 이해할 수 있다는 것을 먼저 말씀드립니다.

원래 2011년에 타 출판사에서 웹 관련 시리즈를 기획하면서 클라이언트 개발과 서버 개발을 모두 다루기로 했습니다. 처음에는 클라이언트를 자바스크립트와 제이쿼리, 서버 개발을 Ruby와 Rails로 다뤄보면 어떨까 생각했었지만, Rails는 책이 팔리지 않는다는 이유로 거부되어 다른 책을 집필하게 되었는데요. 그때만 해도 우리나라에서 Rails에 대한 관심은 거의 없다시피 했습니다. 그리고 이것은 2015년 11월 현재 기준으로 우리나라에 Rails 관련 책이 단 한 권도 없다는 것으로 다시금 확인할 수 있습니다.

그러나 최근에는 스타트업을 중심으로 Rails 사용 빈도가 조금씩 늘어나고 있습니다. 또한, '멋쟁이 사자처럼'과 같은 대학생 모임에서도 Rails를 다루기 시작하면서 학생들 사이에서는 많이 알려지는 추세입니다. 이에 발맞춰 마침 시기적절하게 제이펍에서 Rails 책 번역을 맡게 되었습니다.

Rails는 생산 속도가 굉장히 빠르므로 스타트업의 초기 단계에서 프로토타입과 서비스를 만들 때 굉장히 유용합니다. 하지만 성능이 썩 좋지 않다는 말도 나오는데요. 이는 어디까지나 트위터와 같이 성공한 대규모 서비스에서나 해당하는 얘기지, 일반적인 서비스를 개발할 때는 아무 문제 없으리라 생각합니다.

* * *

역자 입장에서는 이 책의 대상 독자는 다른 웹 개발 공부를 한 번이라도 해본 분이었으면 합니다. Rails는 기능이 정말 많은 프레임워크라 아주 많은 기능과 규칙이 서로 맞물리는데요. 이런 이유로 Rails를 첫 웹 개발 프레임워크로 공부해버리면, 흐름을 파악하지 못하고 규칙만 외우다 끝날 수 있기 때문입니다.

또한, 웹 개발 공부를 조금이라도 해본 주변 사람들에게는 반드시 Rails 공부도 해볼 것을

추천하는데요. Rails는 액티브 레코드(Active Record)라는 기능으로 데이터베이스 연결과 관련된, 설계가 정말 아름다운 프레임워크입니다. 그러니 설령 Rails와 관련 없는 웹 개발자라고 해도 한번쯤은 살펴보시면 좋겠습니다.

책에 대해서 간단하게 소개하자면, 이 책은 Rails와 관련된 전체적인 기능을 다루는 책입니다. 책을 진행하면서 무엇을 만들어 나간다기보다는, 무엇을 만들 때 사용할 수 있는 기능들을 하나하나 알려줍니다. 사실, Rails 자체에 기능이 너무 많으므로 이는 당연한 진행 방식입니다. 그리고 그중에서 Rails의 핵심이라고 부르는 액티브 레코드에 대해서는 전체 중 약 20% 정도의 분량을 할애하여 다루고 있습니다.

* * *

참고로, Rails는 항상 처음 설치할 때 버전과 설정 등의 문제가 굉장히 많이 발생하므로 많은 사람들이 시작하는 것부터가 너무 어렵다고 말합니다. 특히, 과거(현재도 다소 그렇지만) 윈도우즈 버전에서 Rails를 설치할 때는 여러 가지 문제가 많을 가능성이 높아 "Rails를 하려면 맥이 꼭 있어야 해."라는 우스갯소리까지 나올 정도였습니다.

그러나 인내심을 갖고 한번 설치하면 정말 재미있게 개발할 수 있는 것이 바로 Rails입니다. 책을 진행하면서 설치가 잘 이루어지지 않는다면 역자 블로그(http://rintiantta.blog.me)에 문의를 남겨주세요. 원래 번역서와 관련된 질문은 따로 받지 않으나, 설치와 관련해서는 가능한 한 답변과 정리를 해드리고자 합니다.

번역을 맡겨 주시고 진행에 도움을 주신 제이펍 장성두 실장님과 현지환 대리님, 책을 편집해주신 이주원 님께 감사의 말씀을 드리며, 베타리딩에 참가해주신 모든 분께도 감사드립니다.

2015년 11월

역자 **윤인성**

베타리더 후기

김정훈(NBT)

Rails를 새로 접하시는 분, 기존에 Rails를 접하셨던 분 모두에게 친절한 설명이 가득한 Rails 도서가 나왔네요. 프론트엔드부터 백엔드까지 Rails로 가능한 모든 기능에 관한 설명이 들어 있습니다. 이 책 한 권이면 다른 레퍼런스가 필요 없을 정도로 충실하네요. 이 책을 통해 많은 분이 Ruby와 Rails의 아름다움에 흠뻑 빠질 수 있었으면 좋겠습니다.

박정춘(Jobplanet)

Rails를 전반적으로 다룬 책입니다. Rails는 다른 프로그래밍 언어에 경험이 있는 개발자라면 가이드 문서와 API 문서를 통해 쉽게 적응할 수 있는 편입니다. 하지만 어떻게 만드는 게 Rails way에 맞는지 혹은 왜 그렇게 하는지 파악하기란 쉽지 않습니다. 이 책은 Rails의 각 기능에 대해 상세한 설명과 예시가 제공되어 초심자가 이해하기도 쉽고, Rails 경험이 있더라도 참고 도서로서 얼마든지 활용할 수 있으므로 강력히 추천합니다.

박조은(NBT)

Rails에 입문하는 초보로서 이 책을 통해 그동안 자세히 이해하여 활용하지 못했던 Rails의 세부 기능에 익숙해질 수 있었습니다. 그리고 Rails의 다양한 장점을 업무에 녹여낼 수 있을 것이라는 기대도 하게 되었습니다. 또한, 번역과 문맥의 흐름도 자연스러워서 Rails를 이해하는 데 많은 도움이 되었습니다. 다른 프레임워크를 사용하다 이직으로 인해 Rails를 접하게 되었습니다. Rails가 다른 프레임워크에 많은 영향을 끼치기도 했고 프레임워크가 가지고 있는 MVC 철학에 대한 이해 덕에 학습에 대한 부담감은 적었지만, 한국어로 된 책을 구하기가 쉽지 않았습니다. 그런 의미에서 이 책의 출판이 Rails로 제품을 개발하고 있는 분들에게 단비가 되어 주고, Rails에 입문하려는 분들에게 진입 장벽을 낮춰줄 수 있을 것이라 기대합니다.

심상용(Streamlyzer)

웹 프레임워크는 규모가 크다 보니 책을 통해서 공부하더라도 전체적인 구조가 머릿속에 잘 들어오지 않는 경우가 많은데, 이 책은 Rails의 구성 요소를 하나씩 잘 짚어 설명해줍니다. 나아가 참고 도서로 사용할 수 있을 정도의 풍부한 정보를 제공하므로 Rails를 처음 접하는 분들에게 한동안 친하게 지낼 수 있는 친구가 되어줄 것입니다.

윤승준(RoRLab)

Rails가 워낙 빠르게 변화하기 때문에 책이 많이 없어 아쉬웠는데, 마치 사막에서 오아시스를 발견하듯이 이 책을 만날 수 있어 기뻤습니다. 참으로 친절하고 유용하며, 설명에 깊이와 충실함까지 더해져 반드시 읽어보라 권하고 싶은 책입니다. 그동안 Rails에 대해 몰랐던 부분을 많이 깨닫게 해줘서 저 또한 무척 감명 깊게 읽었네요. 이 책은 초보자보다는 Rails 프로그래밍을 조금이라도 해본 분에게 더 좋을 것 같습니다. Rails의 교과서처럼 곁에 두고두고 참고하고 싶은 그러한 책입니다.

이상현(SI 개발자)

Ruby를 보면서 자유로움을 넘어 지저분하다는 생각에 '이 언어를 배워야 하는가?'란 회의감마저 들었습니다. 하지만 Rails를 만난 Ruby는 그 어떤 언어보다 재미있고 '역시 Ruby는 꼭 배워야 해'로 생각을 바꿔주네요. 특히, 스캐폴딩 기능은 생성과 동시에 실제 사이트에 곧바로 적용해도 좋겠다는 생각이 듭니다. Ruby가 적은 타이핑으로 많은 기능을 만들 수 있는, 생산성 높은 언어라는 것을 이 책을 통해 다시금 느낄 수 있었습니다. 책도 전체적으로 재미있게 잘 쓰였고 그대로 따라 하기도 무리가 없습니다. 정말 재미있게 읽었습니다.

최아연(숭실대학교)

Rails가 좋다는 말을 많이 들어서 한번쯤은 탐구해보고 싶었는데, 구체적인 예시와 함께 배울 수 있는 책이 나와서 정말 좋네요! 중간중간 초보자들이 흔히 저지를 수 있는 실수에 대해서 점검할 수 있게 짚어줘서 도움이 많이 되는 책입니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.

이 책의 구성

구문

구문은 다음 규칙에 맞춰 적었습니다. '…(말줄임표)'로 감싸진 매개 변수는 생략 가능하다는 것을 나타냅니다.

```
mail_to(address [,name [,opt]])
```

메서드 이름 / 매개 변수

코드 리스트

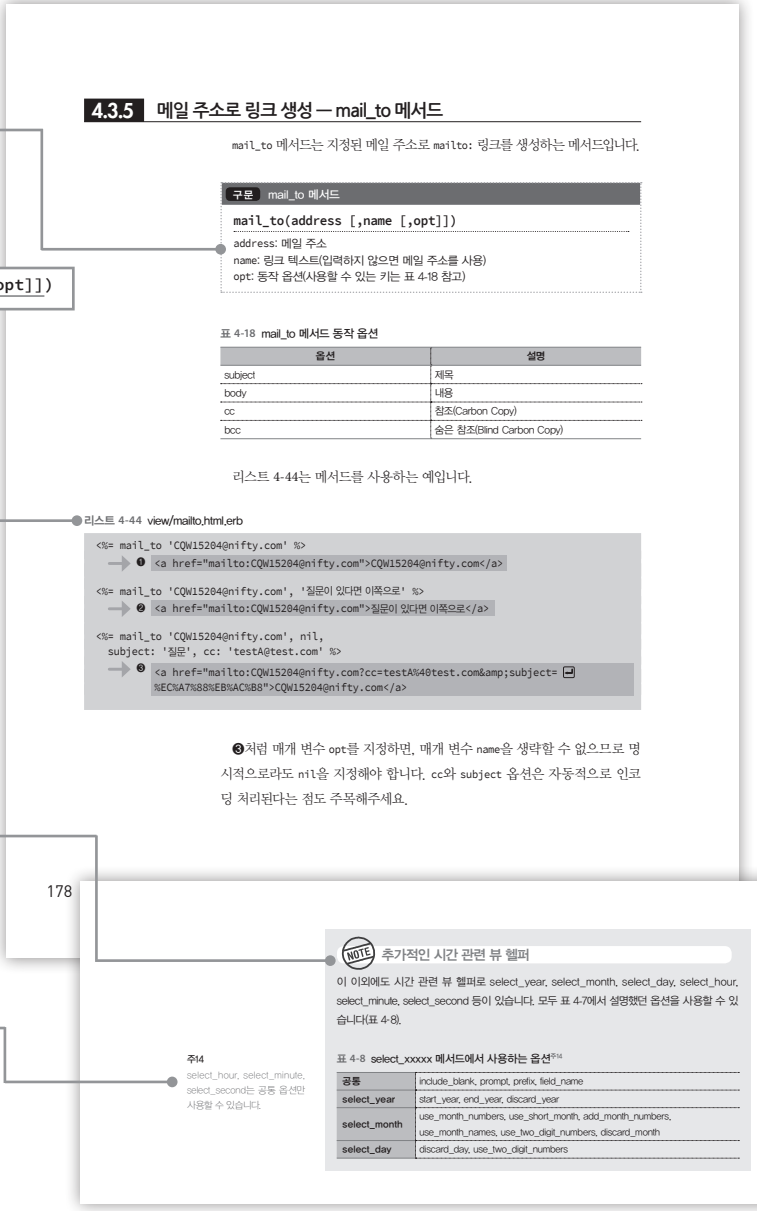
애플리케이션의 소스 코드를 나타냅니다. 지면 관계상 이해할 수 있는 최소한의 코드만 적었습니다. 따라서 전체 코드를 확인하고 싶다면 책과 함께 제공되는 예제 파일을 확인해주세요. 지면 문제로 줄 바꿈이 발생한 부분은 줄 바꿈 기호(↵)로 나타내었습니다.

노트

본문 내용 중 추가로 설명하고 싶은 주의점, 참고, 추가 정보를 나타냅니다.

주석

노트와 마찬가지로 본문에서는 설명하지 못한 추가 정보 또는 초보자가 깜박할 수 있는 포인트 등을 소개합니다. 본문에 적은 번호와 대응되므로 함께 살펴보시기 바랍니다.



4.3.5 메일 주소로 링크 생성 — mail_to 메서드

mail_to 메서드는 지정된 메일 주소로 mailto: 링크를 생성하는 메서드입니다.

구문 mail_to 메서드

```
mail_to(address [,name [,opt]])
```

address: 메일 주소
name: 링크 텍스트(입력하지 않으면 메일 주소를 사용)
opt: 동작 옵션(사용할 수 있는 키는 표 4-18 참고)

표 4-18 mail_to 메서드 동작 옵션

옵션	설명
subject	제목
body	내용
cc	참조(Carbon Copy)
bcc	숨은 참조(Blind Carbon Copy)

리스트 4-44는 메서드를 사용하는 예입니다.

리스트 4-44 view/mailto.html.erb

```
<%= mail_to 'CQW15204@nifty.com' %>
  ❶ <a href="mailto:CQW15204@nifty.com">CQW15204@nifty.com</a>

<%= mail_to 'CQW15204@nifty.com', '질문이 있다면 이쪽으로' %>
  ❷ <a href="mailto:CQW15204@nifty.com">질문이 있다면 이쪽으로</a>

<%= mail_to 'CQW15204@nifty.com', nil,
  subject: '질문', cc: 'testA@test.com' %>
  ❸ <a href="mailto:CQW15204@nifty.com?cc=testA%40test.com&amp;subject=↵
  %EC%A7%88%EB%AC%B8">CQW15204@nifty.com</a>
```

❸처럼 매개 변수 opt를 지정하면, 매개 변수 name을 생략할 수 없으므로 명시적으로라도 nil을 지정해야 합니다. cc와 subject 옵션은 자동적으로 인코딩 처리된다는 점도 주목해주세요.

NOTE 추가적인 시간 관련 뷰 헬퍼

이 이외에도 시간 관련 뷰 헬퍼로 select_year, select_month, select_day, select_hour, select_minute, select_second 등이 있습니다. 모두 표 4-7에서 설명했던 옵션을 사용할 수 있습니다(표 4-8).

표 4-8 select_xxxxx 메서드에서 사용하는 옵션[주14]

공통	include_blank, prompt, prefix, field_name
select_year	start_year, end_year, discard_year
select_month	use_month_numbers, use_short_month, add_month_numbers, use_month_names, use_two_digit_numbers, discard_month
select_day	discard_day, use_two_digit_numbers

주14 select_hour, select_minute, select_second는 공통 옵션만 사용할 수 있습니다.

제 1 장

소개

Ruby on Rails는 Ruby 프로그래밍 언어로 작성된 Ruby 환경에서 동작하는 **웹 애플리케이션 프레임워크**입니다.

이 책에서는 일단 일반적인 프레임워크에 대해서 살펴보고 Rails의 특징과 구체적인 기능에 대해서 설명합니다. 또한, 후반부에서는 Rails를 공부하기 위한 기본적인 환경을 구축하는 방법에 대해서도 설명하겠습니다.

1.1 Rails 프레임워크

무언가 어려운 문제에 직면했을 때 여러분은 어떻게 하시나요? 보통은 처음부터 문제가 무엇인지 하나하나 파악하고 해결해 나가는 방법을 생각해볼 수 있을 것입니다. 하지만 이는 그렇게 효율적인 방법이라고 할 수 없습니다.

대부분의 문제는 나보다 앞서 경험한 사람들이 존재합니다. 따라서 경험한 사람들의 앞선 지혜를 활용하면 문제를 보다 쉽게 해결할 수 있습니다.

이러한 앞선 사람들의 지혜를 '사례'라고 부릅니다. 이러한 것이 정형화되고 축적되면 **프레임워크**라고 불립니다. 프레임워크는 문제를 일반화하고 해결하기 위한 틀이라고 할 수 있습니다.

예를 들어 경영을 위한 프레임워크라고 하면 회사의 현황 분석, 방향성 분석, 수익 구조 분석과 관련된 방법론을 의미합니다. 업계의 구조를 다섯 개의 경쟁 요인으로 나누어서 분석하는 파이브 포스, 내적 요인과 외적 요인으로 기업의 현황을 분석하는 SWOT(Strength Weakness Opportunity Threat) 분석 등을 프레임워크라고 말할 수 있습니다.

프레임워크는 수학에서 사용되는 공식과도 같은 것입니다. 다만 수학과 다른 점은 답이 하나가 아니라는 것입니다. 사용하는 프레임워크에 따라서 만들 수 있는 애플리케이션(답)도 엄청나게 많으며, 상황 또는 환경에 따라 적절한 애플리케이션을 만들 수 있게도 해줍니다.

1.1.1 애플리케이션 프레임워크

애플리케이션 프레임워크도 프레임워크의 일종입니다. 애플리케이션 개발에도 당연히 여러 가지의 설계 방법론이 있습니다. 애플리케이션 프레임워크는 이러한 방법론 중에서도 '재사용이 가능한 클래스'라는 방법을 제공합니다.

개발자는 애플리케이션 프레임워크가 제공하는 기초적인 코드 위에 독자적인 코드를 추가하여 일정한 품질을 가진 애플리케이션을 쉽게 생성할 수 있습

니다. 애플리케이션 코드를 각각의 부품이라고 한다면, 애플리케이션 프레임워크는 그러한 부품들을 연결해주는 마더보드라고 할 수 있습니다(그림 1-1).

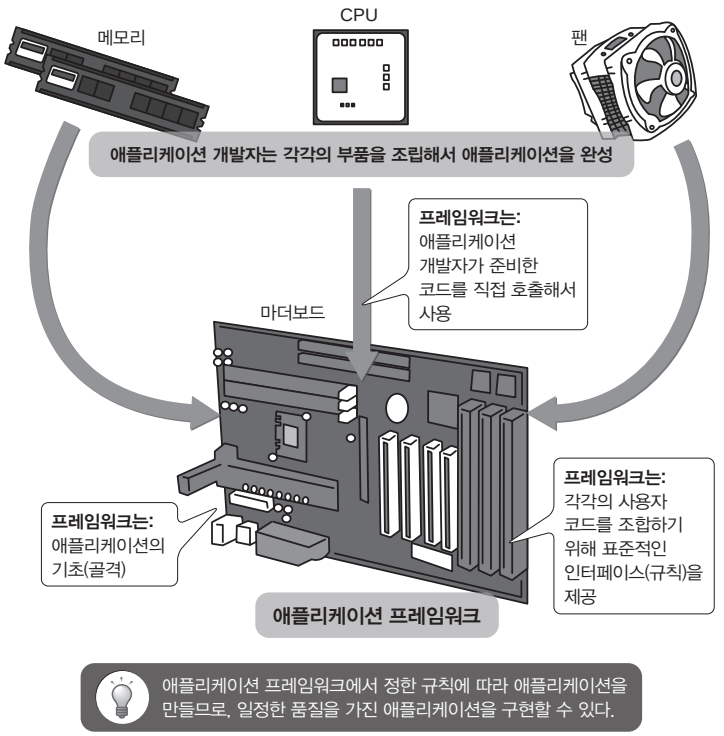


그림 1-1 애플리케이션 프레임워크는 마더보드

'재사용이 가능한 클래스'라고만 이야기하면 기능을 모아둔 **라이브러리**와 무슨 차이가 있는지 구분하기 힘들 것입니다. 실제로 요즘은 라이브러리와 프레임워크를 혼동하는 사람들이 이 두 가지를 섞어서 사용하다 보니 거의 같은 의미로 사용됩니다. 하지만 이 두 가지 사이에는 분명한 차이점이 있습니다.

라이브러리와 프레임워크를 구분하는 방법은 우리가 작성한 코드(사용자 코드)와의 관계가 어떻게 되어 있는지 알면 됩니다(그림 1-2).

우선 라이브러리는 사용자 코드에서 호출되어야 합니다. 라이브러리는 스스로 무언가를 하지 못하므로 문자열 조작 라이브러리, 로그 라이브러리처럼 사용자가 호출할 때 자신의 코드를 실행합니다.

반대로 애플리케이션 프레임워크는 프레임워크 스스로가 사용자 코드를 호출합니다. 애플리케이션 프레임워크는 자신의 라이프 사이클(초기화부터 실제 처리, 종료까지의 흐름)을 모두 직접 관리합니다. 그리고 그 흐름 속에서 개발자는 해당 부분에 사용자 코드를 추가해서 기능을 구현하는 것입니다. 한마디로 애플리케이션 프레임워크는 사용자 코드가 애플리케이션을 지배하는 것이 아니라 프레임워크가 사용자 코드를 직접 지배합니다.

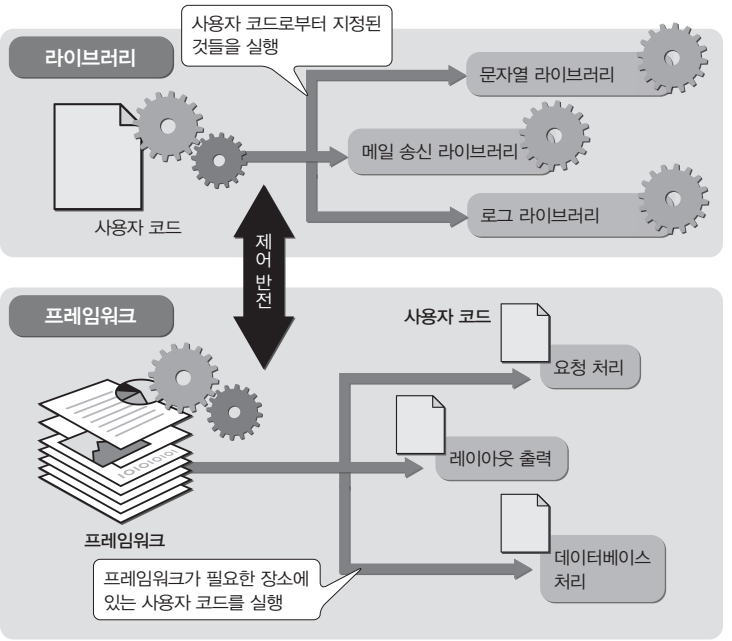


그림 1-2 제어 반전

이렇게 프로그램의 실행 주체가 역전되는 것을 **제어 반전**(Inversion of Control, IoC)이라고 부릅니다. 이 제어 반전이라는 특징이 프레임워크의 본질입니다.

Rails는 이러한 애플리케이션 프레임워크 중에서도 웹 애플리케이션을 개발하기 위한 **웹 애플리케이션 프레임워크**(Web Application Framework, 이하 WAF)입니다[주1]. 이 책의 이후 내용에서는 그냥 프레임워크라고 부르면 대개 WAF를 나타내는 것입니다.

주1
애플리케이션 프레임워크의 종류는 용도에 따라 다양합니다. 데스크톱 애플리케이션을 개발할 때는 데스크톱 GUI 프레임워크, 웹 서비스를 개발할 때는 웹 서비스 프레임워크 등 각각의 영역마다 다른 종류의 프레임워크가 존재합니다.

1.1.2 프레임워크 사용의 장점

프레임워크를 사용하면 다음과 같은 장점이 있습니다.

1 개발 생산성이 향상된다

애플리케이션의 기초가 되는 설계와 기반 코드를 프레임워크가 제공해주므로 개발 생산성이 크게 향상됩니다. 또한 모든 개발자가 동일한 규칙 속에서 작업하도록 강제되므로 코드의 일관성을 유지하기 쉬우며, 일정 수준 이상의 품질을 가진 애플리케이션을 쉽게 만들 수 있습니다.

게다가 각각의 사용자 코드는 서로 독립적으로 구성되므로 팀의 역할을 기능 단위로 분담할 수도 있습니다. 따라서 많은 사람들이 함께 프로젝트를 개발하기에 편리합니다.

2 유지보수성이 뛰어나다

코드에 일관성이 생기면 애플리케이션의 가독성 또한 높아집니다. 따라서 문제가 생기거나 사양을 변경해야 할 때 해당 사항을 쉽게 변경할 수 있다는 장점이 있습니다.

추가로 이후에 동일한 프레임워크를 사용해서 애플리케이션을 만들 때도 이전의 경험 등을 개발이나 보수에 사용할 수 있다는 것도 장점입니다.

3 최신 기술 트랜드에 대응하기 쉽다

최근의 기술은 굉장히 빠르게 변화하고 있습니다. 이러한 것을 스스로 확인하기란 굉장히 어렵습니다. 하지만 프레임워크는 이미 이러한 기술 트랜드를 도입하고 있습니다. 따라서 프레임워크를 사용하는 것만으로도 최신 기술 트랜드에 대응할 수 있습니다.

예를 들어, 보안과 관련된 요구가 높아지고 있는 최근의 경우, 많은 프레임워크들이 보안과 관련된 요구를 이미 수용하여 대응 중입니다. 따라서 프레임워크를 사용하면 보안과 관련된 개발자의 부담을 덜 수 있습니다.

4 일정 이상의 품질을 기대할 수 있다

프레임워크에만 국한되는 것은 아니지만, 일반적으로 공개되어 있는 프레임워크들은 개인이 만든 것보다 신뢰성이 높을 수밖에 없습니다. 오픈 소스로 공개되어 있는 프레임워크는 다양한 애플리케이션에서 사용되며, 많은 사람들이 소스 코드를 테스트하고 수정하는 중입니다. 따라서 일부(나 또는 팀) 사람들만 확인한 코드에 비해 높은 신뢰성을 기대할 수 있습니다[주2].

주2
물론 모든 오픈 소스 소프트웨어가 그렇다는 것은 아닙니다. 제대로 되어 있는 것인지는 스스로 판단해야 합니다.

프레임워크를 사용한다는 것은, 현재 시점에서 가장 확실히 입증된 것들을 사용한다는 의미입니다. 지금까지 든 이유로 인해 프레임워크는 오늘날의 개발에서 빼놓을 수 없는 요소가 되었습니다.

프레임워크 도입의 단점

물론 프레임워크를 사용하는 것이 무조건 좋은 일은 아닙니다. 프레임워크는 규칙의 집합이며, 따라서 이 규칙을 이해하려면 어느 정도의 학습 시간이 필요합니다. 또한, 처음 사용할 경우에는 익숙하지 않으므로 실수가 발생할 수도 있을 것입니다. 따라서 일회용 애플리케이션 또는 소규모 애플리케이션을 만들 때는 프레임워크를 사용하지 않는 방법도 고려해두길 바랍니다.

1.1.3 Ruby에서 사용할 수 있는 프레임워크

이 책에서 다루는 Ruby on Rails(이후에는 간단히 Rails라고 부르겠습니다.)는 Ruby 환경[주3]에서 사용할 수 있는 대표적인 프레임워크입니다. 하지만 Ruby 환경에서 사용할 수 있는 프레임워크가 Rails뿐일리는 없습니다. 다음 Ruby에서는 다음 표 1-1과 같이 다양한 프레임워크를 사용할 수 있습니다.

주3
Ruby는 마츠모토 유키히로(まつもとゆきひろ)가 개발한 객체지향 프로그래밍 언어입니다.

표 1-1 Ruby에서 사용할 수 있는 프레임워크

이름	설명
Ruby on Rails(http://rubyonrails.org/)	이 책에서 설명하고 있습니다
Sinatra(http://www.sinatrarb.com/)	DSL(도메인 고유 언어[주4])을 사용하여 애플리케이션을 간결하게 만드는 것을 목표로 하는 프레임워크입니다
Ramaze(http://ramaze.net/)	모듈화를 지향하므로 라이브러리 선택 자유도가 굉장히 높은 프레임워크입니다

주4
특정 용도를 위해 설계된 언어를 의미합니다. 예를 들어, 데이터베이스 쿼리를 생성하기 위한 용도로 사용하는 SQL(Structured Query Language)도 DSL (Domain Specific Language)의 일종입니다.

표 1-1 Ruby에서 사용할 수 있는 프레임워크(계속)

이름	설명
Padrino(http://www.padrinorb.com/)	시나트라(Sinatra)를 기반으로 헬퍼, 제너레이터, 국제화 대응 등의 기능을 추가한 프레임워크입니다
Pakyow(http://pakyow.com/)	뷰 퍼스트로 간단함을 추구하는 프레임워크입니다
Halcyon(http://halcyon.rubyforge.org/)	SOA(Service Oriented Architecture, 서비스 지향 아키텍처) 애플리케이션을 만드는 것을 목표로 하는 프레임워크입니다

이 책에서는 이러한 다양한 프레임워크 중에서 가장 널리 알려지고, 기능성이 좋으며, 자료도 굉장히 많고, 실적면에서도 우수한 Rails를 다룹니다. Rails의 특징은 다음과 같습니다.

1 MVC 패턴

Rails는 **MVC 패턴(Model-View-Controller 패턴)**이라 불리는 아키텍처를 사용하고 있습니다. MVC 패턴이란, 애플리케이션을 모델(Model, 비즈니스 로직), 뷰(View, 사용자 인터페이스), 컨트롤러(Controller, 모델과 뷰를 제어)와 같이 수행하는 역할로 명확히 구분해서 구성하는 것을 말합니다. 그림 1-3은 MVC 패턴의 전형적인 흐름이자 Rails의 기본적인 실행 방식을 나타낸 것입니다. 일단은 이러한 큰 그림을 이해하도록 합시다.

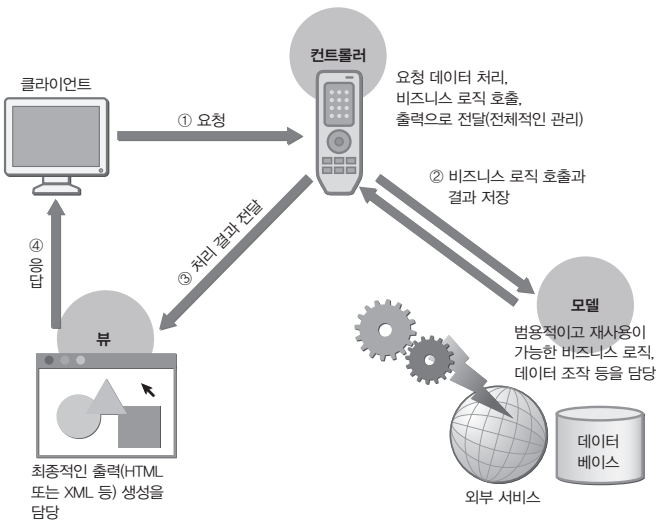


그림 1-3 MVC 패턴

MVC 패턴은 각각의 구성 요소가 명확하게 구분되어 있으므로 다음과 같은 장점이 있습니다.

- 프로그래머와 디자이너의 공동 작업이 쉬워짐
- 디자인과 로직의 수정이 서로에게 영향을 주지 않음(유지보수가 용이)
- 기능 단위 테스트를 독립적으로 실행할 수 있음(테스트 자동화가 쉬워짐)

물론 MVC 패턴은 Rails에서만 사용하는 개념이 아니며, 예전부터 웹 애플리케이션 개발은 MVC 패턴으로 개발하는 것이 일반적이었습니다. 자바(Java), PHP, 펄(Perl), 파이썬(Python) 등으로 프레임워크를 사용해본 적이 있는 독자라면 MVC 패턴과 쉽게 친숙해질 수 있을 것입니다. 반대로 생각하면 Rails를 배워둘 경우, 다른 프레임워크를 이해하는 데 도움이 된다는 의미이기도 합니다.

2 Rails는 개발을 위한 레일을 제공

Rails라는 프레임워크를 이야기할 때 빼놓을 수 없는 것이 바로 Rails의 설계 철학입니다. Rails는 초기 버전부터 다음과 같은 설계 철학을 지키고 있습니다.

- DRY(Don't Repeat Yourself): 같은 코드를 반복하지 말 것
- CoC(Convention over Configuration): 설정보다 규약이 중요

Rails는 소스 코드 내부에서 같은 처리 또는 정의를 반복해서 사용하는 것을 극도로 싫어합니다. 예를 들어, Rails에서는 데이터베이스 스키마 정의를 설정 파일로 별도 작성하지 않아도 됩니다. 그냥 데이터베이스에 테이블을 만드는 것만으로도 Rails가 모두 알아서 해줍니다.

그리고 DRY 원칙을 지탱할 수 있게 해주는 것이 바로 CoC 원칙입니다. 규약(Convention)이란, 간단히 말해 Rails가 미리 정의한 이름을 붙이는 규칙입니다. 예를 들어, `users` 테이블을 읽어 들이려면 `User`라는 이름의 클래스를 이용해야 합니다. 서로 간의 규칙을 따로 정의할 필요 없이, `users`(복수형)와 `User`(단수형)로만 규약에 맞춰 작성하면 Rails가 테이블과 클래스를 연결해줍니다.

Rails는 DRY와 CoC 원칙을 도입함으로써, 개발자가 적은 노력으로 유지보

수가 쉬운 애플리케이션을 개발할 수 있도록 해줍니다. 개발자가 원칙을 지켜 만들도록 하여 바람직한 방향으로 이끌어주는, 길(Rails)이 되는 프레임워크인 것입니다.

Rails의 기본 철학은 이후에 등장한 많은 프레임워크들에게 영향을 주어[주5] Rails의 명성이 한층 더 높아지는 계기가 되었습니다.

주5
Ruby 세계뿐만 아니라, 펄의 Catalyst, PHP의 Symphony 또는 CakePHP, .NET의 ASP.NET MVC 등에서 Rails의 영향을 찾아볼 수 있습니다.

3 풀 스택 프레임워크

Rails는 애플리케이션 개발을 위한 라이브러리는 물론, 코드 생성을 위한 툴 또는 동작 확인을 위한 서버 등을 한꺼번에 제공해주는 풀 스택(모두 들어있는) 프레임워크입니다. 따라서 Rails 하나를 설치하는 것만으로도 애플리케이션 개발에 필요한 모든 환경을 구축할 수 있으므로, 환경을 준비하는 데 소요되는 시간과 노력이 비교적 적습니다. 또한, 라이브러리의 상성 또는 버전 간의 부합성 등을 의식하지 않고 개발을 진행할 수 있습니다.

그림 1-4에 Rails에 포함되어 있는 컴포넌트(라이브러리)를 정리해두었습니다.

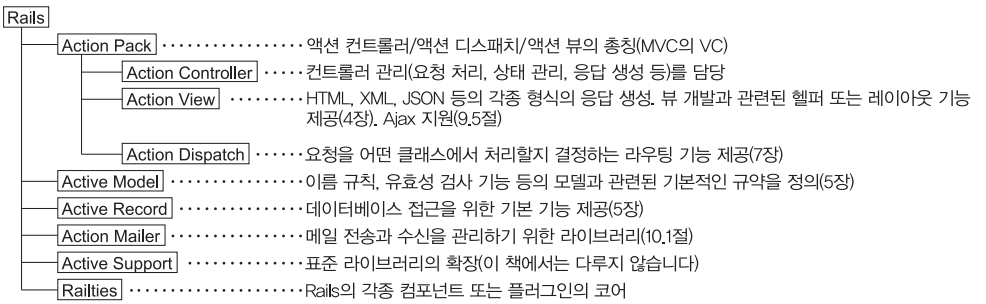


그림 1-4 Rails 라이브러리 구조

Rails 2까지는 이러한 컴포넌트가 의존성을 가지고 있었기에 Rails에서 기본적으로 제공하는 컴포넌트 이외의 것을 사용할 때는 상당한 노력이 필요했습니다.

하지만 Rails 3 이후에서는 모듈 지향(Modularity)을 강력하게 의식하게 되었습니다. 따라서 표준 컴포넌트는 계속 제공되지만, 필요에 따라서 보다 목적에 맞는 컴포넌트로 교체해서 사용할 수 있게 되었습니다.

Rails 3 이후부터는 Rails 2를 이어받으면서도 보다 유연한 설계를 도입함으로써, 다양한 라이브러리를 함께 사용할 수 있는 프레임워크로 진화했습니다.

4 최신 기술 트랜드에 빠르게 대응

Rails 4는 최근의 다양한 기술 트랜드에 대응합니다. 대표적으로는 다음과 같이 세 가지가 있습니다.

- HTML5 대응
- 겸손한(Unobtrusive) 자바스크립트
- RESTful 인터페이스

HTML5는 HTML의 최신 버전입니다. Rails 4에서는 스캐폴딩(Scaffolding) 기능[주6]을 사용할 때 HTML5를 대응하는 마크업으로 페이지가 생성되며, 뷰 헬퍼[주7]가 HTML5 태그에 모두 대응하고 있습니다.

겸손한(Unobtrusive) 자바스크립트는 최근의 자바스크립트 트랜드로, HTML 내부에 자바스크립트를 삽입하지 않는 것을 의미합니다. HTML 내부에 섞여 있는 자바스크립트 코드는 코드의 가독성을 떨어뜨리고 유지보수도 어렵게 만듭니다. Rails 2까지는 (특히) Ajax[주8] 관련 기능을 위해 자동 생성되는 HTML 코드에 복잡한 자바스크립트 코드가 섞여 있었지만, Rails 3부터는 자바스크립트를 외부 라이브러리로 분리해서 HTML 쪽에서 매개 변수 정보만 전달하는 형태로 바뀌었습니다.

RESTful 인터페이스는 리소스의 위치를 URL로 나타내고 리소스에 어떠한 조작(CRUD[주9])을 할지 HTTP 메서드로 나타내는 방법을 의미합니다(자세한 내용은 7장에서 설명하겠습니다). RESTful 인터페이스를 사용하면 의미적으로 구조가 잘 잡힌 URL을 설계할 수 있습니다. RESTful 인터페이스 지원 자체는 Rails 2부터 추가되었지만, Rails 4에서 몇 가지 기능을 더 보강했습니다.

주6
자동적으로 애플리케이션의 기본적인 코드를 생성해주는 기능입니다. 3장에서 자세히 설명합니다.

주7
액션 뷰(Action View)가 제공하는 태그를 생성하기 위한 유틸리티 메서드를 의미합니다. 4장에서 자세히 설명합니다.

주8
자바스크립트를 사용해 풍부한 기능을 가진 애플리케이션(RIA: Rich Internet Application)을 구현 때 사용하는 기술입니다. 9.5절에서 자세히 설명합니다.

주9
Create(생성), Read(추출), Update(변경), Delete(삭제)를 의미합니다.

1.2 Rails 환경 구축

Rails의 개요를 이해했다면 다음 장부터 본격적으로 Rails를 학습하기 앞서 먼저 Rails 애플리케이션을 개발할 수 있는 환경을 구축하도록 합시다.

환경을 구축하는 것은 굉장히 중요합니다. 이후 장에서 코드가 의도한 대로 동작하지 않는다면, 환경과 관련된 설정에 원인이 있을 가능성이 있습니다. 사용하는 버전은 정확한지[주10], 설정하고 있는 옵션이 잘못되지는 않았는지를 꼼꼼하게 확인해 나가며 환경을 구축하기 바랍니다.

주10
책의 원활한 학습을 위해 책과 사용하는 버전을 일치시키는 것을 권합니다.

1.2.1 Rails 프로그래밍에 필요한 소프트웨어

Rails로 애플리케이션을 개발 또는 실행할 때는 그림 1-5처럼 기본적으로 소프트웨어가 필요합니다.

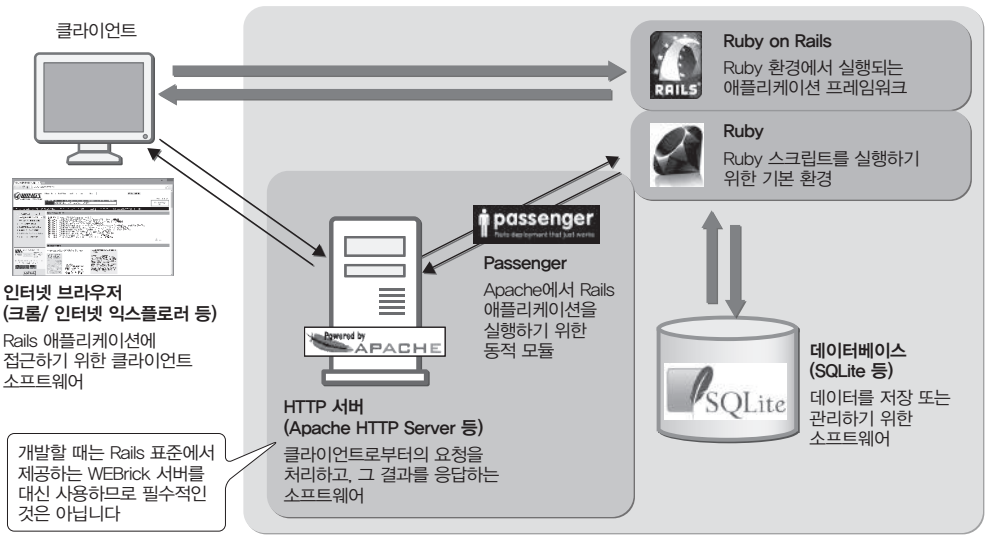


그림 1-5 Rails 프로그래밍에 필요한 소프트웨어

1 Ruby

Ruby on Rails라는 이름 그대로 Rails는 Ruby 환경에서 동작하는 프레임워크입니다. 따라서 Rails를 실행시키려면 Ruby가 설치돼 있어야 합니다.

Ruby 패키지는 Ruby 스크립트 실행 엔진을 포함해 인터렉티브 대화 환경, 표준 라이브러리, 문서 등의 Ruby 애플리케이션의 개발이나 실행에 필요한 소프트웨어들을 포함하고 있습니다.

2 HTTP 서버

반복해서 말하지만 Rails는 웹 애플리케이션을 개발하기 위한 프레임워크입니다. 따라서 Rails 애플리케이션에서 클라이언트의 요청을 받고 응답하려면 이를 매개해주는 HTTP 서버(웹 서버)가 필요합니다.

Rails 애플리케이션을 실행할 때는 Apache HTTP Server(이후에는 간략하게 Apache(아파치)라고 부르겠습니다)와 Nginx 등의 다양한 HTTP 서버를 선택할 수 있습니다. 일단 이 책에서는 개발이나 학습을 목적으로 Rails 표준 HTTP 서버인 WEBrick(웹릭)을 사용하겠습니다.

10.5.1절에서는 실제 배포 환경에서 애플리케이션을 실행하는 예로 Apache + Passenger에서 Rails 애플리케이션을 실행하는 방법을 살펴봅니다.

3 데이터베이스(SQLite)

Rails에서 애플리케이션을 구현한다면 애플리케이션 데이터를 저장하기 위한 데이터 저장소로 데이터베이스를 사용해야 합니다. 데이터베이스 안에도 Oracle 데이터베이스 또는 SQL Server 같은 상용 제품부터, MySQL 또는 PostgreSQL과 같은 오픈 소스 계열 등등 다양한 제품군이 있습니다. Rails에서는 이러한 모든 주요 데이터베이스를 사용할 수 있습니다만, 이 책에서는 Rails의 표준 데이터베이스인 SQLite를 사용합니다[역주1].

역주1
이후에 MySQL 환경으로 옮기는 과정도 살펴봅니다.

4 Ruby on Rails

이 책의 주제인 Rails 자체입니다. 필자가 책을 집필할 때의 최신 안정판인 4.2.0을 사용합니다.

1.2.2 윈도우즈에서의 개발 환경 설정

이 책은 윈도우즈(Windows)8.1 Pro(64비트) 환경을 기준으로 집필되었습니다. 다른 버전을 사용하고 있다면 경로나 메뉴 이름, 또는 일부 조작 형태가 다를 수 있으므로 주의하기 바랍니다.

사용자 계정 컨트롤(이후에는 UAC라고 간단히 표현하겠습니다)를 사용하고 있을 경우 설치 또는 설정할 때 보안 경고와 관련된 대화상자가 나올 것입니다. 이때는 적당히 [확인]을 눌러주세요.

1 Ruby 설치 방법

이 책을 집필하는 시점의 Ruby 안정판의 최신 버전은 2.0.0-p645입니다. 윈도우즈에서 Ruby를 설치할 수 있게 다양한 패키지가 제공되고 있는데요. 그 중에서도 라이브러리 등을 모두 포함하고 있는 Ruby Installer for Windows(이후에는 "Ruby 인스톨러"라고 간단히 표현하겠습니다)를 사용하는 것이 편리합니다. 이 책에서도 Ruby 인스톨러를 사용해서 Ruby를 설치하는 방법을 알아보겠습니다. Ruby 인스톨러는 다음 경로에서 다운로드할 수 있습니다[주11].

주11
해당 페이지는 Ruby 인스톨러 외에도 다양한 패키지가 제공됩니다.

```
http://rubyinstaller.org/downloads/
```

설치 프로그램을 실행하려면 다운로드한 `rubyinstaller-2.0.0-p645.exe`[역주2]를 더블 클릭하면 됩니다. 그러면 그림 1-6처럼 대화상자가 실행되는데, 이 다음부터는 화면의 지시에 따라서 설치를 계속 진행하기 바랍니다.

역주2
숫자가 높으면 무조건 좋을 것이다라는 생각으로 2.1.0 또는 2.2.0을 설치하고 싶다는 생각이 들 수 있는데요. 일단 숫자가 높다고 해서 무조건 최근에 발표되었다는 것은 아닙니다. 또한, sqlite3 젬이 해당 버전들에서는 제대로 작동하지 않으니 가급적 2.0.0의 최신 버전을 사용하기 바랍니다. 또한, 64비트 Ruby가 일부 모듈과 충돌이 일 수 있으므로 64비트 시스템을 사용하고 있더라도 32비트용 루비를 설치해주세요.

[Installation Destination and Optional Tasks] 화면에서는 Ruby 설치 옵션을 설정합니다. 여기에서는 [add Ruby executables to your path]를 선택해두기 바랍니다.

환경 변수 PATH는 명령 프롬프트에서 명령어를 실행할 때 명령어를 실행할 프로그램이 어디에 있는지를 찾기 위한 경로를 나타냅니다. 여기서 PATH 설정을 해두지 않으면 이후에 명령어를 실행할 때 절대 경로를 사용해야 하므로 주의하기 바랍니다. 이어서 [설치] 버튼을 클릭하면 설치가 시작됩니다(그림 1-6).

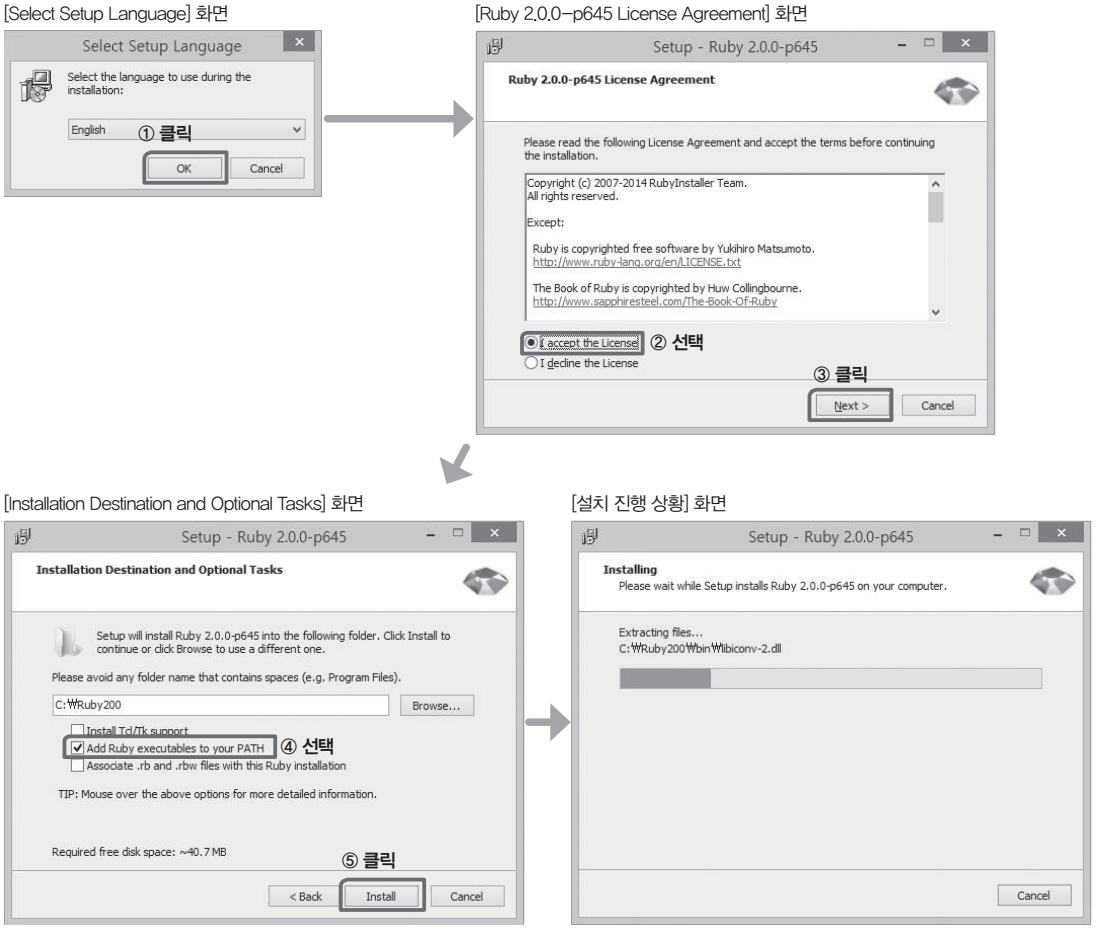


그림 1-6 Ruby 설치 대화상자

설치가 완료되면 설치 완료 화면이 표시됩니다. [Finish] 버튼을 클릭하여 대화상자를 종료해주세요(그림 1-7).

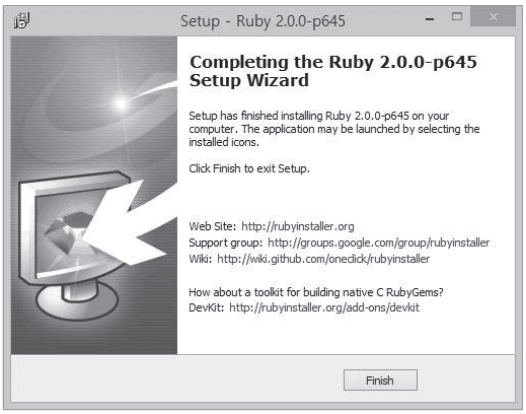


그림 1-7 Ruby 2.0.0-p645 설치 완료 화면

설치가 정상적으로 완료되었다면 Ruby가 정말 제대로 설치되었는지 확인합시다. 시작 화면에서 [Windows 시스템 도구] → [명령 프롬프트]를 선택해주세요. 명령 프롬프트가 실행되면 다음과 같은 명령어를 입력합니다. 설치된 Ruby 버전이 출력된다면 정상적으로 완료된 것입니다.

```
> ruby -v
ruby 2.0.0p645 (2015-04-13) [i386-mingw32]
```

2 SQLite 설치 방법

SQLite 설치는 설치라고 말할 것도 없을 만큼 간단합니다. 그냥 압축 파일을 다운로드하여 압축을 해제하고 특정한 위치로 옮겨주기만 하면 됩니다. 다음 경로에서 `sqlite-shell-win32-x86-3090200.zip`을 다운로드해주세요.

이 책을 집필하는 시점의 SQLite 3 안정판의 최신 버전은 SQLite 3.8.9입니다 책에서도 해당 버전을 사용하도록 하겠습니다.

```
http://www.sqlite.org/download.html
```

`sqlite-shell-win32-x86-3090200.zip`은 커맨드 라인 셸(SQLite 클라이언트)을 포함하고 있는 압축 파일입니다. 압축을 해제하면 `sqlite3.exe`라는 파일이 있을 텐데, 이를 Ruby 바이너리 폴더(`c:\Ruby200\bin`)에 옮겨주세요.

SQLite가 제대로 실행되는지도 명령어로 확인하기 바랍니다.

```
> sqlite3 -version
3.8.9 2015-04-08 12:16:33 8a8ffc862e96f57aa698f93de10dee28e69f6e09
```
현재 버전 출력

Rails에서 SQLite를 사용하기 위한 준비를 마쳤습니다.

3 DevKit 설치 방법

DevKit은 윈도우즈 환경에서 네이티브 C/C++ 확장 모듈을 빌드하기 위한 툴킷입니다. Rails를 설치할 때와 `rails new` 명령어(26쪽)로 애플리케이션을 생성할 때 필수이므로 미리 설치합시다. DevKit은 다음 경로에서 다운로드합니다.

```
http://rubyinstaller.org/downloads/
```

다운로드한 `DevKit-mingw64-32-x.x.x-yyyymmdd-xxxx-sfx.exe`(`x.x.x-yyyymmdd-xxxx`는 버전 번호)를 더블 클릭하여 `c:\Ruby200\devkit`에 압축을 해제하고 다음 명령어를 실행해주세요.

```
> cd c:\Ruby200\devkit
> ruby dk.rb init
> ruby dk.rb install
```

4 Node.js 설치 방법

Rails로 애플리케이션을 생성하려면 Node.js를 미리 설치해둬야 합니다. 이 책을 집필하는 시점의 최신 안정판은 v0.12.2입니다. Node.js 설치 프로그램은 다음 경로에서 다운로드합니다.

```
http://nodejs.org/download/
```

다운로드한 `node-v0.12.2-x64.msi` 파일을 클릭해서 실행하면 설치 프로그

램이 시작됩니다. 그림 1-8처럼 화면에서 설명하는 대로 진행하기 바랍니다.

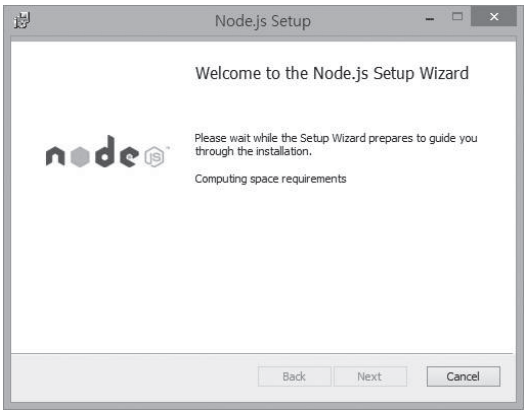


그림 1-8 Node.js 설치 프로그램

설치를 완료하면 완료 화면이 출력됩니다. [Finish] 버튼을 클릭해서 설치 프로그램을 종료해주세요. Node.js가 정상적으로 설치된 것을 확인하려면 명령 프롬프트를 실행하여 다음 명령어를 입력합니다. 버전 정보가 표시되면 Node.js가 정상적으로 설치된 것입니다.

```
> node -v
v0.12.2
```

5 Ruby on Rails 설치 방법

이 책을 집필하는 시점의 Rails 안정판의 최신 버전은 4.2.0입니다. 따라서 이 책을 공부할 때는 원칙적으로 4.2.0 버전을 사용하기 바랍니다. 4.0 계열의 최신 버전을 사용해서 진행해도 됩니다. Rails 설치는 gem 명령어로 수행합니다.

주12
이전 버전의 Rails를 설치하고 싶을 때는 gem install rails -v 4.0.0 처럼 -v 옵션을 사용합니다.

```
> gem install rails[주12]
Fetching: thread_safe-0.3.5.gem (100%)
Successfully installed thread_safe-0.3.5
Fetching: tzinfo-1.2.2.gem (100%)
Successfully installed tzinfo-1.2.2
Fetching: i18n-0.7.0.gem (100%)
Successfully installed i18n-0.7.0
```

```
Fetching: activesupport-4.2.1.gem (100%)
Successfully installed activesupport-4.2.1
…생략…
Installing ri documentation for i18n-0.6.9
Parsing documentation for multi_json-1.8.2
Installing ri documentation for multi_json-1.8.2
Parsing documentation for tzinfo-0.3.38
Installing ri documentation for tzinfo-0.3.38
…생략…
Parsing documentation for rails-4.2.1
Installing ri documentation for rails-4.2.1
Done installing documentation for thread_safe, tzinfo, i18n,
activesupport, rails-deprecated_sanitizer, mini_portile, nokogiri,
rails-dom-testing, loofah, rails-html-sanitizer, erubis, builder,
actionview, rack, rack-test, actionpack, sprockets, sprockets-
rails, bundler, thor, railties, globalid, activejob, mime-types,
mail, actionmailer, arel, activemodel, activerecord, rails after
758 seconds
30 gems installed
```

패키지 다운로드 등에 10분 정도의 시간이 소요됩니다. 위와 같이 표시되면 Rails가 성공적으로 설치된 것입니다. 제대로 설치되었는지 확인할 때는 다음 명령어를 입력합니다.

```
> rails -v
Rails4.2.0
```

이렇게 Rails 버전이 표시되면 Rails가 정상적으로 설치된 것입니다.

gem 명령어의 주요 옵션

일반적으로 Rails 버전은 gem 명령어로 관리합니다. 방금 사용해보았던 install 옵션 외에 표 1-2의 옵션으로 패키지 업데이트나 제거 등을 명령어 한 줄로 수행할 수 있습니다.

표 1-2 gem 명령어의 주요한 옵션(〈Package〉는 패키지 이름)

옵션	설명
gem uninstall 〈Package〉	패키지 제거
gem update 〈Package〉	패키지를 최신 버전으로 변경
gem cleanup 〈Package〉	최신 버전만 남기고 패키지 제거
gem list	설치된 패키지 목록 확인
gem which 〈Package〉	패키지 설치 경로 확인

1.2.3 리눅스에서 개발 환경 설정

리눅스(Linux)에서 개발 환경을 설정하는 방법을 알아봅시다. 이 책에서는 페도라 19(Fedora 19)을 사용합니다. 다른 배포판 또는 버전을 사용하고 있다면 경로 또는 메뉴 이름 등이 다를 수 있으므로 주의합시다.

1 Ruby 설치 방법

이 책을 집필하는 시점의 Ruby 안정판의 최신 버전은 2.0.0–p645입니다. 다음 사이트에 들어가서 Ruby 소스 코드를 다운로드하기 바랍니다.

```
http://www.ruby-lang.org/ko/downloads/
```

다운로드한 ruby-2.0.0-p645.tar.gz를 적당한 폴더로 이동한 후 다음 명령어로 압축을 해제합니다.

```
$ tar zxvf ruby-2.0.0-p645.tar.gz ——— 패키지 압축 해제
```

.tar.gz 파일을 압축 해제하면 /ruby2.0.0-p645와 같은 폴더가 생성됩니다.

이 폴더로 이동한 후 컴파일 조건을 설정합니다. 설정이 완료되면 빌드와 설치를 진행합니다.

```
$ cd ruby-2.0.0-p645
$ ./configure
$ make
$ su
# make install
```

사용하고 있는 환경에 따라서 다르겠지만, 컴파일과 설치에 10분 정도 걸립니다. 모든 과정이 완료되면 Ruby가 /usr/local/bin/ruby에 설치됩니다[주13].

설치가 정상적으로 완료되었다면 이번에는 Ruby가 제대로 설치되었는지 확인할 차례입니다. 셸 프롬프트에 다음과 같은 명령어를 입력해주세요. Ruby 버전이 표시되면 정상적으로 설치된 것입니다.

```
$ ruby -v
ruby 2.0.0-p645 (2015-04-13 revision 43784) [i686-linux]
```

주13
configure를 실행했을 때 설치 대상 경로를 변경하는 바람에 어디에 설치되었는지 잘 모를 경우는 프롬프트에서 which ruby 명령어를 실행해서 설치 경로를 확인할 수 있습니다.

2 SQLite 설치 방법

이 책을 집필하는 시점의 SQLite 안정판의 최신 버전은 SQLite 3.8.9입니다. 다음 페이지에 접속하여 소스 코드를 다운로드하기 바랍니다.

```
http://www.sqlite.org/download.html
```

다운로드한 sqlite-autoconf-3090200.tar.gz를 적당한 폴더로 이동한 후 다음 명령어로 압축을 해제합니다.

```
$ tar zxvf sqlite-autoconf-3090200.tar.gz
```

.tar.gz 파일을 압축 해제하면 /sqlite-autoconf-3090200과 같은 폴더가 생성됩니다. 이 폴더로 이동한 후 컴파일 조건을 설정합니다. 설정이 완료되면 빌드와 설치를 진행합니다.

```
$ cd sqlite-autoconf-3090200
$ ./configure
$ make
$ su
# make install
```

모든 과정이 완료되면 SQLite가 /usr/local/bin/sqlite3에 설치됩니다.

설치가 정상적으로 완료되었다면 프롬프트에서 다음과 같은 명령어를 입력하여 버전이 표시되는지 확인해주세요. 다음과 같이 프롬프트에 버전이 표시되면 SQLite가 정상적으로 설치된 것입니다.

```
$ sqlite3 -version
3.8.9 2015-04-08 12:16:33 27392118af4c38c5203a04b8013e1afdb1cebd0d
```

추가로 Ruby(Rails)에서 SQLite에 접근할 수 있도록 지금 sqlite3 드라이버를 함께 설치해둡시다.

```
# gem install sqlite3
Building native extensions.  This could take a while...
Successfully installed sqlite3-1.3.8
1 gem installed
Installing ri documentation for sqlite3-1.3.8...
Installing RDoc documentation for sqlite3-1.3.8...
```

이렇게 하면 Rails에서 SQLite를 사용할 준비가 모두 완료된 것입니다.

3 Node.js 설치 방법

Rails에서 애플리케이션을 동작시키려면 Node.js가 설치되어 있어야 합니다. 필자가 책을 집필한 시점의 Node.js 버전은 v0.12.2입니다. Node.js의 소스 코드를 다음 경로에서 다운로드해주세요.

```
http://nodejs.org/download/
```

다운로드한 node-v0.12.2.tar.gz를 적당한 폴더로 이동하고 다음 명령어로 압축을 해제합니다.

```
$ tar zxvf node-v0.12.2.tar.gz
```

.tar.gz 파일을 압축 해제하면 /node-v.0.12.2와 같은 폴더가 생성됩니다. 이 폴더로 이동하고 컴파일 조건을 설정합니다. 설정이 완료되면 빌드와 설치를 진행합니다.

```
$ cd node-v0.12.2
$ ./configure
$ make
$ su
# make install
```

모든 과정이 완료되면 /usr/local/bin/node에 Node.js가 설치됩니다.

설치가 정상적으로 완료되었다면 프롬프트에서 다음과 같은 명령어를 입력하여 버전이 표시되는지 확인해주세요. 다음과 같이 표시되면 Node.js가 정상적으로 설치된 것입니다.

```
# node -v
v0.12.2
```

4 Ruby on Rails 설치 방법

이 책을 집필하는 시점의 Rails 안정판의 최신 버전은 4.2.0입니다. 따라서 이 책을 공부할 때는 원칙적으로 4.2.0 버전을 사용하기 바랍니다. 물론 4.0 계열의 최신 버전을 사용해서 진행해도 됩니다. Rails 설치는 gem 명령어로 수행합니다.

```
# gem install rails[주14]
Fetching: i18n-0.6.9.gem (100%)
Successfully installed i18n-0.6.9
Fetching: multi_json-1.8.2.gem (100%)
```

주14
이전 버전의 Rails를 설치하고 싶을 때는 gem install rails –v 4.0.0처럼 –v 옵션을 사용합니다.

```
Successfully installed multi_json-1.8.2
Fetching: tzinfo-0.3.38.gem (100%)
Successfully installed tzinfo-0.3.38
…생략…
Installing ri documentation for rails-4.2.0
28 gems installed
```

패키지 다운로드 등에 10분 정도의 시간이 소요됩니다. 위와 같이 표시되면 Rails가 성공적으로 설치된 것입니다. 제대로 설치되었는지 확인할 때는 다음 명령어를 입력해봅시다.

```
$ rails -v
Rails 4.2.0
```

이렇게 Rails 버전이 표시되면 Rails가 정상적으로 설치된 것입니다.

1.2.4 예제 배치 방법(윈도우즈/리눅스 공통)

이 책에서 사용하는 예제는 제이펍 GitHub(http://github.com/Jpub/RubyonRails4)에서 다운로드할 수 있습니다. 다운로드한 파일을 열면 /railbook과 같은 폴더가 생기는데요. 이 폴더를 적당한 폴더(예를 들면 c:\data)로 옮겨서 다음과 같은 명령어를 실행합니다[주15].

```
> cd c:\data\railbook
> bundle install
```

이렇게 설정하면 서버를 실행해서 모든 예제를 확인해볼 수 있습니다. 물론 코드를 직접 하나하나 입력하면서 진행하면 좋겠지만, 자신의 코드가 잘 동작하지 않거나, 어떤 식으로 동작하는지 빠르게 확인해보고 싶은 독자라면 예제를 활용할 것을 권합니다.

주15
리눅스 환경에서는 db/development.sqlite3에 퍼미션을 666으로 설정하기 바랍니다.

윈도우즈 환경

윈도우즈 환경에서 rails new 명령어로 프로젝트를 생성했다면, Gemfile 파일에 다음과 같은 코드가 추가되어 있을 것입니다. 바로 타임존과 관련된 젬을 설치하는 내용입니다.

```
# Windows does not include zoneinfo files, so bundle the ↵
tzinfo-data gem
gem, 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
```

책을 보는 독자 대부분이 윈도우즈 환경에서 작업할 것이라 여기고 예제 파일에 추가했는데요. 해당 부분이 필요 없는 다른 환경이라면 이 코드를 제거해도 상관없습니다.