Test-Drive ASP.NET MVC

테스트 주도 ASP.NET MVC 프로그래밍

Test-Drive ASP.NET MVC

by Jonathan McCracken

Test-Drive
ASP.NET MVC
테스트 주도 ASP.NET MVC 프로그래밍

초판 1쇄 발행 2011년 6월 3일

지은이 조나단 맥크래컨
옮긴이 장현희 | **펴낸이** 장성두 | **책임편집** 안주연

본문디자인 북아이 | **표지디자인** 미디어픽스

주소 경기도 파주시 교하읍 파주신도시 에이15-1블록 한빛마을 휴먼빌 201-502
전화 070-8201-9010 | **팩스** 02-6280-0405
홈페이지 www.jpub.kr | **펴낸곳** 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호

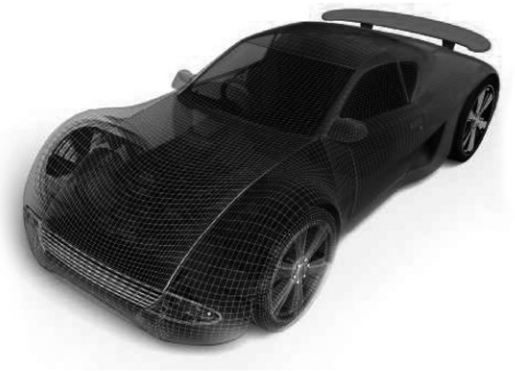
용지 신승지류유통 | **인쇄** 한승문화 | **제본** 춘산제본

ISBN 978-89-94506-08-1 (13560)

값 22,000원

제이펍은 독자 여러분의 책에 관한 아이디어와 원고 투고를 기다리고 있습니다. 책으로 펴내고자 하는 아이디어나 원고가 있으신 분께서는 책에 대한 간단한 개요와 차례, 구성과 저(역)자 약력 등을 메일로 보내주세요. (보내실 곳: jeipub@gmail.com)

The Pragmatic Programmers



Test-Drive ASP.NET MVC

테스트 주도 ASP.NET MVC 프로그래밍

조나단 맥크래컨 지음 | 장현희 옮김

ASP.NET MVC 애플리케이션 구현 방법과 TDD 접근법에 대한 명쾌한 소개!

Jpub 제이펍

차 례

PART 02

애플리케이션의 구현

옮긴이 머리말

테스트 주도 개발(TDD; Test Driven Development) 방법론은 이미 널리 알려진, 그다지 새로운 개념은 아니다. 그러나 그 인지도에 비해 국내에서의 대중화는 아직 충분히 이루어진 것 같지는 않다. 역자가 이 책을 선택하게 된 이유가 바로 여기에 있다.

사실 해외에서 진행되는 다양한 오픈 소스 프로젝트들에서 TDD 방법론의 적용은 이미 대중화의 수준을 넘어 일반화되어 가고 있다. 수십 명의 개발자가 참여하여 충분한 검증을 거치고 대중화의 단계에 들어선 대형 프로젝트부터 개인이 공개하는 간단한 라이브러리 소스에 이르기까지 이제는 테스트 코드가 포함되지 않은 소스 코드를 찾기란 여간 어려운 일이 아니다.

많은 이들이 사용한다는 이유로 무조건 따라갈 필요는 없지만 많은 이들이 공감한다면 그에 상응하는 이유가 있을 것이며, 이 책은 그 이유가 궁금한 독자들의 갈증을 해소해 줄 수 있는 충분한 가치가 있는 책이다.

특히 아직 TDD 방법론을 적용해 본 적이 없거나 이제 막 관심을 갖기 시작한 독자들이라면 이 책은 후회 없는 선택이 될 것이다. 첫 장부터 등장하는 일정 관리 웹 사이트의 소스 코드가 TDD 방법론에 따라 완성되어 가는 과정은 변신 로봇으로 유명한 영화 트랜스포머에서 범블비가 자동차에서 로봇으로 변신하는 장면만큼이나 스펙타클하다.

그렇기에 역자는 이 책의 모든 소스 코드를 단지 눈으로만 보지 말고 직접 입력해 보기를 강력히 권한다. 아직 TDD 방법론에 익숙하지 못한 독자라면 이 책을 통해 TDD 방법론에 입각한 애플리케이션 개발의 기본을 확실히 다질 수 있을 것이며, 이미 익숙한 독자라면 자신의 지식을 다시 한 번 재정리할 수 있는 기회와 함께 또 다른 경험을 얻을 수 있기 때문이다.

이 책을 통해 얻을 수 있는 또 다른 멋진 경험은 이제는 완전히 대중화된 다양한 오픈 소스

프레임워크들을 이용한 ASP.NET MVC 웹 애플리케이션 개발이다. 주로 TDD 방법론에 따라 애플리케이션을 개발하는 데 있어 유용하게 활용할 수 있는 NUnit 테스트 프레임워크나 Rhino Mocks 가상 객체 프레임워크를 비롯하여 NHibernate 프레임워크와 같은 ORM 도구에 이르기까지, 보다 안정적이며 깔끔한 코드를 작성할 수 있는 다양한 프레임워크들의 활용법을 손쉽게 학습할 수 있다.

이 책을 출간하면서 유일하게 걱정스러운 점은 이제 겨우 네 번째 번역서를 출간하는 초보 역자의 어설픈 번역이 이 책의 가치를 독자 여러분께 제대로 전달할 수 있을지 여부이다. 모쪼록 이 책이 대한민국 .NET 개발자들의 기술 향상과 가치관의 변화를 이끌 수 있기를 바라며, 번역하는 동안 물심양면으로 역자를 지원해준 아내 지영과 예린이, 은혁이에게 아빠의 이름이 새겨진 12번째 책을 바친다.

2011년 5월

장현희 _ aspnetmvp@gmail.com

감사의 글

영화와 마찬가지로 책 역시도 표지에 미처 언급되지 못한 많은 사람들의 도움 없이 만들어질 수 없다.

우선 내게 이 책을 쓸 수 있는 기회를 주었으며, 내가 마음에 들어 하는 여러 기술 서적들을 집필하고 출간한 출판자인 데이브와 앤디에게 감사의 말을 전하고 싶다. 이들은 내게 재능 있는 편집자인 수잔나 프팔자를 소개해 주었다. 수잔나는 이 책을 쓰는 내내 내게 조언을 아끼지 않았으며, 그녀의 격려와 건설적인 방향의 피드백 덕분에 이 책을 최고의 내용들로 구성할 수 있었다. "고마워요, 수잔나!"

일개 개발자가 저자로 변신하는 데 롤 모델이 되어준 클린턴 비긴과 마이크 메슨에게 감사한다. 이 책을 쓸 수 있도록 도움을 준 ToughtWorks 사가 주관하는 멤버십 프로그램인 ToughtWorks University 제7기 회원들인 수미트 모거, 크리스넌 네어, 딥시 찬드라무이, 마이클 애글러, 딥시 파우아, 그리고 리스트 위어스머에게도 감사하다고 전하고 싶다. 또한 열정적인 소프트웨어 전문가들과 매일 함께 일할 수 있는 기회를 주신 ToughtWorks 캐나다 지사의 모든 분들께 감사한다.

또한 이 책의 예제 코드와 튜토리얼을 다듬기 위한 검수 작업에 많은 분들이 도움을 주셨다. 내게 파스칼 디버깅을 가르쳐 주었을 뿐 아니라 이 책의 예제 코드를 한 줄 한 줄 검토해 준 나의 오랜 친구 데이비드 카메론, 이 책의 세 번째 파트에 많은 도움을 준 끈기 있는 개발자 스캇 무크, 이 책을 검수하는 동안 끊임없이 하드론 입자 가속기 덕분에 지구가 블랙홀로 빨려 들어가지 않는 이유를 설명해 준 개발자 존 핀레이, 영문법을 가르쳐 준 루마니아의 라두 무레산, ToughtWorker 사의 일원으로 내게 수많은 아이디어를 제공해 준 제니퍼 스미스에게 감사한다. 그 외에도 감수에 참여해 준 많은 분들이 이 프로젝트의 여러 부분에 대해 피

드백을 제공해 주었다. 푸닛 고얄, 테드 뉴워드, 시바 핀나카, 폴 레이머, 라비 쿠마르 파스마시, 싱그루이 페이, 제프 코헨, 조 푼, 엘렌 플루케스와 샤란 카란스에게 감사를 전하고 싶다.

이 책을 쓰는 동안 모든 것을 지원해 준 나의 가족에게 커다란 감사를 드린다. 항상 나를 응원해 준 나의 아내 니키 리키. "리키, 당신은 내가 아는 사람 중에 가장 놀라운 사람이야." 몇 년 전 직접 책을 집필하셨던 나의 아버지 조크 맥크라켄은 꿈을 이루기 위한 나의 노력을 항상 지원해 주셨다. 또한 내게 많은 지도와 영감을 주었던 DK 싱에게도 감사한다.

마지막으로 필자가 이 책을 즐겁게 쓴 만큼 즐겁게 이 책을 읽어줄 독자 여러분에게 감사한다. 이 책이 ASP.NET MVC와 TDD를 학습하기 위한 여러분의 도전에 도움이 되기를 바란다.

조나단 맥크라켄 _ jon@nexicon.ca

이책에 대하여

테스트가 용이하다. 가볍다. 오픈 소스이다. 마이크로소프트 제품이다? 그렇다. ASP.NET MVC는 애자일 소프트웨어 개발자들의 수요를 충족시키기 위해 마이크로소프트가 개발한 오픈 소스 웹 애플리케이션 프레임워크이다. ASP.NET MVC는 2009년 초에 공식적으로 릴리즈된 이후 약 백만 명의 개발자들이 다운로드했으며, 효율적인 개발 모델 덕분에 수많은 조직에 빠르게 확산되었다. 간단히 말하자면 ASP.NET MVC는 웹을 위한 C#이다.

이 책을 통해 ASP.NET MVC 기반의 테스트 주도 접근법을 학습한다면 여러분은 차세대 웹 애플리케이션 개발을 위한 최신의 기술을 갖춘 애자일 개발자로 거듭날 수 있다.

ASP.NET MVC를 특별하게 만드는 것들

마이크로소프트는 두 개의 웹 프레젠테이션 프레임워크를 제공하는데, 그 중 하나는 ASP.NET 웹 폼이며, 다른 하나는 ASP.NET MVC이다. ASP.NET 자체는 공통의 라이브러리와 기능을 제공하며, ASP.NET 웹 폼과 ASP.NET MVC는 이를 토대로 구현된다. 이런 구조 덕분에 기존의 ASP.NET 웹 폼을 사용하는 기존 고객의 요구와 ASP.NET MVC를 위한 미래의 요구를 모두 지원할 수 있다. ASP.NET MVC는 ASP.NET과 상당 부분을 공유하고 있기는 하지만 ASP.NET의 여러 가지 약점을 극복하고 있다. ASP.NET MVC는 웹 애플리케이션을 구현하기 위한 가장 최신의 방법론들을 바탕으로 디자인된 프레임워크로 여러분의 생산성을 놀라울 정도로 향상시킬 수 있다.

ASP.NET 웹 폼과는 차별화된 ASP.NET MVC의 장점은 다음과 같다.

마크업에 대한 완벽한 제어

여러분이 ASP.NET 웹 폼을 이용하여 애플리케이션을 개발해 본 경험이 있다면 인터넷 익스플로러 이외의 다른 브라우저도 지원할 수 있는 ASP.NET 웹 폼 애플리케이션을 구현하는 것이 얼마나 어려운 일인지를 잘 알고 있을 것이다. 그 이유는 ASP.NET이 통상적으로 한 가지 종류의 브라우저를 사용하도록 규정된 인트라넷 웹 애플리케이션을 구현하기에 적합하도록 디자인되었기 때문이다. 그러나 요즘의 대부분의 기업들은 더 이상 한 가지 브라우저만을 사용할 수는 없다. 많은 기업들이 자신들의 파트너나 고객들이 웹을 통해 필요한 업무를 수행할 수 있기를 바라기 때문에 여러 종류의 브라우저를 지원할 필요가 있다.

ASP.NET의 약점은 웹 폼이 스스로 HTML 코드를 만들어 낸다는 점이다. ASP.NET은 페이지에 포함된 웹 컨트롤과 사용자 정의 컨트롤을 통해 복잡한 HTML 코드를 생성한다. 그에 비해 ASP.NET MVC는 보다 간결한 해결책을 제시한다. 약간은 혼돈스러운 웹 폼 뷰 엔진(Web Form View Engine)이라는 이름을 가진 ASP.NET MVC의 기본 뷰 엔진은 개발자가 직접 HTML 마크업을 제어할 수 있는 방법을 제공한다. 이제 더 이상 $와 _ 기호가 붙여진 이상한 ID 값을 볼 필요가 없어졌다. 이런 ID는 오히려 자바스크립트와 같은 클라이언트 스크립팅을 더욱 어렵게 만들 뿐이다. 웹 폼 뷰 엔진에 대해서는 제7장에서 보다 자세히 살펴볼 것이다.

테스트 용이성

테스트를 지원하는 웹 애플리케이션 프레임워크를 이용하면 많은 시간을 절약할 수 있다. ASP.NET 웹 폼 애플리케이션을 개발하는 대부분의 개발자들은 테스트 용이성을 확보하기 위해 모델-뷰-프레젠터(Model-View-Presenter)와 같은 독자적인 패턴들을 활용한다. 그러나 단위 테스트에 대해 잘 모르는 개발자들 입장에서는 테스트를 위한 이 같은 접근법이 명확하지 않다. ASP.NET MVC는 이러한 문제를 해결하기 위해 코드를 테스트하는 명확한 방법을 제공한다. 필자는 이 책을 통해 ASP.NET MVC 애플리케이션을 테스트가 용이하도록 만드는 방법에 대해 심도 깊게 설명할 것이다.

설정이 아닌 규칙에 의존

규칙을 따르는 것 역시 시간을 절약하는 방법 중 하나이다. ASP.NET MVC가 제공하는 규칙들은 설정 파일을 필요로 하지 않으며, 일부 규칙들은 검색 엔진 최적화(SEO: Search

Engine Optimization)와 같은 혜택들을 제공하기도 한다. 예를 들어 ASP.NET MVC로 구현된 URL들은 검색 엔진이 읽고 판단하기가 더욱 쉽다. ASP.NET 웹 폼에서 사용하는 `http://yourblog.com/Blog/Entry.aspx?id=108`과 같은 형식의 URL 대신 ASP.NET MVC에서는 `http://yourblog.com/Blog/Entry/108/MVC-Makes-Search-Engines-Happy`와 같은 형식의 URL을 구현할 수 있다. 물론 ASP.NET 웹 폼에서도 이와 같은 URL 형식을 구현할 수는 있지만 ASP.NET MVC에 비해 직관성이 떨어진다.[주1] ASP.NET MVC를 사용하면 이러한 장점을 자유롭게 활용할 수 있다. ASP.NET MVC가 제공하는 여러 가지 규칙에 대한 보다 자세한 내용은 두 번째 파트인 "애플리케이션의 구현"에서 설명한다.

확장 가능한 아키텍처

규칙과 확장성 사이의 균형을 맞추는 일은 웹 프레임워크를 구현하는 데 있어 매우 복잡한 작업이다. 너무 많은 규칙을 제공하게 된다면 필요할 때 프레임워크를 확장하기가 어렵게 된다. 그 반대의 경우도 마찬가지다. 규칙이 전혀 없다면 팀원들이 제멋대로 코드를 작성하게 될 것이다.

ASP.NET MVC는 이런 면에서 균형을 잘 맞추고 있다. ASP.NET MVC는 강력한 기본 뷰 엔진을 제공하지만 이를 손쉽게 확장하거나 직접 새로운 뷰 엔진을 구현할 수도 있다. 뷰 엔진에 대해서는 섹션 6.2 "사용자 정의 HTML 헬퍼 메서드 구현하기"에서 더 자세히 살펴보기로 한다. ASP.NET MVC는 액션 필터(Action Filter)라고 불리는 기능을 제공하며, 이 기능을 확장하면 트랜잭션 지원과 같은 유용한 기능을 추가로 제공할 수 있다. 액션 필터에 대해서는 섹션 9.4 "사용자 정의 액션 필터의 구현"에서 더 자세히 살펴본다. ASP.NET MVC의 아키텍처는 컨트롤러의 생성을 단 한 곳에서 처리하기 때문에 의존성 주입(DI: Dependency Injection)을 활용하여 이 기능을 확장할 수 있다. 의존성 주입을 활용하면 객체의 동작, 보다 정확히 이야기하면 객체의 동작에 대한 실제 구현을 분리할 수 있다. 우리는 객체의 동작을 생성자에 전달하여 효과적으로 객체 내에 '주입' 할 수 있다. 의존성 주입에 대해서는 섹션 5.1에 있는 "IControllerFactory: 컨트롤러가 탄생하는 곳"에서 살펴보도록 한다.

주1 http://weblogs.asp.net/scottgu/archive/2009/10/13/url-routing-with-asp-net-4-web-forms-vs-2010-and-net-4-0-series.aspx2 http://haacked.com/archive/2010/01/12/ambiguous-controller-names.aspx

마지막으로 ASP.NET MVC는 특정 영속성 프레임워크에 의존하지 않는다(영속성 프레임워크에 대해서는 제1장의 "무엇이든 물어보세요!" 박스를 참고하기 바란다). 사실 ASP.NET MVC에는 영속성 프레임워크가 포함되어 있지 않다. 이는 여러분이 수행할 작업에 적합한 것을 선택할 수 있도록 하기 위함이다. 이 책에서는 가장 대중적인 오픈 소스 영속성 프레임워크인 NHibernate를 이용한다. NHibernate를 사용하는 방법은 제8장 "모델에 영속성 부여하기"에서 설명한다.

왜 테스트 주도 개발이 필요할까?

테스트 주도 개발(TDD: Test-Driven Development)은 우선적으로는 실패하는 테스트 코드를 작성하는 것으로 개발을 시작하는 매우 간단한 프로그래밍 기법이다. 이 기법은 단기간 내에 프로젝트의 표준 기법으로 자리 잡았는데, 그 이유는 TDD를 통해 스스로 작성한 코드에 대한 확신을 가질 수 있기 때문이다. 지금까지 TDD 기법을 적용해 본 경험이 없다면 제2장 "테스트 주도 개발"에서 TDD를 적용하는 방법에 대해 학습할 수 있다. TDD 기법을 적용하면 코드를 작성하는 데는 더 많은 시간이 소요되지만 디버거를 이용해 문제를 찾는 시간은 현저히 줄일 수 있다.

TDD를 적용함으로써 얻을 수 있는 또 다른 장점은 프레임워크를 더욱 빨리 학습할 수 있게 된다는 점이다. 여러분이 작성한 코드가 테스트를 통과했다는 것은 여러분이 올바르게 코드를 작성했다는 것을 의미하며, 프레임워크가 제공하는 테스트들을 더 깊이 분석할 수도 있다. ASP.NET MVC는 오픈 소스이기 때문에 ASP.NET MVC를 더욱 깊이 이해하기 위해 ASP.NET MVC가 제공하는 단위 테스트 코드들을 자유롭게 살펴볼 수 있다.

또한 여러분이 ASP.NET MVC에 막 입문하는 숙련된 TDD 개발자라면 이 책을 통해 어떤 것을 어떻게 테스트할 수 있는지에 대해 학습할 수 있다.

이 책은 누구를 위한 책인가?

이 책은 마이크로소프트 개발자와 비 마이크로소프트 개발자가 모두 읽을 수 있는 책이다. 이 두 부류의 개발자들을 위한 이 책의 목적은 자신의 개발 경험에 맞추어 ASP.NET MVC 애플리케이션을 구현하는 방법을 학습할 수 있도록 돕기 위한 것이다.

마이크로소프트 프레임워크에 대한 오랜 경험을 가진 마이크로소프트 개발자라면 TDD라는 개념에 그다지 익숙하지는 않을 것이다. 이 책에서 사용된 거의 모든 코드는 TDD를 토대로 작성되었으며 TDD 위주로 설명하고 있기 때문에 개발자들은 테스트가 동작하는 방식과 ASP.NET MVC 코드가 동작하는 방식을 모두 학습할 수 있다. 또한 ASP.NET MVC 애플리케이션을 개발할 때 여러분의 시간을 절약하는 데 도움이 되는 몇 가지 도구와 오픈 소스 프로젝트들에 대해서도 학습하게 될 것이다.

비 마이크로소프트 개발자라면 코드를 테스트하는 방법은 이미 익숙하겠지만 이 책을 통해 새로운 언어와 프레임워크를 학습하게 될 것이다. 비록 이 책은 여러분이 C# 언어에 대해 기본적인 지식을 가지고 있음을 전제로 하지만, 각각의 튜토리얼들은 이 코드가 어떤 동작을 수행하며 왜 중요한지를 한 줄 한 줄 자세히 설명해 줄 것이다.

ASP.NET MVC 애플리케이션은 VB.NET으로도 개발할 수는 있지만 이 책의 모든 예제는 C#으로 작성되었다. C# 코드를 읽을 수 있고 VB.NET 코드로 변환이 가능하다면 이 책을 ASP.NET MVC 애플리케이션 개발에 대한 가이드로 활용해도 충분할 것이다.

이 책의 주요 내용

이 책의 첫 번째 파트에서는 ASP.NET MVC 애플리케이션을 구현하는 방법과 TDD 접근법에 대해 소개한다.

두 번째 파트는 예제 애플리케이션의 구현에 중점을 둔다. 여기서는 ASP.NET MVC를 구성하는 컴포넌트 중 테스트가 가능한 핵심 컴포넌트들을 통해 애플리케이션을 구현할 것이며, 이러한 핵심 컴포넌트들과 통합될 수 있는 기본적인 다른 프레임워크들에 대해서도 살펴본다. 제7장 "AJAX와 부분 뷰로 뷰 구성하기"에서는 jQuery를 활용하는 방법에 대해 소개한다.

세 번째 파트는 동일한 애플리케이션을 다른 프레임워크들을 통해 구현하는 방법을 소개한다. ASP.NET MVC의 데이터베이스 액세스는 매우 유연해서 제8장 "모델에 영속성 부여하기"에서는 NHibernate 프레임워크를 활용하는 방법을 학습하게 될 것이며, 제9장 "컨트롤러와 저장소와의 통합"에서는 Castle 프로젝트의 Windsor 컨테이너를 활용하는 방법을 학습한다. 또한 제10장 "REST 웹 서비스 구축하기"에서는 서로 다른 애플리케이션들을 통합하기 위해 REST 웹 서비스를 구현하는 방법에 대해 살펴본다.

네 번째 파트는 많은 개발자들이 고민하고 있는 배포에 대한 이야기를 풀어본다. 제12장 "빌드와 배포"에서 이 부분에 대해 심도 깊게 살펴보도록 하자. 또한 제11장 "보안, 에러 처리, 그리고 로깅"에서는 웹 애플리케이션 구현에 대한 기타 여러 가지 내용들을 다룬다.

이 책을 제대로 활용하기 위해서는 책을 읽는 동안 코드를 직접 작성해 보기를 강력히 권장한다. 그렇게 하면 프레임워크의 개념을 학습하고 각 테스트 코드의 미묘한 차이점을 이해하는 데 도움이 될 뿐 아니라 더 나아가 테스트 주도 개발 방법론을 마스터할 수 있게 된다. 테스트 주도 개발은 여러분이 프로그램 작성에 사용하는 모든 언어에 활용할 수 있는 기법이다. C#, 자바 혹은 루비 등 여러분이 어떤 프로그래밍 언어를 사용하든 테스트 코드를 작성하는 방법을 이해하는 것은 보다 짧은 시간에 높은 품질의 코드를 작성할 수 있는 지름길이다.

ASP.NET MVC 2.0의 새로운 기능들

2009년 3월 ASP.NET MVC 1.0이 발표된 후 레드몬드(옮긴이_마이크로소프트 미국 본사가 위치한 지명)의 개발 팀은 이 프레임워크의 2.0 버전 개발에 박차를 가했다. 혁신보다는 진화를 위해 개발된 2.0 버전에서는 뷰와 모델을 더욱 손쉽게 구현할 수 있게 되었다. ASP.NET MVC 2.0의 새로운 기능에 대해 간략히 살펴보자.

강력하게 형식화된 HTML 핼퍼

새로운 핼퍼들은 컴파일 시점의 오류와 뷰를 구성하는 코드의 양을 줄일 수 있도록 설계되었다. 핼퍼 메서드들은 실행 시점에 속성을 검사하던 것에서 더욱 발전하였다. 예를 들어 ASP.NET MVC 1.0에서는 `Person` 객체의 `Name` 속성을 렌더링하는 텍스트 상자를 아래와

같이 작성하였다.

```
Html.TextBox("Name");
```

이 표준 Html 헬퍼는 텍스트 상자를 렌더링한다. 이 텍스트 상자는 모델 객체의 Name 속성에 연결되어 값을 채우면 모델 자체가 업데이트된다. ASP.NET MVC 2.0에서는 다음과 같이 코드를 작성할 수 있다.

```
Html.EditorFor(person => person.Name);
```

여기서 EditorFor 메서드는 Person 객체를 위한 텍스트 상자를 렌더링하며, Person 객체에 Name 속성이 존재하는지를 컴파일 시점에 검사한다. 컴파일 시점에 검사가 이루어지기 때문에 여러분은 이 코드가 올바르게 실행되는지 여부를 미리 알 수 있다. 또한 뷰가 참조하는 모델의 속성 이름을 변경할 때도 도움이 된다.

```
Html.EditorFor<Person>(person => person);
```

EditorFor() 메서드는 Person 객체의 모든 속성을 검사하고 각 속성을 편집하기 위한 텍스트 상자를 모두 렌더링할 수 있다. 이런 경우, 람다 식에서는 하나의 속성이 아닌 전체 모델 객체를 전달한다. DisplayFor() 헬퍼 메서드를 사용하는 방법은 섹션 1.3 "5분 기초 학습: 오늘의 명언 애플리케이션"에서 살펴보도록 하자.

템플릿 뷰

템플릿 뷰는 강력하게 형식화된 뷰 헬퍼를 토대로 구현된다. ASP.NET MVC 2.0에서는 사용자 정의 뷰를 구현하기 위한 뷰의 템플릿을 정의할 수 있다. 이 기능은 개발자가 올바른 방향으로 개발하고 있는지를 확인하기 위해 고객들로부터 피드백을 받을 필요가 있는 페이지처럼 애플리케이션의 프로토타입을 구현할 때 유용하게 활용할 수 있다. 템플릿을 구현하는 것은 View/Shared 디렉터리에 컨트롤러의 액션과 동일한 이름을 가진 뷰를 만드는 것만큼이나 간단하다. ASP.NET MVC에서는 모델별로 뷰를 만드는 대신 템플릿 뷰를 사용할 수 있다. 이 기능에 대해서는 섹션 4.3에 나오는 "템플릿 뷰를 활용하기"에서 살펴보도록 하자.

데이터 주석

데이터 주석(Annotation)은 모델에 유효성 규칙을 지정하기 위한 방법이다. 예를 들어 사용

자의 이름이 25자 이상을 넘으면 안 되는 경우에는 다음과 같은 특성을 지정하면 된다.

```
[StringLength(25, ErrorMessage="이름은 25자를 초과할 수 없습니다.")]
public string Name { get; set; }
```

StringLength 특성은 문자열의 최대 길이를 25자로 지정하며, ErrorMessage 속성 값은 사용자가 25자를 초과하는 값을 입력했을 때 보여줄 에러 메시지를 지정한다. 데이터 주석에 대한 보다 자세한 내용은 섹션 6.4 "ModelStateDictionary 객체를 이용한 유효성 검사"에서 살펴보도록 하자.

기타 기능들

ASP.NET MVC 2.0에는 Area와 비동기 컨트롤러 및 Html.RenderAction() 등과 같은 유용한 기능들이 새롭게 추가되었다. 이들은 대체로 고급화 혹은 특화된 기능이므로 이 책에서는 언급하지 않는다. Area는 ASP.NET MVC 프로젝트로 대형 웹 애플리케이션을 구현할 경우에 파일 구조를 확장하기 위한 방법이다(필 해크(Phil Haack)의 블로그[주2]에서 이 기능을 활용하는 방법에 대해 확인할 수 있다). 비동기 컨트롤러는 장시간 실행되는 작업을 병렬적으로 실행하기 위한 기능이다. 마지막으로 Html.RenderAction() 메서드는 HTML을 응답 스트림에 보다 효과적으로 출력하기 위한 기능이다.

온라인 리소스

이 책을 위한 웹 사이트 http://pragprog.com/titles/jmasp에서는 다음과 같은 내용들을 확인할 수 있다.

- 이 책의 두 번째 파트 및 세 번째 파트에서 구현하는 예제 애플리케이션을 포함한 모든 소스 코드를 다운로드할 수 있다. 예제 중 GetOrganizedFinal 폴더에서는 최종적으로 구현된 애플리케이션의 소스 코드가 제공된다.
- 정오표가 제공되며, 책에서 발견된 오탈자를 여러분이 직접 등록할 수 있다.
- 다른 ASP.NET MVC 개발자들과 직접 대화할 수 있는 포럼이 제공된다.

주2 http://haacked.com/archive/2010/01/12/ambiguous-controller-names.aspx

- 번역서에 대한 문의는 역자 이메일(aspnetmvp@gmail.com)이나 제이펍 출판사의 이메일(jeipub@gmail.com)로 보내주면 빠른 시간 안에 해결할 수 있도록 하겠다. 번역서에 대한 정오표는 제이펍 출판사(www.jeipub.kr)의 오탈자 정보 카테고리에서 확인할 수 있다.

추가로 이 책의 마지막 섹션 12.3에 나오는 "수고하셨습니다"에서는 여러분의 추가적인 학습에 필요한 또 다른 온라인 리소스들을 소개한다.

이 책의 소스 코드는 여러분의 애플리케이션에서 자유롭게 사용할 수 있다. 그러나 이 책의 일부 예제들은 단지 학습을 위한 것이므로 실제로 서비스될 코드에 적용하기에는 적합하지 않을 수 있음을 명심하기 바란다.

그러면 이제 제1장 "ASP.NET MVC 시작하기"를 통해 간단한 웹 애플리케이션을 구현해 봄으로써 ASP.NET MVC를 알아가 보도록 하자. 이어서 제2장 "테스트 주도 개발"에서는 더욱 효율적인 형태로 애플리케이션을 개발하기 위한 기본적인 내용들을 학습할 것이다. 이러한 내용을 학습하면 이 책의 나머지 내용을 토대로 완벽한 기능을 제공하는 최종 예제 애플리케이션을 구현할 수 있을 것이다.