

ASP.NET Core 공부하기

먼저 아래의 주소에서 비주얼 스튜디오를 설치해야 한다.
윈도우나 macOS에서 다운로드를 받으면 된다.

<https://visualstudio.microsoft.com/ko/thank-you-downloading-visual-studio-mac/?sku=communitymac&rel=16>

무엇을 설치하시겠습니까?



Mac용 Visual Studio

.NET을 사용하여 웹, 모바일, 데스크톱에서 앱과 게임을 만듭니다. Unity, Azure, 기본적으로 포함됩니다.

대상



.NET

오픈 소스, 플랫폼 간 .NET Framework SDK입니다.



Android

Android, Android Wear, Android TV 앱을 만들기 위한 Xamarin SDK입니다. Xamarin.Forms 지원이 포함됩니다.



iOS

iOS, watchOS, tvOS 앱을 만들기 위한 Xamarin SDK입니다. Xamarin.Forms 지원이 포함됩니다.



macOS (Cocoa)

AppKit를 사용하여 네이티브 macOS 앱을 만들기 위한 Xamarin SDK입니다.

취소

Visual Studio와 함께 다운로드한 일부 소프트웨어는 [타사 고지 사항](#) 또는 해당 라이선스에 명시된 것처럼 별도로 허가됩니다. 계속하면 이러한 라이선스에도 동의하게 됩니다.

약 2.3기가 정도 설치된다.

<https://m.blog.naver.com/powerqkrcjfd/222043620745>

자습서: ASP.NET Core 시작



○



○



○



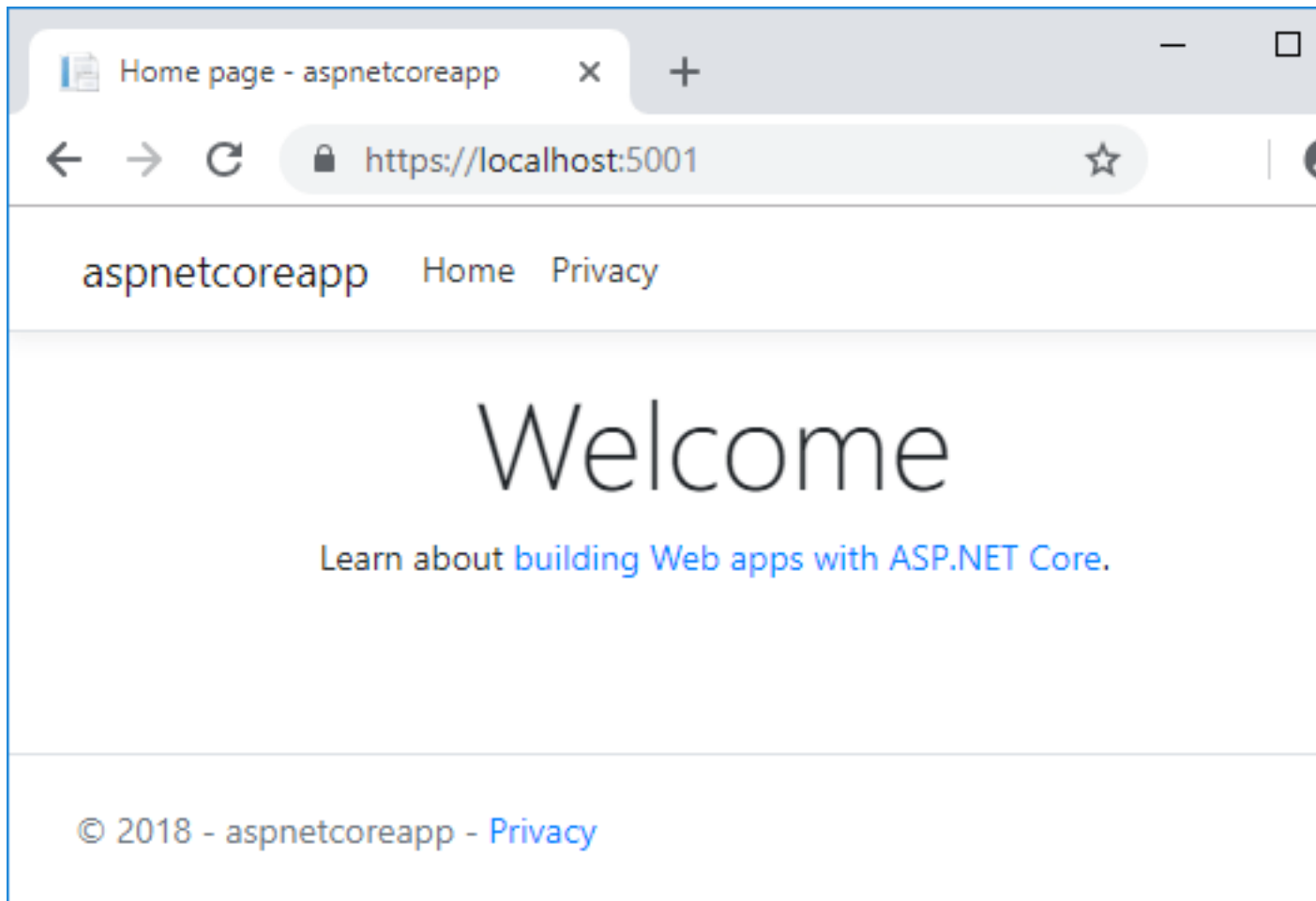
이 페이지가 도움이 되었나요?

이 자습서에서는 .NET Core CLI 를 사용하여 ASP.NET Core 웹 앱을 만들고 실행하는 방법을 보여 줍니다.

다음은 수행하는 방법을 알아봅니다.

- 웹 앱 프로젝트를 만듭니다.
- 개발 인증서를 신뢰합니다.
- 앱을 실행합니다.
- **Razor** 페이지를 편집합니다.

최종적으로 로컬 머신에서 구동되는 웹 앱을 실행할 수 있습니다.



사전 요구 사항

[.NET Core 3.1 SDK](#)

웹앱 프로젝트 만들기

명령 셸을 열고 다음 명령을 입력합니다.

```
.NET CLI 복사  
dotnet new webapp -o aspnetcoreapp
```

이전 명령은

- 새 웹앱을 만듭니다.
- -o aspnetcoreapp 매개 변수는 앱의 원본 파일을 사용하여 *aspnetcoreapp* 라는 이름의 디렉터리를 만듭니다.

개발 인증서 신뢰

HTTPS 개발 인증서 신뢰:

- [Windows](#)
- [macOS](#)
- [Linux](#)

.NET CLI복사

```
dotnet dev-certs https --trust
```

이전 명령으로 인해 다음 메시지가 표시됩니다.

HTTPS 개발 인증서를 신뢰해야 합니다. 인증서를 신뢰할 수 없는 경우 'sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain <<certificate>>' 명령을 실행합니다.

이 명령은 시스템 키 집합에 인증서를 설치하기 위해 암호를 묻는 메시지를 표시할 수 있습니다. 개발 인증서를 신뢰하는 데 동의하는 경우 암호를 입력합니다.

자세한 내용은 [ASP.NET Core HTTPS 개발 인증서 신뢰](#)를 참조하세요.

앱 실행

다음 명령을 실행합니다.

```
.NET CLI복사
cd aspnetcoreapp
dotnet watch run
```

명령 셸에서 앱이 시작되었음을 확인했으면 <https://localhost:5001>로 이동합니다.

Razor 페이지 편집

Pages/Index.cshtml 을 열고 다음에서 강조 표시된 영역처럼 페이지를 수정하고 저장합니다.

CSHTML복사

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
```

```
<div class="text-center">
  <h1 class="display-4">Welcome</h1>
  <p>Hello, world! The time on the server
is @DateTime.Now</p>
</div>
```

https://localhost:5001 로 이동하여 페이지를 새로고치고 변경 내용이 표시되는지 확인합니다.

다음 단계

본 자습서에서는 다음 작업에 관한 방법을 학습했습니다.

- 웹앱 프로젝트를 만듭니다.
- 개발 인증서를 신뢰합니다.
- 프로젝트를 실행합니다.
- 페이지를 수정합니다.

ASP.NET Core 에 대해 자세히 알아보려면 소개에서 권장되는 학습 경로를 참조하세요.

[ASP.NET Core 소개](#)

ASP.NET Core MVC 시작

- 아티클
- 2021. 10. 05.
- 읽는 데 16 분 걸림

•

○





○



○



○



이 페이지가 도움이 되었나요?

작성자: [Rick Anderson](#)

이 자습서에서는 컨트롤러와 보기를 사용하여 ASP.NET Core MVC 웹 배포에 대해 설명합니다. ASP.NET Core 웹 개발이 처음인 경우 이 자습서의 [Razor Pages](#) 버전을 통해 좀 더 쉽게 시작할 수 있습니다. UI 개발은 Razor 페이지, MVC 및 Blazor를 비교하는 [ASP.NET Core UI 선택](#)을 참조하세요.

이 자습서는 컨트롤러와 뷰를 사용한 ASP.NET Core MVC 웹 개발을 설명하는 시리즈의 첫 번째 자습서입니다.

시리즈가 끝나면 영화 데이터를 관리하고 표시하는 앱이 생성됩니다. 다음과 같은 작업을 수행하는 방법을 살펴봅니다.

- 웹앱 만들기
- 모델 추가 및 스캐폴드
- 데이터베이스 작업

- 검색 및 유효성 검사 추가

[예제 코드 살펴보기 및 다운로드 \(다운로드 방법\)](#). 다운로드 예제는 영역을 테스트하기 위한 기초적인 앱을 제공합니다.

사전 요구 사항

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)
 - ASP.NET 및 웹 개발 워크로드가 설치된 [Visual Studio 2022](#)입니다.

웹앱 만들기

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)
 - Visual Studio를 시작하고 새 프로젝트 만들기를 선택합니다.
 - 새 프로젝트 만들기 대화 상자에서 **ASP.NET Core 웹앱(Model-View-Controller) > 다음**을 선택합니다.
 - 새 프로젝트 구성 대화 상자에서 프로젝트 이름으로 **MvcMovie**를 입력합니다. 프로젝트 이름을 *MvcMovie*로 지정해야 합니다. 코드를 복사할 때 대문자 표시가 각 namespace와 일치해야 합니다.
 - 다음을 선택합니다.
 - 추가 정보 대화 상자에서 **.NET 6.0(미리 보기)**을 선택합니다.
 - 생성을 선택합니다.

Additional information

ASP.NET Core Web App (Model-View-Controller)

C#

Linux

Framework [i](#)

.NET 6.0 (Long-term support)

Authentication type [i](#)

None

☒ Configure for HTTPS [i](#)

☐ Enable Docker [i](#)

Docker OS [i](#)

Linux

프로젝트를 만드는 다른 방법은 [Visual Studio에서 새 프로젝트 만들기](#)를 참조하세요.

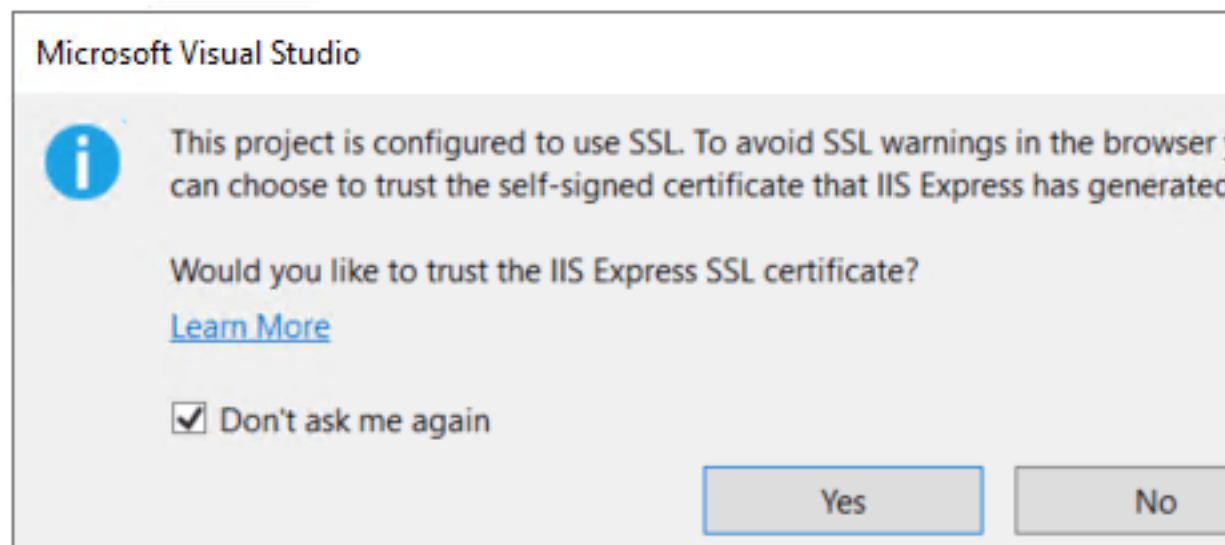
Visual Studio에서는 생성된 MVC 프로젝트에 기본 프로젝트 템플릿을 사용합니다. 생성된 프로젝트는 다음과 같습니다.

- 작동하는 앱입니다.
- 기본 시작 프로젝트입니다.

앱 실행

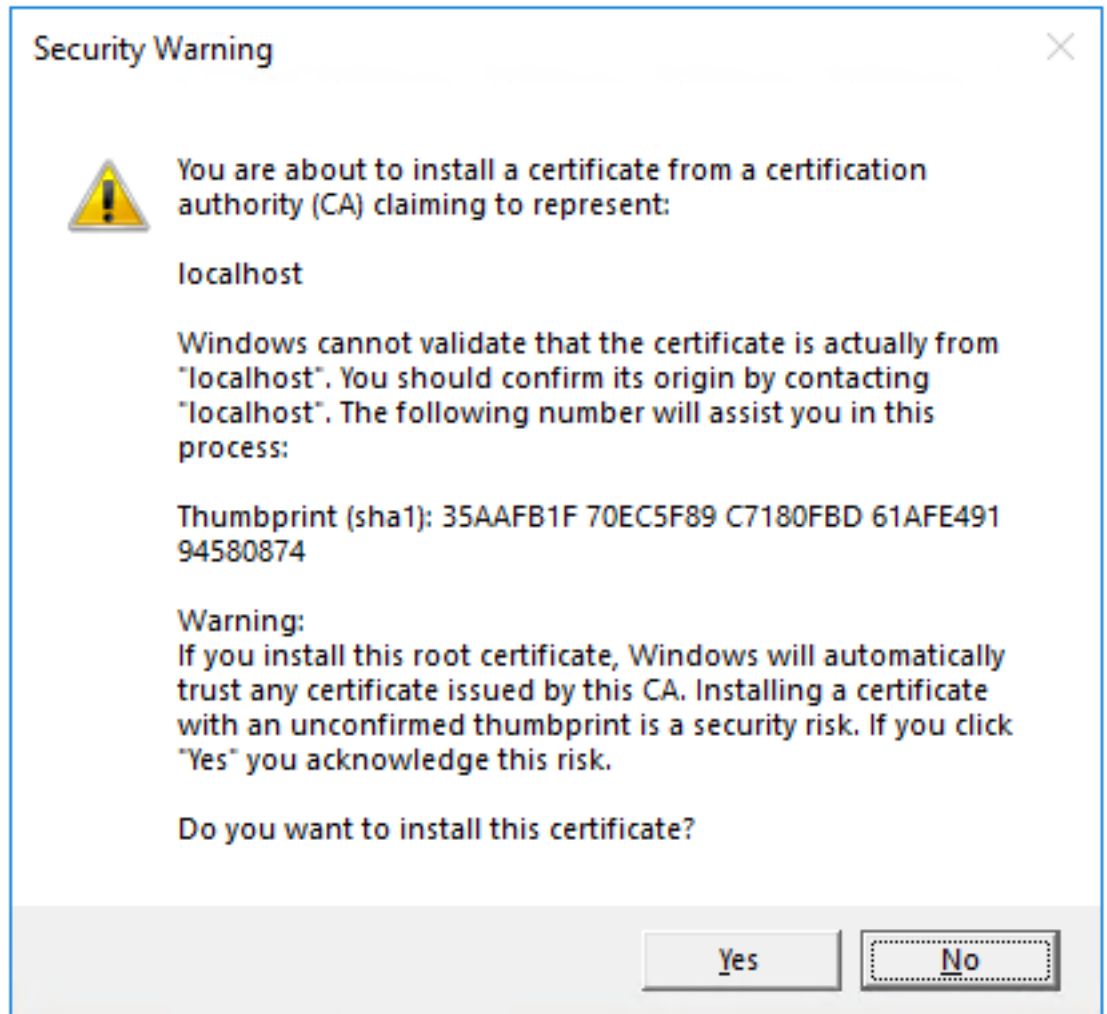
- [Visual Studio](#)
 - [Visual Studio Code](#)
 - [Mac용 Visual Studio](#)
- Ctrl+F5를 선택하여 디버거 없이 앱을 실행합니다.

프로젝트가 SSL을 사용하도록 아직 구성되지 않은 경우 Visual Studio에 다음 대화 상자가 표시됩니다.



IIS Express SSL 인증서를 신뢰하는 경우 **예**를 선택합니다.

다음 대화 상자가 표시됩니다.



개발 인증서를 신뢰하는 데 동의하는 경우 **예** 를 선택합니다.

Firefox 브라우저를 신뢰하는 방법에 대한 자세한 내용은 [Firefox SEC ERROR INADEQUATE KEY USAGE 인증서 오류](#)를 참조하세요.

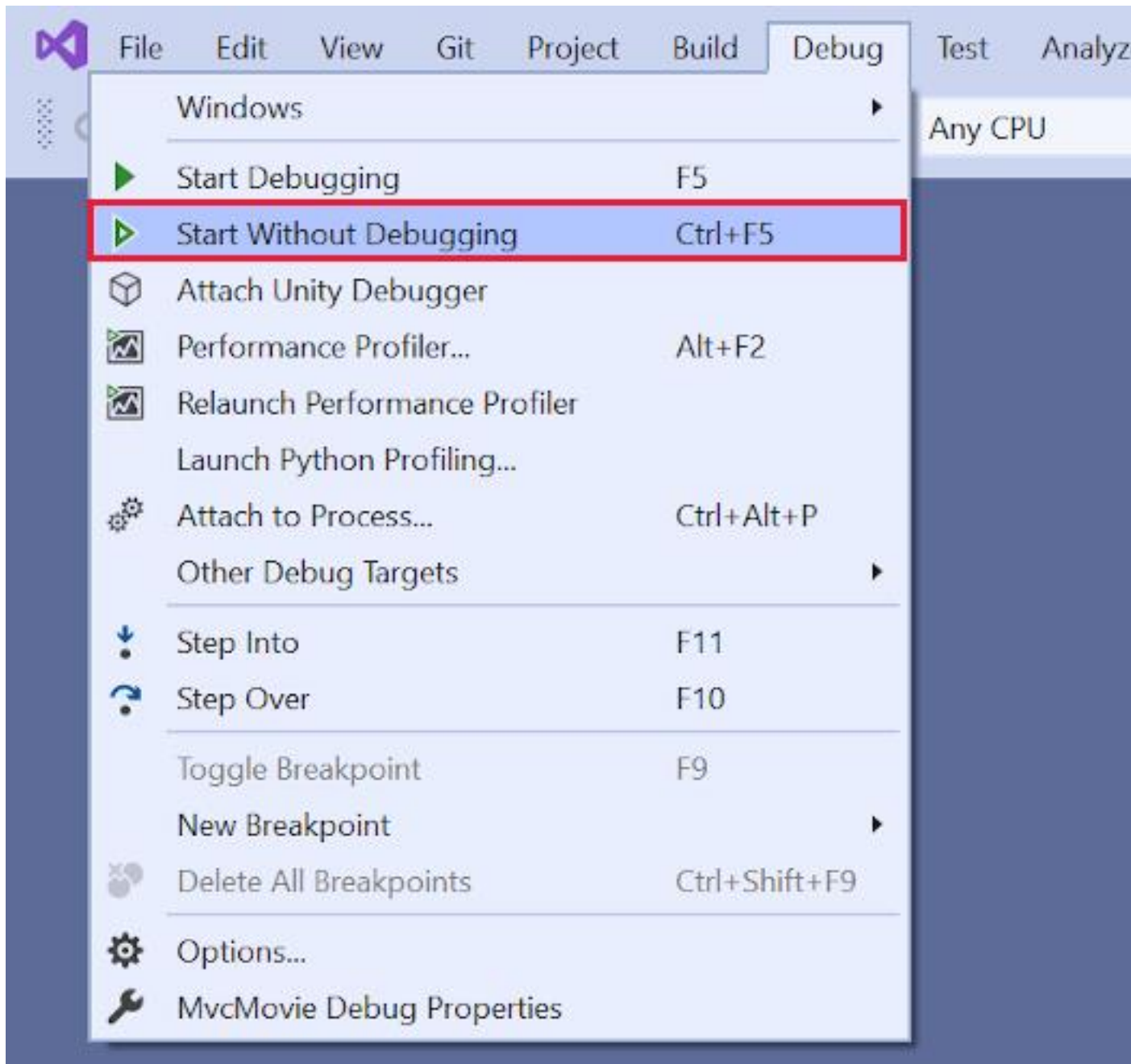
Visual Studio에서는 앱을 실행하고 기본 브라우저를 엽니다.

주소 표시줄에는 localhost:port#이 표시되고 example.com 같은 주소는 표시되지 않습니다. 로컬 컴퓨터의 표준 호스트 이름은 localhost입니다. Visual Studio가 웹 프로젝트를 만들면 임의의 포트가 웹 서버에 사용됩니다.

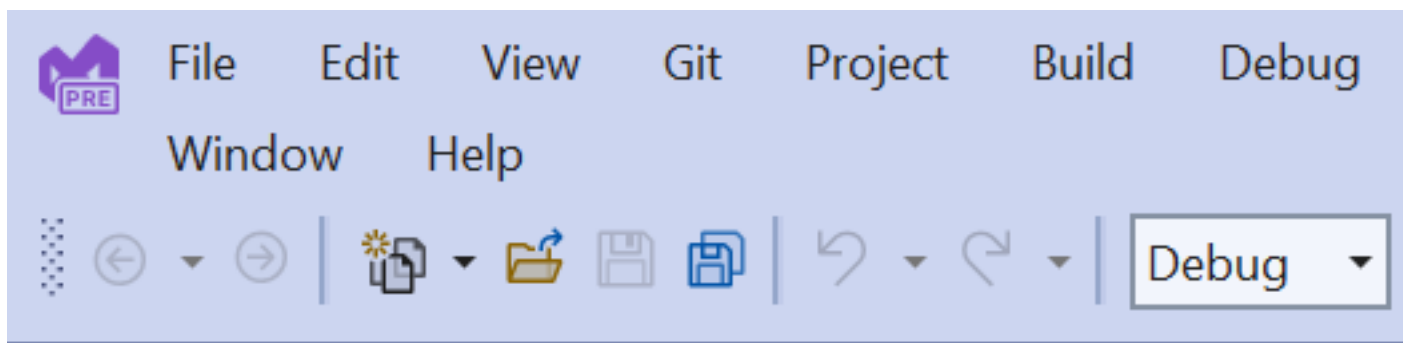
Ctrl+F5를 선택하여 디버깅 없이 앱을 시작하면 다음을 수행할 수 있습니다.

- 코드를 변경합니다.
- 파일을 저장합니다.
- 브라우저를 빠르게 새로 고치고 코드 변경 내용을 확인합니다.

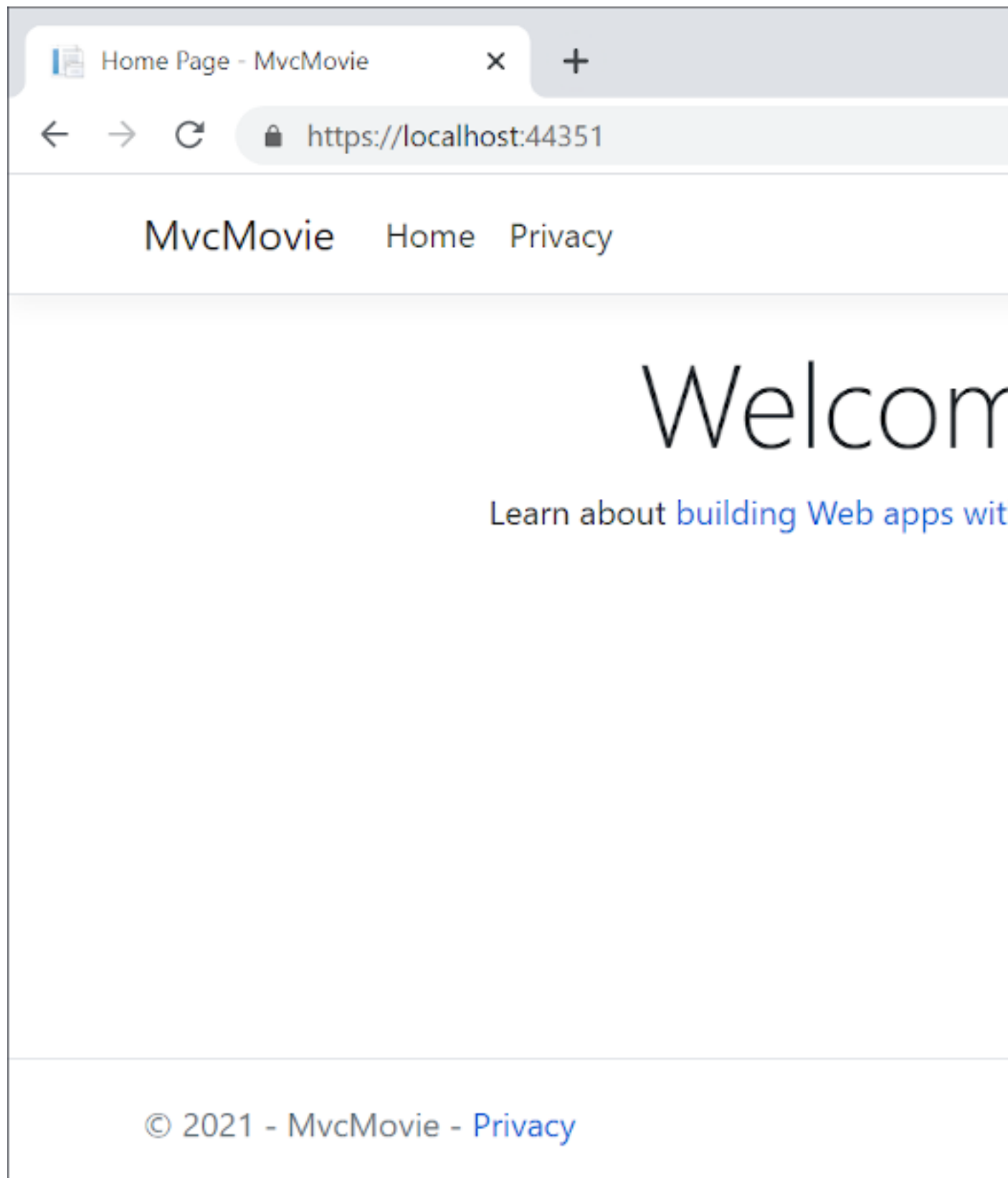
디버그 메뉴에서 앱을 디버그 또는 비 디버그 모드로 시작할 수 있습니다.



도구 모음에서 **MvcMovie** 단추를 선택하면 앱을 디버그할 수 있습니다.



다음 이미지는 앱을 보여줍니다.



- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac Visual Studio](#)

Visual Studio 도움말

- [Visual Studio를 사용하여 C# 코드를 디버그하는 방법 알아보기](#)
- [Visual Studio IDE 소개](#)

이 시리즈의 다음 자습서에서는 MVC에 대해 알아보고 코드 작성을 시작합니다.

[다음: 컨트롤러 추가](#)

권장 콘텐츠

- **3 부. ASP.NET Core MVC 앱에 뷰 추가**

ASP.NET Core MVC에 대한 자습서 시리즈의 3부입니다.
- **2 부. ASP.NET Core MVC 앱에 컨트롤러 추가**

ASP.NET Core MVC에 대한 자습서 시리즈의 2부입니다.
- **자습서: ASP.NET Core에서 Razor Pages 시작**

ASP.NET Core Razor Pages 웹앱을 빌드하는 작업의 기본 사항을 설명하는 자습서 시리즈 중 첫 번째입니다.
-

2 부. ASP.NET Core MVC 앱에 컨트롤러 추가

- 아티클
- 2021. 10. 05.
- 읽는 데 11 분 걸림

•



○



○



○



○



이 페이지가 도움이 되었나요?

작성자: [Rick Anderson](#)

MVC(Model-View-Controller) 아키텍처 패턴은 다음 세 가지 주요 구성 요소로 앱을 구분합니다. **M**odel, **V**iew 및 **C**ontroller. MVC 패턴을 통해 기존 모놀리식 응용 프로그램보다 쉽게 테스트 가능하고 업데이트할 수 있는 앱을 만들 수 있습니다.

MVC 기반 앱에는 다음이 포함됩니다.

- **Models:** 앱의 데이터를 나타내는 클래스입니다. 모델 클래스는 유효성 검사 논리를 사용하여 해당 데이터에 대한 비즈니스 규칙을 적용합니다. 일반적으로 모델 개체는 데이터베이스에서 모델 상태를 검색하고 저장합니다. 이

자습서에서 **Movie** 모델은 데이터베이스에서 영화 데이터를 검색하고 이를 뷰에 제공하거나 수정합니다. 수정된 데이터는 데이터베이스에 기록됩니다.

- **Views:** 보기는 앱의 UI(사용자 인터페이스)를 표시하는 구성 요소입니다. 일반적으로 UI는 모델 데이터를 표시합니다.
- **Controllers(컨트롤러):** 다음을 수행하는 클래스입니다.
 - 브라우저 요청을 처리합니다.
 - 모델 데이터를 검색합니다.
 - 응답을 반환하는 뷰 템플릿을 호출합니다.

MVC 앱에서는 뷰에 정보만 표시됩니다. 컨트롤러는 사용자 입력 및 상호 작용을 처리하고 응답합니다. 예를 들어 컨트롤러는 URL 세그먼트 및 쿼리 문자열 값을 처리하고 이러한 값을 모델에 전달합니다. 모델은 이러한 값을 사용하여 데이터베이스를 쿼리할 수 있습니다. 다음은 그 예입니다.

- `https://localhost:5001/Home/Privacy`: Home 컨트롤러 및 Privacy 작업을 지정합니다.
- `https://localhost:5001/Movies/Edit/5`: Movies 컨트롤러 및 Edit 작업을 사용하여 ID가 5인 동영상을 편집하는 요청으로, 자습서의 뒷부분에서 자세히 설명합니다.

경로 데이터는 자습서의 뒷 부분에서 설명합니다.




MVC 아키텍처 패턴은 앱을 모델, 뷰 및 컨트롤러의 세 가지 주요 구성 요소로 구분합니다. 이 패턴은 문제의 분리를 달성하는 데 도움이 됩니다. UI 논리는 뷰에 속합니다. 입력 논리는 컨트롤러에 속합니다. 비즈니스 논리는 모델에 속합니다. 이렇게 분리하면 다른 코드에 영향을 주지 않고 한 번에 한 구현 측면에서 작업할 수 있기 때문에 앱을 작성할 때 복잡성을 관리하는 데 도움이 됩니다. 예를 들어 비즈니스 논리 코드와 무관하게 보기 코드를 작업할 수 있습니다.

이 자습서 시리즈에서는 동영상 앱을 빌드하면서 이러한 개념을 소개하고 보여줍니다. MVC 프로젝트는 *컨트롤러* 및 *보기*를 위한 폴더를 포함하고 있습니다.

컨트롤러 추가

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)

솔루션 탐색기 에서 컨트롤러 > 추가 > 컨트롤러 를 마우스 오른쪽 단추로 클릭합니다.

-  Controller...
-  Razor Component...
-
-  New Item... Ctrl+Shift+A
-  Existing Item... Shift+Alt+A
- New Scaffolded Item...
-  New Folder
-
-  Application Insights Telemetry...
- Container Orchestrator Support...
-  Docker Support...
-
-  Client-Side Library...
-
-  Machine Learning
-
- New ASP.NET Web API Project...

새 스캐폴드 항목 추가 대화 상자에서 **MVC 컨트롤러 - 비어 있음** > 추가 를 선택합니다.

Add New Scaffolded Item

Installed

Common

API

MVC

Controller

View

Razor Component

Razor Pages

Identity

Layout



MVC Controller - Empty



MVC Controller with read/write access



MVC Controller with views, using Entity Framework

[Click here to go online and find more](#)

새 항목 추가 - MvcMovie 대화 상자에서 HelloWorldController.cs 를 입력하고 추가 를 선택합니다.

Controllers/HelloWorldController.cs 의 내용을 다음 코드로 바꿉니다.

C#복사

```
using Microsoft.AspNetCore.Mvc;  
using System.Text.Encodings.Web;
```

```
namespace MvcMovie.Controllers
```

```
{
```

```
    public class HelloWorldController :  
Controller
```

```
    {
```

```
        //
```

```
        // GET: /HelloWorld/
```

```
        public string Index()
```

```
        {
```

```
            return "This is my default  
action...";
```

```
        }
```

```
        //
```

```
        // GET: /HelloWorld/Welcome/
```

```
        public string Welcome()
```

```
        {
```

```
            return "This is the Welcome  
action method...";
```

```
        }
```

```
    }
```


}

컨트롤러의 모든 `public` 메서드는 HTTP 엔드포인트로 호출할 수 있습니다. 위의 샘플에서 메서드는 모두 문자열을 반환합니다. 각 메서드 앞의 주석에 주목하세요.

HTTP 엔드포인트:

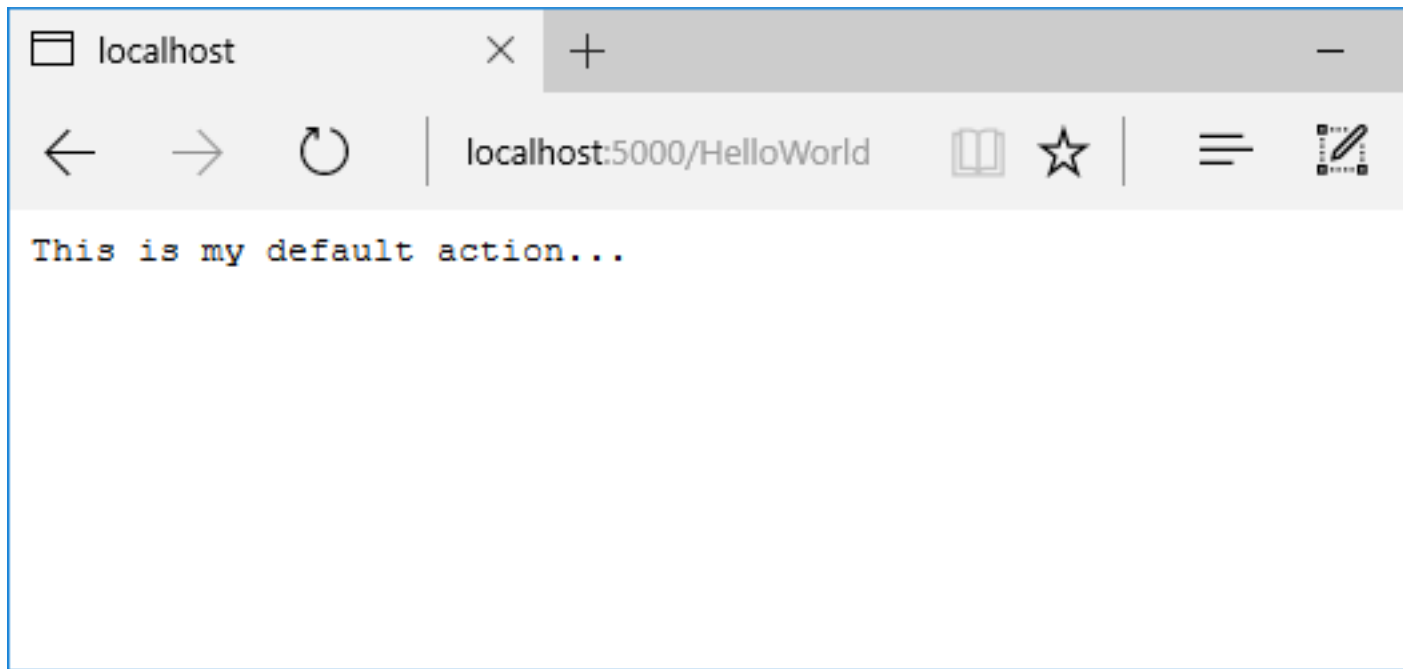
- 웹 애플리케이션에서 대상으로 지정 가능한 URL(예: `https://localhost:5001/HelloWorld`)입니다.
- 다음을 결합합니다.
 - 사용된 프로토콜: HTTPS.
 - TCP 포트를 포함한 웹 서버의 네트워크 위치: `localhost:5001`.
 - 대상 URI: `HelloWorld`.

첫 번째 주석은 이 항목이 기본 URL 에 `/HelloWorld/`를 추가하여 호출되는 [HTTP GET](#) 메서드라고 명시합니다.

두 번째 주석은 URL 에 `/HelloWorld/welcome/`을 추가하여 호출되는 [HTTP GET](#) 메서드를 명시합니다. 자습서 뒷부분에서는 스캐폴딩 엔진을 사용하여 데이터를 업데이트하는 HTTP POST 메서드를 생성합니다.

디버거 없이 앱을 실행합니다.

주소 표시줄의 경로에 “HelloWorld”를 추가합니다. Index 메서드가 문자열을 반환합니다.



MVC 는 들어오는 URL 에 따라 컨트롤러 클래스와 해당 클래스 내의 작업 메서드를 호출합니다. MVC 에서 사용하는 기본 [URL 라우팅 논리](#) 는 다음과 같은 형식을 사용하여 호출할 코드를 결정합니다.

```
/[Controller]/[ActionName]/[Parameters]
```

라우팅 서식은 *Program.cs* 파일에서 설정됩니다.

C#복사

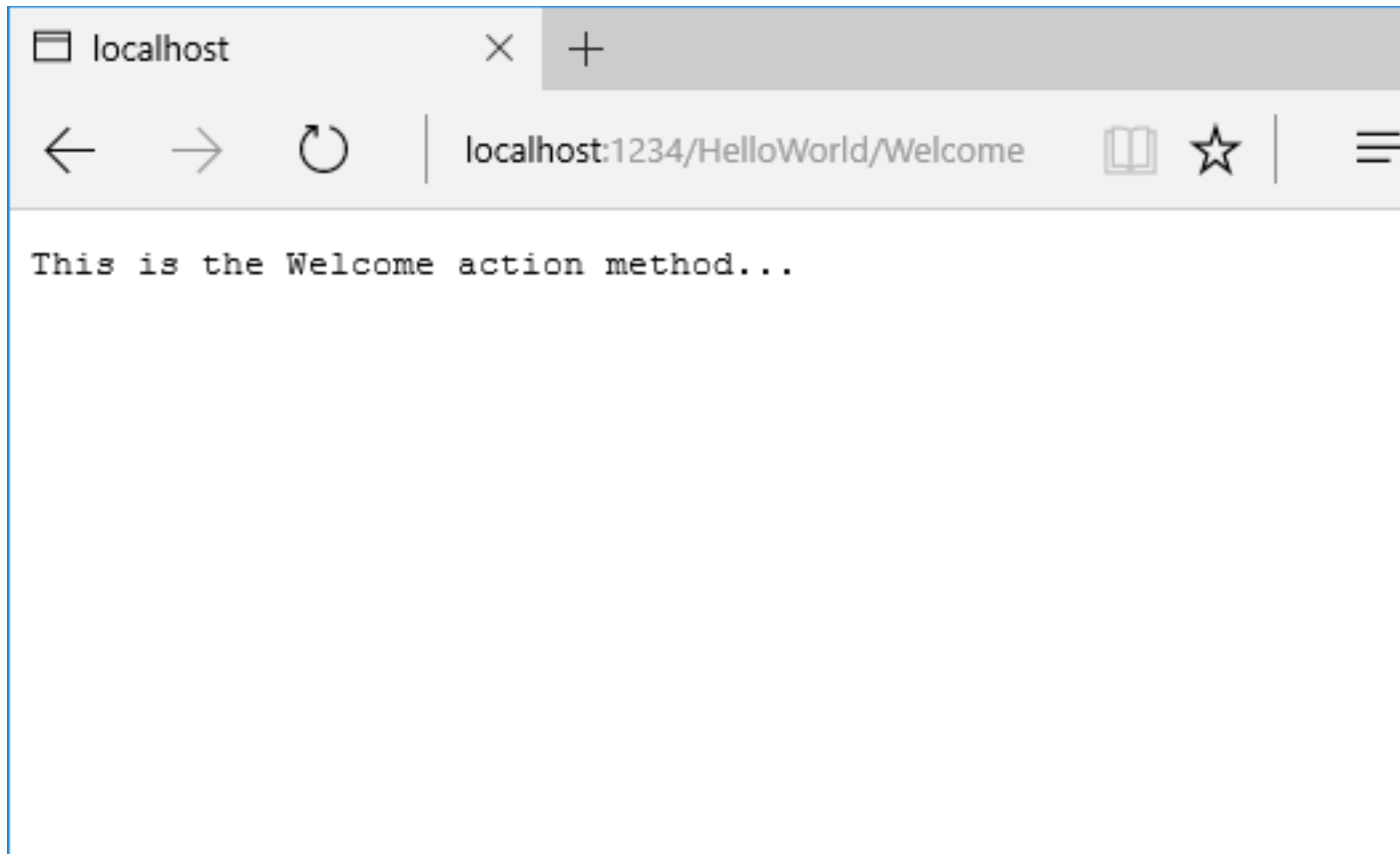
```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

앱으로 이동할 때 URL 세그먼트를 제공하지 않으면 위에 강조 표시된 템플릿 줄에 지정된 “Home” 컨트롤러 및 “Home” 메서드가 기본값으로 사용됩니다. 위 URL 세그먼트는 다음과 같습니다.

- 첫 번째 URL 세그먼트는 실행할 컨트롤러 클래스를 결정합니다. 따라서 localhost:5001/HelloWorld는 **HelloWorld** Controller 클래스에 매핑됩니다.
- URL 세그먼트의 두 번째 부분은 클래스의 작업 메서드를 결정합니다. 따라서 localhost:5001/HelloWorld/Index는 HelloWorldController 클래스의 Index 메서드가 실행되도록 합니다. localhost:5001/HelloWorld로만 이동했음에도 기본적으로 Index 메서드가 호출되었음에 주의하세요. Index는 메서드 이름이 명시적으로 지정되지 않은 경우 컨트롤러에서 호출되는 기본 메서드입니다.
- URL 세그먼트의 세 번째 부분(id)은 경로 데이터입니다. 경로 데이터는 자습서의 뒷 부분에서 설명합니다.

`https://localhost:{PORT}/HelloWorld/Welcome` 으로 이동합니다. {PORT}를 포트 번호로 바꿉니다.

Welcome 메서드가 실행되고 문자열 `This is the Welcome action method...`를 반환합니다. 이 URL의 경우 컨트롤러는 `HelloWorld` 이고 `Welcome` 이 작업 메서드입니다. 아직 URL의 [Parameters] 부분을 사용하지 않았습니다.



URL에서 컨트롤러에 일부 매개 변수 정보를 전달하도록 코드를 수정합니다.
예: `/HelloWorld/Welcome?name=Rick&numtimes=4`.

다음 코드와 같이 `Welcome` 메서드가 두 개의 매개 변수를 포함하도록 변경합니다.

C#복사

```
// GET: /HelloWorld/Welcome/  
// Requires using System.Text.Encodings.Web;  
public string Welcome(string name, int  
numTimes = 1)  
{  
    return HtmlEncoder.Default.Encode($"Hello  
{name}, NumTimes is: {numTimes}");  
}
```

}

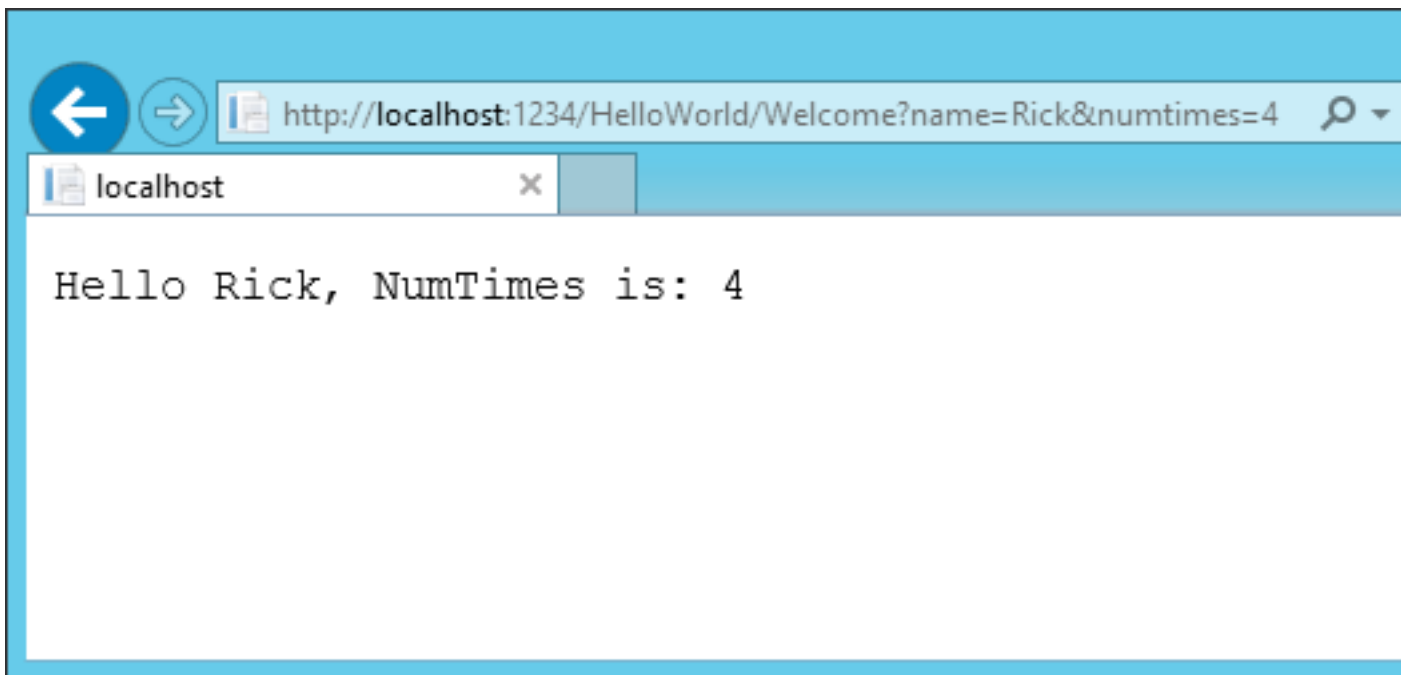
위의 코드는

- C#의 선택적 매개 변수 기능을 사용하여 numTimes 매개 변수에 대해 전달된 값이 없는 경우 해당 매개 변수의 기본값이 1임을 나타냅니다.
- `HtmlEncoder.Default.Encode`를 사용하여 JavaScript 사용과 같은 악의적인 입력으로부터 앱을 보호합니다.
- `$"Hello {name}, NumTimes is: {numTimes}"`에서 [보간된 문자열](#)을 사용합니다.

앱을

실행하고 `https://localhost:{PORT}/HelloWorld/Welcome?name=Rick&numtimes=4` 로 이동합니다. {PORT}를 포트 번호로 바꿉니다.

URL에서 name 및 numtimes에 다른 값을 사용합니다. MVC [모델 바인딩](#) 시스템은 쿼리 문자열에서 메서드의 매개 변수로 명명된 매개 변수를 자동으로 매핑합니다. 자세한 정보는 [모델 바인딩](#)을 참조하세요.



위 이미지는 다음과 같습니다.

- URL 세그먼트 Parameters가 사용되지 않습니다.
- name 및 numTimes 매개 변수는 [쿼리 문자열](#)로 전달됩니다.
- 위 URL에서 ?(물음표)는 구분 기호이며, 다음에 쿼리 문자열이 옵니다.
- & 문자는 필드-값 쌍을 구분합니다.

Welcome 메서드를 다음 코드로 바꿉니다.

C#복사

```
public string Welcome(string name, int ID = 1)
{
    return HtmlEncoder.Default.Encode($"Hello {name}, ID: {ID}");
}
```

앱을 실행하고 `https://localhost:{PORT}/HelloWorld/Welcome/3?name=Rick` URL 을 입력합니다.

위 URL 은 다음과 같습니다.

- 세 번째 URL 세그먼트는 경로 매개 변수 `id`와 일치합니다.
- `Welcome` 메서드는 `MapControllerRoute` 메서드의 URL 템플릿과 일치하는 `id` 매개 변수를 포함하고 있습니다.
- 후행 `?`는 [쿼리 문자열](#)을 시작합니다.

C#복사

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

앞의 예제에서:

- 세 번째 URL 세그먼트는 경로 매개 변수 `id`와 일치합니다.
- `Welcome` 메서드는 `MapControllerRoute` 메서드의 URL 템플릿과 일치하는 `id` 매개 변수를 포함하고 있습니다.
- 후행 `?(id?)`는 `id` 매개 변수가 선택 사항임을 나타냅니다.

[이전: 시작하기](#) [다음: 뷰 추가](#)

Movie App

- Home
- Privacy

@RenderBody()

© 2021 - Movie App - Privacy

@await RenderSectionAsync("Scripts", required: false)

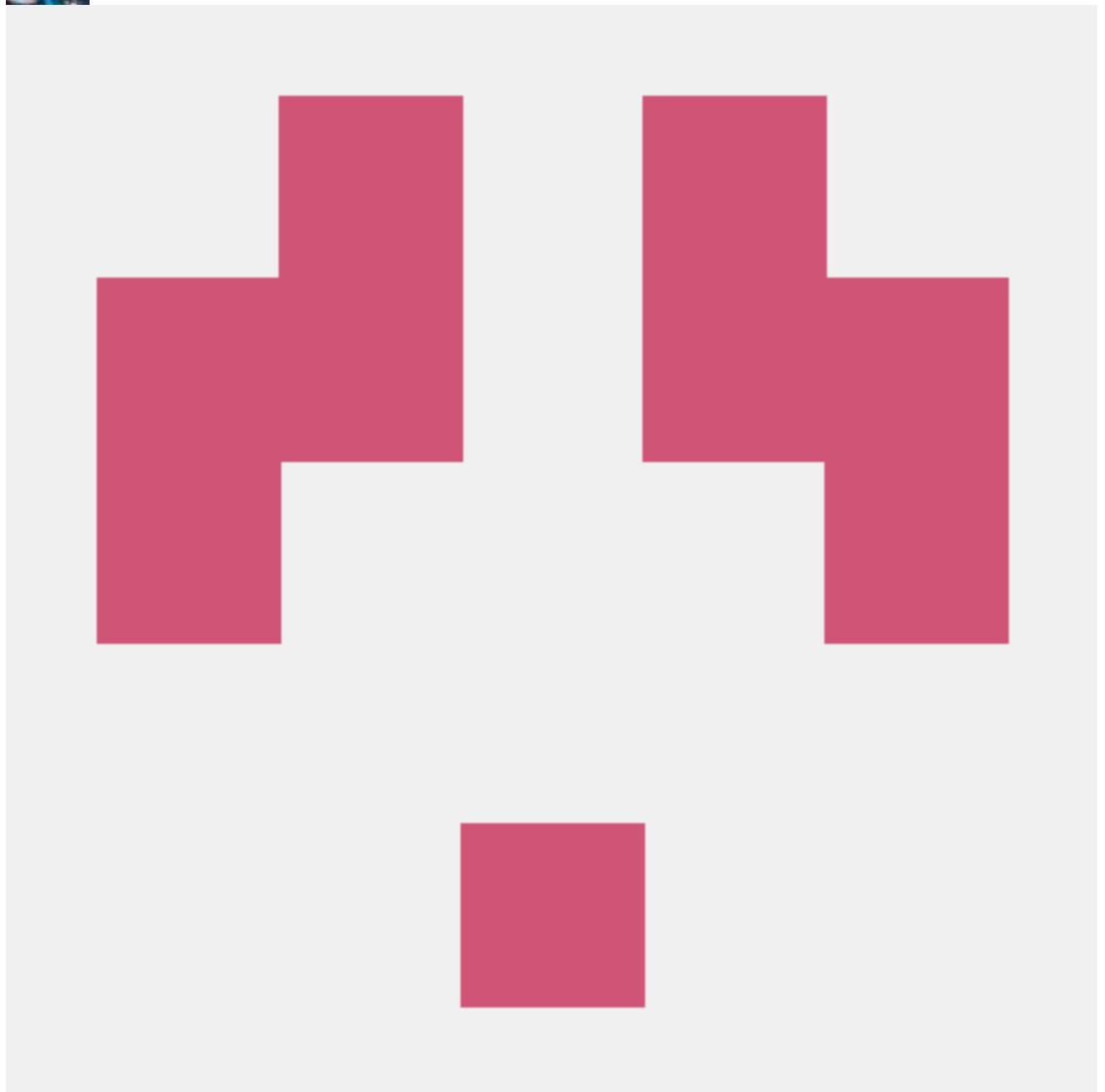
4 부. ASP.NET Core MVC 앱에 모델 추가

- 아티클
- 2021. 08. 17.
- 읽는 데 39 분 걸림

-



-



-



-



이 페이지가 도움이 되었나요?

작성자: [Rick Anderson](#) 및 [Jon P Smith](#).

이 섹션에서는 데이터베이스에서 동영상을 관리하기 위한 클래스를 추가합니다. 이러한 클래스는 **MVC** 앱의 "**M**odel" 부분입니다.

이러한 모델 클래스를 EF Core([Entity Framework Core](#))와 함께 사용하여 데이터베이스 작업을 수행합니다. EF Core는 작성해야 할 데이터 액세스 코드를 간소화하는 ORM(개체-관계형 매핑) 프레임워크입니다.

만들어진 모델 클래스를 ***POCO _** 클래스(***Plain Old CLR Objects**)라고 합니다. **POCO** 클래스는 EF Core에 대한 종속성이 없습니다. 이 클래스는 데이터베이스에 저장되는 데이터의 속성만 정의합니다.

이 자습서에서는 먼저 모델 클래스가 작성되며, EF Core가 데이터베이스를 만듭니다.

데이터 모델 클래스 추가

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)

Models 폴더> 추가> 클래스 를 마우스 오른쪽 단추로 클릭합니다. 파일 이름을 *Movie.cs* 로 지정합니다.

다음 코드로 *Models/Movie.cs* 파일을 업데이트합니다.

```
C#복사
using System;
using System.ComponentModel.DataAnnotations;
```

```
namespace MvcMovie.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string Title { get; set; }

        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get;
set; }
        public string Genre { get; set; }
        public decimal Price { get; set; }
    }
}
```

Movie 클래스에는 데이터베이스에서 기본 키에 필요한 *Id* 필드가 포함되어 있습니다.

ReleaseDate 의 [DataType](#) 특성은 날짜 형식(Date)을 지정합니다. 이 특성을 사용하면:

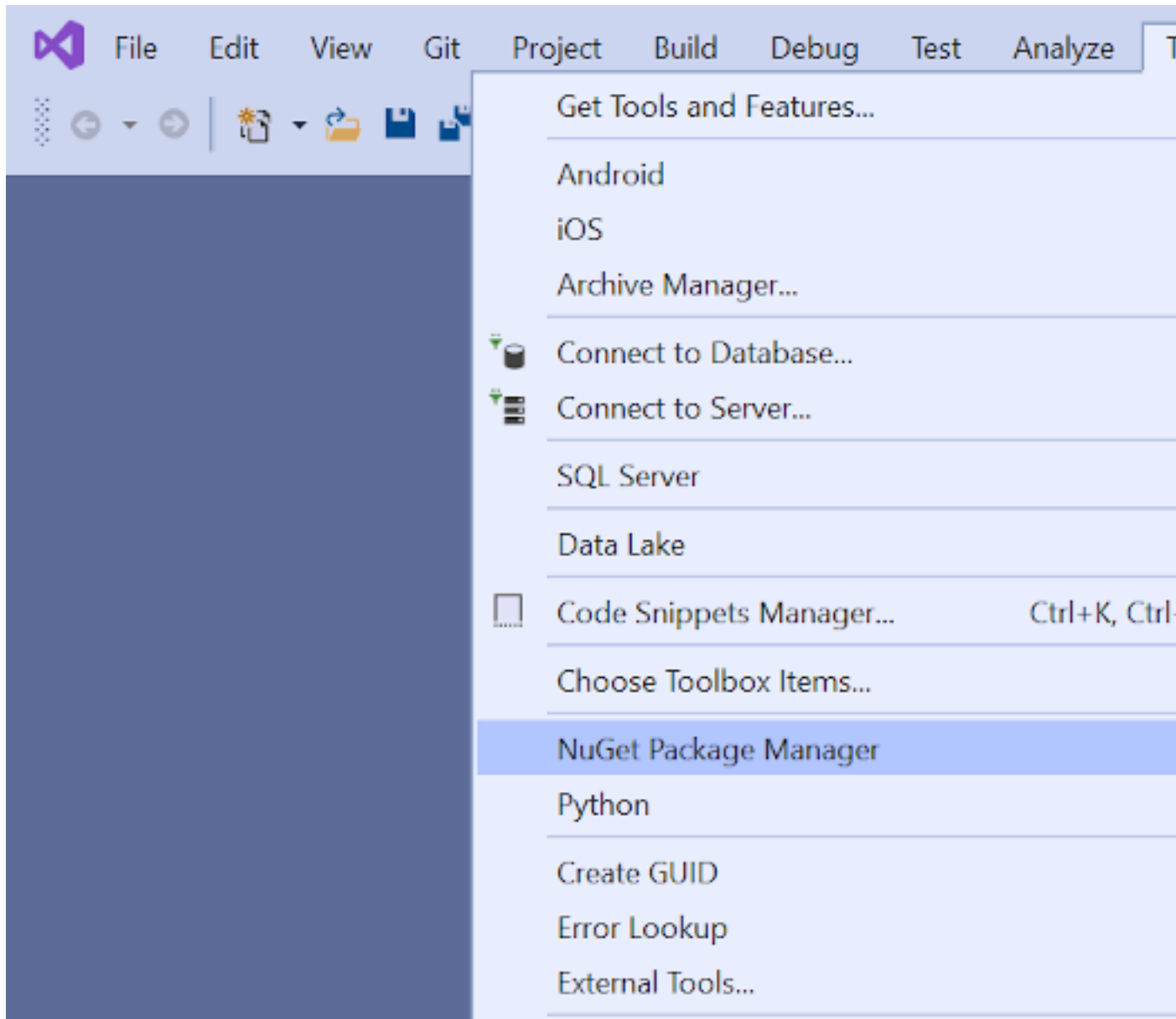
- 사용자가 날짜 필드에 시간 정보를 입력할 필요가 없습니다.
- 시간 정보 없이 날짜만 표시됩니다.

[DataAnnotations](#) 는 이후 자습서에서 다룹니다.

NuGet 패키지 추가

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)

도구 메뉴에서 **NuGet** 패키지 관리자 > **PMC**(패키지 관리자 콘솔) 을 선택합니다.



PMC 에서 다음 명령을 실행합니다.

PowerShell 복사

`Install-Package Microsoft.EntityFrameworkCore.Design`

앞의 명령은 다음을 추가합니다.

- EF Core SQL Server 공급자. 이 공급자 패키지는 EF Core 패키지를 종속성으로 설치합니다.
- 패키지에 사용되는 유틸리티(자습서 뒷부분의 스캐폴딩 단계에서 자동으로 설치됨).

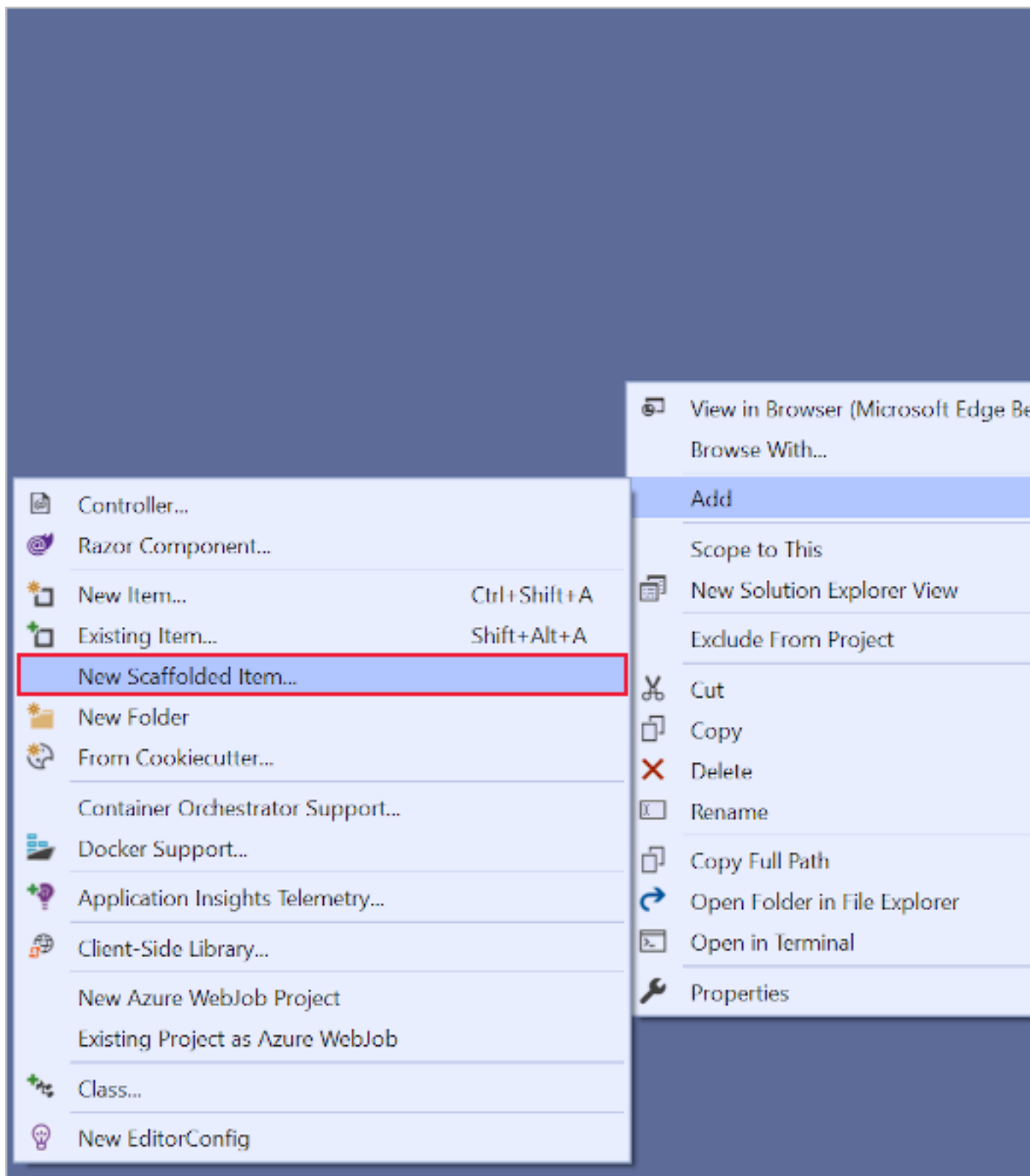
컴파일러 오류에 대한 검사로 프로젝트를 빌드합니다.

영화 페이지 스캐폴드

스캐폴딩 도구를 사용하여 영화 모델에 대한 CRUD(Create, Read, Update 및 Delete) 페이지를 생성합니다.

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)

솔루션 탐색기에서 컨트롤러 폴더를 마우스 오른쪽 단추로 클릭하고 추가 -> 스캐폴드 항목 새로 만들기 를 선택합니다.



스캐폴드 추가 대화 상자에서 보기 포함 MVC 컨트롤러, Entity Framework > 추가 사용을 선택합니다.

Add New Scaffolded Item

Installed

Common

API

▶ MVC

Razor Component

Razor Pages

Identity

Layout



MVC Controller - Empty



MVC Controller with read/write access



MVC Controller with views, using Entity Framework



Razor View - Empty



Razor View

[Click here to go online and find more](#)

다음과 같이 **Entity Framework** 를 사용하여 뷰 포함 MVC 컨트롤러 추가 대화 상자의 작업을 완료합니다.

- 모델 클래스 드롭다운에서 동영상(**MvcMovie.Models**) 을 선택합니다.
- 데이터 컨텍스트 클래스 행에서 + (더하기) 기호를 선택합니다.
 - 데이터 컨텍스트 추가 대화 상자에서 클래스 이름 *MvcMovie.Data.MvcMovieContext* 가 생성됩니다.
 - 추가 를 선택합니다.
- 뷰 및 컨트롤러 이름: 기본값을 유지합니다.
- 추가 를 선택합니다.

Add MVC Controller with views, using Entity Framework

Model class Movie (MvcMovie.Models)

Data context class MvcMovie.Data.MvcMovieContext

Views

☒ Generate views

☒ Reference script libraries

☒ Use a layout page

(Leave empty if it is set in a Razor `_viewstart` file)

Controller name MoviesController

스캐폴딩은 다음과 같이 업데이트를 수행합니다.

- 필요한 패키지 참조를 *MvcMovie.csproj* 프로젝트 파일에 삽입합니다.
- 데이터베이스 컨텍스트를 *Startup.cs* 파일의 *Startup.ConfigureServices*에 등록합니다.

- `appsettings.json` 파일에 데이터베이스 연결 문자열을 추가합니다.

스캐폴딩은 다음을 만듭니다.

- 영화 컨트롤러: `Controllers/MoviesController.cs`
- **Create, Delete, Details, Edit** 및 **Index** 페이지에 대한 Razor 뷰
파일: `Views/Movies/*.cshtml`
- 데이터베이스 컨텍스트 클래스: `Data/MvcMovieContext.cs`

이러한 파일 및 파일 업데이트의 자동 생성을 스캐폴딩이라고 합니다.

데이터베이스가 없으므로 아직 스캐폴드된 페이지를 사용할 수 없습니다. 앱을 실행하고 동영상 **영화 앱** 링크를 선택하면 *데이터베이스를 열 수 없음* 또는 *해당 테이블이 없음: Movie* 오류 메시지가 표시됩니다.

초기 마이그레이션

EF Core [마이그레이션](#) 기능을 사용하여 데이터베이스를 만듭니다. 마이그레이션은 데이터 모델과 일치하도록 데이터베이스를 만들고 수정할 수 있는 도구 모음입니다.

- [Visual Studio](#)
- [Visual Studio Code / Mac용 Visual Studio](#)

도구 메뉴에서 **NuGet 패키지 관리자 > 패키지 관리자 콘솔** 을 선택합니다.

PMC(패키지 관리자 콘솔)에서 다음 명령을 입력합니다.

PowerShell 복사

`Add-Migration InitialCreate`

`Update-Database`

- **Add-Migration**
`InitialCreate: Migrations/{timestamp}_InitialCreate.cs` 마이그레이션 파일을 생성합니다. `InitialCreate` 인수는 마이그레이션 이름입니다. 모든 이름을 사용할 수 있지만 규칙에 따라 마이그레이션을 설명하는 이름을 선택합니다. 이는 첫 번째 마이그레이션이므로 생성된 클래스에는 데이터베이스 스키마를 만드는 코드가 포함되어 있습니다. 데이터베이스 스키마는 `MvcMovieContext` 클래스에 지정된 모델을 기반으로 합니다.
- **Update-Database:** 이전 명령이 만든 최신 마이그레이션으로 데이터베이스를 업데이트합니다. 이 명령은 데이터베이스를 만드는 `Up` method in the `Migrations/{time-stamp}_InitialCreate.cs` 파일을 실행합니다.

`Update-Database` 명령은 다음과 같은 경고를 생성합니다.

No type was specified for the decimal column 'Price' on entity type 'Movie'. This will cause values to be silently truncated if they do not fit in the default precision and scale. Explicitly specify the SQL server column type that can accommodate all the values using 'HasColumnType()'.

위의 경고를 무시하세요. 이 경고는 이후 자습서에서 수정되었습니다.

EF Core 의 PMC 도구에 대한 자세한 내용은 [EF Core 도구 참조 - Visual Studio 의 PMC](#) 를 참조하세요.

앱 테스트

앱을 실행하고 영화 앱 링크를 클릭합니다.

다음과 유사한 예외가 발생하면 [마이그레이션 단계](#)를 누락했을 수 있습니다.

- [Visual Studio](#)
- [Visual Studio Code / Mac용 Visual Studio](#)

콘솔복사

```
SqlException: Cannot open database "MvcMovieContext-1" requested by the login. The login failed.
```

참고

Price 필드에 소수점을 입력하지 못할 수도 있습니다. 소수점으로 쉼표(",")를 사용하는 비 영어 로캘 및 비 US-English 날짜 형식에 대해 [jQuery 유효성 검사](#)를 지원하려면 앱을 세계화해야 합니다. 세계화 지침은 [이 GitHub 문제](#)를 참조하세요.

생성된 데이터베이스 컨텍스트 클래스 및 등록 검사

EF Core에서는 데이터 액세스가 모델을 통해 수행됩니다. 모델은 엔터티 클래스 및 데이터베이스와의 세션을 나타내는 컨텍스트 개체로 구성됩니다. 컨텍스트 개체를 사용하여 데이터를 쿼리하고 저장할 수 있습니다. 데이터베이스 컨텍스트는 [Microsoft.EntityFrameworkCore.DbContext](#)에서 파생되며 데이터 모델에 포함할 엔터티를 지정합니다.

스캐폴딩은 *Data/MvcMovieContext.cs* 데이터베이스 컨텍스트 클래스를 만듭니다.

C#복사

```
using Microsoft.EntityFrameworkCore;
using MvcMovie.Models;

namespace MvcMovie.Data
{
    public class MvcMovieContext : DbContext
    {
```

```

    public MvcMovieContext (DbContextOptions<MvcMovieContext>
options)
        : base(options)
    {
    }

    public DbSet<Movie> Movie { get; set; }
}

```

위의 코드는 데이터베이스의 영화를 나타내는 [DbSet<Movie>](#) 속성을 만듭니다.

ASP.NET Core 는 [DI\(종속성 주입\)](#)를 사용하여 만들어집니다. 데이터베이스 컨텍스트와 같은 서비스는 `Startup` 에서 `DI` 에 등록해야 합니다. 이러한 서비스가 필요한 구성 요소는 생성자 매개 변수를 통해 해당 서비스를 제공합니다.

`Controllers/MoviesController.cs` 파일에서 생성자는 [종속성 주입](#)을 사용하여 컨트롤러에 `MvcMovieContext` 데이터베이스 컨텍스트를 삽입합니다. 컨트롤러의 각 [CRUD](#) 메서드에서 해당 데이터베이스 컨텍스트를 사용합니다.

스캐폴딩은 `Startup.ConfigureServices` 에서 다음과 같이 강조 표시된 코드를 생성했습니다.

- [Visual Studio](#)
- [Visual Studio Code](#)
- [Mac용 Visual Studio](#)

C#복사

```

public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddDbContext<MvcMovieContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("MvcMovieContext")));
}

```

[ASP.NET Core 구성 시스템](#)은 "MvcMovieContext" 데이터베이스 연결 문자열을 읽습니다.

생성된 데이터베이스 연결 문자열 검사

스캐폴딩은 `appsettings.json` 파일에 연결 문자열을 추가했습니다.

- [Visual Studio](#)
- [Visual Studio Code / Mac용 Visual Studio](#)

JSON복사

```

{
  "Logging": {

```



```

    "LogLevel": {
        "Default": "Information",
        "Microsoft": "Warning",
        "Microsoft.Hosting.Lifetime": "Information"
    },
    "AllowedHosts": "*",
    "ConnectionStrings": {
        "MvcMovieContext":
"Server=(localdb)\mssqllocaldb;Database=MvcMovieContext-1;Trusted_Connection=True;MultipleActiveResultSets=true"
    }
}

```

로컬 개발의 경우 [ASP.NET Core 구성 시스템](#)은 `appsettings.json` 파일에서 `ConnectionString` 키를 읽습니다.

InitialCreate 클래스

`Migrations/{timestamp}_InitialCreate.cs` 마이그레이션 파일을 확인합니다.

C#복사

```

public partial class InitialCreate : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Movie",
            columns: table => new
            {
                Id = table.Column<int>(type: "int", nullable: false)
                    .Annotation("SqlServer:Identity", "1, 1"),
                Title = table.Column<string>(type: "nvarchar(max)", nullable:
true),
                ReleaseDate = table.Column<DateTime>(type: "datetime2",
nullable: false),
                Genre = table.Column<string>(type: "nvarchar(max)", nullable:
true),
                Price = table.Column<decimal>(type: "decimal(18,2)", nullable:
false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Movie", x => x.Id);
            });
    }

    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "Movie");
    }
}

```

```
}  
}
```

위의 코드에서

- `InitialCreate.Up`은 Movie 테이블을 만들고 `Id`를 기본 키로 구성합니다.
- `InitialCreate.Down`은 Up 마이그레이션에 의해 변경된 스키마를 되돌립니다.

컨트롤러에서 종속성 주입

`Controllers/MoviesController.cs` 파일을 열고 생성자를 확인합니다.

C#복사

```
public class MoviesController : Controller  
{  
    private readonly MvcMovieContext _context;  
  
    public MoviesController(MvcMovieContext context)  
    {  
        _context = context;  
    }  
}
```

생성자는 [종속성 주입](#)을 사용하여 컨트롤러에 데이터베이스 컨텍스트(`MvcMovieContext`)를 주입합니다. 컨트롤러의 각 [CRUD](#) 메서드에서 해당 데이터베이스 컨텍스트를 사용합니다.

Create 페이지를 테스트합니다. 데이터를 입력하고 제출합니다.

Edit, **Details** 및 **Delete** 링크를 테스트합니다.

강력한 형식의 모델 및 `@model` 지시문

이 자습서의 앞부분에서는 컨트롤러가 `ViewData` 사전을 사용하여 데이터 또는 개체를 보기에 전달하는 방법을 살펴보았습니다. `ViewData` 사전은 정보를 보기에 전달하기 위한 편리한 런타임 바인딩 방법을 제공하는 동적 개체입니다.

MVC는 보기에 강력한 형식의 모델 개체를 전달하는 기능도 제공합니다. 강력한 형식의 방법을 사용하면 컴파일 시에 코드 검사를 수행할 수 있습니다. 스캐폴딩 메커니즘은 `MoviesController` 클래스 및 뷰에서 강력한 형식의 모델을 전달했습니다.

`Controllers/MoviesController.cs` 파일에서 생성된 `Details` 메서드를 확인합니다.

C#복사

```
// GET: Movies/Details/5  
public async Task<IActionResult> Details(int? id)  
{
```

```

    if (id == null)
    {
        return NotFound();
    }

    var movie = await _context.Movie
        .FirstOrDefaultAsync(m => m.Id == id);
    if (movie == null)
    {
        return NotFound();
    }

    return View(movie);
}

```

일반적으로 id 매개 변수는 경로 데이터 변수로 전달됩니다. 예를 들어 `https://localhost:5001/movies/details/1` 은 다음을 설정합니다.

- 컨트롤러를 `movies` 컨트롤러로 설정(첫 번째 URL 세그먼트)
- 작업을 `details`로 설정(두 번째 URL 세그먼트)
- id를 1로 설정(마지막 URL 세그먼트)

id 는 다음 예제와 같이 쿼리 문자열을 통해 전달할 수 있습니다.

`https://localhost:5001/movies/details?id=1`

id 값이 제공되지 않는 경우에 id 매개 변수가 [nullable 형식](#)(`int?`)으로 정의됩니다.

[람다 식](#)을 `FirstOrDefaultAsync` 메서드에 전달하여 경로 데이터 또는 쿼리 문자열 값과 일치하는 영화 엔터티를 선택합니다.

C#복사

```

var movie = await _context.Movie
    .FirstOrDefaultAsync(m => m.Id == id);

```

영화를 찾으면 Movie 모델의 인스턴스를 Details 보기에 전달합니다.

C#복사

```

return View(movie);

```

`Views/Movies/Details.cshtml` 파일의 내용을 확인합니다.

CSHTML복사

```

@model MvcMovie.Models.Movie

```

```

@{
    ViewData["Title"] = "Details";
}

```

```

<h1>Details</h1>

```

```

<div>
  <h4>Movie</h4>
  <hr />
  <dl class="row">
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Title)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Title)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.ReleaseDate)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.ReleaseDate)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Genre)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Genre)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Price)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.Price)
    </dd>
  </dl>
</div>
<div>
  <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
  <a asp-action="Index">Back to List</a>
</div>

```

보기 파일 맨 위에 있는 `@model` 문은 보기에 필요한 개체 형식을 지정합니다. 영화 컨트롤러가 만들어질 때 다음 `@model` 문이 포함됩니다.

CSHTML 복사

`@model MvcMovie.Models.Movie`

`@model` 지시문을 사용하면 컨트롤러가 보기에 전달한 영화에 액세스할 수 있습니다. `Model` 개체는 강력한 형식입니다. 예를 들어 *Details.cshtml* 보기의 코드는 각 영화 필드를 강력한 형식의 `Model` 개체를 사용하여 `DisplayNameFor` 및 `DisplayFor` HTML 도우미에 전달합니다. 또한 `Create` 및 `Edit` 메서드와 보기도 `Movie` 모델 개체를 전달합니다.

Index.cshtml 보기 및 영화 컨트롤러의 *Index* 메서드를 확인합니다.
코드가 *View* 메서드를 호출할 때 *List* 개체를 생성하는 방법에 유의하세요. 이 코드는 *Index* 작업 메서드에서 보기로 *Movies* 목록을 전달합니다.

C#복사

```
// GET: Movies
public async Task<IActionResult> Index()
{
    return View(await _context.Movie.ToListAsync());
}
```

영화 컨트롤러가 만들어질 때 스캐폴딩은 다음 *@model* 문을 *Index.cshtml* 파일의 맨 위에 포함시킵니다.

CSHTML복사

```
@model IEnumerable<MvcMovie.Models.Movie>
```

@model 지시문을 사용하면 강력한 형식인 *Model* 개체를 사용하여 컨트롤러가 보기에 전달한 영화 목록에 액세스할 수 있습니다. 예를 들어 *Index.cshtml* 보기에서 코드는 강력한 형식의 *Model* 개체에 대해 *foreach* 문을 사용하여 영화를 반복합니다.

CSHTML복사

```
@model IEnumerable<MvcMovie.Models.Movie>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ReleaseDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Genre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Price)
            </th>
        </tr>
    </thead>
</table>
```

```

        <th></th>
    </tr>
</thead>
<tbody>
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReleaseDate)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Genre)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Price)
        </td>
        <td>
            <a asp-action="Edit" asp-route-
id="@item.Id">Edit</a> |
            <a asp-action="Details" asp-route-
id="@item.Id">Details</a> |
            <a asp-action="Delete" asp-route-
id="@item.Id">Delete</a>
        </td>
    </tr>
}
</tbody>
</table>

```

Model 개체가 IEnumerable<Movie> 개체로서 강력한 형식이 지정되었으므로 루프의 각 항목은 Movie 형식입니다. 여러 가지 이점이 있지만 컴파일러는 코드에 사용된 형식이 유효한지도 검사합니다.

Entity Framework Core 의 SQL 로깅

로깅 구성은 일반적으로 *appsettings.Environment.json* 파일의 Logging 섹션에서 제공됩니다. SQL 문을 로깅하려면 "Microsoft.EntityFrameworkCore.Database.Command": "Information"을 *appsettings.Development.json* 파일에 추가합니다.

JSON복사

```

{
  "ConnectionStrings": {
    "DefaultConnection":
    "Server=(localdb)\mssqllocaldb;Database=MyDB-
2;Trusted_Connection=True;MultipleActiveResultSets=true"
  },

```

```
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft": "Warning",
    "Microsoft.Hosting.Lifetime": "Information"
  },
  "Microsoft.EntityFrameworkCore.Database.Command": "Information"
},
"AllowedHosts": "*"
}
```

위의 JSON 을 사용하면 SQL 문이 명령줄과 Visual Studio 출력 창에 표시됩니다.

자세한 내용은 [.NET Core 및 ASP.NET Core 의 로깅](#) 및 이 [GitHub 이슈](#)를 참조하세요.

추가 자료

- [태그 도우미](#)
- [세계화 및 지역화](#)

[이전: 뷰 추가](#) [다음: SQL 사용](#)

권장 콘텐츠

- **7 부. ASP.NET Core MVC 앱에 검색 추가**

ASP.NET Core MVC 에 대한 자습서 시리즈의 7 부입니다.
- **3 부. ASP.NET Core MVC 앱에 뷰 추가**

ASP.NET Core MVC 에 대한 자습서 시리즈의 3 부입니다.
- **6 부. ASP.NET Core 의 컨트롤러 메서드 및 보기**

6 부. ASP.NET Core MVC 앱에 모델 추가

-

(스캐폴딩 생성 과정에서 5.0, 6.0 버전에 관련된 충돌
발생 22/1/14일 맥용 비주얼 스튜디오에서 체크)