Add a new field to an ASP.NET Core MVC app

12/13/2018 • 4 minutes to read • +8

In this article

By Rick Anderson

In this section Entity Framework Code First Migrations is used to:

- Add a new field to the model.
- Migrate the new field to the database.

When EF Code First is used to automatically create a database, Code First:

- Adds a table to the database to track the schema of the database.
- Verifies the database is in sync with the model classes it was generated from. If they aren't in sync, EF throws an exception. This makes it easier to find inconsistent database/code issues.

Add a Rating Property to the Movie Model

Add a `Rating` property to *Models/Movie.cs*:

C# Copy

```
public class Movie
{
    public int Id { get; set; }
    public string Title { get; set; }
```

```
    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }
    public string Genre { get; set; }

    [Column(TypeName = "decimal(18, 2)")]
    public decimal Price { get; set; }
    public string Rating { get; set; }
}
```

Build the app (Ctrl+Shift+B).

Because you've added a new field to the `Movie` class, you need to update the binding white list so this new property will be included. In *MoviesController.cs*, update the `[Bind]` attribute for both the `Create` and `Edit` action methods to include the `Rating` property:

C# Copy

```
[Bind("Id,Title,ReleaseDate,Genre,Price,Rating")]
```

Update the view templates in order to display, create, and edit the new `Rating` property in the browser view.

Edit the */Views/Movies/Index.cshtml* file and add a `Rating` field:

HTML Copy

```
<thead>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Movies[0].Title)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Movies[0].ReleaseDate)
```

```
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Movies[0].Genre)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Movies[0].Price)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Movies[0].Rating)
        </th>
        <th></th>
    </tr>
</thead>
<tbody>
    @foreach (var item in Model.Movies)
    {
    <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>
            @Html.DisplayFor(modelItem => item.Rating)
        </td>
        <td>
            <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
```

Update the */Views/Movies/Create.cshtml* with a `Rating` field.

Visual Studio / Visual Studio for Mac | Visual Studio Code

You can copy/paste the previous "form group" and let intelliSense help you update the fields. IntelliSense works with Tag Helpers.

```
        </div>
        <div class="form-group">
            <label asp-for="Title" class="col-md-2 control-label"></label>
            <div class="col-md-10">
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger" />
            </div>
        </div>
        <div class="form-group">
            <label asp-for="R" class="col-md-2 control-label"></label>
            <div class="col-
                <input asp-f                    n-control" />
                <span asp-va                    class="text-danger" />
            </div>
        </div>
        <div class="form-gro
            <div class="col-                    ">
                <input type=                    e" class="btn btn-default" />
            </div>
        </div>
    </div>
</form>
```

GetHashCode
GetType
ID
Price
Rating
ReleaseDate
Title
ToString

Update the `SeedData` class so that it provides a value for the new column. A sample change is shown below, but you'll want to make this change for each `new Movie`.

C# Copy

```
new Movie
{
    Title = "When Harry Met Sally",
    ReleaseDate = DateTime.Parse("1989-1-11"),
    Genre = "Romantic Comedy",
```

```
    Rating = "R",
    Price = 7.99M
},
```

The app won't work until the DB is updated to include the new field. If it's run now, the following `SqlException` is thrown:

```
SqlException: Invalid column name 'Rating'.
```

This error occurs because the updated Movie model class is different than the schema of the Movie table of the existing database. (There's no `Rating` column in the database table.)

There are a few approaches to resolving the error:

1. Have the Entity Framework automatically drop and re-create the database based on the new model class schema. This approach is very convenient early in the development cycle when you're doing active development on a test database; it allows you to quickly evolve the model and database schema together. The downside, though, is that you lose existing data in the database — so you don't want to use this approach on a production database! Using an initializer to automatically seed a database with test data is often a productive way to develop an application. This is a good approach for early development and when using SQLite.

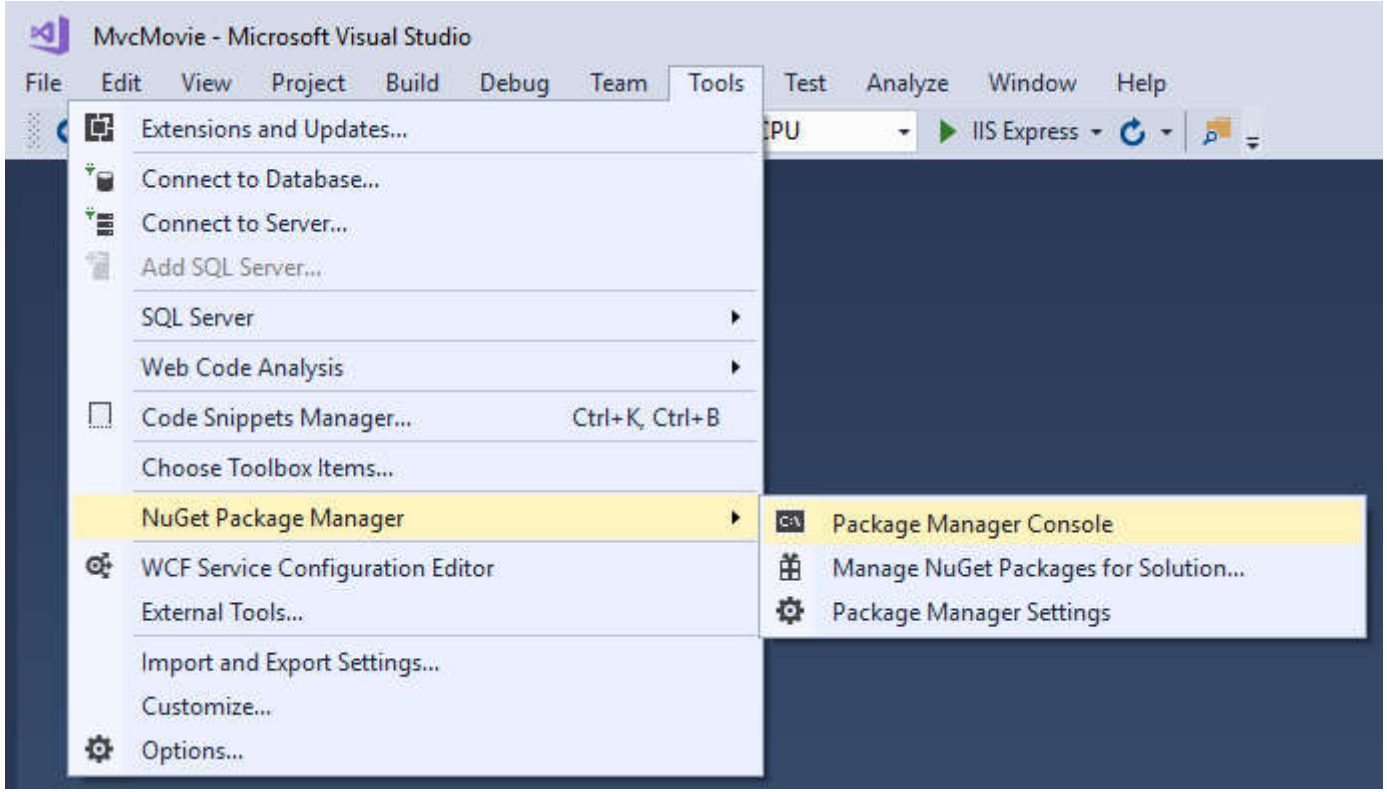
2. Explicitly modify the schema of the existing database so that it matches the model classes. The advantage of this approach is that you keep your data. You can make this change either manually or by creating a database change script.

3. Use Code First Migrations to update the database schema.

For this tutorial, Code First Migrations is used.

Visual Studio | Visual Studio Code / Visual Studio for Mac

From the **Tools** menu, select **NuGet Package Manager > Package Manager Console**.



In the PMC, enter the following commands:

PowerShell Copy

```
Add-Migration Rating
Update-Database
```

The `Add-Migration` command tells the migration framework to examine the current `Movie` model with the current `Movie` DB schema and create the necessary code to migrate the DB to the new model.

The name "Rating" is arbitrary and is used to name the migration file. It's helpful to use a meaningful name for the migration file.

If all the records in the DB are deleted, the initialize method will seed the DB and include the `Rating` field.

Run the app and verify you can create/edit/display movies with a `Rating` field. You should add the `Rating` field to the `Edit`, `Details`, and `Delete` view templates.

Previous Next