MVC Step 8 (https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/new-field[)](https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-model)

Adding a New Field

In this section you'll use [Entity Framework](http://docs.efproject.net/en/latest/platforms/aspnetcore/new-db.html) Code First Migrations to add a new field to the model and migrate that change to the database.+

When you use EF Code First to automatically create a database, Code First adds a table to the database to help track whether the schema of 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 track down issues at development time that you might otherwise only find (by obscure errors) at run time.+

## Adding a Rating Property to the Movie Model

Open the Models/Movie.cs file and add a Rating property:+

Copy

C#

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; }

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 also 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:3

Copy

C#

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

You also need to 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:+

Copy

HTML

<table class="table">

<tr>

<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].ReleaseDate)

</th>

<th>

@Html.DisplayNameFor(model => model.movies[0].Title)

</th>

<th>

@Html.DisplayNameFor(model => model.movies[0].Rating)

</th>

<th></th>

</tr>

<tbody>

@foreach (var item in Model.movies)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Genre)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

@Html.DisplayFor(modelItem => item.ReleaseDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.Title)

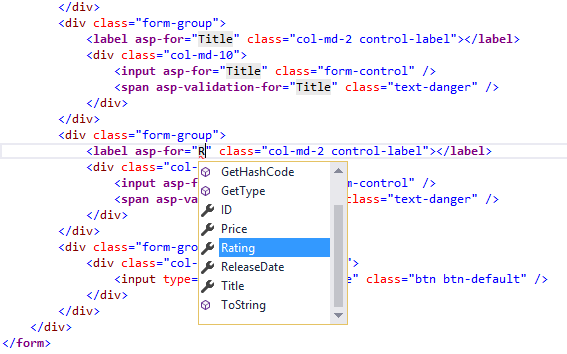
</td>

<td>

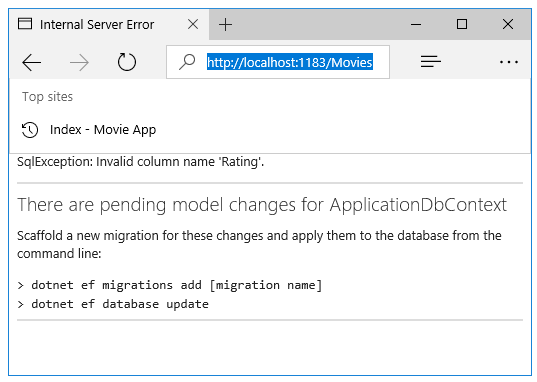
@Html.DisplayFor(modelItem => item.Rating)

</td>

Update the /Views/Movies/Create.cshtml with a Rating field. You can copy/paste the previous "form group" and let intelliSense help you update the fields. IntelliSense works with [Tag Helpers](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro).+

+

The app won't work until we update the DB to include the new field. If you run it now, you'll get the following SqlException:+

+

You're seeing this error 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 are 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.
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, we'll use Code First Migrations.+

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.+

Copy

C#

new Movie

{

Title = "When Harry Met Sally",

ReleaseDate = DateTime.Parse("1989-1-11"),

Genre = "Romantic Comedy",

Rating = "R",

Price = 7.99M

},

Build the solution then open a command prompt. Enter the following commands:4

Copy

console

dotnet ef migrations add Rating

dotnet ef database update

The migrations add 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 step.+

If you delete all the records in the DB, the initialize will seed the DB and include the Rating field. You can do this with the delete links in the browser or from SSOX.2

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