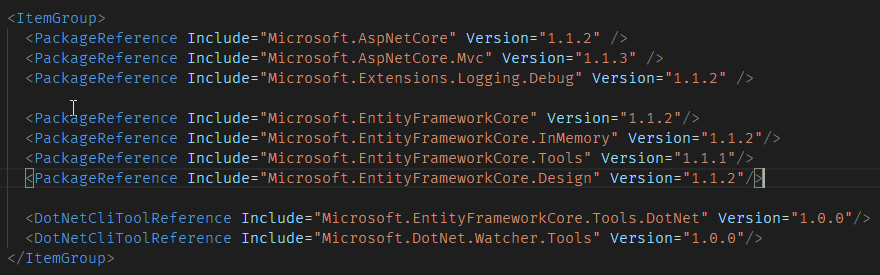
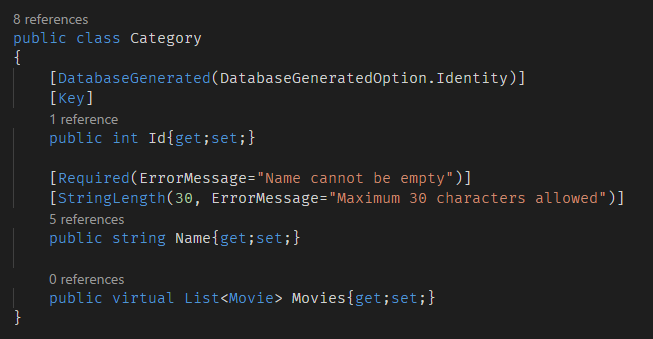
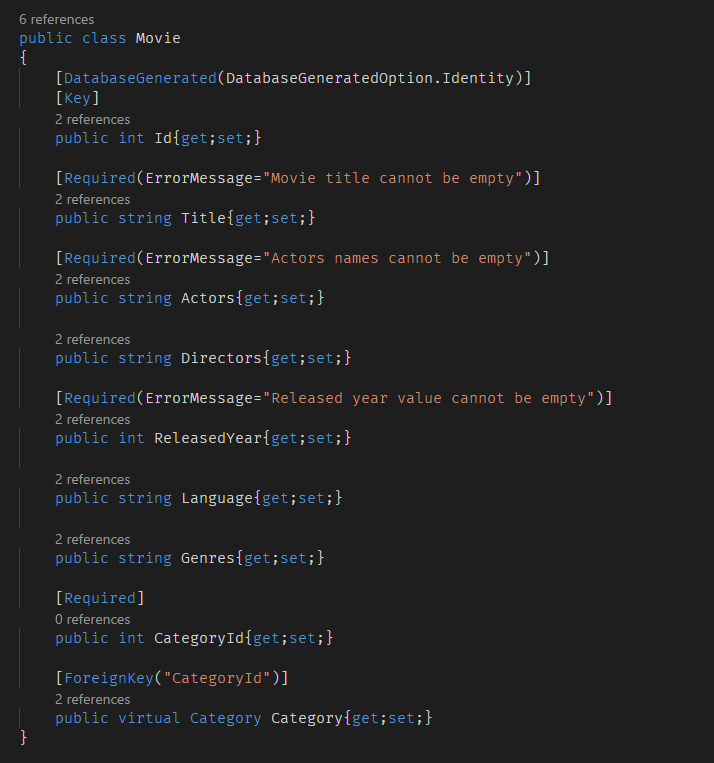
Creating ASP.NET Core WEB API Project

1. Open Visual studio Code. Set a folder as project folder and open the Integrated terminal by Views>Terminal.
2. Run the “**dotnet new webapi**” command to generate a web api project.
3. Add the following packages to the .csproj file and restore the packages by running “**dotnet restore**” command.

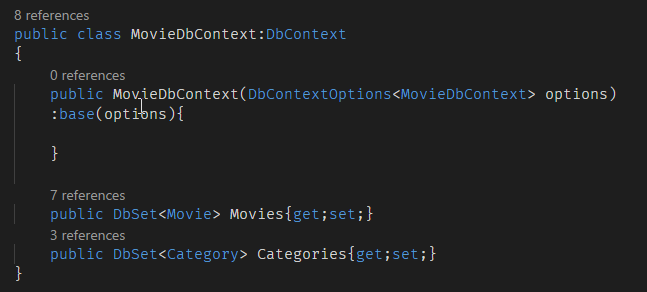


1. Add two models to the “Models” folder. Category class and Movie class .

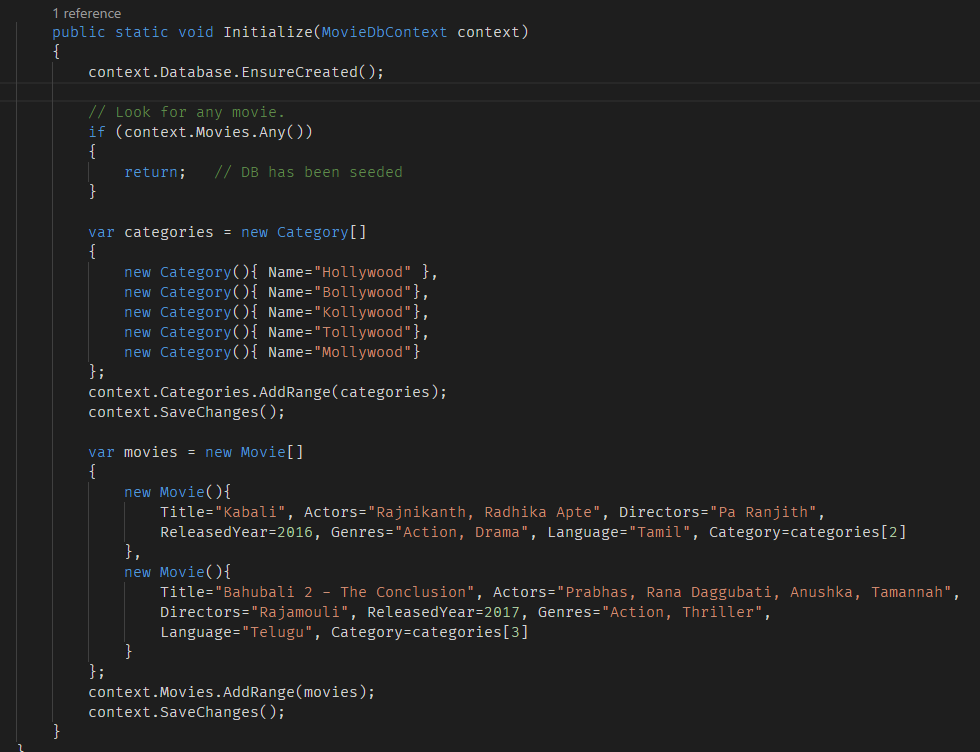




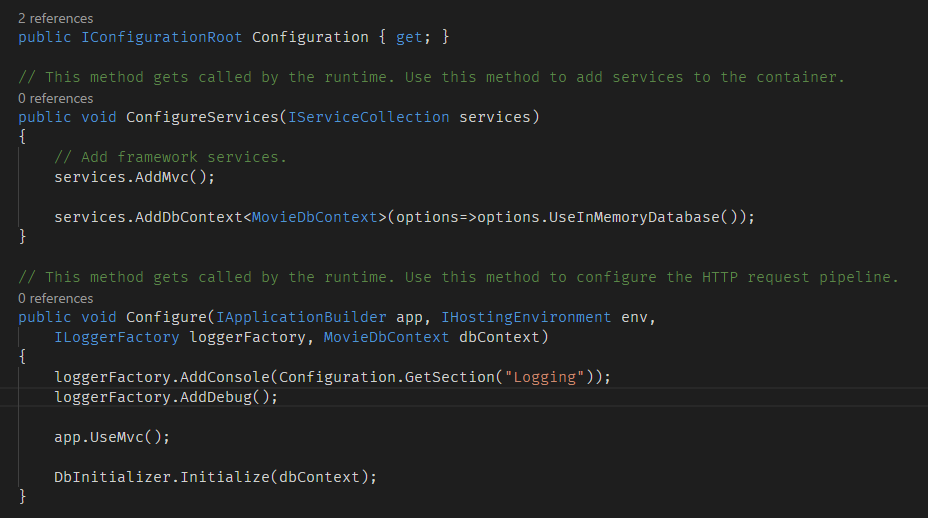
1. Add a **DbContext** class in the **Data** folder.



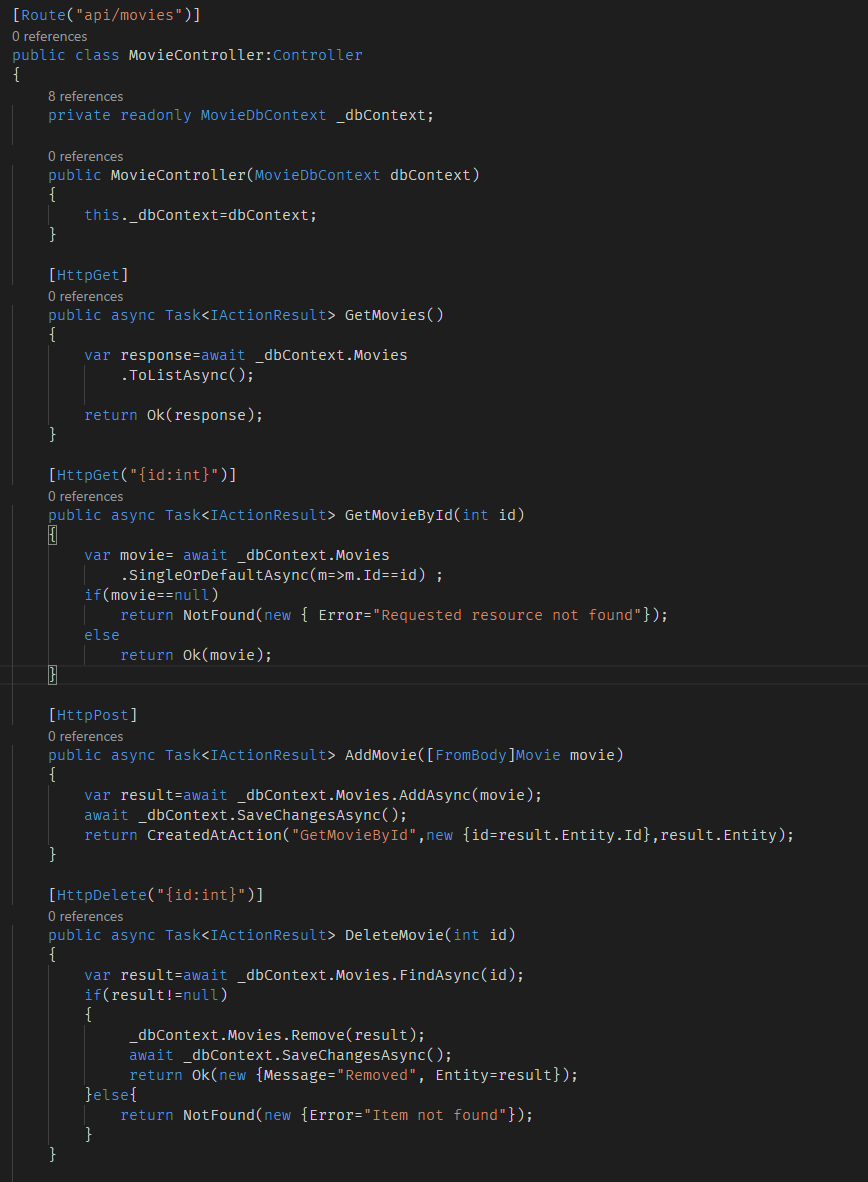
1. Add a **DbInitializer** to set default values to the database. Create the **DbInitializer** in the **Data** folder.

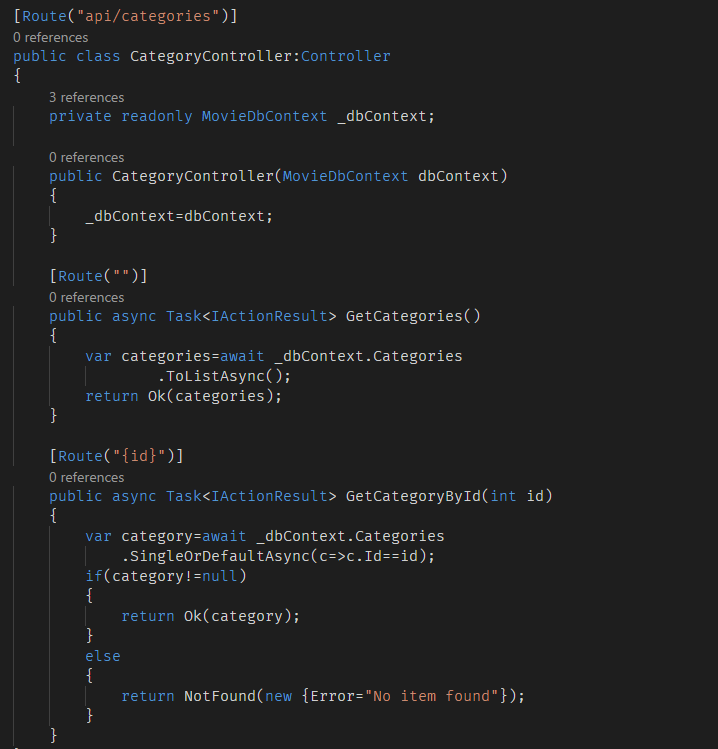


1. Update the **startup** class to add the services and call the **DbInitializer** to seed the database.



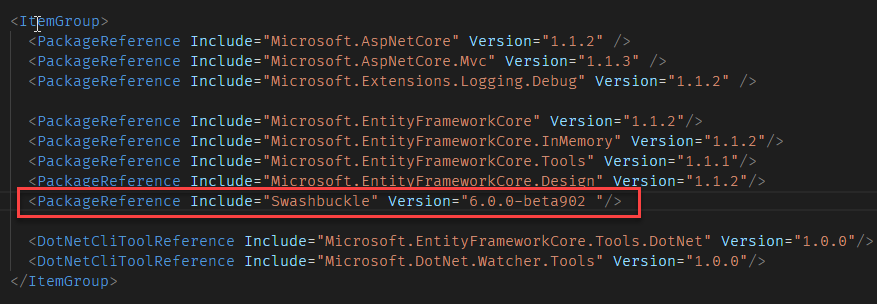
1. Create a “MovieController” and “CategoryController” in the Controllers folder.





**Adding Swagger metadata to Web API**

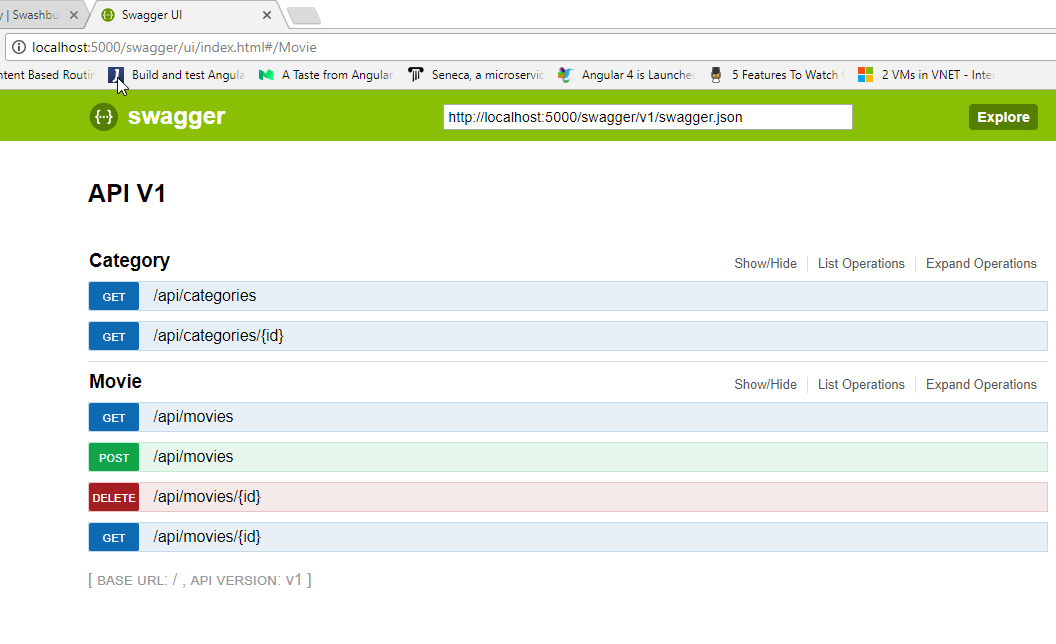
1. Open the .**csproj** file and add the **Swashbuckle** package reference.



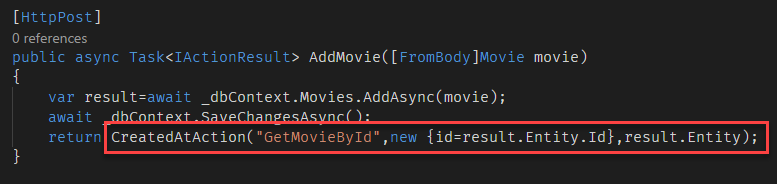
1. Add the swagger service to the **ConfigureServices**() method of the **Startup** Class.



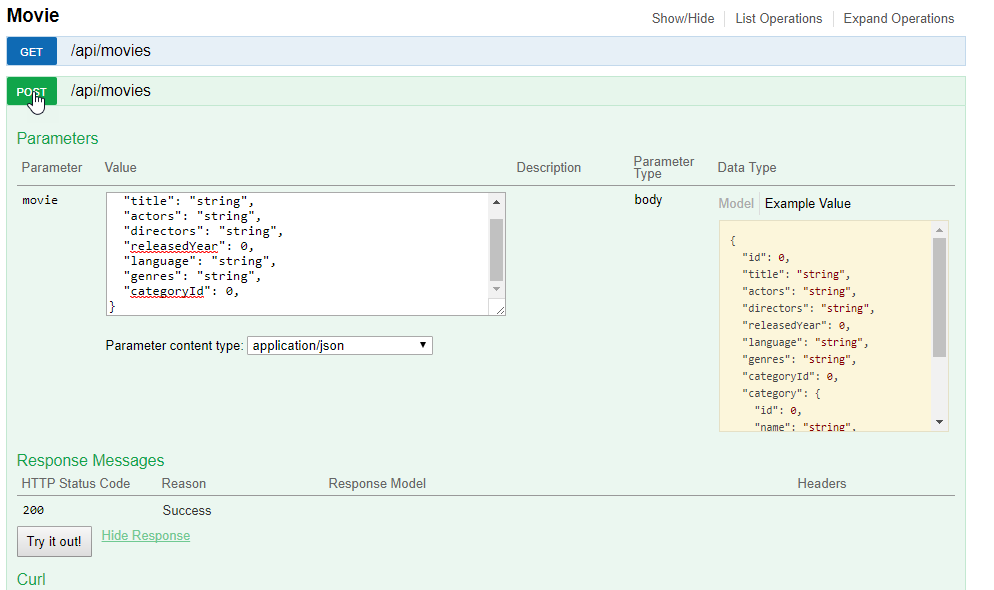
1. Start the project and navigate to the address <http://localhost:5000/swagger/ui>. It opens the index.html with list of methods.



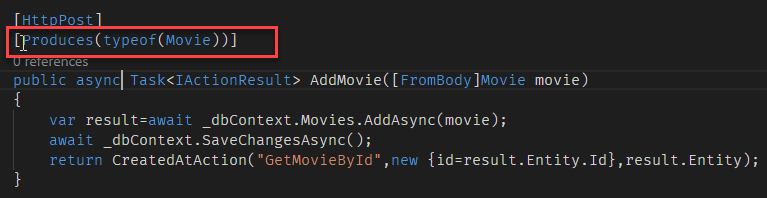
1. It also gives the swagger documents in the URL <http://localhost:5000/swagger/v1/swagger.json>.
2. In the swagger UI you can test all GET, POST, PUT and DELETE methods. In our **MovieControllers** POST action we returns an **IActionResult** type. The methods return a **CreatedAtAction** type which is a subclass of **IActionResult** interface.



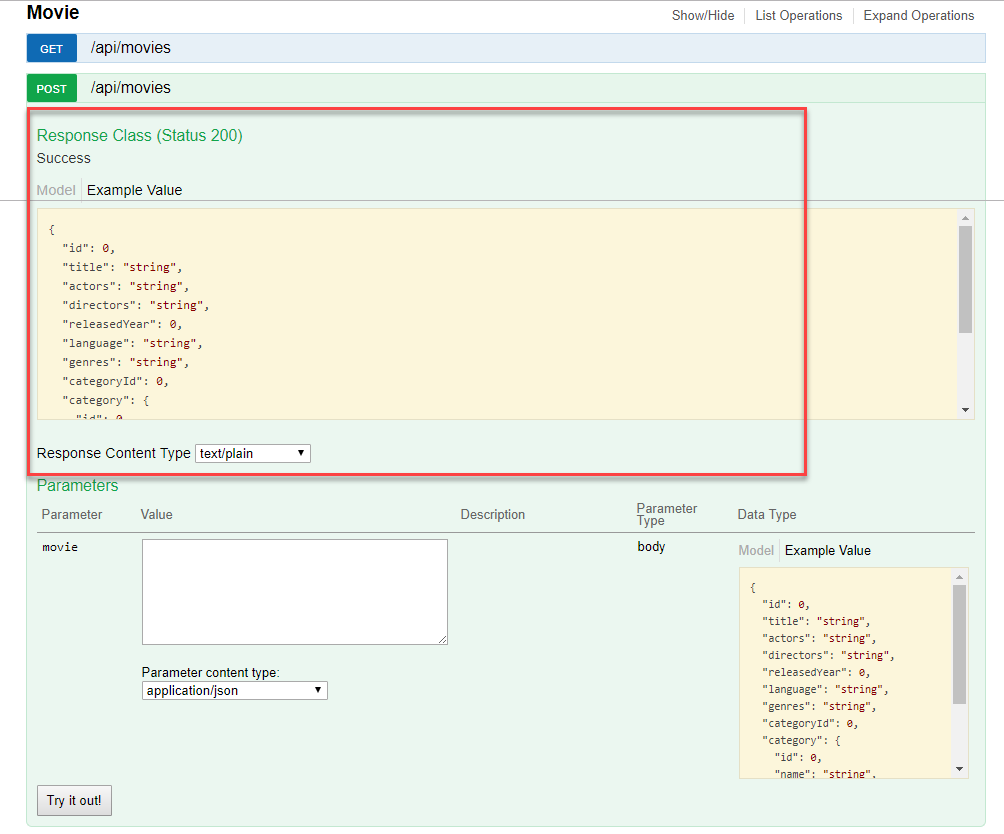
1. Because the return type is **IActionResult** we will not get the complete metadata about the result type of the action.



1. So, we can add “**Produces**” attribute to specify the output generated by the POST action.

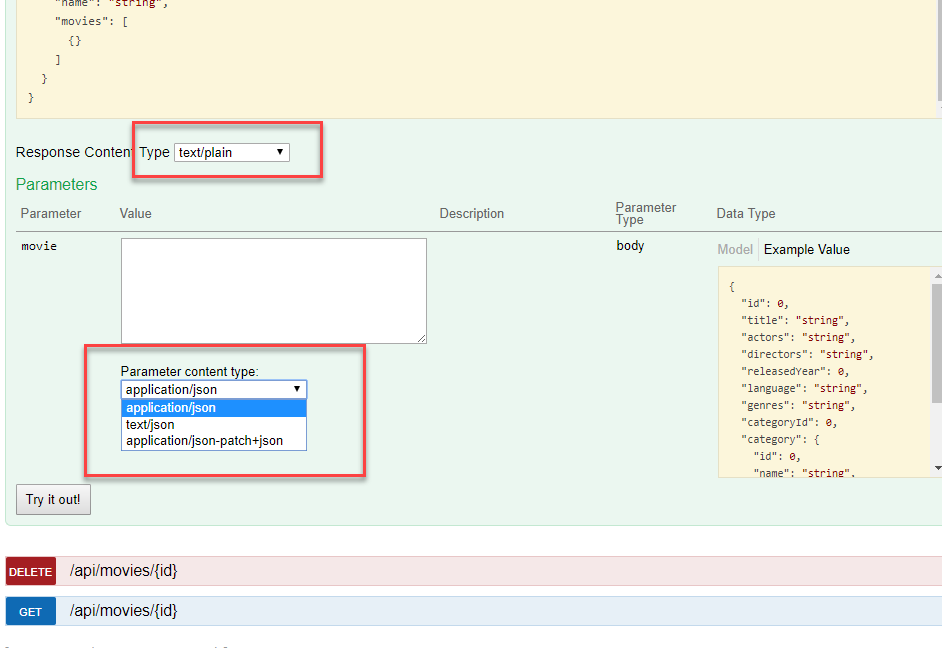


1. It creates response metadata in the swagger documentation

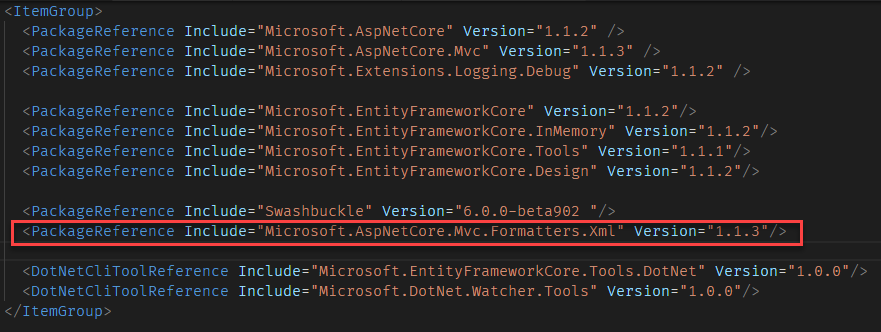


**Support for XML formatters**

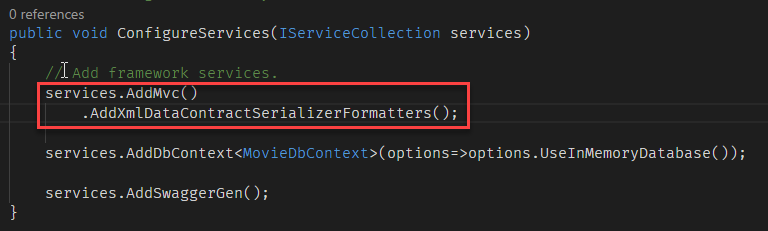
1. By Default ASPNET Core Web API gives support for only for JSON.



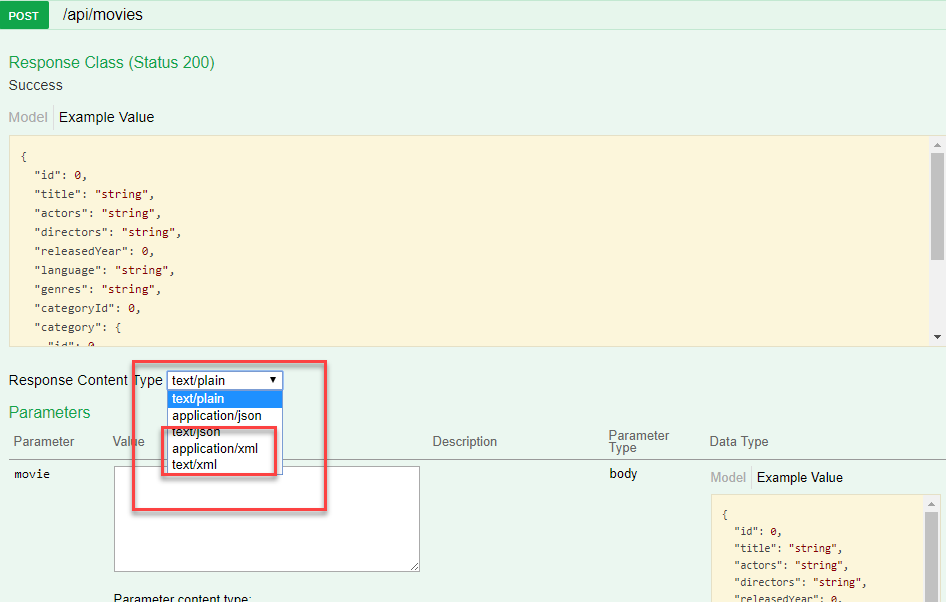
1. To add support for XML, You need to add the formatters packages for ASP.NET Core. Add the “**Microsoft.AspNetCore.Mvc.Formatters.Xml**” package to the .csproj file.



1. Configure the **XmlDataContractSerializerFormatters**() in **ConfigureServices**() method of Startup class. You can call **AddXmlDataContractSerializerFormatters()** method after the **AddMvc()** method.

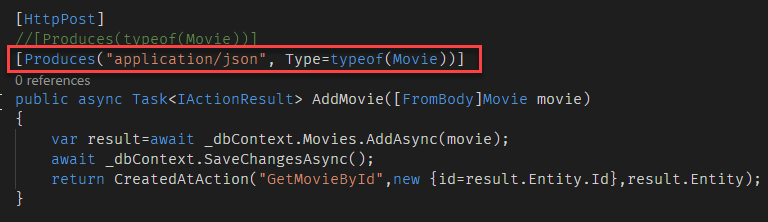


1. Run the project and open the Swagger UI, Now, you can see the XML formatters in the response type list.

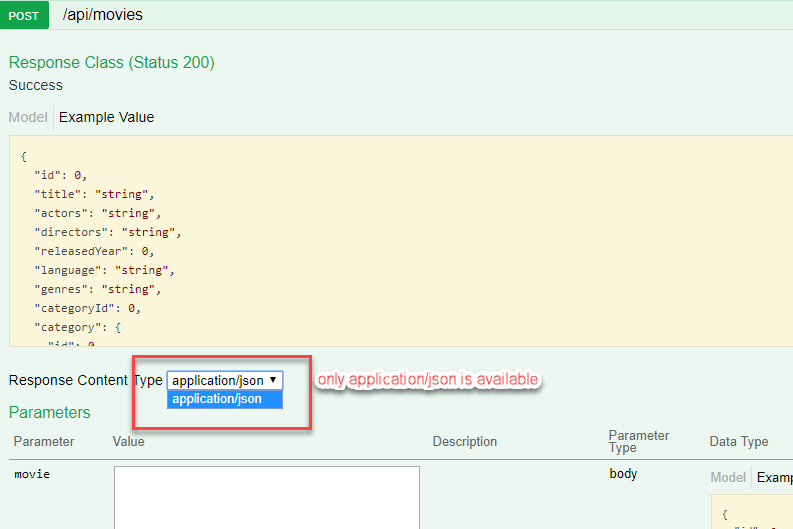


1. You can configure your action to produce only a specific type of format, eg: Movie POST action always produce only **application/json**.

To set that inside the produces attribute we can set the format type and response type.

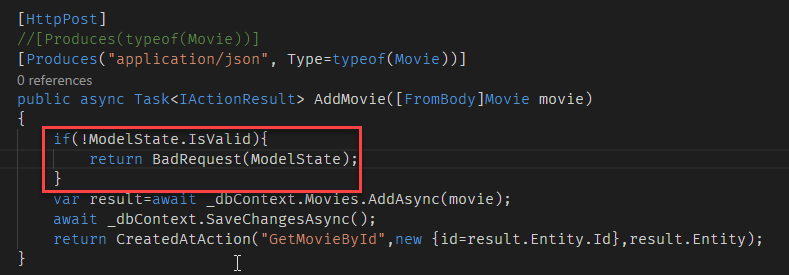


1. Run the application and test the post method using swagger UI. You can see only **application/json** in the response types list.



**Model validation and errors**

1. To enable model validations, you can add data annotations to the model class.
2. After adding data annotations, you can check whether the model is valid or not while performing POST or PUT actions.



1. Test the validation configuration, to do so, open the swagger UI and test the POST action with some invalid model values.

