**Using ASP.NET Core to create a Web API and integrating with Swagger**

1.Add a model -- data mapping

2.Add a database context -- entity framework

3.Register the database context -- using dependency injection (DI)

4.Scaffold a controller -- create a cotroller by using code generation

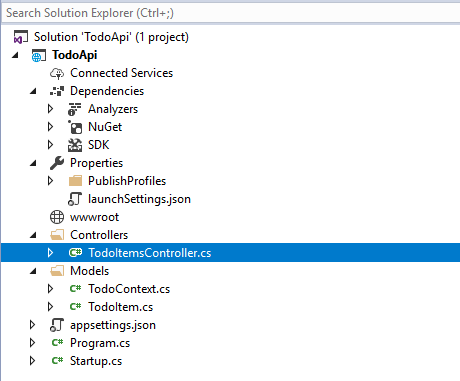
5.Update controller -- make it clean

6.Working with Swagger

Prerequisite

* Dot net core SDK 2.1
* VS studio 2017 Community

Code structure explanation



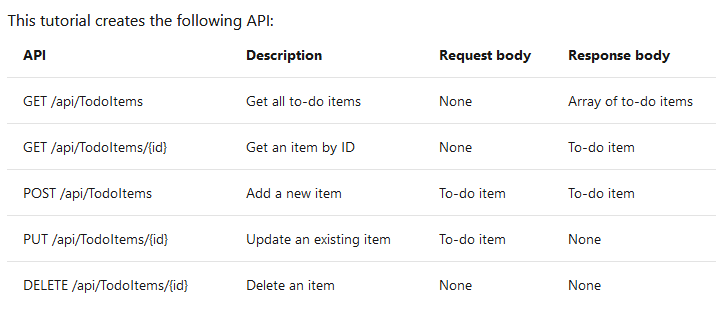
Dependency injection registration;

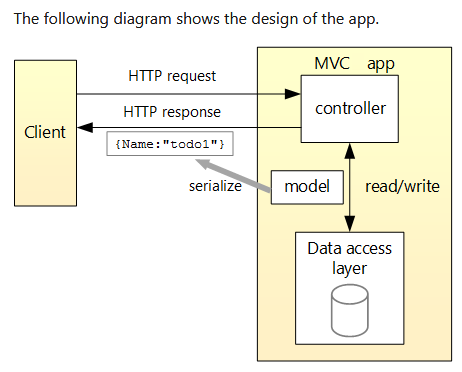
Swagger UI registration;

Entry point

Database connection string

Indicate iis express and port number





**In-memory DB**

Visual studio -> New Project -> ASP.NET Core web application -> API Template.

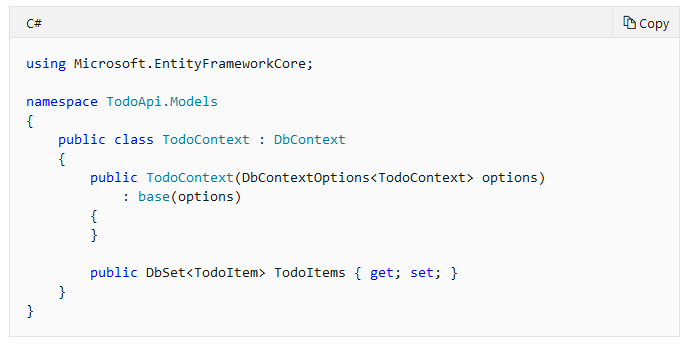
1. **Add a model**

A model is a set of classes that represent the data that the app manages. The model for this app is a single TodoItem class.

1. **Add a database context**

Nuget Manager -> **Microsoft.EntityFrameworkCore.SqlServer v2.1.1**

The database context is the main class that coordinates Entity Framework functionality for a data model. It includes the **DbSet** properties for the Model class.

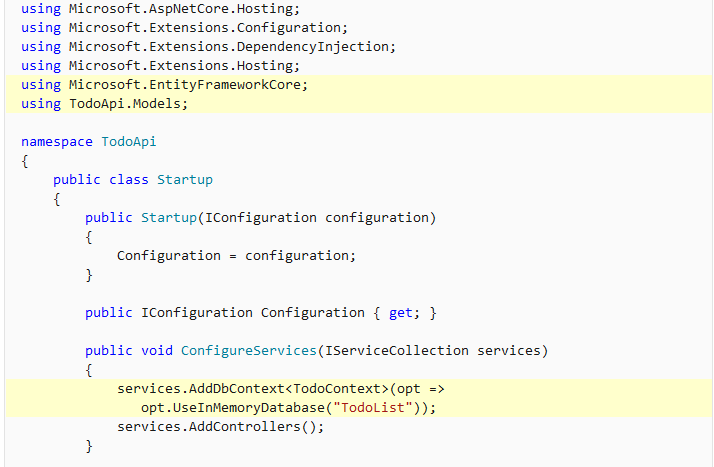


1. **Register the database context**

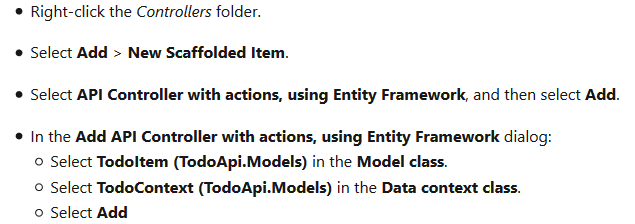
In ASP.NET Core, services such as the DB context must be registered with the [dependency injection (DI)](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.0) container. The container provides the service to controllers.

Update *Startup.cs* with the following highlighted code:

Here we Specify that the database context will use an in-memory database, and the DB name is “TodoList”



1. **Scaffold a controller**

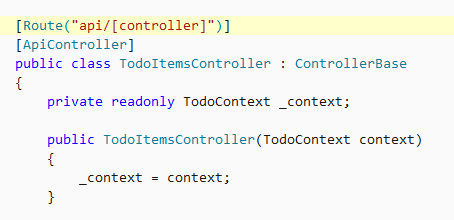


* Defines an API controller class without methods.
* Decorates the class with the **[ApiController]** attribute. This attribute indicates that the controller responds to web API requests.
* Uses Dependency Injection to inject the database context (TodoContext) into the controller. The database context is used in each of the CRUD methods in the controller.

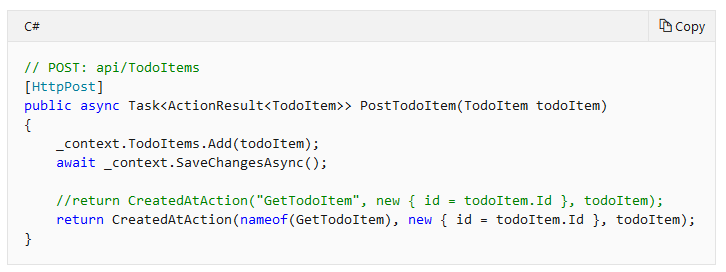
1. **Update controller**

Standard API controller should have

* Route attribute
* API Controoler



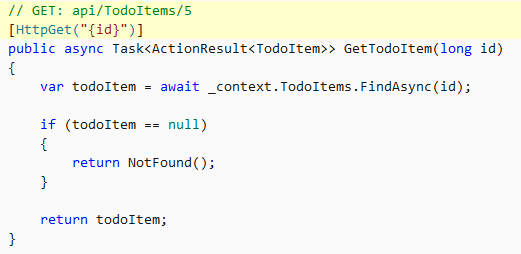
Create item in the database method – HTTP POST

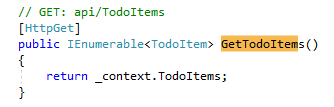


The method gets the value of the to-do item from the body of the HTTP request.

The [CreatedAtAction](https://docs.microsoft.com/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.createdataction) method:

* Returns an HTTP 201 status code if successful. HTTP 201 is the standard response for an HTTP POST method that creates a new resource on the server.
* Adds a [Location](https://developer.mozilla.org/docs/Web/HTTP/Headers/Location) header to the response. The Location header specifies the [URI](https://developer.mozilla.org/docs/Glossary/URI) of the newly created to-do item.
* References the GetTodoItem action (a method) to create the Location header's URI. The C# nameof keyword is used to avoid hard-coding the action name in the CreatedAtAction call.



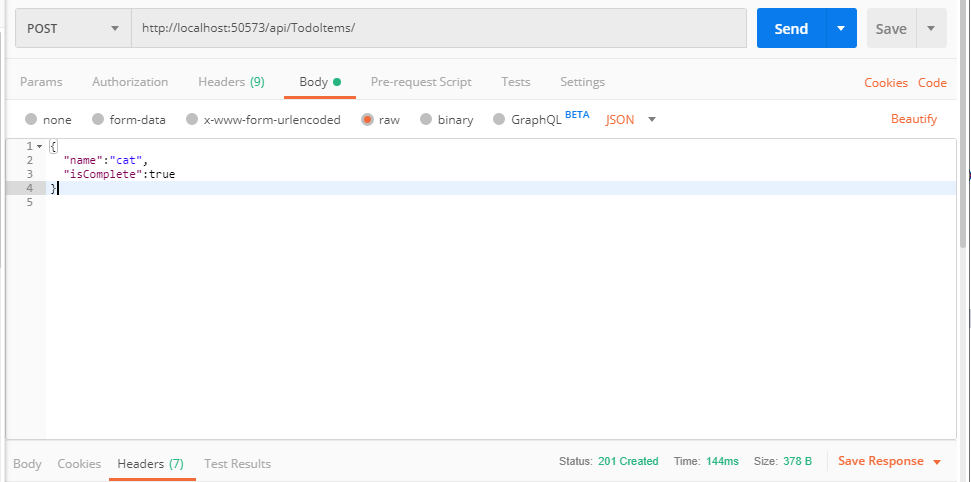


The return type of the GetTodoItems and GetTodoItem methods is **ActionResult<T> type**. ASP.NET Core automatically serializes the object to JSON and writes the JSON into the body of the response message. The response code for this return type is 200, assuming there are no unhandled exceptions. Unhandled exceptions are translated into 5xx errors.

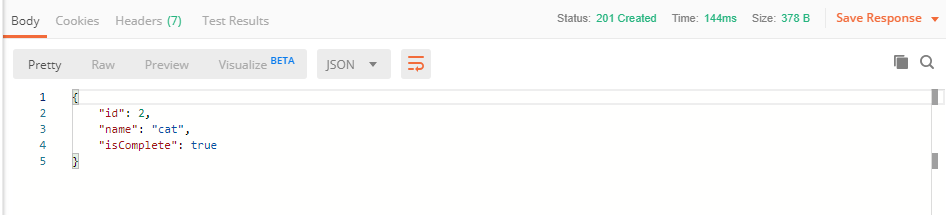
ActionResult return types can represent a wide range of HTTP status codes. For example, GetTodoItem can return two different status values:

* If no item matches the requested ID, the method returns a 404 Not Found error code.
* Otherwise, the method returns 200 with a JSON response body. Returning item results in an HTTP 200 response.

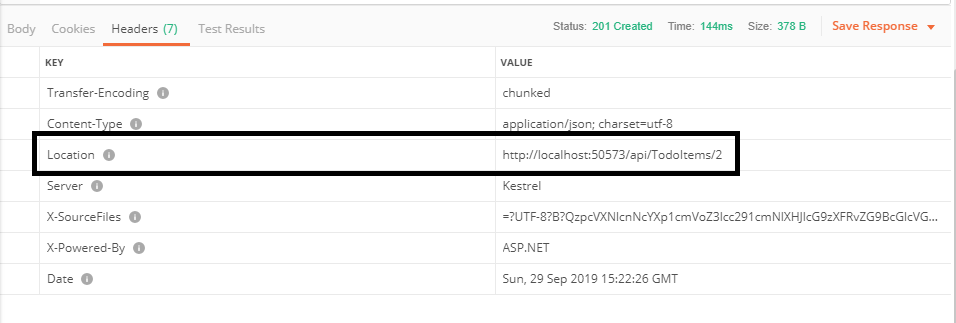
Request in postman



Response in postman, check body

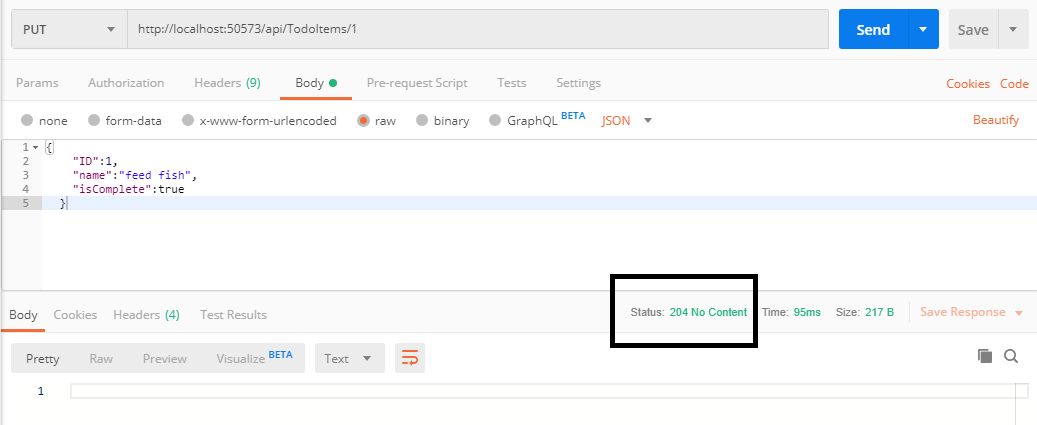


Response in postman, check header



Update a particular record in the database – HTTP PUT







1. **Working with Swagger**

Install Nuget Package

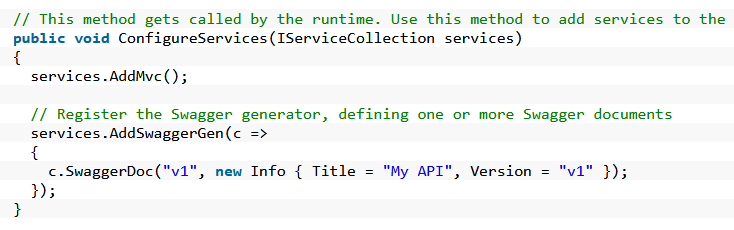
Swashbuckle.AspNetCore v2.1.1

Microsoft.AspNetCore.Hosting.Abstractions v2.1.1

Microsoft.VisualStudio.Web.BrowserLink v2.1.1

Add the Swagger generator to the service collection after services.AddMvc();

Enable the middleware for serving the generated JSON document after app.UseStaticFiles();  
  
Add the below background yellow lines in your  **Startup file**.





Build and open browser, Go to : <http://localhost:50573/swagger>

