Simple minimal API with .Net 9

Minimal APIs are architected to create HTTP APIs with minimal dependencies. They're ideal for microservices and apps that want to include only the minimum files, features, and dependencies in ASP.NET Core.

|  |  |  |  |
| --- | --- | --- | --- |
| **API** | **Description** | **Request body** | **Response body** |
| GET /todoitems | Get all to-do items | None | Array of to-do items |
| GET /todoitems/complete | Get completed to-do items | None | Array of to-do items |
| GET /todoitems/{id} | Get an item by ID | None | To-do item |
| POST /todoitems | Add a new item | To-do item | To-do item |
| PUT /todoitems/{id} | Update an existing item | To-do item | None |
| DELETE /todoitems/{id} | Delete an item | None | None |

# Create an API project

* Start Visual Studio 2022 and select **Create a new project**.
* In the **Create a new project** dialog:
  + Enter Empty in the **Search for templates** search box.
  + Select the **ASP.NET Core Empty** template and select **Next**.

A screenshot of a project

Description automatically generated

* Name the project *TodoApi* and select **Next**.
* In the **Additional information** dialog:
  + Select **.NET 9.0**
  + Uncheck **Do not use top-level statements**
  + Select **Create**

# Examine the code

The Program.cs file contains the following code:

A screenshot of a computer program

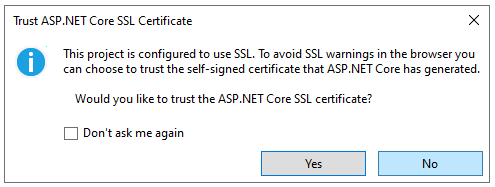
Description automatically generated

The preceding code:

* Creates a WebApplicationBuilder and a WebApplication with preconfigured defaults.
* Creates an HTTP GET endpoint / that returns Hello World!

# Run the app

In the first-time of running this type of app, VS displays these two messages. Select YES.





Visual Studio launches the Kestrel web server and opens a browser window.

Hello World! is displayed in the browser. The Program.cs file contains a minimal but complete app.

Close the browser window.

A screen shot of a computer

Description automatically generated

# Add NuGet packages

* From the **Tools** menu, select **NuGet Package Manager > Manage NuGet Packages for Solution**.
* Select the **Browse** tab.
* Select **Include Prelease**.
* Enter **Microsoft.EntityFrameworkCore.InMemory** in the search box, and then select Microsoft.EntityFrameworkCore.InMemory.
* Select the **Project** checkbox in the right pane and then select **Install**.
* Follow the preceding instructions to add the Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore package.

A screenshot of a computer

Description automatically generated

# The model and database context classes

* In the project folder, create a file named Todo.cs with the following code:

A screen shot of a computer

Description automatically generated

The preceding code creates the model for this app. A *model* is a class that represents data that the app manages.

* Create a file named TodoDb.cs with the following code:

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  class TodoDb : DbContext  {  public TodoDb(DbContextOptions<TodoDb> options)  : base(options) { }  public DbSet<Todo> Todos => Set<Todo>();  } |

The preceding code defines the *database context*, which is the main class that coordinates Entity Framework functionality for a data model. This class derives from the Microsoft.EntityFrameworkCore.DbContext class.

# Add the API code

* Replace the contents of the Program.cs file with the following code:

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  var builder = WebApplication.CreateBuilder(args);  builder.Services.AddDbContext<TodoDb>(opt => opt.UseInMemoryDatabase("TodoList"));  builder.Services.AddDatabaseDeveloperPageExceptionFilter();  var app = builder.Build();  app.MapGet("/todoitems", async (TodoDb db) =>  await db.Todos.ToListAsync());  app.MapGet("/todoitems/complete", async (TodoDb db) =>  await db.Todos.Where(t => t.IsComplete).ToListAsync());  app.MapGet("/todoitems/{id}", async (int id, TodoDb db) =>  await db.Todos.FindAsync(id)  is Todo todo  ? Results.Ok(todo)  : Results.NotFound());  app.MapPost("/todoitems", async (Todo todo, TodoDb db) =>  {  db.Todos.Add(todo);  await db.SaveChangesAsync();  return Results.Created($"/todoitems/{todo.Id}", todo);  });  app.MapPut("/todoitems/{id}", async (int id, Todo inputTodo, TodoDb db) =>  {  var todo = await db.Todos.FindAsync(id);  if (todo is null) return Results.NotFound();  todo.Name = inputTodo.Name;  todo.IsComplete = inputTodo.IsComplete;  await db.SaveChangesAsync();  return Results.NoContent();  });  app.MapDelete("/todoitems/{id}", async (int id, TodoDb db) =>  {  if (await db.Todos.FindAsync(id) is Todo todo)  {  db.Todos.Remove(todo);  await db.SaveChangesAsync();  return Results.NoContent();  }  return Results.NotFound();  });  app.Run(); |

The following highlighted code adds the database context to the dependency injection (DI) container and enables displaying database-related exceptions:

The DI container provides access to the database context and other services.

# Test posting data

Select **View** > **Other Windows** > **Endpoints Explorer**.

A screenshot of a computer program

Description automatically generated

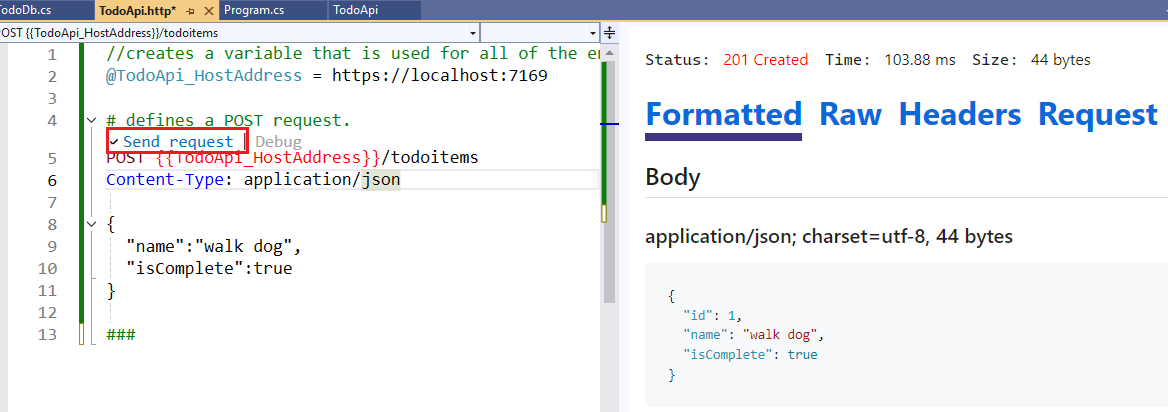
Right click on POST, select Generate Request

A screenshot of a computer

Description automatically generated

|  |
| --- |
| //creates a variable that is used for all of the endpoints.  @TodoApi\_HostAddress = https://localhost:7169  # defines a POST request.  POST {{TodoApi\_HostAddress}}/todoitems  Content-Type: application/json  {  "name":"walk dog",  "isComplete":true  }  ### |

Click “Send Request”



# Examine the GET endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| **API** | **Description** | **Request body** | **Response body** |
| GET /todoitems | Get all to-do items | None | Array of to-do items |
| GET /todoitems/complete | Get all completed to-do items | None | Array of to-do items |
| GET /todoitems/{id} | Get an item by ID | None | To-do item |

A screenshot of a computer screen

Description automatically generated

Complete all other tests by using the guide from the references.

# Reference

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-9.0&tabs=visual-studio>