



CREATE A WEB API WITH ASP.NET CORE AND MONGODB

DISCLAIMER

The material presented here is not original. It might point to several websites. This paper clarifies how MongoDB and the ASP.NET Core Web API interact. Original images extracted from the author's source code while running the project. All software's are community edition.

Vadivel Rangasamy
+91 9787778365

Table of Contents

About this Document	2
Introduction.....	2
MongoDB.....	2
MongoDB download link	2
Install MongoDB Community Edition on Windows.....	2
Visual Studio 2022.....	2
Visual Studio download link.....	2
Create a database to MongoDB	3
Create the ASP.NET Core web API project	5
Add an entity model.....	6
Add a configuration model	7
Add a CRUD operations service	8
Add a controller	9
Test the web API	10
Create / Insert.....	11
Read.....	12
Update	12
Delete	13
Not yet to resolve	14
Source code	14

About this Document

In this document help to Creates a web API that runs Create, Read, Update, and Delete (CRUD) operations on a MongoDB NoSQL database.

Introduction

MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License which is deemed non-free by several distributions.¹

MongoDB download link

https://www.mongodb.com/try/download/community?tck=docs_server

Install MongoDB Community Edition on Windows

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/>

Visual Studio 2022

Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.²

Visual Studio download link

<https://visualstudio.microsoft.com/vs/>

You can follow the above URLs instruction to install the software to your PC. The current experimental system has installed the MongoDB and VS2022 software's to Windows 10 operating system.

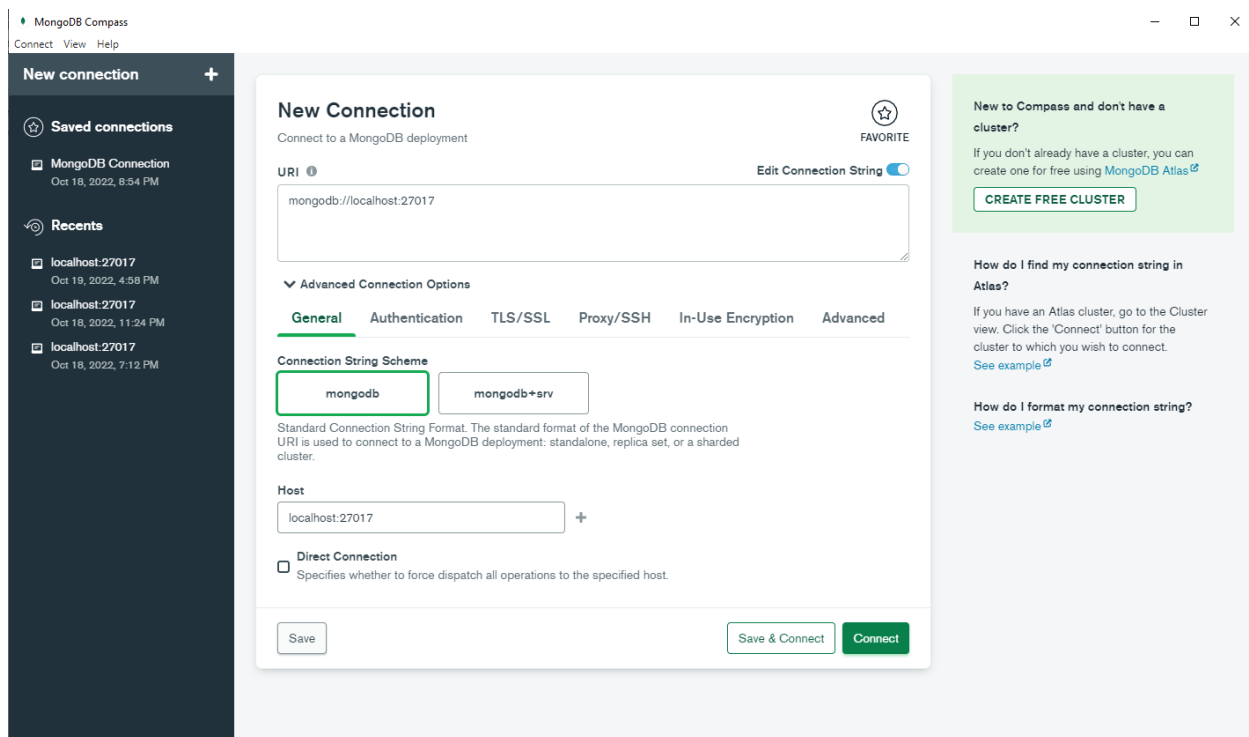
¹ <https://www.mongodb.com/what-is-mongodb>

² <https://visualstudio.microsoft.com/vs/>

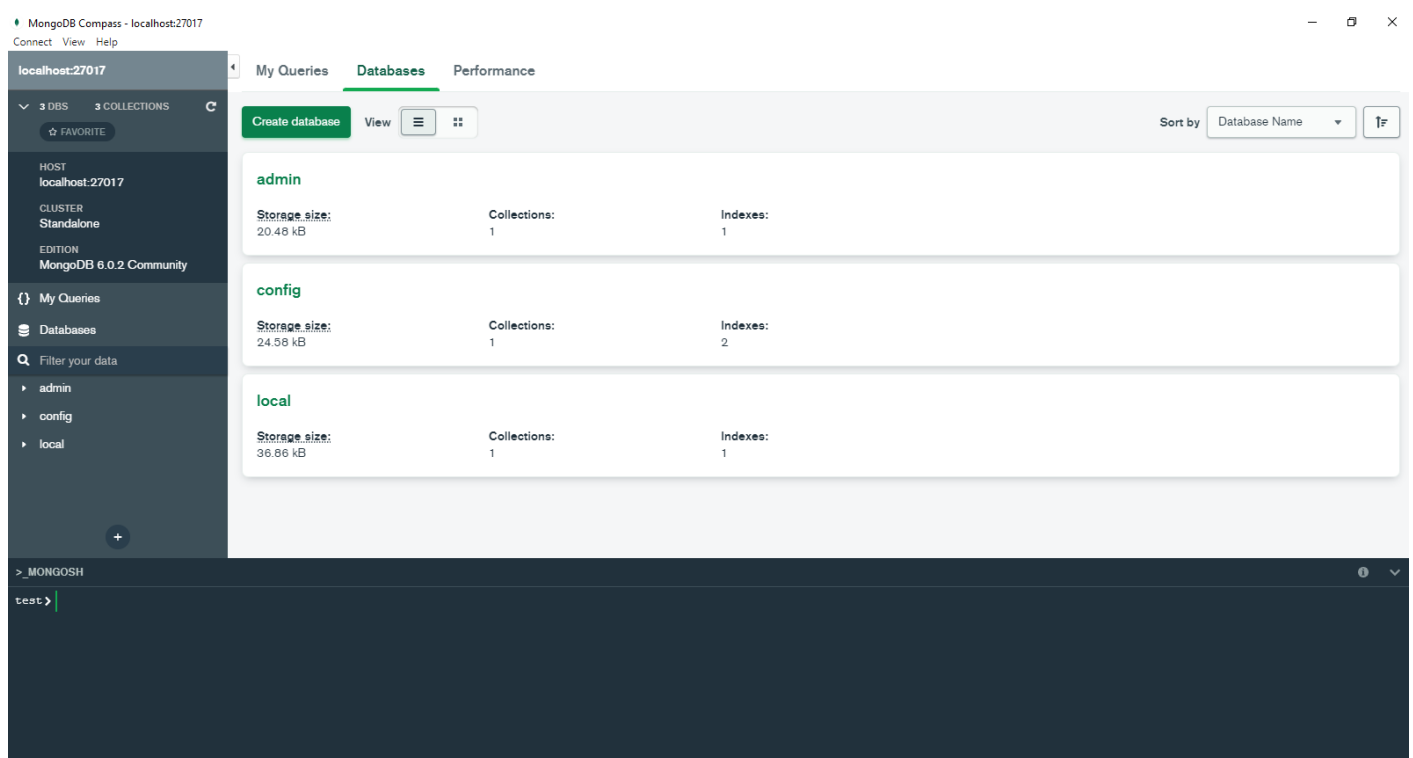
Create a database to MongoDB

Most of the MongoDB commands are recommended to use Ms-DOS prompt. You can install additionally MongoDB Compass IDE to visually execute the MongoDB commands.

The MongoDB default logical port is 27017 (This logical port may be differed system to system). Use “Connect” button to connect the MongoDB.



Once you successfully connected then you landed on the “Database” tab. Use “Create database” button or Shell prompt to create MongoDB database and collection. In this example “ArticleDB” is database and “Articles” is collection.



Create Database

Database Name

ArticleDB

Collection Name

Articles

Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)

Capped Collection

☐ Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. [Learn More](#)

Use Custom Collation

☐ Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks. [Learn More](#)

Time-Series

☐ Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

Clustered Collection

☐ Clustered collections store documents ordered by a user-defined cluster key. [Learn More](#)

Cancel

Create Database

Database and collection have been successfully created

MongoDB Compass - localhost:27017

Connect View Help

localhost:27017

4 DBS 4 COLLECTIONS

FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 6.0.2 Community

{ } My Queries

Databases

Filter your data

ArticleDB

Articles

admin

config

local

+

> _MONGOSH

test >

My Queries Databases Performance

Create database View

admin

Storage size: 20.48 kB

Collections: 1

Indexes: 1

ArticleDB

Storage size: 4.10 kB

Collections: 1

Indexes: 1

config

Storage size: 24.58 kB

Collections: 1

Indexes: 2

local

Storage size:

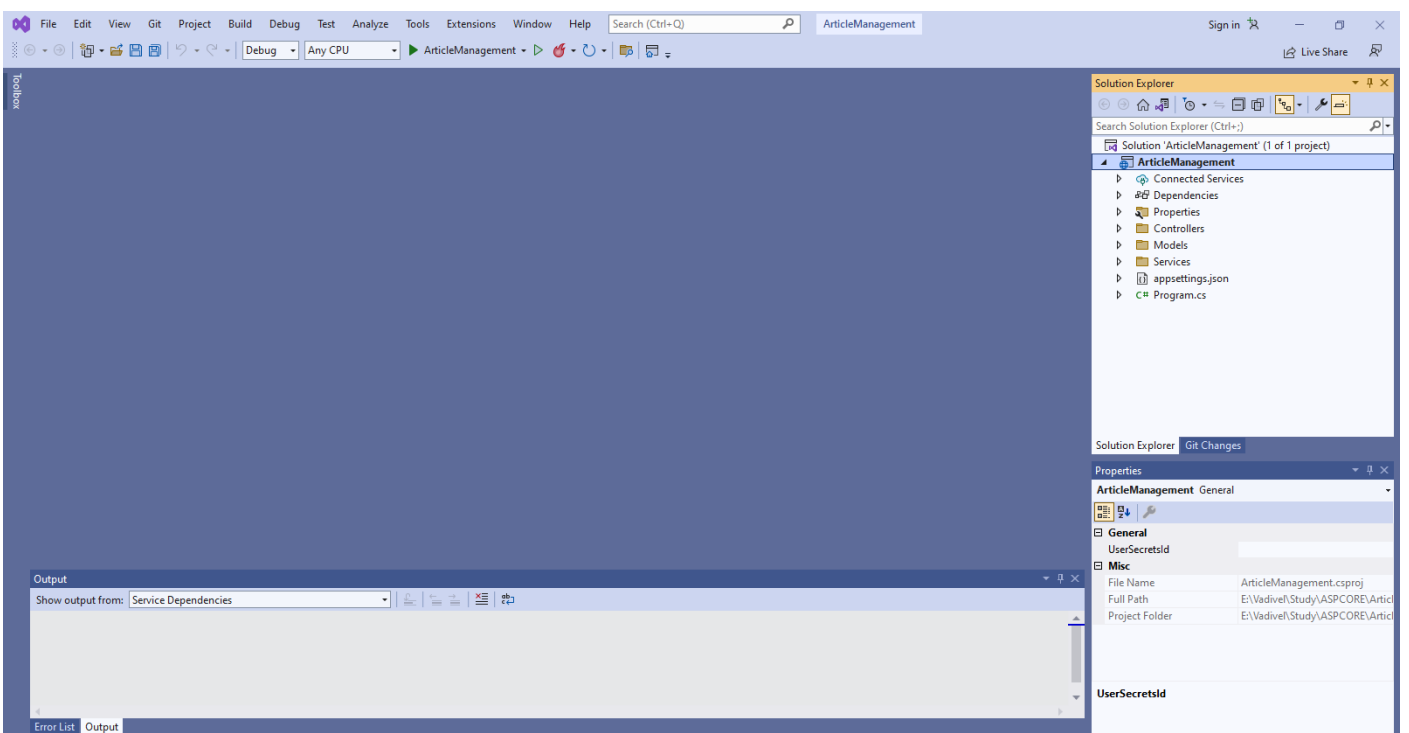
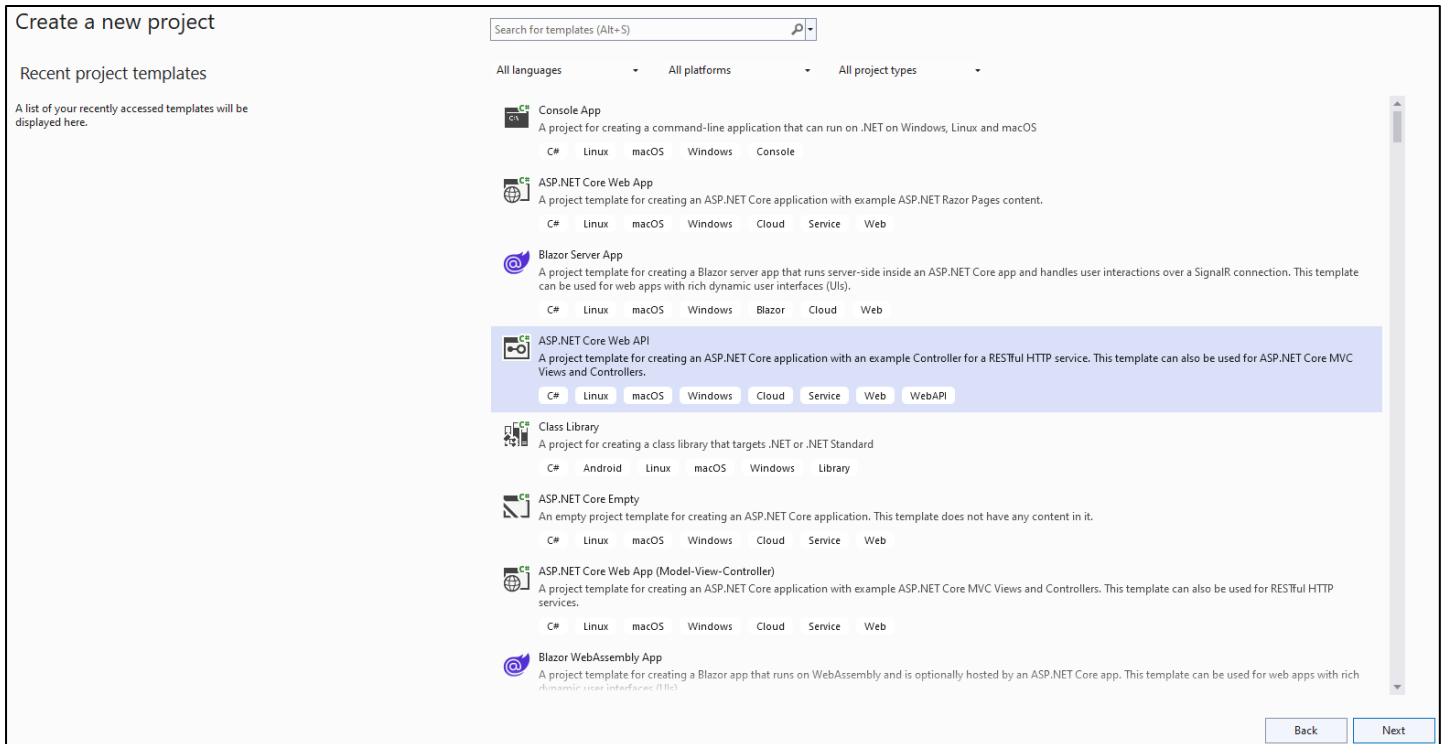
Collections:

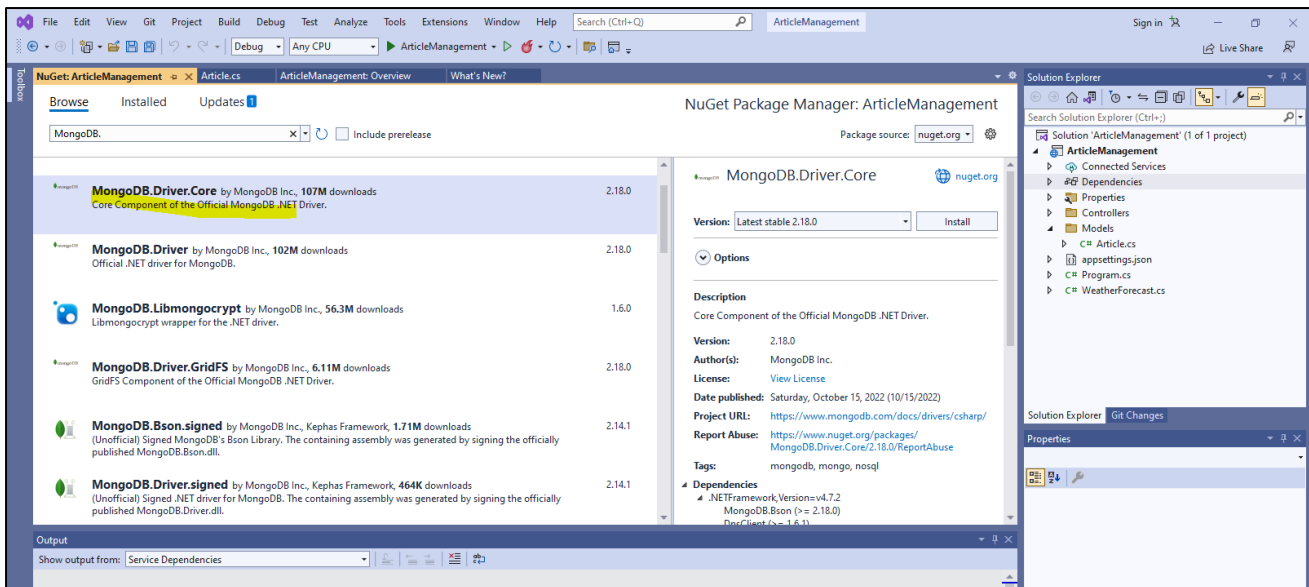
Indexes:

Create the ASP.NET Core web API project

Once you successfully installed the Visual Studio 2022 then you can see the below screenshot you select the **“ASP.NET Core Web API”**

1. Go to File > New > Project.
2. Select the ASP.NET Core Web API project type and select Next.
3. Name the project “ArticleManagement” and select Next.
4. Select the .NET 6.0 (Long-term support) framework and select Create.
5. Install the .NET driver for MongoDB to Visual studio 2022 project with help of “NuGet”





Add an entity model

1. Add a Models directory to the project root
2. Add a "Article" class to the Models directory with the following code

```
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
namespace ArticleManagement.Models
{
    10 references
    public class Article
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        6 references
        public string? ArticleId { get; set; }

        [BsonElement("ArticleName")]
        0 references
        public string ArticleName { get; set; } = null!;
        0 references
        public string Description { get; set; } = null!;
        0 references
        public string Author { get; set; } = null!;
    }
}
```

In the preceding class, the ArticleId property is:

- Required for mapping the Common Language Runtime (CLR) object to the MongoDB collection.
- Annotated with [BsonId] to make this property the document's primary key.
- Annotated with [BsonRepresentation(BsonType.ObjectId)] to allow passing the parameter as type string instead of an ObjectId structure. Mongo handles the conversion from string to ObjectId.

The ArticleName property is annotated with the [BsonElement] attribute. The attribute's value of Name represents the property name in the MongoDB collection.

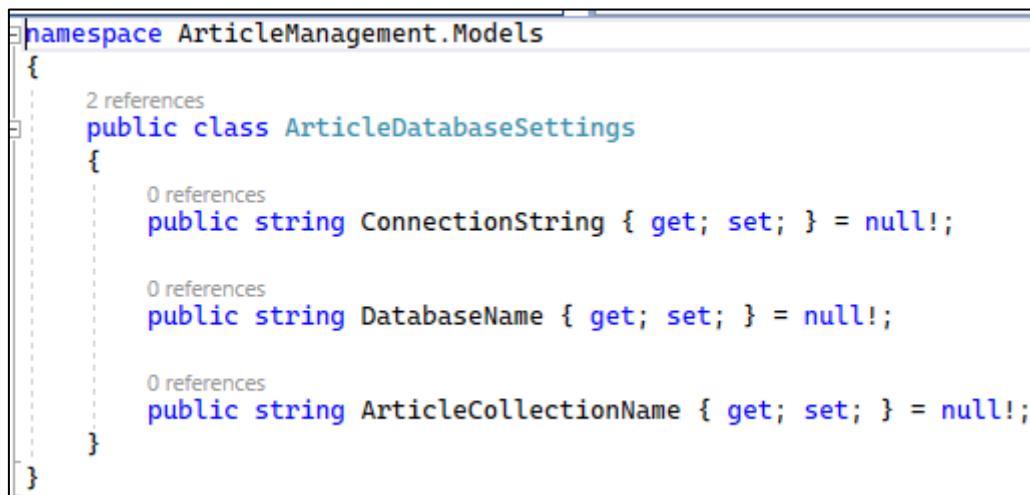
Add a configuration model

1. Add the following database configuration values to appsettings.json file instead of "web.config" in traditional ASP.NET.

A screenshot of a Visual Studio editor window showing the 'appsettings.json' file. The file is open in a tab titled 'appsettings.json'. The schema is listed as 'https://json.schemastore.org/appsettings.json'. The JSON content is as follows:

```
1 {  
2   "ArticleDatabase": {  
3     "ConnectionString": "mongodb://localhost:27017",  
4     "DatabaseName": "ArticleDB",  
5     "ArticleCollectionName": "Articles"  
6   },  
7   "Logging": {  
8     "LogLevel": {  
9       "Default": "Information",  
10      "Microsoft.AspNetCore": "Warning"  
11    }  
12  },  
13  "AllowedHosts": "*"   
14 }
```

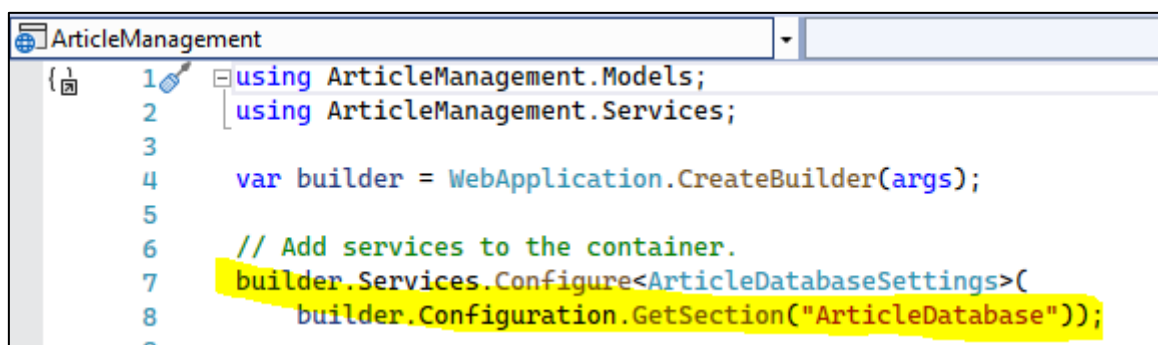
2. Add a "ArticleDatabaseSettings" class to the Models directory with the following code

A screenshot of a Visual Studio editor window showing the 'ArticleDatabaseSettings' class. The file is open in a tab titled 'ArticleDatabaseSettings.cs'. The namespace is 'ArticleManagement.Models'. The class is 'ArticleDatabaseSettings' and it has three public string properties: 'ConnectionString', 'DatabaseName', and 'ArticleCollectionName'. Each property has a getter and setter, and is initialized to 'null!'.

```
namespace ArticleManagement.Models  
{  
    2 references  
    public class ArticleDatabaseSettings  
    {  
        0 references  
        public string ConnectionString { get; set; } = null!;  
  
        0 references  
        public string DatabaseName { get; set; } = null!;  
  
        0 references  
        public string ArticleCollectionName { get; set; } = null!;  
    }  
}
```

The preceding "ArticleDatabaseSettings" class is used to store the appsettings.json file's "ArticleDatabase" property values. The JSON and C# property names are named identically to ease the mapping process.

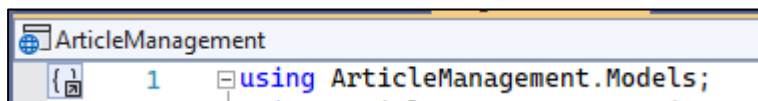
3. Add the following highlighted code to Program.cs

A screenshot of a Visual Studio editor window showing the 'Program.cs' file. The file is open in a tab titled 'Program.cs'. The code is as follows:

```
1 using ArticleManagement.Models;  
2 using ArticleManagement.Services;  
3  
4 var builder = WebApplication.CreateBuilder(args);  
5  
6 // Add services to the container.  
7 builder.Services.Configure<ArticleDatabaseSettings>(  
8     builder.Configuration.GetSection("ArticleDatabase"));
```

In the preceding code, the configuration instance to which the appsettings.json file's "ArticleDatabase" section binds is registered in the Dependency Injection (DI) container. For example, the "ArticleDatabaseSettings" object's ConnectionString property is populated with the ArticleDatabase:ConnectionString property in appsettings.json.

4. Add the following code to the top of Program.cs to resolve the ArticleDatabaseSettings. In Visual studio 2022 automatically add the namespace.

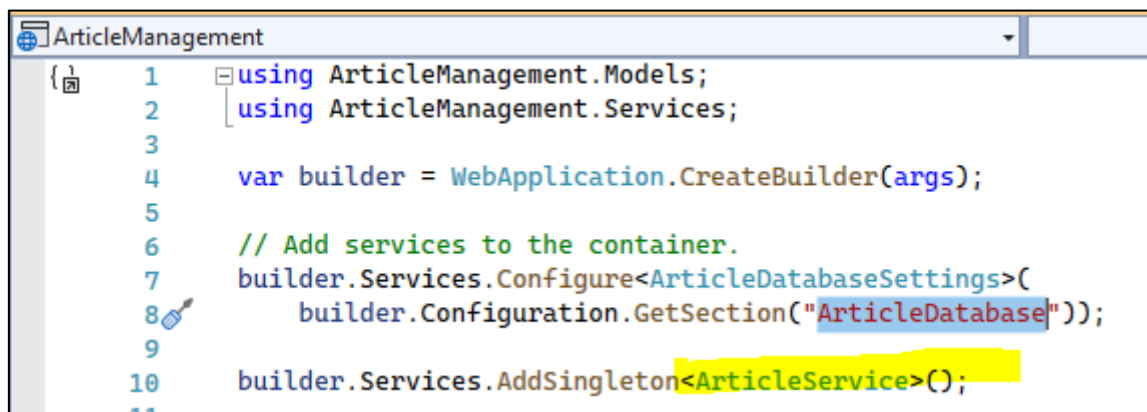


Add a CRUD operations service

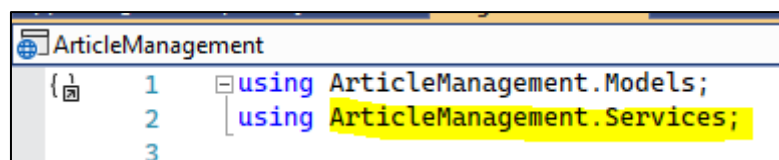
1. Add a Services directory to the project root.
2. Add a "ArticleService" class to the Services directory with the following code



3. Add the following highlighted code to Program.cs



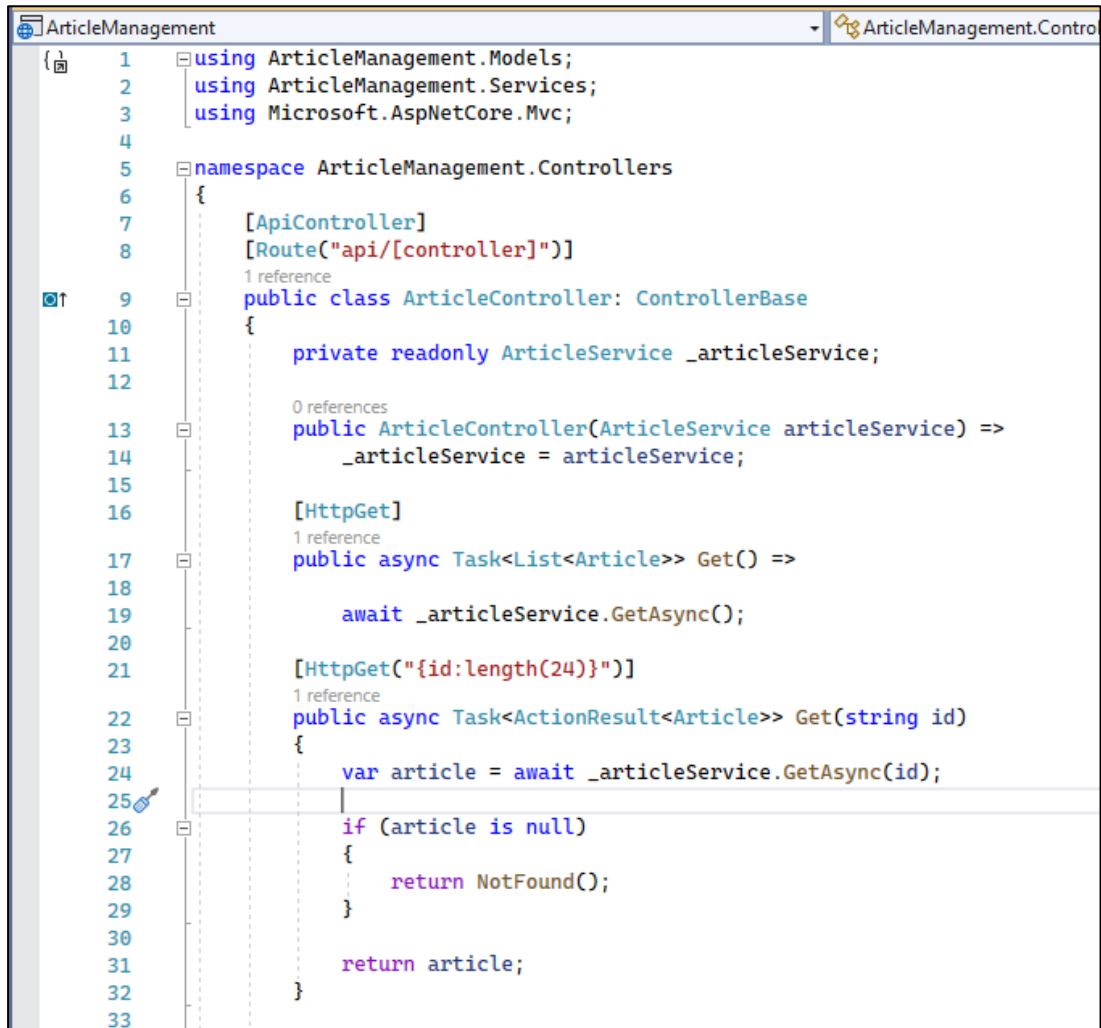
4. Add the following code to the top of Program.cs to resolve the "ArticleService" reference. In Visual studio 2022 automatically add the namespace.



The "ArticleService" class uses the following MongoDB.Driver members to run CRUD operations against the database


Add a controller

1. Add a "ArticleController" class to the Controllers directory with the following code



```
1 using ArticleManagement.Models;
2 using ArticleManagement.Services;
3 using Microsoft.AspNetCore.Mvc;
4
5 namespace ArticleManagement.Controllers
6 {
7     [ApiController]
8     [Route("api/[controller]")]
9     public class ArticleController: ControllerBase
10    {
11        private readonly ArticleService _articleService;
12
13        public ArticleController(ArticleService articleService) =>
14            _articleService = articleService;
15
16        [HttpGet]
17        public async Task<List<Article>> Get() =>
18        {
19            await _articleService.GetAsync();
20        }
21
22        [HttpGet("{id:length(24)}")]
23        public async Task<ActionResult<Article>> Get(string id)
24        {
25            var article = await _articleService.GetAsync(id);
26
27            if (article is null)
28            {
29                return NotFound();
30            }
31
32            return article;
33        }
34    }
```

Test the web API

 Select a definition ArticleManagement v1

ArticleManagement 1.0 OK

View the latest OpenAPI definition here

Article

GET `/api/Article`

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Success	API docs

Media type

application/json

Content-accept header

Example Value Schema

```
{  "articleId": "string",  "articleName": "string",  "description": "string",  "author": "string"}
```

POST `/api/Article`

Parameters

Try it out

No parameters

Request body

application/json

Example Value Schema

```
{  "articleId": "string",  "articleName": "string",  "description": "string",  "author": "string"}
```

Responses

Code	Description	Links
200	Success	API docs

GET `/api/Article/{id}`

Parameters

Try it out

Name	Description
id <small>required</small>	

string (path)

id

Responses

Code	Description	Links
200	Success	API docs

Media type

application/json

Content-accept header

Example Value Schema

```
{  "articleId": "string",  "articleName": "string",  "description": "string",  "author": "string"}
```

PUT `/api/Article/{id}`

Parameters

Try it out

Name	Description
id <small>required</small>	

string (path)

id

Request body

application/json

Example Value Schema

```
{  "articleId": "string",  "articleName": "string",  "description": "string",  "author": "string"}
```

Responses

Code	Description	Links
200	Success	API docs

DELETE `/api/Article/{id}`

Parameters

Try it out

Name	Description
id <small>required</small>	

string (path)

id

Responses

Code	Description	Links
200	Success	API docs

Schemas

Article >

Create / Insert

URL: <https://localhost:7068/api/article> (The URL may be differed system to system)

The screenshot shows the Postman interface for a POST request to `https://localhost:7068/api/article`. The request body is a JSON object with the following fields:

```
{  "articleId": "634edf085eeec043ecb385ae",  "articleName": "Elcom",  "description": "We help organisations build world-class intranet, portal, website, learning management and digital workplace solutions - giving staff the tools they need to deliver exceptional digital customer and employee experiences.",  "author": "Vadivel"}
```

The response status is **201 Created** with a time of 1884 ms and a size of 541 B. The response body is the same JSON object as the request body.

The screenshot shows the MongoDB Compass interface for the `ArticleDB.Articles` collection. The collection contains three documents:

```
{  "_id": ObjectId('634edf085eeec043ecb385ae'),  "ArticleName": "Elcom",  "Description": "We help organisations build world-class intranet, portal, website, lea...",  "Author": "Vadivel"}{  "_id": ObjectId('634edf085eeec043ecb385aa'),  "ArticleName": "MongoDB",  "Description": "MongoDB is a source-available cross-platform document-oriented databas...",  "Author": "Vadivel Rangasamy"}{  "_id": ObjectId('634edf085eeec043ecb385ab'),  "ArticleName": "VisualStudio",  "Description": "Visual Studio is an integrated development environment (IDE) from Micr...",  "Author": "Vadivel Rangasamy"}
```

Read

URL: <https://localhost:7068/api/article/634edf085eeec043ecb385aa> (The URL may be differed system to system)

The screenshot shows the Postman interface for a GET request to the endpoint `https://localhost:7068/api/article/634edf085eeec043ecb385aa`. The request is successful, returning a 200 OK status with a response time of 159 ms and a size of 572 B. The response body is displayed in JSON format, showing the article details.

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

```
1 {
2   "articleId": "634edf085eeec043ecb385aa",
3   "articleName": "MongoDB",
4   "description": "MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB
5     uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License which is
6     deemed non-free by several distributions.",
7   "author": "Vadivel Rangasamy"
8 }
```

Update

URL: <https://localhost:7068/api/Article/634edf085eeec043ecb385aa> (The URL may be differed system to system)

The screenshot shows the Postman interface for a PUT request to the endpoint `https://localhost:7068/api/Article/634edf085eeec043ecb385aa`. The request is successful, returning a 204 No Content status with a response time of 23 ms and a size of 81 B. The response body is displayed in JSON format, showing the updated article details.

Body

```
1 {
2   "articleName": "VisualStudio - Update",
3   "description": "Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as
4     websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows
5     Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.",
6   "author": "Vadivel Rangasamy-CBE"
7 }
```

204 No Content

The server successfully processed the request, but is not returning any content.

Delete

URL: <https://localhost:7068/api/Article/634edf085eeec043ecb385ae> (The URL may be differed system to system)

The screenshot shows an API client interface with a tab for 'Delete Article'. The URL is `https://localhost:7068/api/article/634edf085eeec043ecb385ae`. The method is 'DELETE'. The status is '204 No Content', with a time of '118 ms' and size of '81 B'. A tooltip explains: '204 No Content. The server successfully processed the request, but is not returning any content.'

DELETE `https://localhost:7068/api/article/634edf085eeec043ecb385ae` Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Co

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (2) Test Results

Pretty Raw Preview Visualize Text

1

Status: 204 No Content Time: 118 ms Size: 81 B Save Response

204 No Content

The server successfully processed the request, but is not returning any content.


The screenshot shows the MongoDB Compass interface. The left sidebar shows the database structure: 'localhost:27017' with '4 DBS' and '4 COLLECTIONS'. The 'ArticleDB' database is selected, and the 'Articles' collection is highlighted. The main panel shows the 'Documents' tab for 'ArticleDB.Articles'. A filter is applied: `_id: '634edf085eeec043ecb385aa'`. The document is highlighted in yellow. The document details are:

```
{
  "_id": ObjectId('634edf085eeec043ecb385aa'),
  "ArticleName": "VisualStudio - Update",
  "Description": "Visual Studio is an integrated development environment (IDE) from Micr...",
  "Author": "Vadivel Rangasamy-CBE"
}
```

The document is successfully deleted, as indicated by the '204 No Content' status in the API client screenshot.

Not yet to resolve

1. Directly used connection string `"var mongoClient = new MongoClient("mongodb://localhost:27017");"` instead of articleDatabaseSettings object `"var mongoClient = new MongoClient(articleDatabaseSettings.Value.ConnectionString);"`.



```
4 using MongoDB.Driver;
5 namespace ArticleManagement.Services
6 {
7     4 references
8     public class ArticleService
9     {
10         private readonly IMongoCollection<Article> _articlesCollection;
11
12         0 references
13         public ArticleService(
14             IOption<ArticleDatabaseSettings> articleDatabaseSettings)
15         {
16             bool isMongoLive = MongoDBConnectionProbe("mongodb://localhost:27017", "ArticleDB"); //database.RunCommandAsync((Command<BsonDocument>)"(ping:1)").Wait(1000);
17             if (isMongoLive)
18             {
19                 var mongoClient = new MongoClient("mongodb://localhost:27017"); //(articleDatabaseSettings.Value.ConnectionString);
20                 var mongoDatabase = mongoClient.GetDatabase("ArticleDB"); //(articleDatabaseSettings.Value.DatabaseName);
21                 _articlesCollection = mongoDatabase.GetCollection<Article>("Articles"); //(articleDatabaseSettings.Value.ArticleCollectionName);
22             }
23             else
24             {
25                 // couldn't connect
26             }
27
28         1 reference
29         public async Task<List<Article>> GetAsync() =>
30             await _articlesCollection.Find(_ => true).ToListAsync();
31     }
32 }
```

2. The web site is not published to a specific web server. I have directly run the website with auto generated URLs

<https://localhost:7068/> to <https://testwebapi.com>

Source code

URL: <https://drive.google.com/file/d/1HJmpwRbUJZm8PKIysM1GPnPI55vCDbbT/view?usp=sharing>