

Bài giảng tích hợp:

ENTITY FRAMEWORK CORE

Faculty of IT

Email: smdat@hueic.edu.vn

ENTITY FRAMEWORK CORE

Chương 1. Tổng quan về Entity Framework Core

Chương 2. Truy vấn trong Entity Framework Core

Chương 3. Razor Page với Entity Framework Core

Chương 4. Asp.Net Core MVC với Entity Framework Core

Chương 5: Web API

1. Giới thiệu

- **API là gì?**

- API là viết tắt của Application Programming Interface – phương thức trung gian kết nối các ứng dụng và thư viện khác nhau.
- Nó cung cấp khả năng truy xuất đến một tập các hàm hay dùng, từ đó có thể trao đổi dữ liệu giữa các ứng dụng.
- Thi thoảng vẫn có người lầm tưởng API là một ngôn ngữ lập trình nhưng thực ra, API chỉ là các hàm hay thủ tục thông thường. Các hàm này được viết trên nhiều ngôn ngữ lập trình khác nhau.
 - *Để hiểu rõ hơn API là gì, hãy tưởng tượng bạn đang ngồi trong một nhà hàng, trước mặt bạn là menu để gọi thức ăn. Nhà bếp là một phần của “hệ thống”, nơi sẽ chuẩn bị những món ăn mà bạn gọi. Tuy nhiên, làm thế nào để nhà bếp biết được bạn muốn ăn món nào? Và làm sao để họ phân phối thức ăn đến bàn của bạn? Đây là lúc cần đến sự xuất hiện của người phục vụ, đóng vai trò như API.*
 - *Người phục vụ (hay API) sẽ nhận yêu cầu từ bạn và truyền đạt với nhà bếp (hệ thống) những thứ cần làm. Sau đó người phục vụ sẽ phản hồi ngược lại cho bạn, trong trường hợp này, họ sẽ mang thức ăn sau khi nhà bếp hoàn thành đến tận bàn cho bạn.*

1. Giới thiệu

• 4 đặc điểm nổi bật của API

- API sử dụng mã nguồn mở, dùng được với mọi client hỗ trợ XML, JSON
- API có khả năng đáp ứng đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content forma.... Bạn có thể sử dụng các host nằm trong phần ứng dụng hoặc trên IIS.
- Mô hình web API dùng để hỗ trợ MVC như: unit test, injection, ioc container, model binder, action result, filter, routing, controller. Ngoài ra, nó cũng hỗ trợ RESTful đầy đủ các phương thức như: GET, POST, PUT, DELETE các dữ liệu.
- Được đánh giá là một trong những kiểu kiến trúc hỗ trợ tốt nhất với các thiết bị có lượng băng thông bị giới hạn như smartphone, tablet...

1. Giới thiệu

- **Ứng dụng của API**

- **Web API:** Là hệ thống API được sử dụng trong các hệ thống website, chẳng hạn: Google, Facebook... Hầu hết các website đều cung cấp hệ thống API cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. Đa số Web API được thiết kế theo tiêu chuẩn [RESTful](#).
- **API trên hệ điều hành:** Windows hay Linux có rất nhiều API. Họ cung cấp các tài liệu API là đặc tả các hàm, phương thức cũng như các giao thức kết nối. Nó giúp [lập trình viên](#) có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành.
- **API của thư viện phần mềm (framework):** API mô tả và quy định các hành động mong muốn mà các thư viện cung cấp. Một API có thể có nhiều cách triển khai khác nhau, giúp cho một chương trình viết bằng ngôn ngữ này có thể sử dụng được thư viện viết bằng ngôn ngữ khác.

1. Giới thiệu

- **Web API là gì?**
 - **Web API** là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường ở dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS.
- **Những điểm nổi bật của Web API:** Web API hỗ trợ restful đầy đủ các phương thức: Get/Post/put/delete dữ liệu. Nó giúp xây dựng các HTTP service một cách rất đơn giản và nhanh chóng. Nó cũng có khả năng hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content format.
 - **Tự động hóa sản phẩm:** Với **web API**, chúng ta sẽ tự động hóa quản lý công việc, cập nhật luồng công việc, giúp tăng năng suất và tạo hiệu quả công việc cao hơn.
 - **Khả năng tích hợp linh động:** API cho phép lấy nội dung từ bất kỳ website hoặc ứng dụng nào một cách dễ dàng nếu được cho phép, tăng trải nghiệm người dùng. API hoạt động như một chiếc cổng, cho phép các công ty chia sẻ thông tin được chọn nhưng vẫn tránh được những yêu cầu không mong muốn.

1. Giới thiệu

- **Những điểm nổi bật của Web API:**
 - **Cập nhật thông tin thời gian thực:** API có chức năng thay đổi và cập nhật thay đổi theo thời gian thực. Với công nghệ này, dữ liệu sẽ được truyền đi tốt hơn, thông tin chính xác hơn, dịch vụ cung cấp linh hoạt hơn.
 - **Có tiêu chuẩn chung để sử dụng:** Bất kỳ người dùng, công ty nào sử dụng cũng có thể điều chỉnh nội dung, dịch vụ mà họ sử dụng. Hỗ trợ đầy đủ các thành phần **MVC** như: routing, controller, action result, filter, model binder, IoC container, [dependency injection](#), unit test.

1. Giới thiệu

• Web API hoạt động như thế nào?

1. Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi request dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.
2. Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trả về kết quả.
3. Server trả về kết quả theo định dạng JSON hoặc XML thông qua giao thức HTTP/HTTPS.
4. Tại nơi yêu cầu ban đầu là ứng dụng web hoặc ứng dụng di động, dữ liệu JSON/XML sẽ được parse để lấy data. Sau khi có được data thì thực hiện tiếp các hoạt động như lưu dữ liệu xuống Cơ sở dữ liệu, hiển thị dữ liệu...

1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

- **Tạo project**

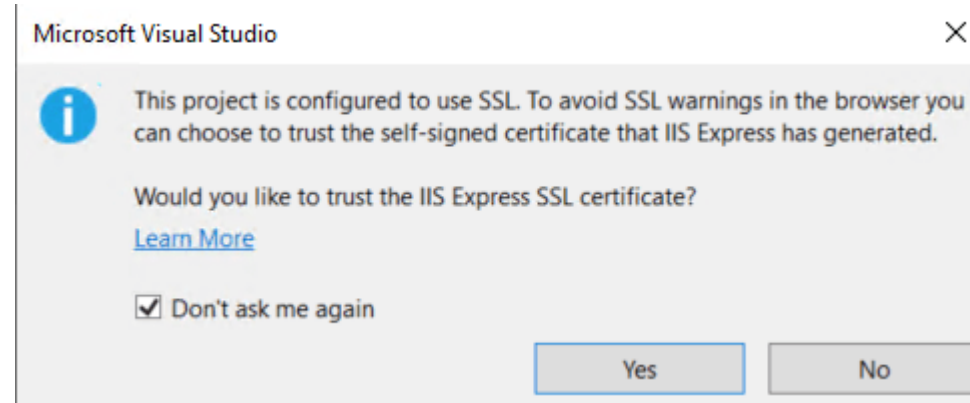
- Từ menu **File**, chọn new **Project**
- Chọn **ASP.NET Core Web API** và chọn **Next**
- Đặt tên project: **ToDoAPI** và click **Create**
- Trong hộp thoại **Create a new ASP.NET Core Web Application**, chọn **.Net Core** và **ASP.NET Core 5.0**. Chọn nút **Create**.

1. Giới thiệu

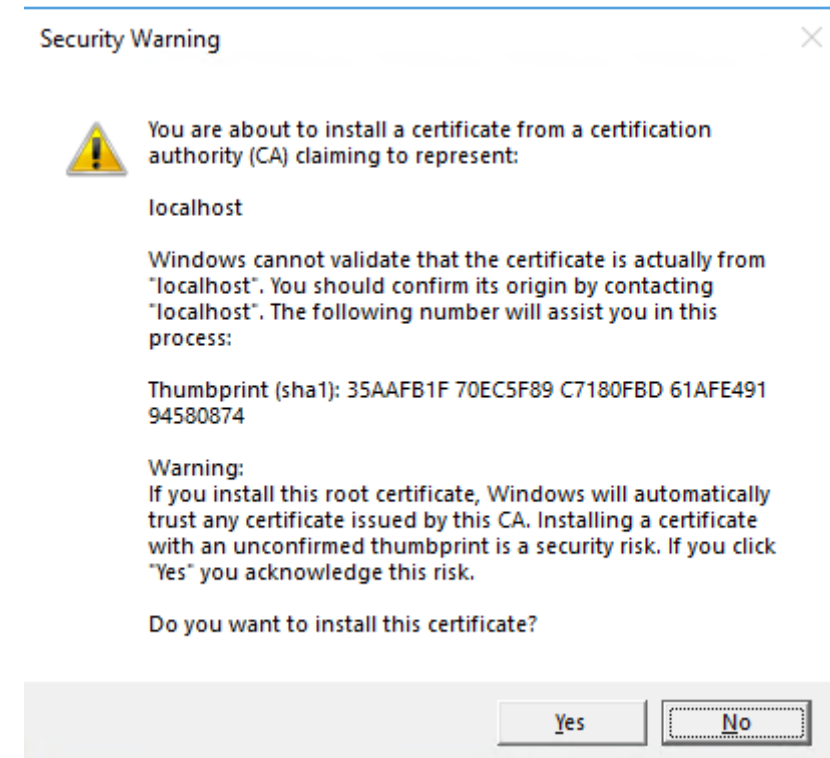
2. Tạo một web API với ASP.NET CORE

• Test project

- Project này tạo API dự báo thời tiết với sự hỗ trợ Swagger.
 - Ấn Ctrl + F5 để chạy mà không cần debugger
 - Visual sẽ hiển thị hộp thoại khi project không được cấu hình để sử dụng SSL



- Click Yes để xác nhận IIS Express SSL
- Hộp thoại sau sẽ xuất hiện:



1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

• Test project

- Trang Swagger */swagger/index.html* sẽ được hiển thị.
- Chọn **GET** > **Try it out** > **Execute**. Một trang được hiển thị gồm:
 - Lệnh Curl để test WeatherForecast API
 - URL để test WeatherForecast API
 - Phần response code, body, và headers.
 - Một danh sách xổ xuống với loại media và giá trị, lược đồ ví dụ
- Swagger được sử dụng để tạo các trang trợ giúp và tài liệu hữu ích cho các web API.
- Copy và paste Request URL vào trình duyệt: <https://localhost:<port>/WeatherForecast>. Đoạn JSON sẽ được trả về

• Test project

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

```
[
  {
    "date": "2019-07-16T19:04:05.7257911-06:00",
    "temperatureC": 52,
    "temperatureF": 125,
    "summary": "Mild"
  },
  {
    "date": "2019-07-17T19:04:05.7258461-06:00",
    "temperatureC": 36,
    "temperatureF": 96,
    "summary": "Warm"
  },
  {
    "date": "2019-07-18T19:04:05.7258467-06:00",
    "temperatureC": 39,
    "temperatureF": 102,
    "summary": "Cool"
  },
  {
    "date": "2019-07-19T19:04:05.7258471-06:00",
    "temperatureC": 10,
    "temperatureF": 49,
    "summary": "Bracing"
  },
  {
    "date": "2019-07-20T19:04:05.7258474-06:00",
    "temperatureC": -1,
    "temperatureF": 31,
    "summary": "Chilly"
  }
]
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

- **Cập nhật launchUrl**

- Trong *Properties\launchSettings.json*, cập nhật *launchUrl* từ **swagger** thành **api/todoitems**

JSON

```
"launchUrl": "api/todoitems",
```

- Bởi vì Swagger sẽ bị xóa, đánh dấu trước đó sẽ thay đổi URL được khởi chạy thành phương thức GET của controller được thêm ở phần sau.

1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

- **Thêm một class model**

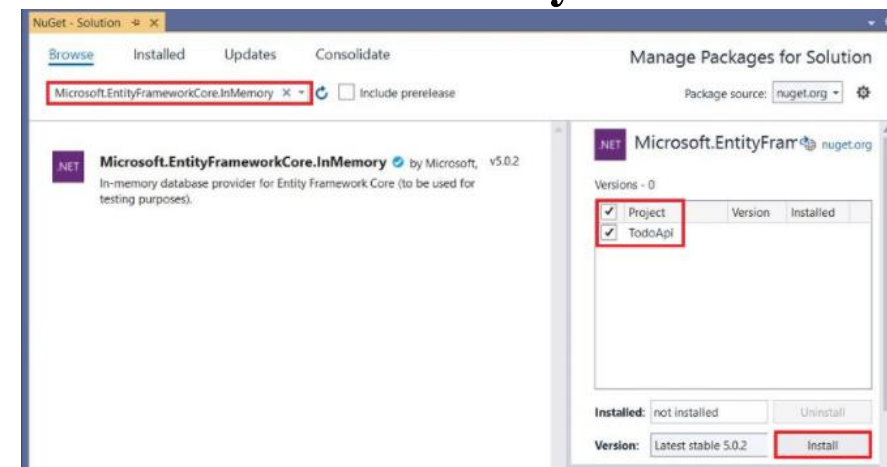
- Tạo thư mục Models, và tạo class *TodoItem.cs*

```
public class TodoItem
{
    public long Id { get; set; }
    public string Name { get; set; }
    public bool IsComplete { get; set; }
}
```

- **Thêm database context**

- **Thêm NuGet packages**

- Từ menu Tools, chọn **NuGet Package Manager > Manage NuGet Packages for Solution.**
 - Chọn thẻ **Browse**, gõ ở ô search: **Microsoft.EntityFrameworkCore.InMemory**
 - Tiến hành **Install.**



1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

- **Thêm database context**

- **Tạo TodoContext database context**

- Click phải vào thư mục Models, chọn Add -> Class. Tạo class *TodoContext*

```
public class TodoContext : DbContext {  
    public TodoContext(DbContextOptions<TodoContext> options)  
        : base(options)  
    {  
    }  
    public DbSet<TodoItem> TodoItems { get; set; }  
}
```

- **Đăng ký database context**

- Trong Startup.cs, cập nhật

```
public void ConfigureServices(IServiceCollection services) {  
    services.AddControllers();  
    services.AddDbContext<TodoContext>(opt =>  
        opt.UseInMemoryDatabase("TodoList"));  
    //services.AddSwaggerGen(c => {  
    //    c.SwaggerDoc("v1", new OpenApiInfo { Title =  
    "TodoApi", Version = "v1" });  
    //});  
}
```


1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

- **Thêm database context**
 - **Đăng ký database context**

- Trong Startup.cs, cập nhật

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        //app.UseSwagger();
        //app.UseSwaggerUI(c =>
            c.SwaggerEndpoint("/swagger/v1/swagger.json", "TodoApi
v1"));
    }
    app.UseHttpsRedirection();
    app.UseRouting();
    app.UseAuthorization();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

- **Scaffold một controller**

- Click phải thư mục Controllers, chọn **Add > New Scaffolded Item.**
- Chọn **API Controller with actions, using Entity Framework**, và nhấn **Add**
- Trong hộp thoại **Add API Controller with actions, using Entity Framework:**
 - Chọn **TodoItem (TodoApi.Models)** trong **Model class**
 - Chọn **TodoContext (TodoApi.Models)** trong **Data context class.**
 - Chọn **Add**

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

• Cập nhật phương thức tạo PostTodoItem

```
// POST: api/TodoItems [HttpPost]
public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItem
todoItem)
{
    _context.TodoItems.Add(todoItem);
    await _context.SaveChangesAsync();
    //return CreatedAtAction("GetTodoItem", new { id =
todoItem.Id }, todoItem);
    return CreatedAtAction(nameof(GetTodoItem),
        new { id = todoItem.Id }, todoItem);
}
```

1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

- **Cài đặt Postman:** sử dụng Postman để test web API
 - Cài Postman: <https://www.postman.com/downloads/>
 - Mở web app
 - Mở Postman
 - Tắt chế độ **SSL certificate verification**.
 - Từ menu File, -> **Settings (General tab)**, tắt **SSL certificate verification**
 - Lưu ý: sau khi test Controller xong thì mở lại chế độ này.

1. Giới thiệu

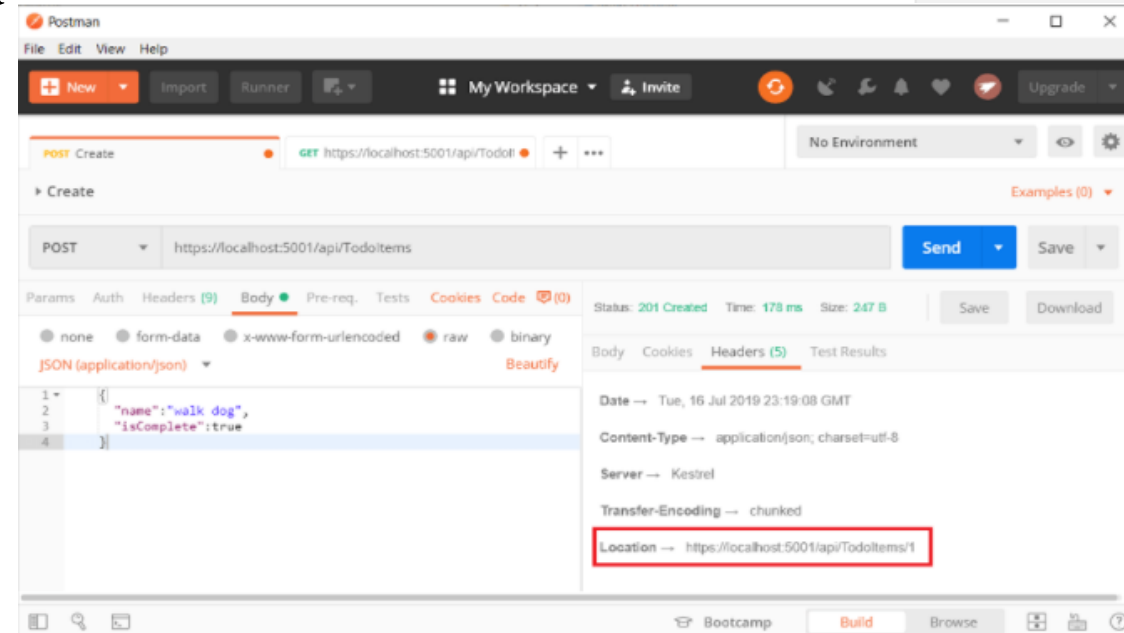
2. Tạo một web API với ASP.NET CORE

• Test PostTodoItem with Postman

- Tạo mới 1 request
- Thiết lập 1 phương thức HTTP từ **POST**
- Thiết lập **URI**: <https://localhost:<port>/api/todoitems>. Ví dụ: <https://localhost:5001/api/todoitems>
- Chọn thẻ **Body**
- Chọn radio button là **raw**
- Chọn kiểu là **JSON (application/json)**.
- Trong phần thân request, gõ JSON cho item to-do:
- Nhấn nút Send

JSON

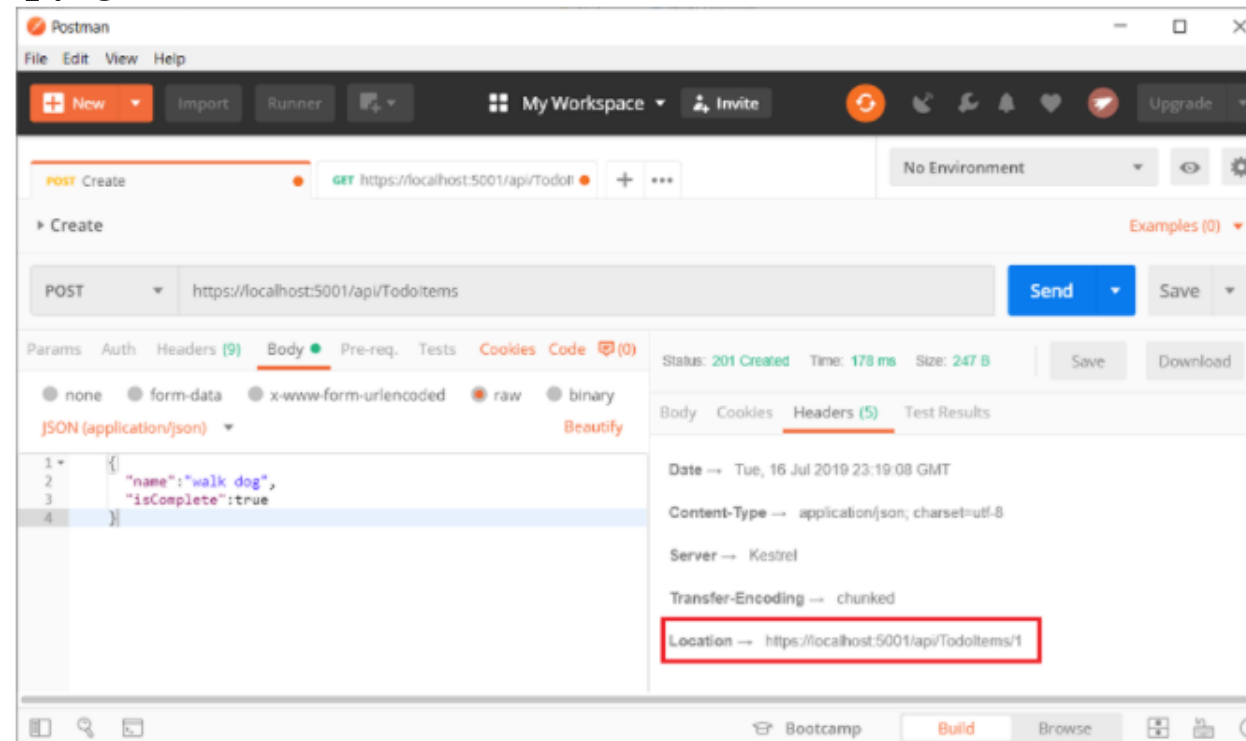
```
{  "name": "walk dog",  "isComplete": true}
```



1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

• Test the location header URI

- Phần **location header URI** được test trong trình duyệt. Copy và paste location header URI vào trong trình duyệt
 - Chọn thẻ **Headers** trong khung **Response**
 - Copy giá trị **Location header**



- Thiết lập **HTTP** là **GET**
- Thiết lập URI thành <https://localhost:5001/api/todoitems/1>
- Nhấn nút **Send**

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE

- **Examine the GET methods**
 - URI của GET có 2 dạng:
 - GET /api/todoitems
 - GET /api/todoitems/{id}
 - **Ví dụ:**
 - https://localhost:5001/api/todoitems
 - https://localhost:5001/api/todoitems/1
 - Khi thực hiện GetTodoItems thì được

JSON

```
[  
  {  
    "id": 1,  
    "name": "Item1",  
    "isComplete": false  
  }  
]
```


1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

• Test Get với Postman

- Tạo một request mới
- Thiết lập phương thức HTTP là GET
- Thiết lập request URI: `https://localhost:<port>/api/todoitems`. Ví dụ: <https://localhost:5001/api/todoitems>
- Thiết lập chế độ xem **Two pane view** trong Postman
- Nhấn nút Send

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

- Cài đặt MongoDB

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

- Tạo database: BookStoreDB

- Bảng: Books

- Nhập dữ liệu:


```
{  
  "Name" : "Design Patterns",  
  "Price" : 54.93,  
  "Category" : "Computers",  
  "Author" : "Ralph Johnson"  
}
```

```
{  
  "Name" : "Clean Code",  
  "Price" : 43.15,  
  "Category" : "Computers",  
  "Author" : "Robert C. Martin"  
}
```

- 1. Giới thiệu
- 2. Tạo một web API với ASP.NET CORE
- 3. Web API với MongoDB


- **Tạo ASP.NET Core web API project**
 - **File > New > Project**
 - Chọn loại **ASP.NET Core Web Application** project, nhấn Next
 - Đặt tên project: BooksAPI, và nhấn Create
 - Chọn framework **.NET Core** và **ASP.NET Core 5.0**. Chọn mẫu API và nhấn Create
 - Trong cửa sổ **Package Manager Console**, gõ lệnh:

Install-Package MongoDB.Driver -Version {VERSION}
 - Hoặc




MongoDB.Bson by MongoDB Inc., 63,5M downloads
MongoDB's Official Bson Library.

2.13.2




MongoDB.Driver.Core by MongoDB Inc., 59,5M downloads
Core Component of the Official MongoDB .NET Driver.

2.13.2




MongoDB.Driver by MongoDB Inc., 58,3M downloads
Official .NET driver for MongoDB.

2.13.2




MongoDB.Libmongocrypt by MongoDB Inc., 23,4M downloads
Libmongocrypt wrapper for the .NET driver.

1.2.3



MongoDB.Driver.GridFS by MongoDB Inc., 3,45M downloads
GridFS Component of the Official MongoDB .NET Driver.

2.13.2



MongoDB.Driver

nuget.org

Versions - 1

<input checked="" type="checkbox"/>	Project	Version	Installed
<input checked="" type="checkbox"/>	BookStoreAPI	2.13.2	2.13.2

Installed: 2.13.2

Uninstall

Version: Latest stable 2.13.2

Install

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

- **Tạo model**

- Tạo thư mục Models
- Tạo class Book trong thư mục này

```
public class Book
{
    [BsonId]
    [BsonRepresentation(BsonType.ObjectId)]
    public string Id { get; set; }
    [BsonElement("Name")]
    public string BookName { get; set; }
    public decimal Price { get; set; }
    public string Category { get; set; }
    public string Author { get; set; }
}
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

- Cấu hình model

- Mở file *appsettings.json*

```
{  
  "BookstoreDatabaseSettings": {  
    "BooksCollectionName": "Books",  
    "ConnectionString": "mongodb://localhost:27017",  
    "DatabaseName": "BookstoreDb"  
  },  
  .....  
}
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

• Cấu hình model

- Thêm file *BookstoreDatabaseSettings.cs* trong thư mục Models

```
public class BookstoreDatabaseSettings :  
    IBookstoreDatabaseSettings  
{  
    public string BooksCollectionName { get; set; }  
    public string ConnectionString { get; set; }  
    public string DatabaseName { get; set; }  
}  
  
public interface IBookstoreDatabaseSettings  
{  
    string BooksCollectionName { get; set; }  
    string ConnectionString { get; set; }  
    string DatabaseName { get; set; }  
}
```

- Lớp *BookstoreDatabaseSettings* dùng để lưu trữ giá trị các thuộc tính *BookstoreDatabaseSettings* trong file *appsettings.json*. Tên các thuộc tính trong C# và JSON phải giống nhau để dễ dàng trong quá trình kết nối.

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

- **Cấu hình model**

- Cập nhật lại **Startup.ConfigureServices**

```
public void ConfigureServices(IServiceCollection services)
{
    // requires using Microsoft.Extensions.Options
    services.Configure<BookstoreDatabaseSettings>(
        Configuration.GetSection(nameof(BookstoreDatabaseSettings)));

    services.AddSingleton<IBookstoreDatabaseSettings>(sp =>
        sp.GetRequiredService<IOptions<BookstoreDatabaseSettings>>().
Value);
    services.AddControllers();
}
```


1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

• Tạo dịch vụ cho hoạt động CRUD

- Tạo thư mục *Services*

- Tạo class *BookService* trong thư mục này

```
public class BookService {  
    private readonly IMongoCollection<Book> _books;  
    public BookService(IBookstoreDatabaseSettings settings) {  
        var client = new MongoClient(settings.ConnectionString);  
        var database = client.GetDatabase(settings.DatabaseName);  
        _books = database.GetCollection<Book>(settings.BooksCollectionName);  
    }  
    public List<Book> Get() =>  
        _books.Find(book => true).ToList();  
    public Book Get(string id) =>  
        _books.Find<Book>(book => book.Id == id).FirstOrDefault();  
    public Book Create(Book book)  
    {  
        _books.InsertOne(book);  
        return book;  
    }  
    public void Update(string id, Book bookIn) =>  
        _books.ReplaceOne(book => book.Id == id, bookIn);  
    public void Remove(Book bookIn) =>  
        _books.DeleteOne(book => book.Id == bookIn.Id);  
    public void Remove(string id) =>  
        _books.DeleteOne(book => book.Id == id);  
}
```

- **Tạo dịch vụ cho hoạt động CRUD**
 - Cập nhật *Startup.ConfigureServices*

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<BookstoreDatabaseSettings>(
        Configuration.GetSection(nameof(BookstoreDatabaseSettings)));
    services.AddSingleton<IBookstoreDatabaseSettings>(sp =>
        sp.GetRequiredService<IOptions<BookstoreDatabaseSettings>>().
Value);

    services.AddSingleton<BookService>();
    services.AddControllers();
}
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

- **Tạo controller**

- Tạo class *BooksController* trong thư mục *Controllers*

```
namespace BooksApi.Controllers {  
    [Route("api/[controller]")]  
    [ApiController]  
    public class BooksController : ControllerBase {  
        private readonly BookService _bookService;  
        public BooksController(BookService bookService) {  
            _bookService = bookService;  
        }  
        [HttpGet]  
        public ActionResult<List<Book>> Get() {  
            => _bookService.Get();  
        }  
        [HttpGet("{id:length(24)}", Name = "GetBook")]  
        public ActionResult<Book> Get(string id) {  
            var book = _bookService.Get(id);  
            if (book == null) {  
                return NotFound();  
            }  
            return book;  
        }  
    }  
}
```

Chương 5. Web API

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

```
[HttpPost]
public ActionResult<Book> Create(Book book) {
    _bookService.Create(book);
    return CreatedAtRoute("GetBook", new {
        id = book.Id.ToString() }, book);
}

[HttpPut("{id:length(24)}")]
public IActionResult Update(string id, Book bookIn) {
    var book = _bookService.Get(id);
    if (book == null) {
        return NotFound(); }
    _bookService.Update(id, bookIn);
    return NoContent();
}

[HttpDelete("{id:length(24)}")]
public IActionResult Delete(string id) {
    var book = _bookService.Get(id);
    if (book == null) {
        return NotFound(); }
    _bookService.Remove(book.Id);
    return NoContent();
}
}
```

1. Giới thiệu
2. Tạo một web API với ASP.NET CORE
3. Web API với MongoDB

• Test the web API

- Chạy thử chương trình
- <http://localhost:<port>/api/books> thì sẽ cho đoạn JSON sau

```
JSON

[
  {
    "id": "5bfd996f7b8e48dc15ff215d",
    "bookName": "Design Patterns",
    "price": 54.93,
    "category": "Computers",
    "author": "Ralph Johnson"
  },
  {
    "id": "5bfd996f7b8e48dc15ff215e",
    "bookName": "Clean Code",
    "price": 43.15,
    "category": "Computers",
    "author": "Robert C. Martin"
  }
]
```

- <http://localhost:<port>/api/books/{id here}> thì sẽ cho kết quả

```
JSON

{
  "id": "{ID}",
  "bookName": "Clean Code",
  "price": 43.15,
  "category": "Computers",
  "author": "Robert C. Martin"
}
```

Thank you!

1. Giới thiệu

2. Tạo một web API với ASP.NET CORE

• Routing and URL paths

- Thuộc tính [HttpGet] biểu thị một phương thức phản hồi yêu cầu HTTP GET. Đường dẫn URL cho mỗi phương thức được xây dựng như sau:

- Bắt đầu với template string trong thuộc tính Route của controller

```
[Route("api/[controller]")]
```

```
[ApiController]
```

```
public class TodoItemsController : ControllerBase {  
    private readonly TodoContext _context;  
    public TodoItemsController(TodoContext context)  
    {  
        _context = context;  
    }  
}
```

- Thay thế [controller] bằng tên của controller, theo quy ước là tên lớp của controller trừ đi hậu tố "Controller". Đối với ví dụ này, tên lớp controller là TodoItemsController , vì vậy tên controller là “TodoItems”. Routing của ASP.NET Core không phân biệt chữ hoa chữ thường.
- Nếu thuộc tính [HttpGet] có dạng route (Ví dụ: [HttpGet ("products")]), thì nối theo vào đường dẫn. Ví dụ này không sử dụng dạng mẫu này.