

This README provides instructions on how to set up and run the E-commerce API, an application for calculating the total cost of a list of watches. It also outlines the approach taken and potential areas for improvement.

Author: Feng Wei

Setup and Run the Application

Prerequisites

- .NET 5 SDK
- Visual Studio, Visual Studio Code, or any preferred code editor

Steps

1. Unzip the zip file.

2. Open the Project:

Open the project in your preferred code editor.

3. Build and Run:

Build and run the project by using the built-in tools in your code editor.

Expected output example:

WebApplication1 > WebApplication3 > bin > Debug > net5.0 >

Sort View ...

Name	Date modified	Type	Size
ref	11/7/2023 7:04 PM	File folder	
appsettings.Development.json	11/7/2023 5:09 PM	JSON File	1 KB
appsettings.json	11/7/2023 5:09 PM	JSON File	1 KB
BCGdv.deps.json	11/7/2023 7:04 PM	JSON File	105 KB
BCGdv.dll	11/7/2023 7:04 PM	Application exten...	15 KB
BCGdv.exe	11/7/2023 7:04 PM	Application	123 KB
BCGdv.pdb	11/7/2023 7:04 PM	Program Debug D...	21 KB
BCGdv.runtimeconfig.dev.json	11/7/2023 7:04 PM	JSON File	1 KB
BCGdv.runtimeconfig.json	11/7/2023 7:04 PM	JSON File	1 KB
BCGdv.Views.dll	11/7/2023 7:04 PM	Application exten...	39 KB
BCGdv.Views.pdb	11/7/2023 7:04 PM	Program Debug D...	22 KB

4. API Endpoint:

The API will be accessible at <http://localhost:8080/checkout>.

5. Use API:

You can make POST requests to the /checkout endpoint with a list of watch IDs to calculate the total cost.

POST <http://localhost:8080/checkout>

<http://localhost:8080/checkout>

POST <http://localhost:8080/checkout>

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Headers 8 hidden

Key	Value
<input checked="" type="checkbox"/> Content-Type	application/json
<input checked="" type="checkbox"/> Accept	application/json
Key	Value

Example

- example1

```
1  ✓ [
2  "001",
3  "001",
4  "001"
5  ]
```

Expected results:

```
1  {
2  "price": 200
3  }
```

- Example2:

```
[
"001",
"001",
"001",
"002",
"002",
"003",
"004"
]
```

Output:

```
1  ✓ {
2  "price": 400
3  }
```

Approach

- Framework: The application is built using ASP.NET Core, a robust and cross-platform framework for building web APIs.
-
- Data Structure: A watch catalog is defined in the “CheckoutController” with watch details such as ID, unit price, and discount rules.
-
- API Endpoint: The /checkout endpoint accepts a POST request with a list of watch IDs, calculates the total cost, and returns the result.
-
- Discount Logic: A custom discount calculation logic is implemented based on the discount rules provided.
-
- Testing: The application includes unit tests to verify the correctness of individual components and ensure that the discount logic works as expected.

Potential Improvements

- Error Handling: Implement comprehensive error handling to provide meaningful error responses to clients when issues occur.
- Authentication and Authorization: Add authentication and authorization mechanisms to secure the API.

Git commit history

I did not use git this time.

Automated testing

Automated testing is an important part of the solution.

Unit Tests: Unit tests are included to ensure the correctness of individual components, including the discount calculation logic.