**BasketCartPrototype (Web API Endpoint)**

**GitHub Link:** <https://github.com/LambrosP/BasketCartPrototype>

**Target FrameWork:** .NET Framework 4.7.2

**Projects:**

* **BasketCartPrototype.Data**

Within this Project I have the StockItems class which is representing a list of Products (it is going to be stored in Memory).

I am keeping in memory all the available Products within **List<Product> Products** as well as the List of Baskets which is keeping in memory all the current baskets for every customer. For that purpose I use the CacheManager implementation in order to handle in memory both Available Products and Baskets.

Additionally, within this project I Implement the BasketContext and ProductContext classes which are responsible for Products and Baskets manipulation.

* **BasketCartPrototype.Core**

Within this project I have all the entities as well as the models. Also, I have the interfaces for Cache and Services implementations.

* **BasketCartPrototype.Service**

Within this project I have the implementation of the services.

* **BasketCartPrototype.API**

Within this project I have all the endpoints under the BasketCartController Class.

The whole project was based on asynchronous calls and for that I use the TPL (Task Parallel Library) and async / await. Additionally for DI (Dependency Injection) pattern I use the unity container framework.

Below is the List of **API EndPoints**:

1. **AddBasketItemAsync**

Adding a Product Item in the memory List. It takes as a parameter BasketItemViewModel (customerId, ProductId, Quantity). If the customer does not exist within the DB, it is inserting the customer and the corresponding product item. If yes, then insert the Product Item for that customer and the relevant quantity. It updates the total balance for TotalItems and TotalSub within the customer’s Basket as well as the quantity of the specific product in Products List.

1. **UpdateBasketItemAsync**

Updates an existing Product Item’s quantity for an existing customer. It updates the total balance for TotalItems and TotalSub within the customer’s Basket as well as the quantity of the specific product in Products List .It takes as a parameter BasketItemViewModel (customerId, ProductId, Quantity).

1. **RemoveBasketItemAsync**

Removes an existing Product item for a specific customer. Additionally, It updates the total balance for TotalItems and TotalSub within the customer’s Basket as well as the quantity of the specific product in Producst List. It takes as a parameter BasketItemViewModel (customerId, ProductId, Quantity).

1. **ClearCustomerBasketAsync**

Clears the basket of a specific customer. It updates the total balance for TotalItems and TotalSub within the customer’s Basket as well as the quantity of the specific product in Producst List. It takes as a parameter BasketItemViewModel (customerId, ProductId, Quantity).

1. **GetCustomerBasketItemsAsync**

Gets all the Basket Items for a specific Customer. It takes as a parameter the customerId.

1. **GetAvailableProductsInfoAsync**

Gets a list of the available Product Items (Info) within the memory.

1. **GetProductItemAsync**

Gets information for a specific Product Item. It takes as a parameter the productId.

**ConsumeShoppingCartAPI (Console Application Project)**

**GitHub Link:** <https://github.com/LambrosP/ConsumeShoppingCartAPI>

In order to run the application you have to enter the uriString of the RESTful Endpoint once the application is launched (within the console).

**Example:** <http://localhost:53350/>

**Target FrameWork:** .NET Framework 4.7.2

I use the WepApi.Client in order to consume all the RESTful EndPoints of the application.

I created the following scenario in order to test the project.

1. //Add Product Item = 1 for Customer = 1 (Quantity = 100)
2. //Add Product Item = 1 for Customer = 2 (Quantity = 100)
3. //Update Product Item = 1 for Customer = 1 (Quantity = 200)
4. //Get Customer Basket Items (customerId = 1)
5. //Get Customer Basket Items (customerId = 2)
6. //Remove Product Item = 1 for Customer = 1 (Total Quantity = 300)
7. //Clear Basket for Customer = 1
8. //Clear Basket for Customer = 2
9. //Get Available Products Info
10. //Get Product = 1 Info

All the relevant messages are presenting within the console once the application will be executed.