SugarRush

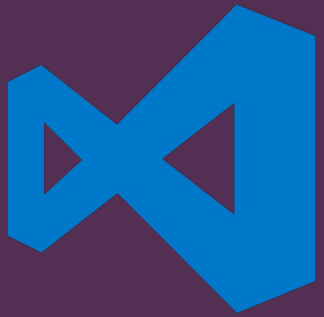
A Sweet Web Shop

Presented By

Jayson Lennon

Technologies & Tools

- C# Programming Language
- Visual Studio Code
- .NET Core
- ASP.NET Core
- Entity Framework Core
- SQLite
- SASS
- Git



Documentation

```dotnet
/// <summary>
/// Retrieve all the submitted orders made by a customer.
/// <remarks>
/// The orders returned will only be ones that have been submitted for processing.
/// Open orders will not be returned.
/// </remarks>
/// </summary>
/// <param name="userId">The user id to query.</param>
/// <returns>IEnumerable of Tuple containing the Order and quantity of items within the order.</returns>
2 references
IEnumerable<Tuple<Order, int>> IOrder.GetSubmittedOrders(Guid userId)
{
    return _context.Orders
        .Where(o => o.User.UserId == userId)
        .Where(o => o.TimeSubmitted != null)
        .Select(o =>
            new Tuple<Order, int>(
                o,
                o.OrderLineItems
                    .Where(li => li.Order.OrderId == o.OrderId)
                    .Sum(li => li.Quantity)
            )
        )
        .AsEnumerable();
}

/// <summary>
/// Retrieve all the other lines within an order.
/// </summary>
/// <param name="userId">The user id of the order.</param>
/// <param name="orderId">The order id to query.</param>
/// <returns>IEnumerable of OrderLineItem.</returns>
11 references
IEnumerable<OrderLineItem> IOrder.GetOrderLines(Guid userId, Guid orderId)
{
    return _context.OrderLineItems
        .Include(ol => ol.Product)
        .Where(ol => ol.Order.OrderId == orderId)
        .Where(ol => ol.Order.User.UserId == userId)
        .Select(ol => ol)
        .OrderBy(ol => ol.Product.Name)
        .AsEnumerable();
}
```

```html
<script>
window.addEventListener("load", function() {
    // Applies highlighting to raw text using spans.
    // Returns a new element that may be inserted into the DOM.
    //
    // The highlighter will apply a highlight to every string present
    // in the arrayOfSubstrToHighlight array.
    //
    // classToApply is the clasname to be applied to the highlighted
    // characters.
    function highlight(text, arrayOfSubstrToHighlight, classToApply) {
        // This function works by creating an array of booleans
        // corresponding to the length of the input text.
        //
        // Whenever a letter in the text should be highlighted, the
        // corresponding index in the boolean array is switched to
        // true incidating that the letter should be highlighted.
        //
        // Once all substrings have been searched in the text,
        // elements are created for each letter and a
        // class is applied to the ones with true entries in
        // the boolean array.
        let container = document.createElement("span");

        // Generate boolean array indicating whether a letter in the
        // text should be highlighted.
        let highlightIndices = [];
        for (let i = 0; i < text.length; i++) {
            highlightIndices.push(false);
        }

        // Iterate through each substring
        for (let i = 0; i < arrayOfSubstrToHighlight.length; i++) {
            let substr = arrayOfSubstrToHighlight[i];
            let matchIndex = text.toLowerCase().indexOf(substr);
            if (matchIndex != -1) {
                // For each letter in the original text, set the corresponding
                // value in highlightIndices to true.
                for (let c = matchIndex; c < matchIndex + substr.length; c++) {
                    highlightIndices[c] = true;
                }
            }
        }
```

Testing

```
jaysonLennon-repo1/Project1 on  master [$!] took 3s
❯ just test
cd Tests && dotnet test
TestOrderRepository.cs(476,27): warning xUnit1013: Public
Test run for /home/jayson/data/dev/revature/jaysonLennon-r
Microsoft (R) Test Execution Command Line Tool Version 16.
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 23
     Passed: 23
 Total time: 3.0987 Seconds
```

```
0 references | Run Test | Debug Test
public async void RejectsOrderWhenOutOfStock() ···

[Fact]
0 references | Run Test | Debug Test
public async void RejectsEmptyOrders() ···

[Fact]
0 references | Run Test | Debug Test
public async void RejectsOrdersWithInvalidIds() ···

[Fact]
0 references | Run Test | Debug Test
public void GetsSubmittedOrders() ···

[Fact]
0 references | Run Test | Debug Test
public void GetsOrderLines() ···

[Fact]
0 references | Run Test | Debug Test
public void RejectsOrderLinesWithBadIds() ···

[Fact]
0 references | Run Test | Debug Test
public async void SetsLineItemQuantity() ···

[Fact]
0 references | Run Test | Debug Test
public async void RejectsLineItemAdjustmentWhenItExceedsStock() ···

[Fact]
0 references | Run Test | Debug Test
public async void RejectsLineItemAdjustmentWithBadIds() ···

[Fact]
0 references | Run Test | Debug Test
public async void AddsLineItem() ···

[Fact]
0 references | Run Test | Debug Test
public async void RejectsNewLineItemWhenItExceedsStock() ···

[Fact]
```

MVC

- Using MVC pattern in ASP.NET Core
- Strongly-typed views with view models
- View logic limited:
  - Login status
  - Special messages (errors, empty data)

Data

- Data access using repository pattern
  - Repositories implemented using interfaces
- Database is in 3NF
  - Built using EFCore code-first

```
namespace StoreApp.Data
{
    /// <summary>
    /// Context used to access the database.
    /// </summary>
    85 references
    public class StoreContext : DbContext
    {
        /// <summary>The Products table.</summary>
        3 references
        public DbSet<Entity.Product> Products { get; set; }

        /// <summary>The Users table.</summary>
        14 references
        public DbSet<Entity.User> Users { get; set; }

        /// <summary>The Locations table.</summary>
        3 references
        public DbSet<Entity.Location> Locations { get; set; }

        /// <summary>The LocationInventories table.</summary>
        8 references
        public DbSet<Entity.LocationInventory> LocationInventories { get; set; }
        /// <summary>The Orders table.</summary>

        11 references
        public DbSet<Entity.Order> Orders { get; set; }
        /// <summary>The OrderLineItems table.</summary>

        8 references
        public DbSet<Entity.OrderLineItem> OrderLineItems { get; set; }

        /// <summary>The Addresses table.</summary>
        4 references
        public DbSet<Entity.Address> Addresses { get; set; }

        /// <summary>The table for Line1 of addresses.</summary>
        1 reference
        public DbSet<Entity.AddressLine1> AddressLine1s { get; set; }

        /// <summary>The table for Line2 of addresses.</summary>
        1 reference
        public DbSet<Entity.AddressLine2> AddressLine2s { get; set; }
```

Implementation Details

- Exception handling
- CSRF prevention
- Logging
- No public fields
- Dependency injection
- Loose coupling of repositories, entities, domain models, and business logic

```
int inStock;
try
{
    inStock = await _context.LocationInventories
        .Where(li => li.Product.ProductId == lineItem.Product.ProductId)
        .Where(li => li.Location.LocationId == order.Location.LocationId)
        .SumAsync(li => li.Quantity);
}
catch (NullReferenceException)
{
    return SetLineItemQuantityResult.ProductMissing;
}

if (lineItem == null) return SetLineItemQuantityResult.ProductMissing;
```

```
}
case StoreApp.Repository.PlaceOrderResult.OutOfStock:
{
    this._logger.LogWarning($"An order failed to be submitted due to insuffic
    this.SetFlashError("Unable to place order: Some items are out of stock.")
    return RedirectToAction("PlaceOrderError", "Checkout");
}
case StoreApp.Repository.PlaceOrderResult.NoLineItems:
{
    this._logger.LogWarning($"An order failed to be submitted due to not havi
    this.SetFlashError("Unable to place order: There are no items in your ord
    return RedirectToAction("PlaceOrderError", "Checkout");
}
case StoreApp.Repository.PlaceOrderResult.OrderNotFound:
{
    this._logger.LogCritical($"An order failed to be submitted due to missing
    this.SetFlashError("Unable to place order: No order was found.");
    return RedirectToAction("PlaceOrderError", "Checkout");
}
```

Demo

```
jaysonLennon-repo1/Project1 on master [$!]
❯ dotnet run
```

Questions?

```
SELECT Answer FROM Batch
WHERE Member = 'Jayson';
```