Technical Specifications Document

Greatwide Truckload Management Demo App

Developed By: Jeremy Willhelm
jtwillhelm@gmail.com
https://github.com/CodeslayerJay/GreatwideAppDemo

Setup:

I have included the sample AdventureWorks scripts to create and seed the database. In the folder “DBSchema\AdventureWorks” locate the script file “instawdb.sql” and follow the instructions included in the script. The Demo Application ConnectionString points to the default database “AdventureWorks”. If creating a different database name then you would have to update the Database name in appsettings.json under ConnectionStrings.

Connecting to the database:

The application uses a local version of SqlLite. You may need to change your connection string so it points to your local SQL Server. You can find the connection string in “appsettings.json” under ConnectionStrings.

Additional Scripts:

Included is a script for seeind product reviews for testing average ratings, view ratings, reviews, etc on the product details page.

In SSMS open the script file “DB_ProductReviewSeeds” found in DBSchema folder. Press F5 to execute the script.

Domain/Data Model

The entities in the domain are a basic model of the database. For the purposes of this demo I have excluded all the other tables and relational data that was not required.

Architecture

The application is built on a three layer architecture approach. The domain is strictly system/c# it has no dependencies. The domain logic is encapsulated so it can be moved and re-used with any other data layer or ui layer. The data layer is labeled under “Infrastructure”, this layer can include external 3rd party dependencies like smtp emailers, text message senders, packages, etc. The UI is a ASP.NET Core MVC application, that has WebAPI built in. Although the demo application makes use of MVC, there is an example API endpoint for getting products as a JSON result at: localhost:yourportnumber/api/products.

Keynotes

-The application uses the Repository/Unit Of Work pattern. It wraps EF DbContext, but still relatively easy to change contexts since the application is decoupled using repositories/unit of work interfaces via Dependency Injection / IoC.

-Validation is performed using FluentValidation.

-The application uses a Specification Pattern to encapsulate business logic/rules/specifications. The model has db model configuration and validation, however the specifications are used for the domain layer. Some of the business rules used as example are taken from constraints on the AdventureWorks database object.

-The UI design used a simple Bootstrap template for demo purposes.

-Added additional feature/functionality to add new reviews to a product for testing purposes. This does not include the ability to delete or edit a review.

-For simplicity most of the code for Product/ProductReview is in the same controller/service, for larger projects best practice would be to move each into separate controllers/services.

-There is a single unit test to test for getting average rating of a product. This demonstrates unit testing, etc.

-Enjoy the Demo! =)