Company ABC: Architecture Document

Prepared by Connor Moreside

Contents

Requirements

User Stories

Required User Stories

1. As a user, I want to be able to add a new product to the system.
2. As a user, I want to be able to view the details of a product.
3. As a user, I want to be able to delete a product from the system.
4. As a user, I want to be able to view a paginated list of products.
5. As a user, I want to be able to search for products by any of the fields.
6. As a user, I want to be able to export a list of products to a PDF.

Future User Stories

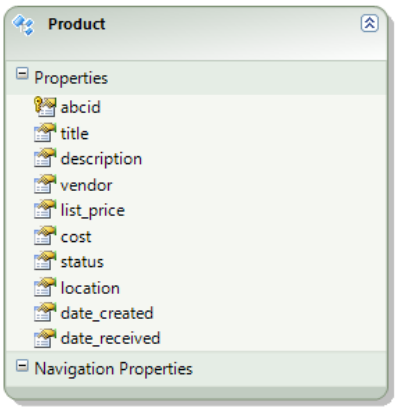
1. As a user, I want to be able to sort a list of products by its fields
2. As a user, I want to be able to customize the fields I see while viewing the list of products.
3. As a user, I want to see my previous searches.
4. As a user, I want to be able to customize the number of products I see on the page.

Functional Requirements

1. Search must be fast! < 5 seconds to return.
2. Must be able to search by any field or any combination of fields
3. Must be able to generate PDF of products
4. Must use MVC 4
5. Must use NIRIX color scheme
6. Must be "simple, concise, and easy to navigate."
7. Nicely formatted PDF Report.

Diagrams

SQL Diagram (Entity)



Architecture and Design

The solution is split into 4 projects:

- CompanyABC.Domain - Contains the entity / domain models, entity framework mapping, searching capabilities
- CompanyABC.Tests - Contains unit tests for the whole project.
- CompanyABC.WebUI - Contains the web front-end (MVC portion).
- CompanyABC.Utility - Contains any sort of utility classes; namely, the PDF generation capabilities.

3rd Party Libraries and Frameworks Used

- Entity Framework (ORM capabilities)
- jQuery, jQuery UI, and jQuery Validation
- Bootstrap (Fast UI design)
- html5shiv (Legacy IE compatibility)
- Ninject (Dependency Injection)
- PagedList and PagedList.Mvc (Paging of results)
- PDFsharp + MigraDoc (PDF generation)
- Moq (Mocking)

Note: NuGet was used to install the packages. Make sure the NuGet package manager extension is installed.

Design Choices

Search

The search functionality in CompanyABC.Domain.Search.ProductSearchService may be a little confusing to read at first. I chose to manually build the LINQ expression tree instead of using the fluent API. I thought it would be cleaner than the alternative, but it's debatable. If set a break point somewhere in there, you can see the LINQ tree being built. Might make it a little bit easier to understand.

The nice thing about building the expression tree manually is if you have any conditional logic which might change the number of statements in the query, you can avoid duplicating the query and then adding the additional statement.

Dependency Injection

I chose the Ninject IoC container for a number of reasons, but primarily for its compatibility with MVC4 and its ease of use. To view the mapping of dependency interfaces to implementations, see CompanyABC.WebUI.Infrastructure.NinjectDependancyResolver.cs.

Localization

As a general rule, I try to avoid putting string literals in the middle of code. I like to put them into resource files (resx) or a static class of constants. I tried to move most of the string literals into a satellite assembly, but it was giving me nothing but grieve, so I created a facade to avoid it. See CompanyABC.WebUI.LocalizedMessageService. This class should also probably be moved into the Utility project so all other subprojects can share it.

Bootstrap

I chose to use Bootstrap for most of the design. Made it easy to create a nice looking interface in a short period of time.

Running the solution

1. Make sure you change the connection string in the Web.config file to point to the database file. Right now, it is a hard coded absolute path, so update accordingly!

Known Issues

1. When exporting a list of products that have long descriptions, the pages get all out of place. Would need to play around with the PDF generation framework to learn how to fix the problem.
2. When generating the PDF from a list of products, there is NO indicator of the progress. Need to add a dialog to or something. Right now, this is not very user friendly.
3. After you perform a search, you click on either edit or details of a product and then try to navigate back, you lose your search place. Again, not very user friendly.
4. In some browsers, the UI datepicker widget doesn't render properly. I think there is a wrongly set setting in my BundleConfig.cs file

Future Improvements

1. Refactor the PDF generation code. Right now it is pretty awful :(Need to break up the method calls into smaller, well-named methods.
2. Refactor some of the embedded styles in Razor files into their own style sheets.
3. Complete the "localization" effort started. Tried to put any constant strings into resource files so localization could be performed.
4. Refactor ProductsController's dependencies into a new container so the construction parameter list isn't so long...
5. Write a more comprehensive suite of tests. Right now, it is pretty bare. I would write more if I had more time.