**Quinlan v1**

Written By Jeff Campbell

The Quinlan application is an effort to manage the inventory of a sports card collection that my client has inherited. It will also provide tools to bundle parts of the inventory together as sellable products, if my client decides to do so.

Overview

The application is written using .NET Core v3 and utilizes the following:

|  |  |
| --- | --- |
| Server Side | Client Side |
| - C#  - ASP.Net Core  - Entity Framework Core  - Dependency Injection Framework  - MVC  - Web API  - LINQ  - T-SQL | - HTML5  - CSS3  - Razor with Tag Helpers  - Partial Views  - Angular  - TypeScript  - jQuery  - JavaScript |

Projects

The solution contains the following projects (layers) for the initial release. :

|  |  |
| --- | --- |
| Quinlan.Data | Data models  Entity Framework  Code value, Data access, and Query services |
| Quinlan.Domain | Domain models  CRUD, Domain Search, and Helper services |
| Quinlan.Initialize | Seed services  Import Files  Import SQL |
| Quinlan.MVC | MVC website  View Models  Page, Form, and Helper services |
| Quinlan.SPA | SPA website  Web API  API services |

Unit Testing

All functionality has been developed in service classes which can be individually tested. The corresponding unit tests utilize an InMemory instance of the database which are initialized using Seed services in the INITIALIZE project. Tests are performed on a separate instance of the InMemory database context.

Note that InMemory databases are persisted for the life of the unit test run. So each project has a separate unit test project to limit the number of InMemory databases persisted in memory.

Quinlan.Data

The DATA project contains the Entity Framework context, the data model, and the data access services used to query and update the database. The data model has been written to achieve a relational database.

- Contexts

QdbContext => E/F context that defines the following DbSets and generates the

corresponding database tables. Code table values are seeded

when the database is created.

|  |  |  |
| --- | --- | --- |
| Data Tables | Code Tables | Import Tables |
| Collectibles  Colleges  Grades  People  PersonSports  Products  Sets  Teams | CardTypes  CollectibleStatuses  CollectibleTypes  League  ProductStatuses  Sports | ImportCollectibles  ImportColleges  ImportPeople  ImportTeams |

- Exceptions => Exception classes that are specific to the data access process

InvalidCodeValueChange

InvalidIdException

- Interfaces => Interfaces for data access services.

|  |  |
| --- | --- |
| ICodeService | Service to access code values |
| IDataService | Service to manage a data source |
| IDatabaseService |  |
| IQueryService | Service to query a data source |
| ICollectibleQueryService | Custom query service for Collectibles |

- Data Models => POCO classes used to define context and database

CardType

Collectible

CollectibleStatus

CollectibleType

College

Grade

ImportCollectible

ImportCollege

ImportPerson

ImportTeam

League

Person

PersonSport

Product

ProjectStatus

Set

Sport

Team

- Filter Models => filter options for query services

CollectibleFilterOptions

PersonFilterOptions

TeamFilterOptions

- Code Services => Static classes that handle code values.

CardTypeCodeService

CollectibleStatusCodeService

CollectibleTypeCodeService

LeagueCodeService

ProductStatusCodeService

SportCodeService

- Data Services => Classes that inherit from IDataService

CollectibleDataService

CollegeDataService

PersonDataService

ProductDataService

TeamDataService

- Query Services => Classes that inherit from IQueryService

CollectibleQueryService

PersonQueryService

TeamQueryService

Quinlan.Initialize

The INITIALIZE project contains functionality to seed the database for development and testing purposes. Note that the initial production database will be a copy of the seeded development database which contains all of the Q’s recorded inventory.

- InitFiles => CSV files containing initialization data.

Cards.csv

Colleges.csv

People.csv

Teams.csv

- Services

SeedService

- SQL

Add2DB.sql => Import script that pulls data from InitFiles into an empty database

Quinlan.Domain

The DOMAIN project contains the domain model as well as services that are used

to hydrate those classes.

- Exceptions => Exception classes that are specific to the domain

DeleteNotSupportedException

GetNotSupportedException

ItemNotFoundException

PostNotSupportedException

PutNotSupportedException

- Interfaces

ICollectibleSearchService

ICrudService

ISearchService

ISummaryService

- Domain Models

Card

CardType

College

DataSummary

Figurine

ImportCard

League

Magazine

Person

Sport

SubTotal

Team

- Filter Options Models

CardSearchFilterOptions

FigurineSearchFilterOptions

MagazineSearchFilterOptions

PersonSearchFilterOptions

TeamSearchFilterOptions

- Search Models

CardSearch

FigurineSearch

MagazineSearch

PersonSearch

TeamSearch

- CRUD Services

CardService

FigurineService

LeagueService

MagazineService

PersonService

SportService

TeamService

- Helper Services

QueryFilterService

- Database Services

CollectibleSummaryService

- Search Services

CardSearchService

FigurineSearchService

MagazineSearchService

PersonSearchService

TeamSearchService

Quinlan.MVC

The MVC project contains an ASP.Net Core website using the built-in MVC functionality. Note the following:

1. Controllers have very little functionality. They call a Page service to generate a View Model and pass that to the corresponding View. Try/Catch surrounds all logic with exceptions written to the logger.
2. Models have no functionality. All functionality to hydrate models is handled by MVC services.
3. Views utilize Razor and Tag Helpers to render pages. Partial views utilized for shared search criteria section. More are likely to be written.
4. No JavaScript has been written. All functionality is server-side

- Controllers

CardsController

FigurinesController

HomeController

LeaguesController

MagazinesController

PeopleController

SportsController

TeamsController

- Exceptions

tbd

- Interfaces => Interfaces are defined for each type of page

|  |  |
| --- | --- |
| IDetailsService | Details page shows cards related to a specific data point such as sport or person |
| IEditService | Edit page allows a data point such as a person or team to be modified |
| IFormService |  |
| IHomePageService |  |
| IIndexService | Index page allows data points such as people or teams to be listed and filtered |
| IViewService | View page displays information for a single data point such as a card |

- Binding VMs => Used to receive data from a form in a submitted from an HTML Form

CardFilterOptionsViewModel

CardViewModel

CollegeViewModel

FigurineFilterOptionsViewModel

FigurineViewModel

LeagueViewModel

MagazineFilterOptionsViewModel

MagazineViewModel

PersonFilterOptionsViewModel

PersonViewModel

SportViewModel

TeamFilteroptionsViewModel

TeamViewModel

- ListItem VMs => Models for data to be listed on a page.

CardListItemViewModel

FigurineListItemViewModel

MagazineListItemViewModel

PersonListItemViewModel

TeamListItemViewModel

- Page VMs => View models for page level data bindings.

CardEdit

CardView

CollegeDetails

CollegeEdit

FigurineIndex

Home

LeagueDetails

MagazineIndex

PersonDetails

PersonEdit

PersonIndex

SportDetails

Summary

TeamDetails

TeamIndex

- Shared Models => View models that are shared across pages

CardSearchViewModel

SearchTotalsViewModel

SubTotalViewmodel

- Form Services

CardFormService

CollegeFormService

PersonFormService

TeamFormService

- Helper Services

FormatService

MvcService

SearchFilterService

- Page Services => generate the data bindings required to render a page.

CardEditService

CardViewService

CollegeDetailsService

CollegeEditService

FigurineIndexService

HomePageService

LeagueDetailsService

MagazineIndexService

PersonDetailsService

PersonEditService

PersonIndexService

SportDetailsService

TeamDetailsService

TeamEditService

TeamIndexService

- Views => Views which contain the markup and razor code

Cards

* Edit
* View

Colleges

* Details
* Edit

Figurines

* Index

Home

* Details
* Index

Leagues

* Details

Magazine

* Index

People

* Details
* Edit
* Index

Sports

* Details

Teams

* Details
* Edit
* Index

Shared Views

* \_FilterOptions
* \_Layout
* \_SearchTotals

Quinlan.SPA

Single Page Application (SPA) written using ASP.Net Core for the back-end and Angular/TypeScript for the front-end.

- ClientApp

tbd

- Controllers

SportsController

- Interfaces

IApiService

- Models

CardDTO

SportDTO

- Services

SportsApiService

Quinlan.Data.Test

- Interfaces

ICodeServiceTests

IDataServiceTests

IQueryServiceTests

- Code Service Tests

CardTypeCodeServiceTests

CollectibleStatusCodeServiceTests

CollectibleTypeCodeServiceTests

LeagueCodeServiceTests

ProductStatusCodeServiceTests

SportCodeServiceTests

- Data Service Tests

CollectibleDataServiceTests

CollegeDataServiceTests

PersonDataServiceTests

ProductDataServiceTests

TeamDataServiceTests

- Query Service Tests

CollectibleQueryServiceTests

PersonQueryServiceTests

TeamQueryServiceTests

Quinlan.Domain.Test

- Interfaces

ICrudServiceTests

ISearchServiceTests

- CRUD Service Tests

CardServiceTests

FigurineServiceTests

LeagueServiceTests

MagazineServiceTests

PersonServiceTests

SportServiceTests

TeamServiceTests

- Database Service Tests

CollectibleSummaryServiceTests

- Helper Service Tests

QueryFilterServiceTests

- Search Service Tests

CardSearchServiceTests

FigurineSearchServiceTests

MagazineSearchServiceTests

PersonSearchServiceTests

TeamSearchServiceTests

Quinlan.MVC.Test

- Interfaces

IDetailsServiceTests

IEditServiceTests

IIndexServiceTests

- Page Service Tests

CardEditServiceTests

CardViewServiceTests

CollegeDetailsServiceTests

CollegeEditServiceTests

FigurineIndexServiceTests

HomePageServiceTests

LeagueDetailsServiceTests

MagazineIndexServiceTests

PersonDetailsServiceTests

PersonEditServiceTests

PersonIndexServiceTests

SportDetailsServiceTests

TeamDetailsServiceTests

TeamEditServiceTests

TeamIndexServiceTest

- Helper Service Tests

FormatServiceTests

MvcServiceTests

SearchFilterServiceTests

- Form Service Tests

CardFormServiceTests

CollegeFormServiceTests

PersonFormServiceTests

TeamFormServiceTests

Quinlan.SPA.Test

- Interfaces

tbd

- Tests

tbd

Quinlan.Initialize.Test

- Tests

SeedServiceTests

- SQL

VerifyDB.sql