|  |  |
| --- | --- |
| Bank | Bank details |
| BankAccount | Bank Account details |
| Transaction / ScheduledTransaction / Movement  Transfer | Defines a movement of money from one account to another. May be one-off or repeating (e.g. monthly).  Name  FromBankAccount, ToBankAccount,  Amount, AmountTolerence  StartDate, EndDate, Frequency |
| Cashflow | Defines the starting point for a cashflow projection.  Name, List of bankAccount, Date, Opening Balance |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Feature \ Module** | **Bank** | **BankAccount** | **Transfer** | **BankTree** | **Cashflow** |
| **Delete multipe ids** | Y | Y | Y | X | y |
| **VM Item to wrap Entity** | Y | Y | Y | X | Y |
| **Agent class to handle Add/Edit** | Y | Y | Y | X | Y |
| **VM Edit to wrap Entity** | Y | y | Y | X | Y |
| **Remove Sets from Item VM** | Y | Y | Y | X | Y |
| **Remove VM mappings** | Y | Y | Y | x | Y |
| **Maybe move entity props to base Item and Editor classes?** |  |  |  |  |  |
|  |  |  |  |  |  |
| **Make account name property same?** |  |  |  |  |  |
| **Tasks for background** |  |  |  |  |  |

**TO DO**

|  |  |
| --- | --- |
| Try Attlisan SourceTree GIT UI tool | / |
| CashflowENgineC - refactor t.Frequency == "Monthly" code. | / |
| Add AggregatedProjectionItemsGeneratorFactory  Move factory into Engine – **TEST THIS** | / |
| Remove need for the enum?  Have started converting ProjectMode combo to DataIdName. Why default to empty - done  Need to pass Code into Generate method - done  **What about passing in the actual generator object instead of it’s code? This would need to be select by the combobox. Maybe have a generic ViewModel wrapper? Or just use actual object for combo?** | / |
| Schedule info:  Pull out schedule props into Schedule  Schedule.GetDesc  AutoMapper – mappings and ConstructUsing  Inject with calculators  Factory for Schedule | / |
| Use Windsor in IntergrationTests | / |
| All ViewModels (or even all classes?) should be instantiated via a factory so that they have ability to be injected? YES + Entities  Does Windsor factory interface allow an object to be constructed with a supplied parameter (e.g. an entity) and injected parameters? YES |  |
| Separate MappiingCreator classes for each repository.  If implement IMappingCreator – then should just work? | / |
|  |  |
| Replace Enums with types? |  |
| How to have bespoke repository methods  IRepositoryBespokeCashflow instead? Depends on requirements  Remove ICashflowRepository? done  CashflowRepository to implement segregated interfaces instead? done |  |
| Exception handling – ErrorHandlingInterceptor uses IReporitory which is applied to CashflowRepository which means in intercepts errors which should be handled by DbEntityValidation decorator!  Look at AOP? |  |
| Library-specific unit test projects? |  |
| Get source code from SOLID book from Github |  |
| How to avoid having to register decorators for each Entity?  Can we mix decoorators and open generics in Windsor? |  |
| Refactor mode code using SOLID ?  ReportRender abstract class? |  |
| Schedule Description in Edit dialog with show/hide button/expander? |  |
| Try:  Generic segrageted repository interfaces, with decorators.  e.g.  IRead<TEntity>  IDelete<TEntity>  ISave<TEntity>  or IrepositoryRead<T> and IrepositoryWrite<T>?  Would need an Entity base type - **done**.  Could create generic decorators e.g.:  SaveAuditing  DeleteLogging  Transactions  A combined interface at the leaf level – which interacts directrly with the ORM  Started with CashflowRepository  Have created IRepositoryRead<TEntity>, RepositoryReadDecoratorTest<TEntity> class, and injected into container. It works! Have also changed dependency in CashflowViewModel to use IreporsitoryRead<Cashflow>.  Next is to create IRepositoryWrite and do the same.  Will ICashflowRepository then just be used for non-CRUD operations?  CashflowRepostority still just inplements IcashfowRepository !!  In which project should the decorators live?  Do we need to inject for each entity type? Yes until I research if it’s possible to mix generics and decorators in Windsor. |  |
| CashflowEngine unit tests:  ProjectionTransferGenerator – in progress  TransferDirectionGenerator  TransferFrequencyDateCalculatorMonthly  AggregatedProjectionItemsGeneratorMonthlySummary  AggregatedProjectionItemsGeneratorMonthlyDetail |  |
| Refactor CommonRepository.HandleDbEntityValidationException |  |
| Summary->Righ-click Show Detail option? |  |
| Separate period rows? E.g. with underline? |  |
| Try more composing/embedding of WPF views/models |  |
|  |  |
|  |  |
| Pressing Enter on OK in Add/Edit will re-open the dialog  Double-click to adjust column width also open Edit dialog |  |
|  |  |
|  |  |
| @variable ? |  |
|  |  |
|  |  |
|  |  |
|  |  |

Current - ??CashflowProjection GenerateProjection(Cashflow cashflow, DateTime endDate, decimal threshold, mode)

**CashflowProjection/or list of items? GenerateProjection(accounts, startDate, endDate, openingBalance, threshold, mode)**

List<CashflowProjectionTransfer> xfers = GenerateProjectionTransfers(accounts, start, end)

List<CashflowProjectionItem> cpis = GenerateAggregatedProjectionItems(xfers,mode)

List<CashflowProjectionItem> ApplyOpeningBalanceAndThreshold(cpis, openbal, threshold)

--

List<CashflowProjectionTransfer> GetCashflowProjectionTransfers(accounts, start, end)

List<CashflowProjectionItem> GetCashflowProjectionItems(cpts, openingBalance, threshold, mode)

CashflowProjectionItem

PeriodName

PeriodStartDate

PeriodEndDate

Item

In/Out/Balance

BalanceState

Modes=Detail, Monthly Summary

Monthly Summary aggregates by Category (or item if Category==None).

**Cashflow Engine refactor**

Engine A

Enum parameter to main Generate method.

Selection inside engine.

Engine B

Factory with CreateDetail and CreateMonthlySummary methods

VeiwModel shows list of Enums

Mode selection is made in ViewModel

Engine C

Register list of IAggregatedProjectGenerator in container

ViewModel to use this list for the combo

Pass the selection as a parameter to the main engine Generate method

Selection made by user

**Notes**

When to create an interface for a class?

Anything added to the DI container. E.g. Services, Factories

New entity process

|  |  |
| --- | --- |
| Create table | Y |
| Generate EF entity | Y |
| Core entity | Y |
| AutoMapper Mappings for EF->Core | Y |
| Unit test for mappings | Y |
| Repository | Y |
| Integration Test for repo | Y |
| Item ViewModels |  |
| Mappings for Core -> Item VM |  |
| Unit tests for Item VM mappings |  |
| List ViewModel (without editor) |  |
| Add to DI |  |
| List View |  |
| Editor ViewModel |  |
| Mappings for Core -> Editor VM |  |
| Unit tests for Editor VM mappings |  |
| VM Editor factory interface |  |
| Add editor to List VM |  |
| Editor View |  |
|  |  |
|  |  |

Notes

**Cross-cutting concerns – advice from Rational Geek**

<http://rationalgeek.com/blog/architectural-guidance-cross-cutting-concerns/>

|  |  |
| --- | --- |
| Logging and instrumentation | Log4net |
| Security | .NET Framework  <https://msdn.microsoft.com/en-us/library/z164t8hs(v=VS.100).aspx> |
| Error Handling | MS Exception Handling Application Block  <https://msdn.microsoft.com/en-us/library/ff664698(v=PandP.50).aspx> |
| Validation | Use whatever validation technology exists in the UI framework you are using |
| Caching | System.Runtime.Caching  <http://msdn.microsoft.com/en-us/library/system.runtime.caching(VS.100).aspx> |
| Transaction Management | System.Transactions namespace |
| AOP | <https://www.postsharp.net/features> |
|  |  |

**Model.Context.tt**

public <#=code.Escape(container)#>(string nameOrConnectionString="name=<#=container.Name#>")

: base(nameOrConnectionString)