Penggunaan Library

# Prerequisite

Untuk menggunakan library, kita memerlukan beberapa hal

1. Runtime Library
   1. BackEnd Library collection
      1. digunakan sebagai penunjang library untuk semua Business Process dan Controller serta API yg ada di back end
   2. Common Library collection
      1. digunakan sebagai penunjang library untuk semua shared class yg dipakai di Back End maupun Front End
   3. FrontEnd Library collection
      1. Digunakan sebagai penunjang library untuk semua class yg ada di Front End
2. Realta Nuget
   1. Realta.BackCollectionNuGet
      1. Merupakan collection dari Nuget package yg kita pakai di BackEnd Library
   2. Realta.CommonCollectionNuGet
      1. Merupakan collection dari Nuget package yg kita pakai di Common Library
   3. Realta.FrontCollectionNuGet
      1. Merupakan collection dari Nuget package yg kita pakai di FrontEnd Library

# Schema penggunaan Library

Common Library

BackEnd Library

Front Library

use

use

use

use

use

use

use

use

use

use

use

use

use

use

use

use

use

use

call

Blazor Menu Project

Main API Project

Service Project per program

Service Project per program

Back Project per program

Back Project per program

Blazor Front Project per program

Blazor Front Project per program

Synch

Common Project

Common Project per program

Common Project

Common Project per program

Back

Front

# Pembuatan Standard Project

Seperti schema penggunaan library, aplikasi kita dibagi menjadi

1. Penamaan Program
   1. Program memakai nama MMTXXXyy
      1. MM merupakan kode Modul, misalkan SA (system administrator modul)
      2. T merupakan Type program
         1. M=Maintenance
         2. R=Report
         3. T=Transaction
         4. I=Inquiry
         5. B=Batch
      3. XXX merupakan nomor program, dimulai dengan yy = 00
         1. 00100
      4. YY merupakan anak dari program induk
         1. 00110
   2. Contoh penamaan program
      1. Header
         1. SAM00100
      2. Detail
         1. SAM00110
2. Back
   1. API Project
      1. Merupakan project Back yg dipakai sebagai penampung dari koleksi dari sebuah kumpulan Program di BackEnd, biasa dibagi per Modul atau application(kalau aplikasi kecil)
         1. Contoh penamaan berdasarkan Modul nya API-AppId-MM
            1. AppId merupakan Application ID, misalkan Rhapsody, Bimasakti
            2. MM merupakan Modul
         2. Contoh penamaan berdasarkan application saja API-ESS
            1. ESS merupakan application Id dari Employee Self Service
   2. Service Project
      1. Merupakan project yang menampung semua controller header maupun detail yang mewakili suatu program
      2. Penamaan project berdasarkan nama program induk ditambahkan Service
         1. MMTXXXyyService
         2. Contoh nya SAM0100Service
      3. Didalam project Service tsb terdiri dari 1 atau lebih controller berdasarkan dari pembagian Header dan Detail dari program tsb.
         1. Controller2 ini nantinya akan di-ekspose ke API sesuai dengan nama Controller beserta method2 nya
         2. Penamaan Class Controller berdasarkan nama program header maupun detail ditambahkan dengan Controller
         3. Contoh penamaan class Controller yg ada di suatu project service
            1. SAM00100Controller
            2. SAM00110Controller
   3. Back Project
      1. Merupakan project yg menampung semua business process header maupun detail yang mewakili sebuah program
      2. Penamaan project berdasarkan nama program induk ditambahkan Back
         1. MMTXXXyyBack
         2. Contohnya SAM00100Back
      3. Didalam project back tsb terdiri dari 1 atau lebih class business process berdasarkan dari pembagian Header dan Detail dari program tsb
         1. Penamaan Class Business Process berdasarkan nama program header maupun detail ditambahkan dengan Cls
         2. Contoh penamaan class Business proceed yg ada di project Back
            1. SAM00100Cls
            2. SAM00110Cls
3. Common
   1. Common Project
      1. Merupakan shared project yang dipakai oleh BackEnd dan FrontEnd
      2. Penamaan project berdasarkan nama program induk ditambahkan Common
         1. MTXXXyyCommon
         2. Contohnya SAM00100Common
4. Front
   1. Menu Project
      1. Merupakan project utama front yang berisi konfigurasi menu dan penanganan peletakan program yg dipanggil dari menu
   2. Front Project
      1. Merupakan project yg menampung semua page front dari header maupun detail yang mewakili sebuah program
      2. Penamaan project berdasarkan nama program induk ditambahkan Front
         1. MMTXXXyyFront
         2. Contohnya SAM0100Front

# Exception

Exception dipakai sebagai field exception standard message kita untuk dapat kita tampilkan dari BackEnd sampai ke API

Dalam pembuatan project ini mempunyai simulasi menampilkan list customer yg akan dihasilkan oleh simulasi back, dimana nantinya fungsi untuk menghasilkan list customer ini akan mempunyai parameter jumlah customer yg akan dihasilkan dan pada jumlah tertentu akan melempar exception supaya kita dapat melihat simulasi untuk menghasilkan data customer dan exception dapat diatur dari parameter fungsi tsb.

Struktur dari result API kita secara standard adalah

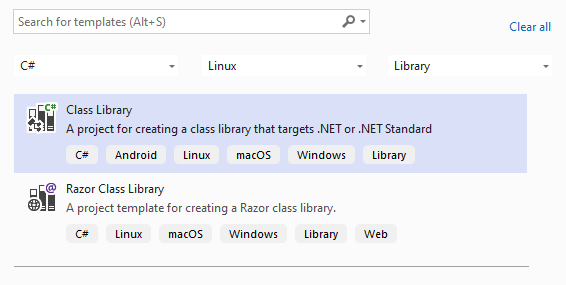
1. Data
   1. Merupakan field untuk menampung data
   2. Akan berisi null
      1. kalau ada exception
2. Errors
   1. Merupakan field yg berisi koleksi dari Exception yg terbentuk
   2. Akan berisi null kalau tidak ada exception

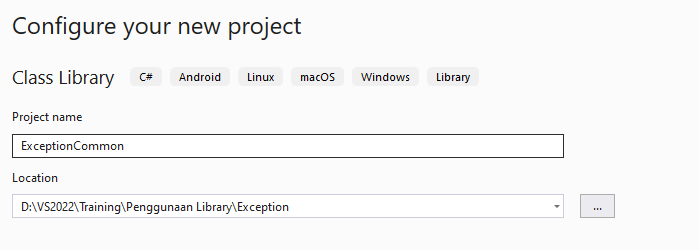
## Pembuatan Project Exception di Common

### ExceptionCommon

Merupakan project Common yg akan dipakai oleh Back dan Front.

Project tsb berupa Data yg akan dihasilkan oleh Back juga interface yg akan diimplementasikan di API dan dipakai atau dipanggil oleh Front





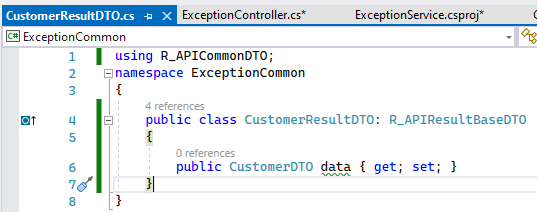


1. Penamaan dan Type Project
   1. Penamaan ExceptionCommon
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Standard 2.1
2. Reference Assembly R\_CommonFrontBackAPI
   1. Digunakan untuk Result DTO
   2. Digunakan untuk Interface

#### Entity

Untuk Entity yg dihasilkan dapat dibagi 2

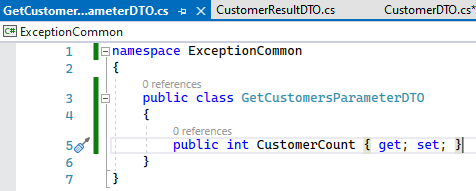
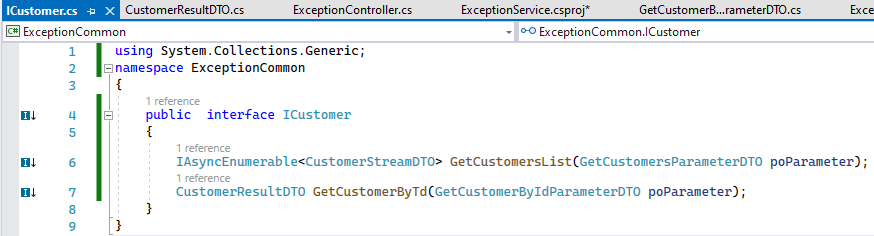
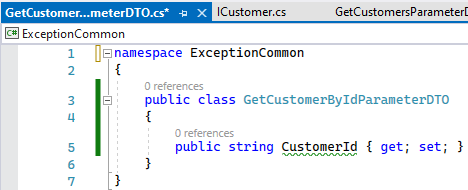
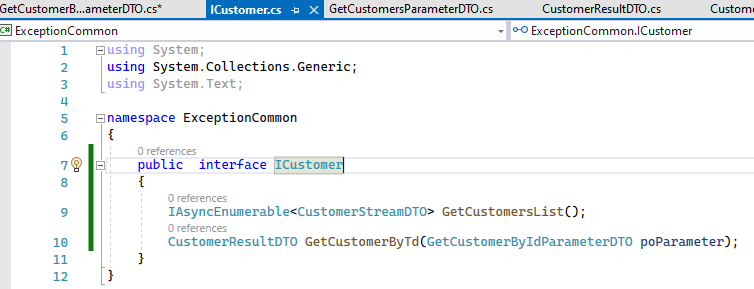
1. Entity untuk streaming
   1. CustomerStreamDTO
   2. Biasanya merupakan DTO yg bersifat singkat untuk keperluan dari data yg sifatnya collection
   3. Graphical user interface, text, application, email

      Description automatically generated
2. Entity untuk satuan
   1. CustomerResultDTO
      1. Inherits dari R\_APIResultBaseDTO
      2. R\_APIResultBaseDTO merupakan bagian dari R\_APICommonDTO
      3. Buat property untuk data
         1. Mempunyai type CustomerDTO
   2. 
   3. CustomerDTO
   4. Graphical user interface, text, application

      Description automatically generated

#### Interface

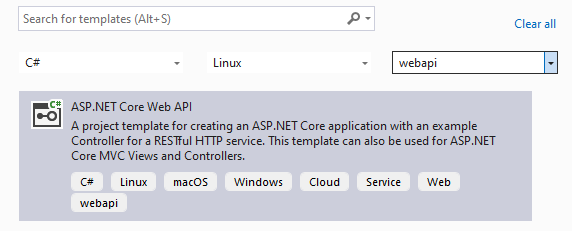
Untuk interface kita akan membuat 2 fungsi

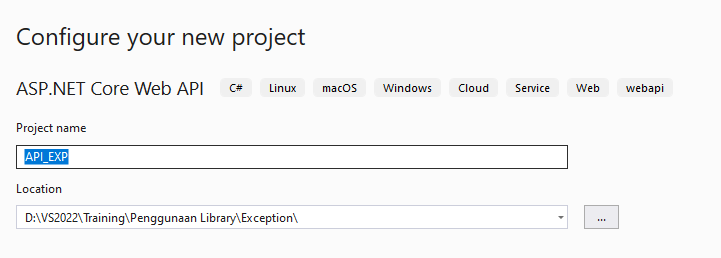
1. Fungsi untuk mendapatkan sejumlah data customer yg tergantung dari parameter yg dikirimkan
   1. Fungsi GetCustomers
      1. Merupakan fungsi streaming untuk mendapatkan sejumlah collection Customer Data
   2. Mempunyai parameter GetCustomersParameterDTO
   3. 
   4. Mempunyai interface yg menghasilkan IAsyncEnumerable<[dto]>
   5. 
2. Fungsi untuk mendapatkan data customer sesuai dengan Id Customer yg dikirimkan dari parameter
   1. Fungsi GetCustomerById
      1. Merupakan fungsi untuk mendapatkan sebuah data customer
   2. Mempunyai parameter GetCustomerByIdParameterDTO
   3. 
   4. Mempunyai interface yg menghasilkan CustomerResultDTO
   5. 

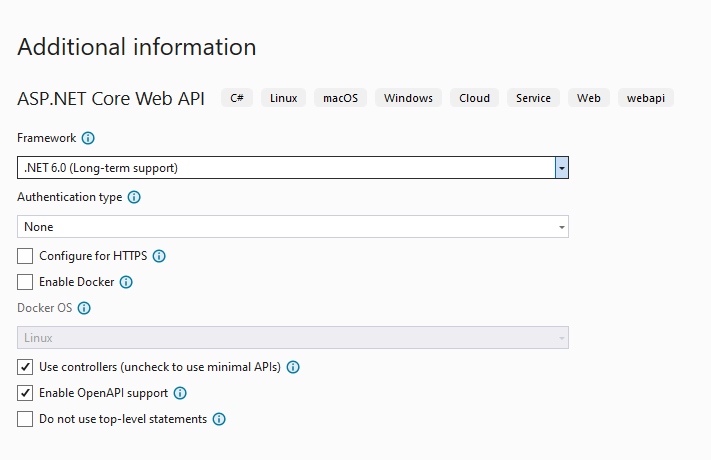
## Pembuatan project Exception di Back sesuai dengan standard project

### API\_EXP

Merupakan project API Exception



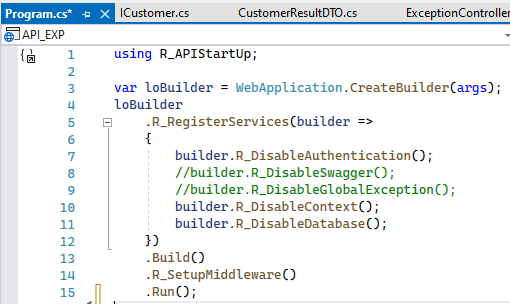




1. Penamaan dan Type Project
   1. Penamaan API\_EXP
   2. Language C#
   3. Platform Open (Linux)
   4. Type project ASP.NET Core Web API
   5. Framework NET 6.0
   6. HTTP
2. Delete default Controller dan Model WeatherForecast
3. Reference Project
   1. Reference Service Project
      1. ExceptionService
4. Reference semua collection assembly baik Back maupun Common
   1. Semua assembly yg ada di back directory sudah termasuk common directory
   2. Reference semua assembly di directory “RuntimeLibrary/Back” semua assembly yg mempunyai awalan R\_ saja
5. Menggunakan Private realta Nuget
   1. Berikan nama Realta Nuget
   2. Address https://www1.realta.net/realtanuget/nuget
   3. Pakai Realta.BackCollectionNuget
6. Graphical user interface, text, application, email

   Description automatically generatedGraphical user interface, application

   Description automatically generatedText

   Description automatically generated
7. Perubahan pada Class Program.cs
   1. using R\_APIStartUp
   2. Untuk penggunaan StartUp ada 5 hal yg dapat di-disable
      1. Authentication
      2. Swagger
      3. GlobalException
      4. Context
      5. Database
   3. Karena kita akan menggunakan Exception dan Swagger, maka kita akan melakukan disable untuk point yg lain
   4. 

### ExceptionBack

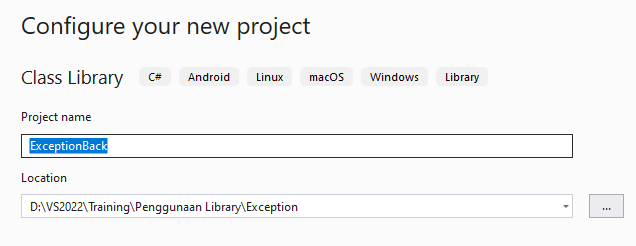
Merupakan Project yg berupa kumpulan bisnis process untuk menghasilkan data yg diminta oleh Controller API, dimana sudah dituangkan fungsi2 nya pada Interface yg ada di common.

Project ini akan berhubungan dengan database, karena tujuan dari contoh peoject ini adalah untuk keperluan Exception, maka Project back ini tidak akan berhubungan dengan database, tapi hanya menghasilkan data dengan cara mensimulasikan-nya.

Mempunyai Class dengan nama ExceptionCls dan mempunyai fungsi sesuai dengan interface yang ada di common

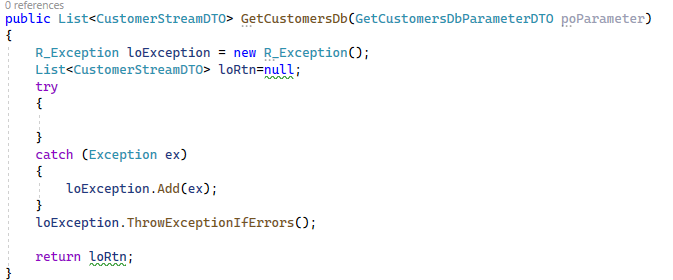
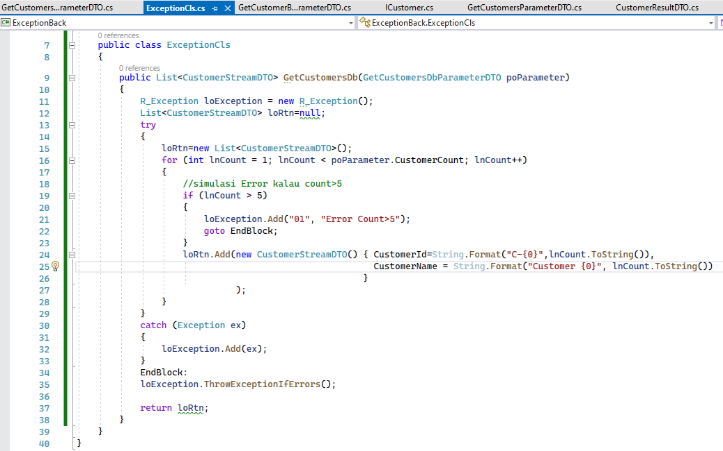
Graphical user interface, text, application

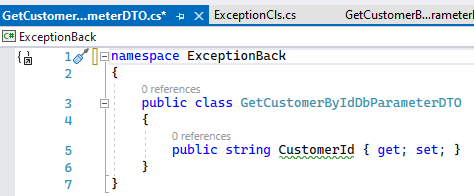
Description automatically generated



Graphical user interface, text, application, email

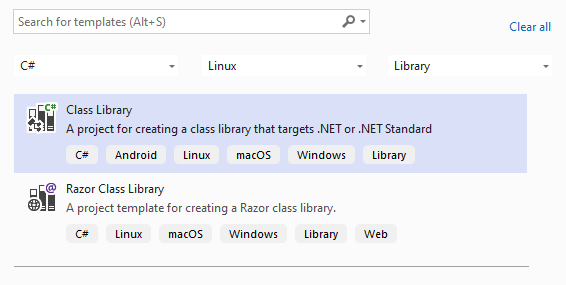
Description automatically generated

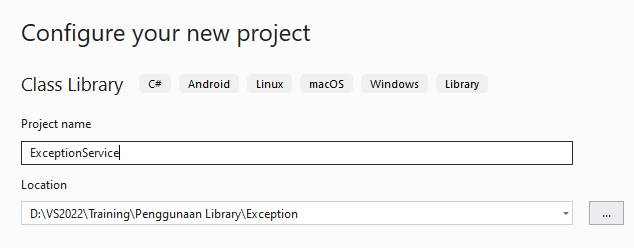
1. Penamaan dan Type Project
   1. Penamaan ExceptionBack
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APICommon
      1. Dipakai untuk utility umum
      2. Dipakai untuk Exception framework
         1. Merupakan standard Exception yg dipakai untuk mendapatkan koleksi exception yg terjadi dan melakukan throw exception kalau memang terjadi exception
      3. 
   2. Assembly R\_APIBackEnd
      1. Dipakai untuk berhubungan dengan database
   3. Project ExceptionCommon
      1. Untuk reference entity yg ada di common
         1. CustomerStreamDTO
         2. CustomerDTO
3. Penamaan Class
   1. Sesuai dengan project, kata Back diubah menjadi Cls
   2. ExceptionCls
   3. Mempunyai fungsi untuk menghasilkan data sesuai dengan interface yg akan diimplementasikan oleh controller di project Service
   4. GetCustomersDb
      1. 
      2. Menghasilkan collection dari CustomerStreamDTO
      3. Mempunyai parameter GetCustomersDbParameterDTO
         1. Biasanya parameter dibedakan dengan parameter yg ada di common project karena parameter y gada di Cls tidak boleh mengandung data2 yg didapat dari BackEnd Global Variable misalkan CompanyId
         2. GetCustomersDbParameterDTO
         3. 
   5. GetCustomerByIdDb
      1. Graphical user interface, text, application, email

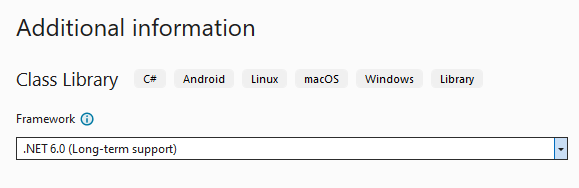
         Description automatically generated
      2. Menghasilkan CustomerDTO
      3. Mempunyai parameter yang dibedakan dart parameter y gada di common project
      4. GetCustomerByIdDbParameterDTO
      5. 

### ExceptionService

Merupakan project service yg berisi sekumpulan controller yg akan mendukung API sesuai dengan interface nya







1. Penamaan dan Type Project
   1. Penamaan ExceptionService
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Edit project file dan tambahkan
   1. <ItemGroup>
   2. <FrameworkReference Include="Microsoft.AspNetCore.App" />
   3. </ItemGroup>
3. Reference Assembly
   1. R\_CommonFrontBackAPI
      1. Untuk Interface yg digunakan oleh Ptoject ExceptionCommon
   2. R\_APICommon
      1. Untuk ExceptionFramework
4. Reference Project
   1. ExceptionCommon
   2. ExceptionBack
5. Buat Class Controller
   1. Class Controller tergantung dari banyak nya class Back
      1. Ubah internal menjadi public
   2. Buat dengan penamaan Business Process ditambahkan dengan Controller
      1. ExceptionController
   3. Using
      1. R\_Common
      2. R\_CommonFrontBackAPI
      3. Microsoft.AspNetCore.Mvc
   4. Tambahkan annotation di class
      1. [Route("api/[controller]/[action]")]
      2. [ApiController]
   5. Inherit dan interface
      1. Inherit ControllerBase
      2. Interface sesuai dengan business object interface nya, contohnya
         1. Untuk ExceptionController
            1. ICustomer
   6. Tambahkan annotation di method [HttpPost]
   7. Untuk method yg meng-implementasikan IAsyncEnumerable
      1. Tambahkan async di depan nya
      2. Tidak boleh memakai try block karena akan memakai yield di foreach

# Context

Context merupakan channel yg dapat kita pakai untuk meneruskan informasi atau data dari Client ke API project yg nanti nya dapat kita akses di Controller, dimana data tsb dapat merupakan informasi tambahan atau berupa parameter tambahan yg tidak melewati sebuah fungsi tapi akan di-passing melewati Header saat melakukan post ke API

## Beberapa model Context

1. Internal Context
   1. Internal context dipakai sebagai channel data yg mepunyai sifat global di client dan data tsb secara otomatis akan diatur oleh front framework saat login dan nantinya akan diteruskan ke API dan dapat digunakan di controller.
2. Public Context
   1. Public Context atau kita biasa sebut dengan context dipakai sebagai channel data yg mempunyai sifat data per program atau kelompok program, dimana ada kebutuhan menyalurkan data dari client ke API secara public per program atau kelompok program dan nantinya akan diteruskan ke API dan dapat digunakan di controller.
   2. Penggunaan context ini tidak diatur oleh framework tetapi diatur oleh masing2 developer yg menangani program atau kelompok program dan untuk mempermudah penamaan, biasanya akan dibuatkan constant di project common per masing2 program atau kelompok program sehingga dapat dikenali baik di front saat melakukan assign context maupun di back saat melakukan access context
3. Streaming Context
   1. Streaming Context dipakai sebagai channel data yg mempunyai memperlengkapi streaming API, dimana terkadang kalau kita memakai component di bagian client, component tsb sudah mempunyai pasangan nya di bagian API saat melakukan Post, sehingga kita tidak dapat melempar data saat post, untuk hal tsb channel tsb dipakai.
   2. Nantinya channel tsb dipakai pada filter component grid, dimana terkadang ada grid yg melakukan paging atau sorting di bagian server dan akan melakukan action post saat melakukan paging atau sorting dan component tsb sudah mempunyai pasangan parameter body nya di bagian API saat melakukan post
   3. Sama seperti public context, penggunaan context tsb diatur oleh masing2 developer dan bukan per program tapi lebih spesifik lagi per grid dan untuk mempermudah penamaan, biasanya akan dibuatkan constant di project common project per masing2 program sehingga dapat dikenali baik di front saat melakukan assign context maupun di back saat melakukan access context

## Gambaran Program

Department Id

Sales Data (dalam Grid)

|  |  |
| --- | --- |
| Sales ID | Sales Name |
|  |  |
|  |  |
|  |  |

Order Data per Sales (dalam Grid)

|  |  |
| --- | --- |
| Order ID | Order Date |
|  |  |
|  |  |
|  |  |

Gambar diatas adalah gambaran program yang menampilkan 2 buah kumpulan data yaitu

1. Kumpulan Salesman data dalam sebuah grid
   1. Kumpulan Salesman data tsb merupakan data salesman dalam sebuah company dalam sebuah departemen tertentu
   2. Mempunyai filter
      1. Company ID
      2. Department ID
2. Kumpulan Order data dalam sebuah grid
   1. Kumpulan Order berdasarkan Salesman yg dipilih pada grid salesman dalam sebuah company dalam sebuah departemen tertentu
   2. Mempunyai filter
      1. Company ID
      2. Department ID
      3. Salesman ID

### Context yang dibutuhkan berdasarkan Program

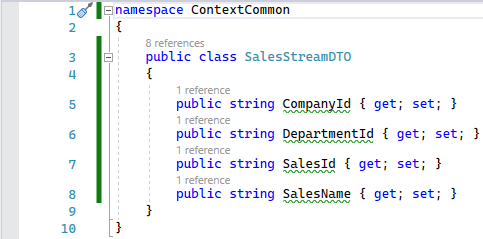
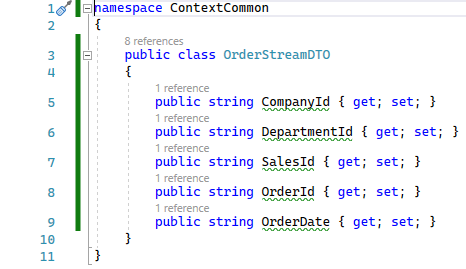
Karena semua data dalam bentuk Grid, dimana kita tidak dapat menambahkan parameter sebagai filter untuk mendapatkan data tsb, maka kita memakai context sebagai channel untuk menyalurkan filter data yg kita butuhkan sebagai pengganti parameter.

1. Internal Context
   1. Company Id
   2. Merupakan internal context yg disediakan oleh front framework pada waktu login (meminta input company id)
2. Public Context
   1. Department Id
   2. Merupakan public context karena merupakan filter pada 1 program page tsb, dimana baik Sales data maupun Order data harus difilter berdasarkan input department
3. Streaming Context
   1. Untuk Sales Data
      1. Sales Count
         1. Dipakai untuk melakukan simulasi jumlah sales data yg akan ditampilkan
   2. Untuk Order Data
      1. Sales Id
         1. Dipakai untuk mendapatkan Order dari Salesman dengan Id tsb
      2. Order Count
         1. Dipakai untuk melakukan simulasi jumlah order data yg akan ditampilkan

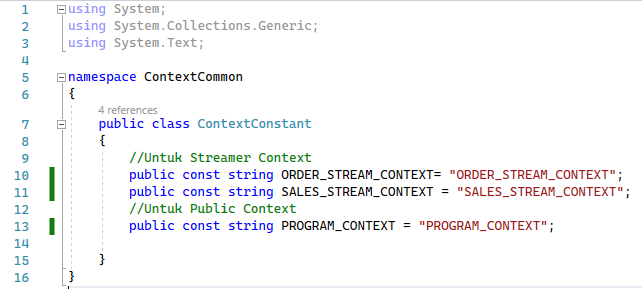
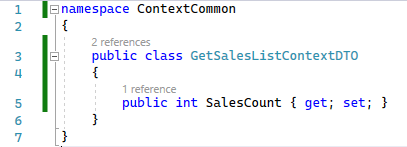
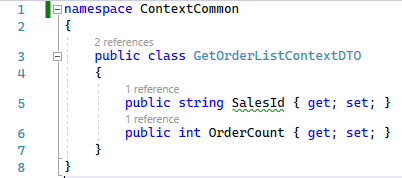
## ContextCommon

1. Penamaan dan Type Project
   1. Penamaan ContextCommon
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Standard 2.1
2. Reference Assembly R\_CommonFrontBackAPI
   1. Digunakan untuk Result DTO
   2. Digunakan untuk Interface

### Entity

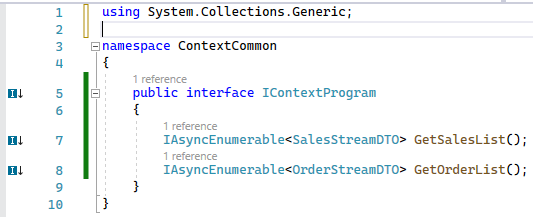
1. SalesStreamDTO
   1. 
2. OrderStreamDTO
   1. 

### Context

1. ContextConstant
   1. Merupakan kumpulan penamaan context untuk mempermudah saat meng-akses context
   2. 
2. ProgramContextDTO
   1. Merupakan DTO untuk menampung context data untuk sebuah program, biasa nya penamaan nya mengikuti nama program nya
   2. 
3. GetSalesListContextDTO
   1. Merupakan DTO untuk menampung context data yang dibutuhkan untuk mendapatkan data Salesman melalui function GetSalesList
   2. 
4. GetOrderListContextDTO
   1. Merupakan DTO untuk menampung context data yang dibutuhkan untuk mendapatkan data Order melalui function GetOrderList
   2. 

### Interface

Untuk interface kita akan membuat 2 fungsi

1. GetSalesList
2. GetOrderList
3. 

## ContextBack

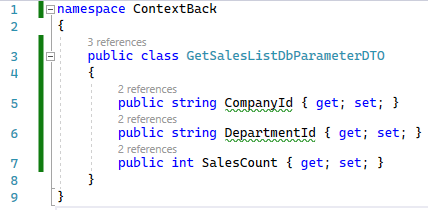
Merupakan Project yg berupa kumpulan bisnis process untuk menghasilkan data yg diminta oleh Controller API, dimana sudah dituangkan fungsi2 nya pada Interface yg ada di common.

Untuk semua fungsi yg ada di class tsb mempunyai parameter DTO tanpa melibatkan context

Data context yg datang dari front hanya akan sampai di controller dan akan diterjemahkan kedalam parameter DTO sesuai dengan fungsi2 nya masing2 sehingga bisnis proses y gada di class tsb tidak akan dipengaruhi oleh teknologi Back Interface yang dipakai (misalkan API, MVC, Web Service)

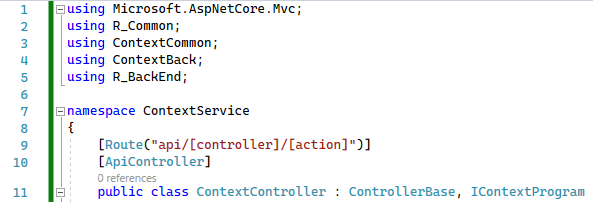
Mempunyai Class dengan nama ContextCls dan mempunyai fungsi sesuai dengan interface yang ada di common

1. Penamaan dan Type Project
   1. Penamaan ContextBack
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APICommon
      1. Dipakai untuk utility umum
      2. Dipakai untuk Exception framework
   2. Assembly R\_APIBackEnd
      1. Dipakai untuk berhubungan dengan database
   3. Project ContextCommon
3. Penamaan Class
   1. Sesuai dengan project, kata Back diubah menjadi Cls
   2. ContextCls
   3. Mempunyai fungsi untuk menghasilkan data sesuai dengan interface yg akan diimplementasikan oleh controller di project Service
   4. GetSalesListDb
      1. Text

         Description automatically generated with low confidence
      2. Menghasilkan collection dari SalesStreamDTO
      3. Mempunyai parameter
         1. GetSalesListDbParameterDTO
         2. 
   5. GetOrderListDb
      1. 
      2. Menghasilkan collection dari OrderStreamDTO
      3. Mempunyai parameter
         1. GetOrderListDbParameterDTO
         2. 

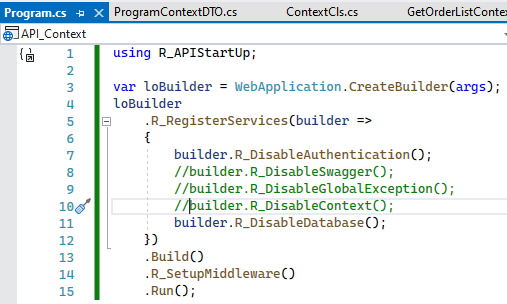
## ContextService

Merupakan project service yg berisi sekumpulan controller yg akan mendukung API sesuai dengan interface nya

1. Penamaan dan Type Project
   1. Penamaan ContextService
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Edit project file dan tambahkan
   1. <ItemGroup>
   2. <FrameworkReference Include="Microsoft.AspNetCore.App" />
   3. </ItemGroup>
3. Reference Assembly
   1. R\_CommonFrontBackAPI
      1. Untuk Interface yg digunakan oleh Ptoject ContextCommon
   2. R\_APICommon
      1. Untuk ExceptionFramework
4. Reference Project
   1. ContextCommon
   2. ContextBack
5. Buat Class Controller
   1. Class Controller tergantung dari banyak nya class Back
   2. Buat dengan penamaan Business Process ditambahkan dengan Controller
      1. ContextController
   3. Using
      1. R\_Common
      2. R\_CommonFrontBackAPI
      3. Microsoft.AspNetCore.Mvc
   4. Tambahkan annotation di class
      1. [Route("api/[controller]/[action]")]
      2. [ApiController]
   5. Inherit dan interface
      1. Inherit ControllerBase
      2. Interface sesuai dengan business object interface nya
   6. 
   7. Terjemahkan Context menjadi parameter pada back class
      1. 
   8. Karena untuk menghasilkan stream kita memakai yield sehingga kita tidak dapat memakai try catch block
      1. Kita dapat mengubah menjadi 2 fungsi
         1. HttpPost function tanpa menggunakan async dan dapat menggunakan try block
         2. Private function menggunakan async dan yield sehingga tidak dapat memakai try catch block
      2. 

## API\_Context

Merupakan project API Context

1. Penamaan dan Type Project
   1. Penamaan API\_EXP
   2. Language C#
   3. Platform Open (Linux)
   4. Type project ASP.NET Core Web API
   5. Framework NET 6.0
   6. HTTP
2. Reference Project
   1. Reference Service Project
      1. ContextService
3. Reference semua collection assembly baik Back maupun Common
   1. Semua assembly yg ada di back directory sudah termasuk common directory
   2. Reference semua assembly di directory “RuntimeLibrary/Back” semua assembly yg mempunyai awalan R\_ saja
4. Menggunakan Private realta Nuget
   1. Pakai Realta.BackCollectionNuget
5. Perubahan pada Class Program.cs
   1. using R\_APIStartUp
   2. Untuk penggunaan StartUp ada 5 hal yg dapat di-disable
      1. Authentication
      2. Swagger
      3. GlobalException
      4. Context
      5. Database
   3. Karena kita akan menggunakan Exception, Swagger dan Context, maka kita akan melakukan disable untuk point yg lain
   4. 

## Assign dan akses Context Data

### Assign Context

Context data akan di-assign oleh front end application dan mempunyai utility untuk melakukan assign context.

Pada waktu melakukan post data, maka context tsb akan diterjemahkan menjadi Header key dan value, dari sisi API Post terjemahan dari data context di Header tergantung dari model context tsb.

1. Internal Context
   1. companyId akan diterjemahkan ke header dengan
      1. Key
         1. \_internalcontext\_company\_id
      2. Value
         1. Berupa json data
            1. TypeFullName

Merupakan type data dari internal context

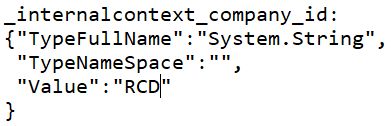
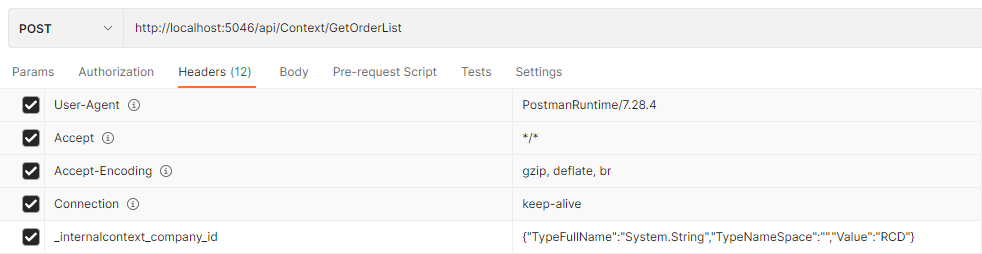
* + - * 1. TypeNameSpace

Merupakan Namespace dari Type data, kalau berupa primitive data maka akan diisi “”

* + - * 1. Value

Merupakan value dari data

Kalau primitive value dan string, maka akan diberi tada petik 2

* + 1. 
    2. Pada Postman Header
       1. 

1. Public Context
   1. Department Id akan diterjemahkan menjadi public context dengan menggunakan DTO ProgramContextDTO
   2. Karena public context dapat berupa multiple context, maka akan diterjemahkan ke header dengan
      1. Key
         1. R\_CustomContext
      2. Value
         1. Merupakan collection context dictionary <string,Object>
            1. Dalam dictionary

Key

PROGRAM\_CONTEXT

Value

TypeFullName

Merupakan type data dari DTO context

ContextCommon.ProgramContextDTO

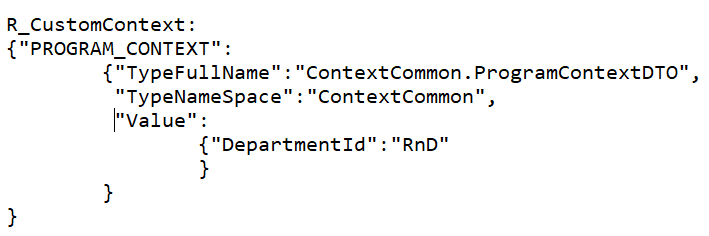
TypeNameSpace

Merupakan Namespace dari DTO context

ContextCommon

Value

Merupakan value dari data

* + 1. 
    2. Pada Postman Header
       1. 

1. Streaming Context
   1. SalesCount akan diterjemahkan menjadi streaming context dengan menggunakan DTO GetSalesListContextDTO
   2. Karena streaming context dapat berupa multiple context, maka akan diterjemahkan ke header dengan
      1. Key
         1. R\_StreamingContext
      2. Value
         1. Merupakan collection context dictionary <string,Object>
            1. Dalam dictionary

Key

SALES\_STREAM\_CONTEXT

Value

TypeFullName

Merupakan type data dari DTO context

ContextCommon.GetSalesListContextDTO

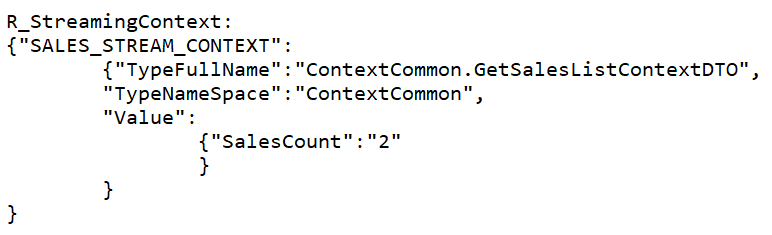
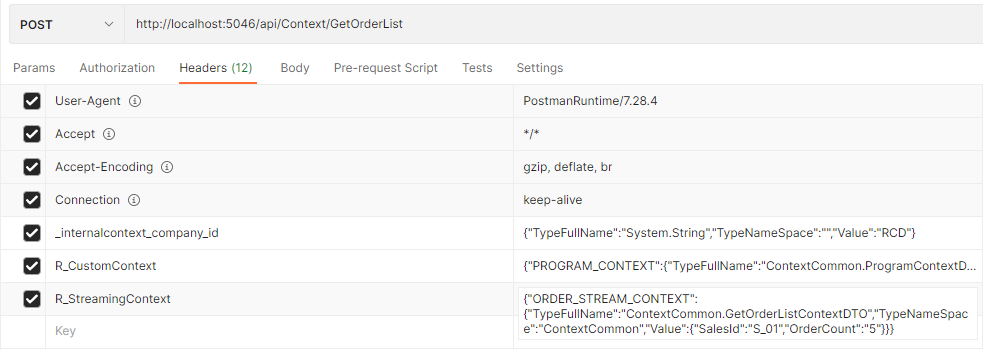
TypeNameSpace

Merupakan Namespace dari DTO context

ContextCommon

Value

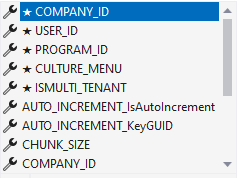
Merupakan value dari data

* + 1. 
    2. Pada Postman Header
       1. 

### Akses Context

Context akan di-akses di bagian backEnd pada Controller yang nantinya akan diteruskan ke Back Class dalam bentuk parameter.

Untuk meng-akses context tergantung dari model context

1. Internal Context
   1. Internal Context dapat kita akses menggunakan Class R\_BackGlobalVar yang ada di R\_BackEnd
   2. Internal Context akan diterjemahkan menjadi property pada class R\_ BackGlobalVar, sebagai contoh R\_BackGlobalVar.COMPANY\_ID
   3. Contoh dari property internal context y gada di class R\_ BackGlobalVar
      1. 
2. Public Context
   1. Public context dapat di-akses melalui class R\_Utility yg ada di R\_Common
   2. Public Context dapat diakses melalui method R\_GetContext<Type DTO>
   3. 
3. Streaming Context
   1. Streaming context dapat di-akses melalui class R\_Utility yg ada di R\_Common
   2. Streaming Context dapat diakses melalui method R\_GetStreamingContext<Type DTO>
   3. 

# CRUD

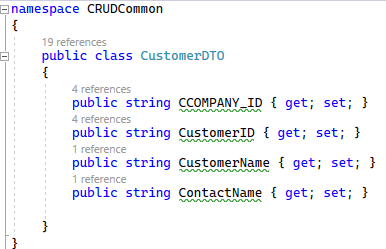
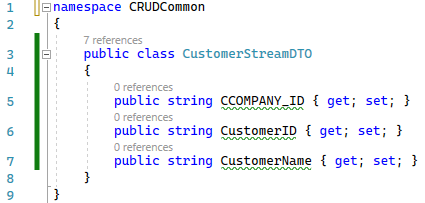
Secara umum kita sering kali melakukan suatu transaksi yg melibatkan action CRUD (Create, Update, Delete) pada sebuah entity atau record.

Untuk hal ini, kita menyediakan framework BackEnd yg dapat dipakai untuk melakukan proses CRUD.

## CRUDCommon

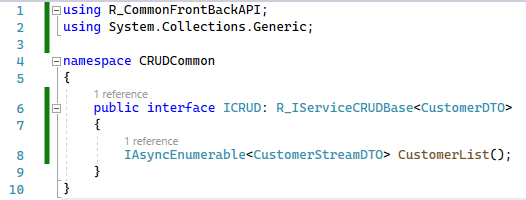
1. Penamaan dan Type Project
   1. Penamaan CRUDCommon
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Standard 2.1
2. Reference Assembly R\_CommonFrontBackAPI
   1. Digunakan untuk Result DTO
   2. Digunakan untuk Interface

### Entity

1. CustomerDTO
   1. Digunakan sebagai entity Single yg dipakai untuk melakukan action CRUD
   2. Pada entity ini semua property harus lengkap termasuk yg ada di Context internal seperti Company Id
   3. Entity ini akan digunakan oleh Controller maupun oleh Back, sehingga tidak memerlukan DTO tambahan pada Back
   4. Karena Entity ini akan digunakan di Back, maka penamaan property harus disesuaikan dengan database, misalnya CompanyId akan disesuaikan dengan database menjadi CCOMPANY\_ID
   5. Semua property yg disesuaikan dengan database bersifat Case Sensitive, jadi huruf besar atau kecil harus sama
   6. 
2. CustomerStreamDTO
   1. Digunakan sebagai entity collection, dimana pada entity tsb biasa nya jumlah field nya tidak sebanyak entity single nya
   2. Entity ini akan digunakan oleh Controller maupun oleh Back, sehingga tidak memerlukan DTO tambahan pada Back
   3. Karena Entity ini akan digunakan di Back, maka penamaan property harus disesuaikan dengan database, misalnya CompanyId akan disesuaikan dengan database menjadi CCOMPANY\_ID
   4. Semua property yg disesuaikan dengan database bersifat Case Sensitive, jadi huruf besar atau kecil harus sama
   5. 

### Interface

Karena kita sudah mendukung action CRUD di framework backend, maka untuk interface

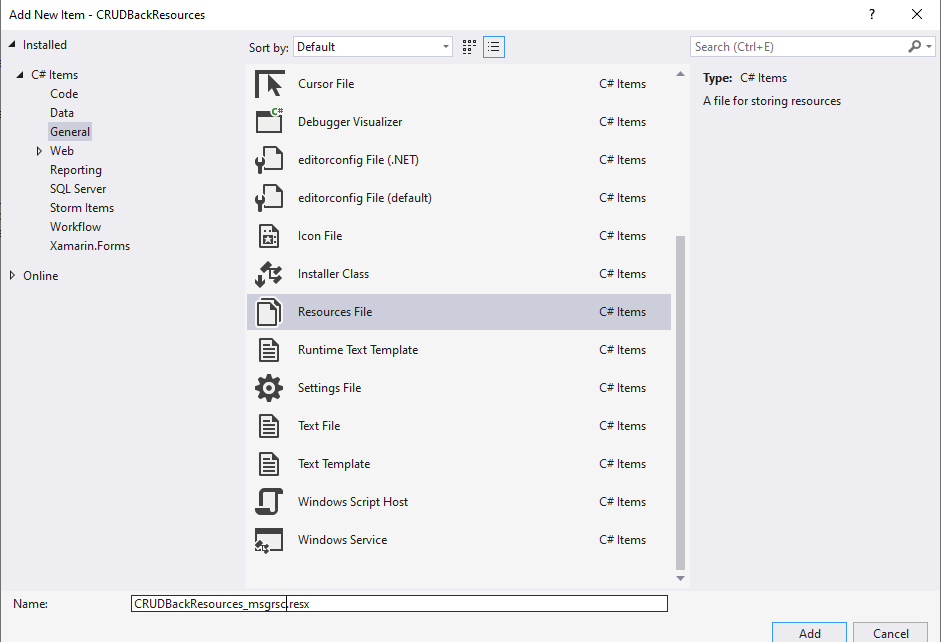
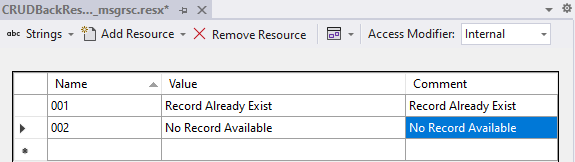
1. Kita meng-inherits dari R\_IServiceCRUDBase<[Entity]>
2. Dimana Entity nya adalah CustomerDTO
3. Kita menambahkan fungsi untuk mendapatkan collection CustomerStreamDTO
   1. CustomerList
4. 

## CRUDBackResources

Merupakan project Resources yg dipakai untuk menghasilkan message yg dapat kita pakai sebagai message validasi.

Project ini diperlukan terpisah karena Message ini dapat dipakai untuk multi language dan nantinya dapat dikerjakan terjemahan nya oleh orang yg berbeda tanpa harus mengganggu project Back nya sendiri.

Karena Project resources ini akan diintegrasikan dengan library yg sudah dipersiapkan untuk menghasilkan message multi language sesuai dengan language yg dikirim dari dari front, maka project ini mempunyai beberapa aturan dan cara penamaan nya.

1. Penamaan dan Type Project
   1. Penamaan CRUDBackResources
      1. Sesusaikan dengan nama project Back nya ditambahkan Resources dibelakang nya
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework NET 6.0
2. Tambahkan Class dengan nama Resources\_Dummy\_Class
3. Tambahkan resources dengan nama CRUDBackResources\_msgrsc
   1. Penamaan Resources sesuai dengan nama project ditambahkan “\_msgrsc” dibelakang nya
   2. 
4. Kita dapat mengisi Name dan Value nya sesuai dengan pengaturan kita
   1. Name harus unique untuk 1 resources
   2. Untuk mempermudah penterjemah resources, maka lebih baik kita menuliskan comment sama dengan value nya
   3. 

## CRUDBack

Merupakan Project yg berupa kumpulan bisnis process untuk menghasilkan data yg diminta oleh Controller API, dimana sudah dituangkan fungsi2 nya pada Interface yg ada di common.

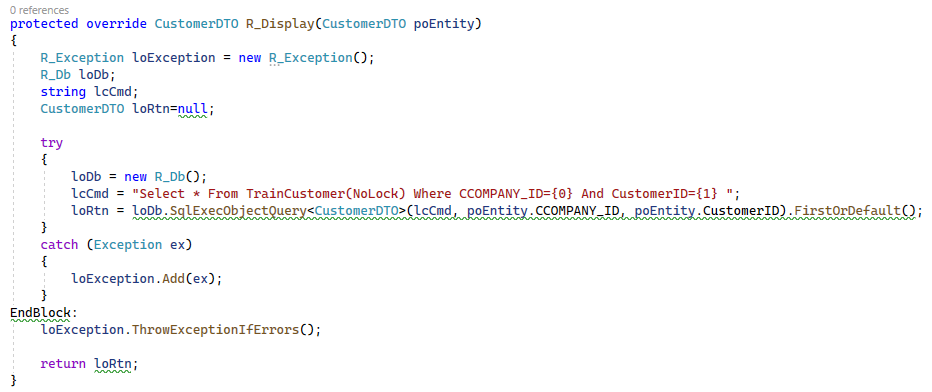
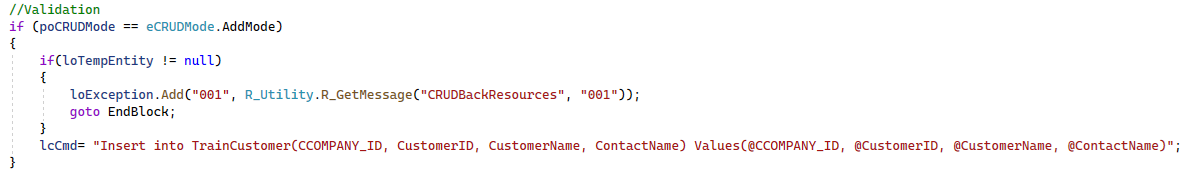
Untuk semua fungsi yg ada di class tsb mempunyai parameter DTO tanpa melibatkan context

Untuk CRUD, kita sudah menyediakan library dari Interface samapi ke Back, untuk Back kita dapat meng-inherits dari R\_BusinessObject<[Entity]>

Entity di Back ini tidak harus sama dengan Entity yg kita pakai di Service, tapi secara umum kalau Interface yg diimplementasikan di service mengarah pada Back dengan relasi 1-1 maka kita memakai entity yg sama.

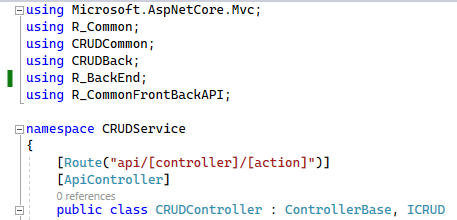
Mempunyai Class dengan nama CRUDCls dan mempunyai fungsi sesuai dengan interface yang ada di common

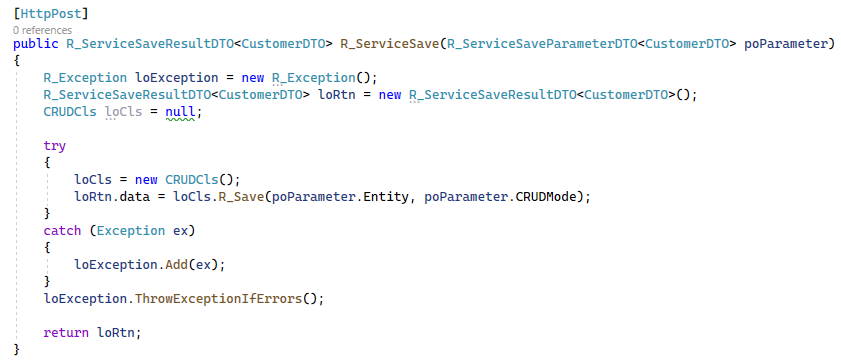
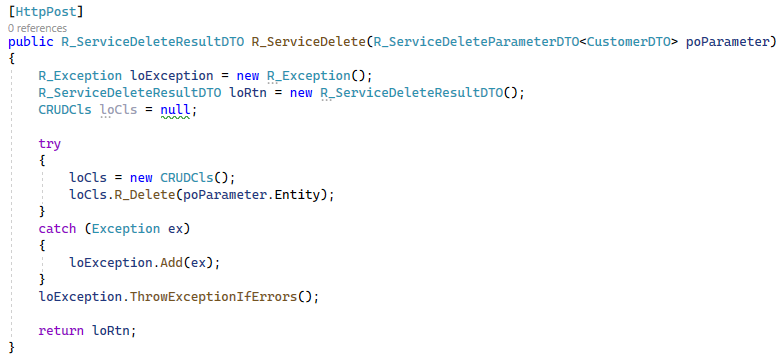
1. Penamaan dan Type Project
   1. Penamaan CRUDBack
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APICommon
      1. Dipakai untuk utility umum
      2. Dipakai untuk Exception framework
   2. Assembly R\_APIBackEnd
      1. Dipakai untuk berhubungan dengan database
   3. Assembly R\_CommonFrontBackAPI
      1. Dipakai untuk enumeration CRUD library
   4. Project CRUDCommon
   5. Project CRUDBackResources
3. Penamaan Class
   1. Sesuai dengan project, kata Back diubah menjadi Cls
   2. CRUDCls
   3. Mempunyai fungsi untuk menghasilkan data sesuai dengan interface yg akan diimplementasikan oleh controller di project Service
   4. Inherits dari R\_BusinessObject<[Entity]>
   5. Graphical user interface, text, application

      Description automatically generated
   6. Setelah kita meng-inherits, maka ada 3 method yg harus kita override
4. Method2 yg akan di-Override
   1. R\_Display
      1. Merupakan method Read bagian dari CRUD yg menghasilkan Entity dan mempunyai parameter entity
         1. 
      2. Method ini dipakai untuk menghasilkan single object entity (dalam hal ini CustomerDTO) yang kita dapatkan dari database dengan kondisi yg kita dapatkan dari parameter yg berupa entity dg type yg sama (dalam hal ini poEntity)
      3. 
   2. R\_Saving
      1. Merupakan method Create dan Update bagian dari CRUD yang mempunyai 2 parameter
         1. poNewEntity yg merupakan parameter entity yang informasinya akan disimpan ke database
         2. poCRUDMode yg merupakan parameter enumeration untuk menentukan operation penyimpanan apakah Create atau Update
      2. 
      3. Method ini dipakai untuk melakukan penyimpanan data tergantung dari mode operation-nya juga pada method ini dapat dilakukan validasi terlebih dahulu, kalau memang validasi nya gagal, maka kita dapat melakukan throw exception pada method tsb.
      4. Untuk Validasi, kita dapat memasukkan exception collection dari message yg kita sudah buat pada resources
         1. 
      5. Kalau sudah lolos dari validation, maka kita dapat memakai utility R\_Db untuk melakukan saving ke database
      6. 
   3. R\_Delete
      1. Merupakan method Delete bagian dari CRUD yang mempunyai parameter entity yg dipakai sebagai informasi untuk mendapatkan record yg akan di-delete di database.
      2. Dalam method ini kita juga dapat melakukan validasi terlebih dahulu dan melakukan throw Exception kalau memang validasi gagal
      3. 
5. Penambahan method untuk menghasilkan collection entity
   1. CustomerListDb
      1. Merupakan method untuk menghasilkan collection dari stream entity dalam bentuk list
      2. Mempunyai parameter Entity untuk melakukan pengelompokan sejumlah record dari database
      3. 

## CRUDService

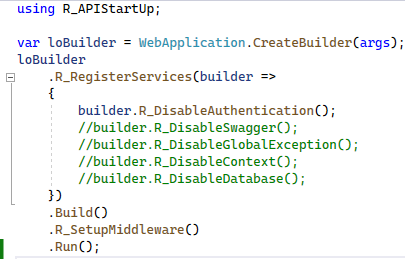
Merupakan project service yg berisi sekumpulan controller yg akan mendukung API sesuai dengan interface nya

1. Penamaan dan Type Project
   1. Penamaan ContextService
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Edit project file dan tambahkan
   1. <ItemGroup>
   2. <FrameworkReference Include="Microsoft.AspNetCore.App" />
   3. </ItemGroup>
3. Reference Assembly
   1. R\_APIBackEnd
      1. Dipakai untuk mendapatkan internal context y gada di R\_BackGlobalVar
   2. R\_APICommon
      1. Untuk ExceptionFramework
   3. R\_CommonFrontBackAPI
      1. Untuk Interface yg digunakan oleh Project CRUDCommon
4. Reference Project
   1. CRUDCommon
   2. CRUDBack
5. Buat Class Controller
   1. Class Controller tergantung dari banyak nya class Back
   2. Buat dengan penamaan Business Process ditambahkan dengan Controller
      1. CRUDController
   3. Using
      1. R\_BackEnd
      2. R\_Common
      3. R\_CommonFrontBackAPI
      4. Microsoft.AspNetCore.Mvc
   4. Tambahkan annotation di class
      1. [Route("api/[controller]/[action]")]
      2. [ApiController]
   5. Inherit dan interface
      1. Inherit ControllerBase
      2. Interface sesuai dengan business object interface nya
   6. 
   7. R\_ServiceGetRecord
      1. Result type R\_ServiceGetRecordResultDTO<[Entity]>
      2. Parameter type R\_ServiceGetRecordParameterDTO<[Entity]>
      3. Akan memanggil method R\_GetRecord di project Back nya dan akan melakukan passing parameter dengan entity yg sama atau bisa juga berbeda, Entity didapat dari parameter method ini
      4. Graphical user interface, text, application, email

         Description automatically generated
   8. R\_ServiceSave
      1. Result type R\_ServiceSaveResultDTO<[Entity]>
      2. Parameter type R\_ServiceSaveParameterDTO<[Entity]>
      3. Akan memanggil method R\_Save di project Back nya dan akan melakukan passing parameter
         1. Entity bisa dengan entity yg sama atau bisa juga berbeda, Entity didapat dari parameter method ini
         2. CRUDMode, didapat dari parameter method tsb
      4. 
   9. R\_ServiceDelete
      1. Result type R\_ServiceDeleteResultDTO
      2. Parameter type R\_ServiceDeleteParameterDTO<[Entity]>
      3. Akan memanggil method R\_Delete di project Back nya dan akan melakukan passing parameter dengan entity yg sama atau bisa juga berbeda, Entity didapat dari parameter method ini
      4. 
   10. CustomerList
       1. Menghasilkan streaming Collection dg type IAsyncEnumerable<[Entity]>
       2. Tidak ada parameter yg dikirim, untuk pembatasan kriteria kita menggunakan context
       3. Akan memanggil method yg sudah dibuat di Back Class hanya saja method Collection List dari Back Class akan diterjemahkan menjadi 2 function public dan private spy API yg dipanggil dapat menghasilkan streaming data
       4. 

## API\_CRUD

Merupakan project API CRUD

1. Penamaan dan Type Project
   1. Penamaan API\_CRUD
   2. Language C#
   3. Platform Open (Linux)
   4. Type project ASP.NET Core Web API
   5. Framework NET 6.0
   6. HTTP
2. Reference Project
   1. Reference Service Project
      1. CRUDService
3. Reference semua collection assembly baik Back maupun Common
   1. Semua assembly yg ada di back directory sudah termasuk common directory
   2. Reference semua assembly di directory “RuntimeLibrary/Back” semua assembly yg mempunyai awalan R\_ saja
4. Menggunakan Private realta Nuget
   1. Pakai Realta.BackCollectionNuget
5. Perubahan pada Class Program.cs
   1. using R\_APIStartUp
   2. Untuk penggunaan StartUp ada 5 hal yg dapat di-disable
      1. Authentication
      2. Swagger
      3. GlobalException
      4. Context
      5. Database
   3. Karena kita akan menggunakan Exception, Swagger dan Context, Database maka kita akan melakukan disable untuk point yg lain
   4. 
6. Penambahan pada AppSetting
   1. Karena kita akan berhubungan dengan database, maka kita akan tambahkan konfigurasi database di bagian AppSettings.json
   2. Kita dapat melakukan setting untuk beberapa database di bagian R\_DBConfigs, semua nya akan dijelaskan pada dokumentasi R\_Db
   3. 

# Batch Process and Upload

Kita juga mengenal transaksi yg bersifat processing beberapa record dari front dikirim ke back secara bulk (many record) juga proses untuk melakukan upload file yg cukup besar sehingga file tsb tidak dapat disimpan dengan cara CRUD untuk single record atau untuk file yg besar.

## Database Part

### Framework

#### Table

Ada beberapa table yg akan dipakai untuk melakukan proses Upload maupun Batch Process

1. GST\_SPLIT\_UPLOAD
   1. Table ini dipakai untuk menampung Data yg berupa ByteArray yang terpotong2 KeyGUID sebagai headernya
2. GST\_UPLOAD\_PROCESS\_STATUS
   1. Table ini dipakai untuk melaporkan status dari suatu proses berdasarkan KeyGUID
3. GST\_UPLOAD\_ERROR\_STATUS
   1. Table ini dipakai untuk melaporkan Error validasi business process yg terjadi berdasarkan KeyGUID

#### Stored Procedure dan Function

Ada beberapa SP dan function yg digunakan untuk melakukan proses upload dan Batch

1. RFN\_CombineByte
   1. Merupakan function yg dipakai untuk melakukan penggabungan kumpulan beberapa Data ByteArray menjadi sebuah kumpulan ByteArray berdasarkan key tertentu, function tsb merupakan function Dll yg didapat dengan cara meregister dll ke SQL
   2. Parameter
      1. CompanyId
      2. UserId
      3. KeyGUID
   3. Return ByteArray
2. RSP\_WriteUploadProcessStatus
   1. Merupakan procedure yang dipakai untuk menuliskan status proses berdasarkan key tertentu ke table GST\_UPLOAD\_PROCESS\_STATUS
   2. Parameter
      1. CompanyId
      2. UserId
      3. KeyGUID
      4. Step
      5. Status
      6. Finish (0=Process, 1=Success, 9=Fail)
3. RSP\_Write\_Error
   1. Merupakan procedure yang dipakai untuk menuliskan error suatu proses berdasarkan key tertentu ke table GST\_UPLOAD\_ERROR\_STATUS
   2. Parameter
      1. RECID
         1. Akan diisi dengan Recid dari process
      2. Error\_Code
         1. Code dari Error nya
      3. Detail
         1. Detail error message

### Training

#### Table

1. TestEmployeeAttachment
   1. Dipakai untuk melakukan testing Attachment
2. TestEmployeeTable
   1. Dipakai untuk melakukan testing SaveBatch
3. TestCompanyTable
   1. Dipakai untuk melakukan validasi data pada testing SaveBatch
4. TestSexTable
   1. Dipakai untuk melakukan validasi data pada testing SaveBatch

#### Stored Procedure

1. SampleProcessBatch
   1. Dipakai untuk melakukan testing Process Batch
2. SaveBatchEmployee
   1. Dipakai untuk melakukan testing SaveBatch

## Library Part

### Common Library Part

Pada R\_CommonFrontBackAPI library ditambahkan beberapa structure yang dipakai untuk proses upload dan batch

#### Structure R\_KeyValue

Merupakan structure yang dipakai untuk menyimpan pasangan Key dan Value

1. Property
   1. Key berupa String
   2. Value berupa Object

#### Structure R\_File

Merupakan structure yang dipakai untuk menyimpan informasi File

1. Property
   1. FileId
      1. Merupakan Id dari sebuah File
   2. FileDescription
      1. Merupakan Description dari sebuah File
   3. FileExtension
      1. Merupakan Extension dari sebuah file

#### Structure R\_UploadAndProcessKey

Merupakan structure yang dipakai sebagai key pada proses upload dan batch

1. Property
   1. COMPANY\_ID
   2. USER\_ID
   3. KEY\_GUID

#### Structure R\_UploadPar

Merupakan structure yang dipakai sebagai parameter pada waktu mau menjalankan proses upload file maupun Attach File

1. Property
   1. COMPANY\_ID
      1. Company saat Login
   2. USER\_ID
      1. User saat Login
   3. File
      1. Type R\_File
   4. FilePath
      1. Merupakan nama full path dari file yang akan di-upload
   5. ClassName
      1. Merupakan Class yang akan dipanggil di bagian backend untuk melakukan proses upload atau attachment
   6. UserParameters
      1. Merupakan kumpulan parameter user yang akan dikirimkan ke back
      2. Type List(Of R\_KeyValue)

#### Structure R\_BatchParameter

Merupakan structure yang dipakai sebagai parameter pada waktu mau melakukan proses batch

1. Property
   1. COMPANY\_ID
      1. Company saat Login
   2. USER\_ID
      1. User saat Login
   3. ClassName
      1. Merupakan Class yang akan dipanggil di bagian backend untuk melakukan proses upload atau attachment
   4. BigObject
      1. Merupakan object yang bersifat besar yang akan dikirim ke belakang untuk diproses, biasanya berupa sekumpulan DTO
      2. Type Object
   5. UserParameters
      1. Merupakan kumpulan parameter user yang akan dikirimkan ke back
      2. Type List(Of R\_KeyValue)

#### Structure R\_ErrorStatusReturn

Merupakan structure yang dipakai untuk mengembalikan error status yang berisi validasi bisnis proses, biasanya berupa kumpulan error status

1. Property
   1. SeqNo
      1. Urutan dari Error, biasa juga dipakai untuk field Seqno yang ada di File Excel yang akan diproses
   2. ErrorMessage
      1. Message Error

### Front Library Part

Pada R\_ProcessAndUploadFront library ditambahkan beberapa class yang dipakai untuk proses upload dan batch

#### Class R\_ProcessAndUploadClient

Merupakan class yang di-instantiate dan dipakai di bagian front untuk melakukan proses upload, Attachment File dan batch process

##### Instantiate

Pada waktu melakukan instantiate, ada 2 option

1. New(pcHttpClientName, pcRequestServiceEndPoint, poProcessProgressStatus, plSendWithContext, plSendWithToken)
   1. pcHttpClientName
      1. Type string
      2. Nama dari httpclint
      3. Default
         1. “DEFAULT”
   2. pcRequestServiceEndPoint
      1. Type string
      2. Nama dari Service point untuk process
      3. Default
         1. "api/helper/R\_ProcessAndUpload"
   3. poProcessProgressStatus
      1. Type R\_IProcessProgressStatus
      2. merupakan instant dari interface R\_IProcessProgressStatus yang nantinya akan dipakai untuk melakukan proses monitoring status dan error
      3. Default
         1. null
   4. plSendWithContext
      1. Type Boolean
      2. Merupakan parameter flag untuk menentukan apakah process tsb akan memakai context
      3. Kalau memakai context, maka context akan dibaca dari ContextHeader yg akan dimasukkan pada parameter saat kita meng-create R\_HTTPClient object yg akan kita pakai namanya di parameter pcHttpClientName saat melakukan instantiate
      4. Default
         1. True
   5. plSendWithToken
      1. Type Boolean
      2. Merupakan parameter flag untuk menentukan apakah process tsb akan memakai token
      3. Kalau memakai token, maka token akan didapat dari Token Repository
      4. Default
         1. True

##### Method R\_AttachFile

merupakan method yang dipakai untuk melakukan attach sebuah file yang nantinya akan disimpan dalam bentuk record bersama dengan id dan extension file tsb

1. Parameter
   1. Upload parameter
      1. Type R\_UploadPar
      2. Lihat R\_UploadPar di bagian Common part

##### Function R\_ProcessFile

Merupakan function yang dipakai untuk melakukan proses sebuah file secara asynchronous, biasanya berupa Excel File

1. Parameter
   * 1. ProcessFile Parameter
        1. Type R\_UploadPar
        2. Lihat R\_UploadPar di bagian Common part
     2. Final Step
        1. Merupakan jumlah step dalam proses tsb
   1. Return
      1. KeyGUID
         1. Merupakan GUID yang akan dipakai sebagai key unique id proses tsb

##### Function R\_BatchProcess

Merupakan function yang dipakai untuk melakukan proses batch secara asynchronous

1. Parameter
   * 1. Batch Parameter
        1. Type R\_BatchParameter
        2. Lihat R\_BatchParameter di bagian Common part
     2. Final Step
        1. Merupakan jumlah step dalam proses tsb
   1. Return
      1. KeyGUID merupakan GUID yang akan dipakai sebagai key unique id proses tsb

#### Interface R\_IProcessProgressStatus

##### Method ProcessComplete

Merupakan method yang dipakai oleh proses asynchronous, method ini untuk menandakan kalau proses sudah selesai

1. Parameter
   1. KeyGuid merupakan GUID yang akan dipakai sebagai key unique id proses tsb
   2. ProcessResultMode, merupakan status dari proses tsb
      1. Success
      2. Fail

##### Method ProcessError

Merupakan method yang dipakai oleh proses asynchronous, event ini untuk menandakan kalau ada suatu error yg terjadi pada proses tsb

1. Parameter
   * 1. KeyGuid merupakan GUID yang akan dipakai sebagai key unique id proses tsb
     2. Exception, merupakan exception error pada proses tsb

##### Method ReportProgress

Merupakan method yang dipakai oleh proses asynchronous, method ini dipakai untuk melakukan monitoring status dan progress pada proses tsb

1. Parameter
   * 1. lnProgress merupakan integer yang akan dipakai progress step dari proses tsb
     2. lcStatus merupakan string yang akan dipakai untuk melaporkan status dari proses tsb

### Back Library part

Pada R\_APIBackEnd akan ditambahkan Class untuk menangani process upload dan batch

#### Interface R\_IAttachFile

##### Method R\_AttachFile

Merupakan method di bagian Back End yg mempunyai informasi tentang file yg akan di attach bersama dengan informasi lain nya yang akan dipakai untuk melakukan saving ke database

1. Parameter
   1. poAttachFile
   2. Type R\_AttachFilePar
      1. Key
         1. Type R\_UploadAndProcessKey
         2. Lihat bagian Common Part
      2. File
         1. Type R\_File
         2. Lihat bagian Common Part
      3. BigObject
         1. Type Byte Array
         2. Merupakan Object dalam bentuk Byte Array yg dapat dikirimkan dari Front
         3. Tetapi biasanya tidak terpakai untuk attach file
      4. UserParameters
         1. Type
            1. List<R\_KeyValue>
            2. R\_KeyValue lihat bagian Common Part
         2. Parameter tsb dibentuk oleh user sebagai informasi tambahan untuk record yg yang akan di save di table bersama2 dengan file tsb

#### Interface R\_IBatchProcess

##### Method R\_BatchProcess

Merupakan method yg dipakai untuk melakukan Batch Process maupun melakukan Bulk Saving

1. Parameter
   1. poBatchProcessPar
   2. Type R\_BatchProcessPar
      1. Key
         1. Type R\_UploadAndProcessKey
         2. Lihat bagian Common Part
      2. BigObject
         1. Type Byte Array
         2. Merupakan Object dalam bentuk Byte Array yg dapat dikirimkan dari Front
         3. Object tsb yg akan berupa data yg akan diproses secara batch atau di save secara bulk
      3. UserParameters
         1. Type
            1. List<R\_KeyValue>
            2. R\_KeyValue lihat bagian Common Part
         2. Parameter tsb dibentuk oleh user sebagai informasi tambahan untuk melakukan proses batch maupun bulk save

#### Class R\_DB

Merupakan Class yang berhubungan dengan database

##### Method R\_BulkInsert

Merupakan method yang dipakai untuk melakukan insert sekumpulan DTO kedalam sebuah table secara bulk

1. Parameter
   1. Connection, merupakan connection ke database yg berupa type SqlConnection
   2. Nama Table, merupakan nama table yg akan di-insert
   3. ListofDTO merupakan collection dari DTO yang akan disimpan
   4. Mapping merupakan parameter optional Dictionary(String,String) yang dipakai untuk melakukan mapping antara field dto dan field table yang tidak sama namanya

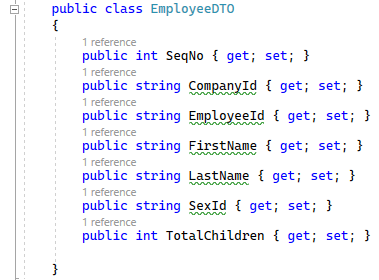
## Pembuatan project

Untuk pembuatan project process, save batch dan upload tidak memerlukan project service karena sudah ada di framework R\_APIBackEnd

### ProcessCommon

1. Penamaan dan Type Project
   1. Penamaan ProCommon
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Standard 2.1
2. Reference Assembly R\_CommonFrontBackAPI
   1. Digunakan untuk Result DTO

#### Entity

1. EmployeeDTO
   1. Digunakan sebagai entity Single yg dipakai untuk melakukan melakukan process Save Batch
   2. Entity ini akan digunakan oleh Controller maupun oleh Back, sehingga tidak memerlukan DTO tambahan pada Back
   3. 

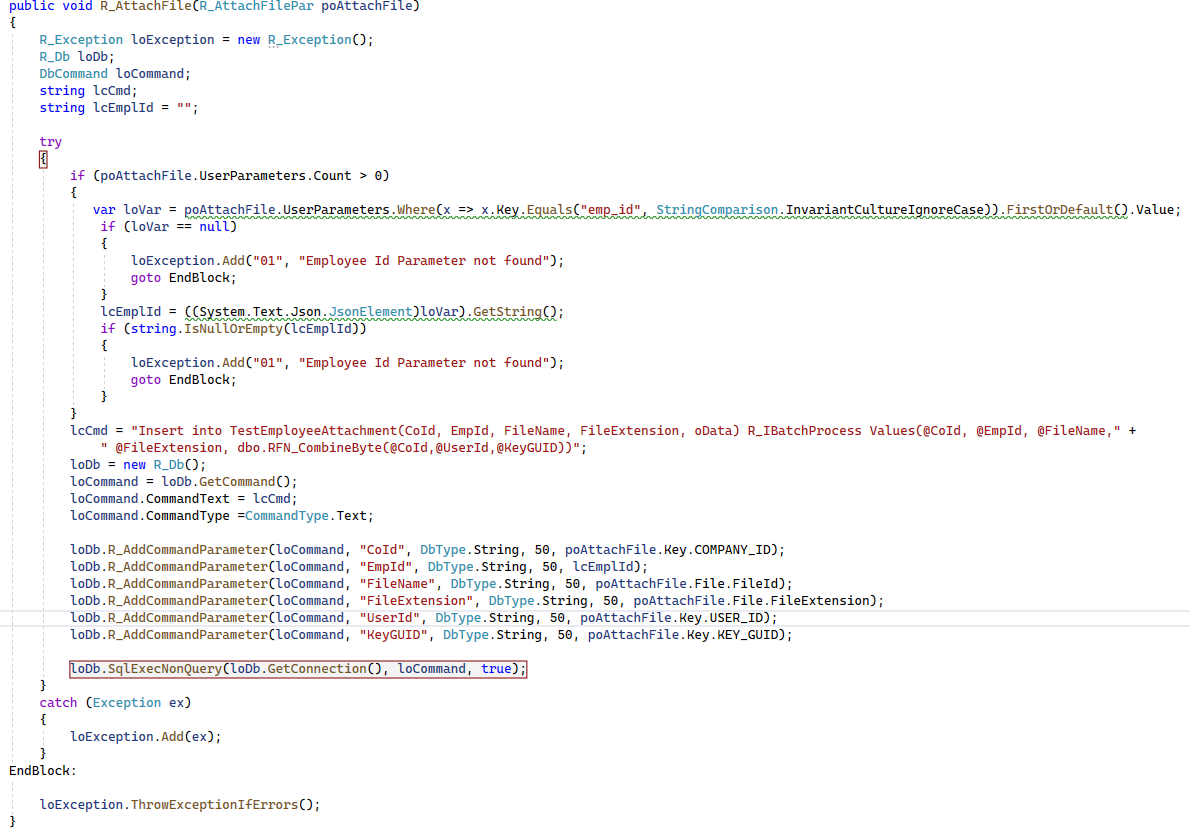
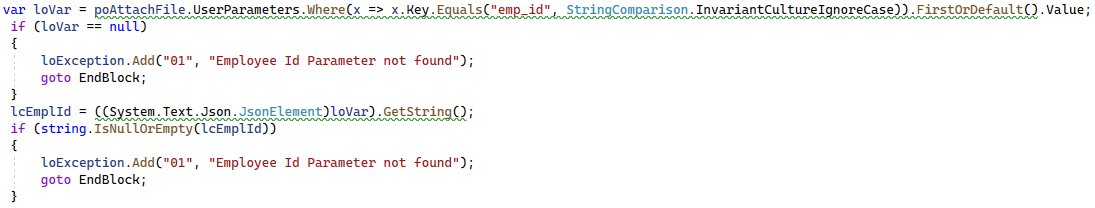
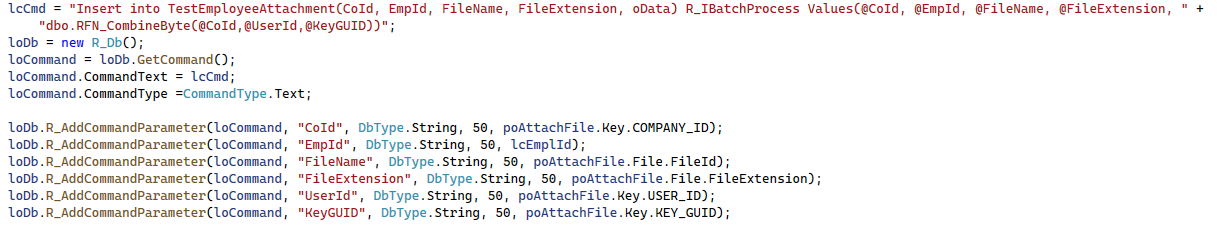
### ProcessBack

Merupakan project khusus yg sudah mempunyai Class2 khusus untuk 3 model bisnis proses.

1. Penamaan dan Type Project
   1. Penamaan ProcessBack
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APICommon
      1. Dipakai untuk utility umum
      2. Dipakai untuk Exception framework
   2. Assembly R\_APIBackEnd
      1. Dipakai untuk berhubungan dengan database
   3. Assembly R\_CommonFrontBackAPI
      1. Dipakai untuk enumeration CRUD library
   4. Project ProcessCommon
   5. Packet Nuget System.Data.SqlClient versi 4.8.5
      1. Digunakan untuk proses SaveBatch karena harus menggunakan Sql Client driver
3. Penamaan Class
   1. Untuk Class kita dapat menggunakan penamaan class dengan bebas, tapi lbh baik disesuaikan dengan nama program nya, karena ini merupakan training, maka saya pisahkan masing2 model bisnis proses menggunakan class yg berbeda

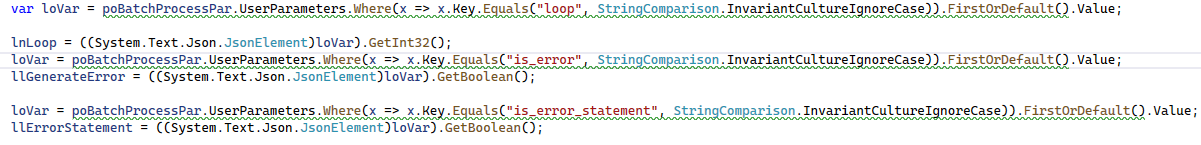
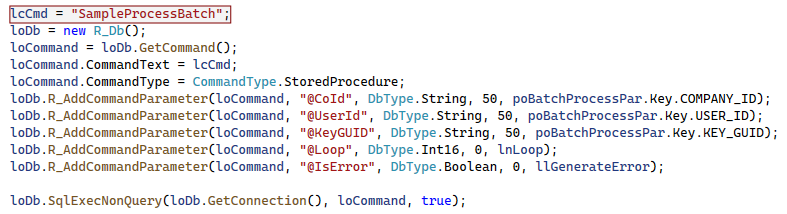
#### Attachment

Untuk melakukan proses Attachment, maka kita dapat membuat sebuah class yg dipakai untuk process attachment File.

1. Penamaan Class AttachFileCls
2. Implementasi dari R\_IAttachFile
   1. Lihat bagian Back Library Part
3. 
4. Untuk mendapatkan UserParameters
   1. Karena UserParameters berType List<R\_KeyValue> maka kita dapat memakai LINQ untuk mendapatkan Value berdasarkan key yg kita cari
   2. Value yg kita dapatkan adalah berupa object, maka kita harus melakukan convert dengan menggunakan System.Text.Json.JsonElement untuk mendapatkan type real yg kita kirim dari front, misalkan type String, maka kita akan mengembalikan dengan GetString()
   3. 
5. Untuk mendapatkan File data berupa Byte Array, maka kita dapat memakai function SQL dbo.RFN\_CombineByte
   1. Dapat dilihat pada Database Part bagian Framework untuk Stored Procedure dan Function
   2. 

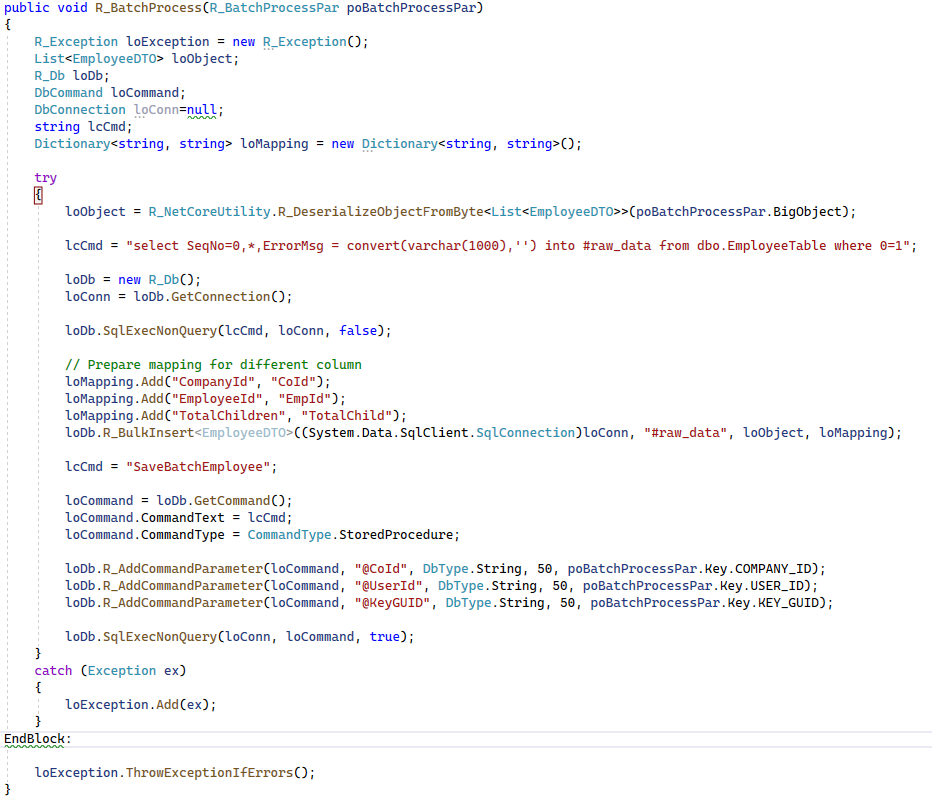
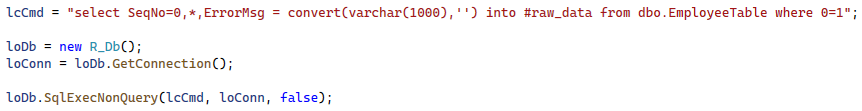
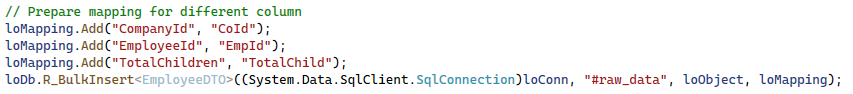
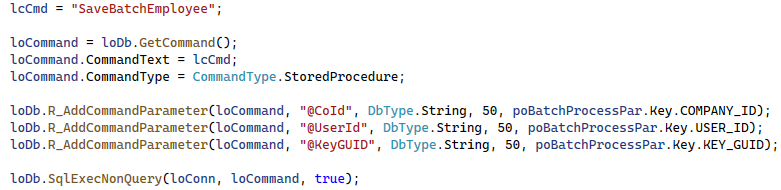
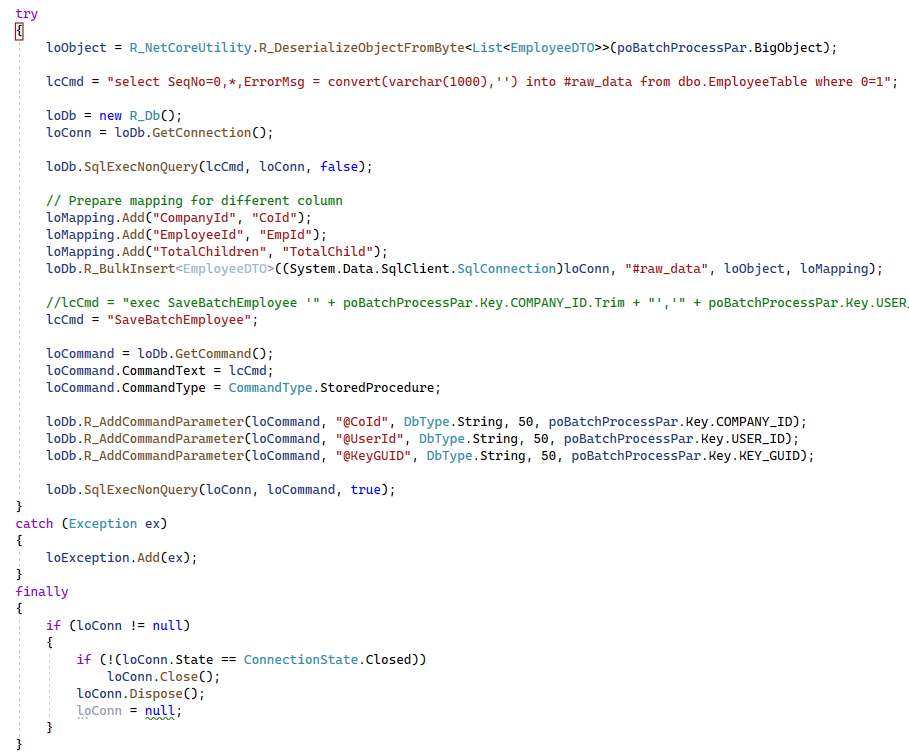
#### Batch Process

Untuk melakukan Batch process, maka kita dapat membuat sebuah class untuk melakukan proses batch

1. Penamaan Class BatchProcessCls
2. Implementasi dari R\_IBatchProcess
   1. Lihat bagian Back Library Part
3. 
4. Untuk mendapatkan UserParameters
   1. Karena UserParameters berType List<R\_KeyValue> maka kita dapat memakai LINQ untuk mendapatkan Value berdasarkan key yg kita cari
   2. Value yg kita dapatkan adalah berupa object, maka kita harus melakukan convert dengan menggunakan System.Text.Json.JsonElement untuk mendapatkan type real yg kita kirim dari front, dari front terdapat 3 parameter
   3. 
   4. Lalu kita menjalankan Stored Procedure dengan menggunakan parameter
   5. 

#### Bulk Save

Procedure ini sebenarnya mempunyai interface yg sama dengan Batch Process hanya saja tujuan nya bukan melakukan proses menggunakan Stored Procedure dengan parameter tertentu melainkan melakukan insert terhadap sekumpulan DTO.

1. Penamaan Class SaveBatchWithBulkCopyCls
2. Implementasi dari R\_IBatchProcess
   1. Lihat bagian Back Library Part
3. 
4. Untuk mendapatkan collection data yg akan dimasukkan ke dalam table
   1. Data tsb akan kita dapatkan dari Property BigObject (Lihat bagian Back Library Part ) dan kita lakukan deserialisasi dengan menggunakan R\_NetCoreUtility.R\_DeserializeObjectFromByte yg ada di R\_CommonFrontBackAPI
   2. 
5. Karena kita mau melakukan Insert ke Table SQL lewat Stored procedure, maka kita harus
   1. Membuat Empty Temporary Table
      1. 
   2. Melakukan mapping antara DTO dan table temporary dan memasukkan DTO Collection kedalam Temporary Table dengan bantuan R\_BulkInsert dari R\_DB
      1. 
   3. Menjalankan Stored Procedure yg dipakai untuk melakukan proses validasi dan memasukkan data kedalam SQL table
      1. 
   4. Langkah untuk melakukan Bulk Insert ini **harus menggunakan connection yg sama dan membiarkan connection dalam keadaan terbuka**
   5. Connection akan ditutup di block finally
      1. 

### Process\_API

Merupakan project API Process Upload

1. Penamaan dan Type Project
   1. Penamaan ProcessAPI
   2. Language C#
   3. Platform Open (Linux)
   4. Type project ASP.NET Core Web API
   5. Framework NET 6.0
   6. HTTP
2. Reference Project
   1. Reference Service Project
      1. ProcessBack
         1. Supaya Back Project dapat di load karena tidak ada project service (Service Upload ada di library)
3. Reference semua collection assembly baik Back maupun Common
   1. Semua assembly yg ada di back directory sudah termasuk common directory
   2. Reference semua assembly di directory “RuntimeLibrary/Back” semua assembly yg mempunyai awalan R\_ saja
4. Menggunakan Private realta Nuget
   1. Pakai Realta.BackCollectionNuget
5. Perubahan pada Class Program.cs
   1. using R\_APIStartUp
   2. Untuk penggunaan StartUp ada 5 hal yg dapat di-disable
      1. Authentication
      2. Swagger
      3. GlobalException
      4. Context
      5. Database
   3. Karena kita akan menggunakan Exception, Swagger dan Context, Database maka kita akan melakukan disable untuk point yg lain
   4. Text

      Description automatically generated
6. Penambahan pada AppSetting
   1. Karena kita akan berhubungan dengan database, maka kita akan tambahkan konfigurasi database di bagian AppSettings.json
   2. Kita dapat melakukan setting untuk beberapa database di bagian R\_DBConfigs, semua nya akan dijelaskan pada dokumentasi R\_Db
   3. Graphical user interface, text, application

      Description automatically generated

### ProcessClientConsole

Merupakan project client front end untuk membantu simulasi 3 model bisnis proses.

1. Penamaan dan Type Project
   1. Penamaan ProcessClientConsole
   2. Language C#
   3. Platform open (Linux)
   4. Type project Console App
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APIClient
      1. Dipakai untuk melakukan request API
   2. Assembly R\_APICommonDTO
      1. Dipakai untuk Common Exception and Result
   3. Assembly R\_CommonFrontBackAPI
      1. Dipakai untuk common Class DTO R\_KeyValue
   4. Assembly R\_AuthenticationEnumAndInterface
      1. Menjadi dependency saat melakukan process client
   5. Assembly R\_ContextEnumAndInterface
      1. Menjadi dependency saat melakukan process client
   6. Assembly R\_ProcessAndUploadFront
      1. Merupakan main assembly untuk melakukan process client
   7. Project ProcessCommon
   8. Packet Nuget Realta.FrontCollectionNuGet
      1. Digunakan untuk mendukung assembly front dan common
      2. Nuget ini sudah termasuk Realta.CommonCollectionNuGet
   9. Package ObjectDumper.NET versi 3.5.2
      1. Paket Nuget tsb dipakai untuk mempermudah melakukan display isi dari class di console

**Class Program**

Merupakan class yg dipakai untuk menjalankan main static method.

Pada class tsb kita dapat mendeklarasikan variable dengan scope class yg nantinya dapat dipakai oleh semua function yg akan ada di dalam class tsb, berikut ini adalah gambaran besar class tsb



Variable

1. loHttpClient
   1. Type HttpClient
   2. Merupakan variable HttpClient yg dibutuhkan untuk meng-create R\_HTTPClient
2. loClient
   1. Type R\_HttpClient
   2. Dipakai untuk melakukan request ke API, class ini untuk mempermudah request berdasarkan nama dari instant nya.

#### Class R\_ProcessAndUploadClient

1. Type R\_ProcessAndUploadClient
2. Merupakan class yg dipakai untuk melakukan 3 model process

**Main sub**

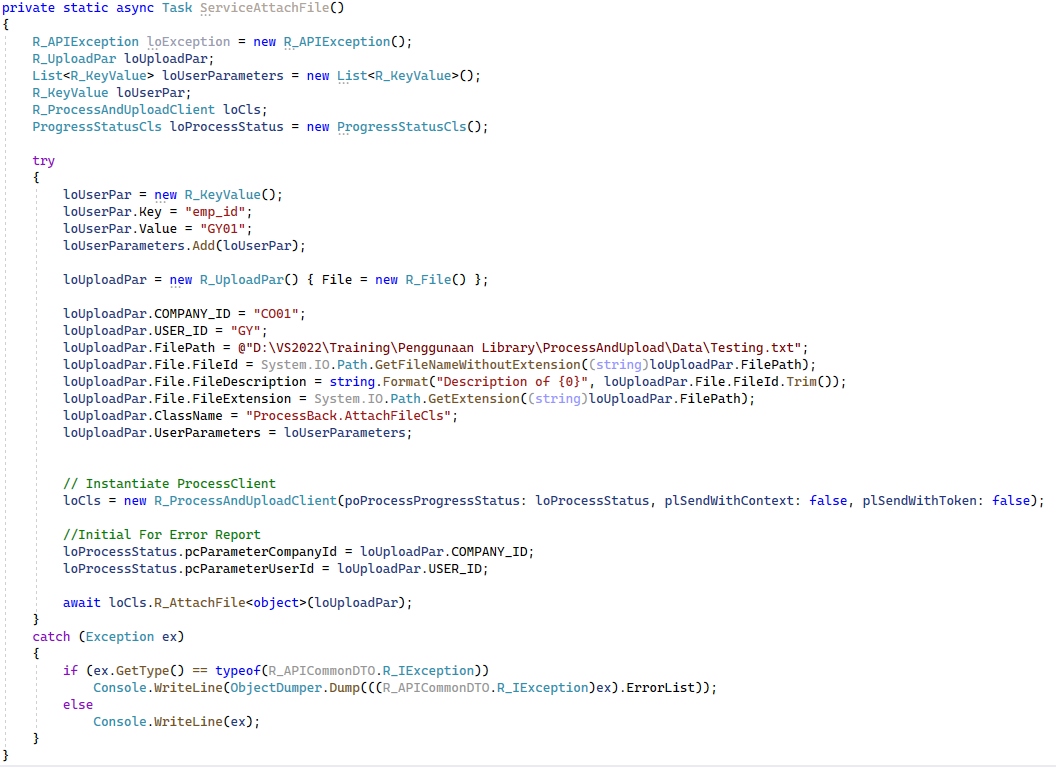
Merupakan static main console sub, dimana sub tsb akan dijalankan oleh project console.

Kita akan membuat 3 function static untuk melakukan implementasi 3 process tsb.

Main program akan melakukan initialisasi semua requirement general yang dibutuhkan sebelum menjalankan process menurut fungsi nya masing2

**Function ServiceAttachFile**

Merupakan function untuk meng-implementasikan Attachment File, function ini akan mengembalikan Task karena kita akan memanggil API secara asynchronous.



Untuk melakukan Proses Attachment File yg besar, maka diperlukan

##### Function R\_AttachFile

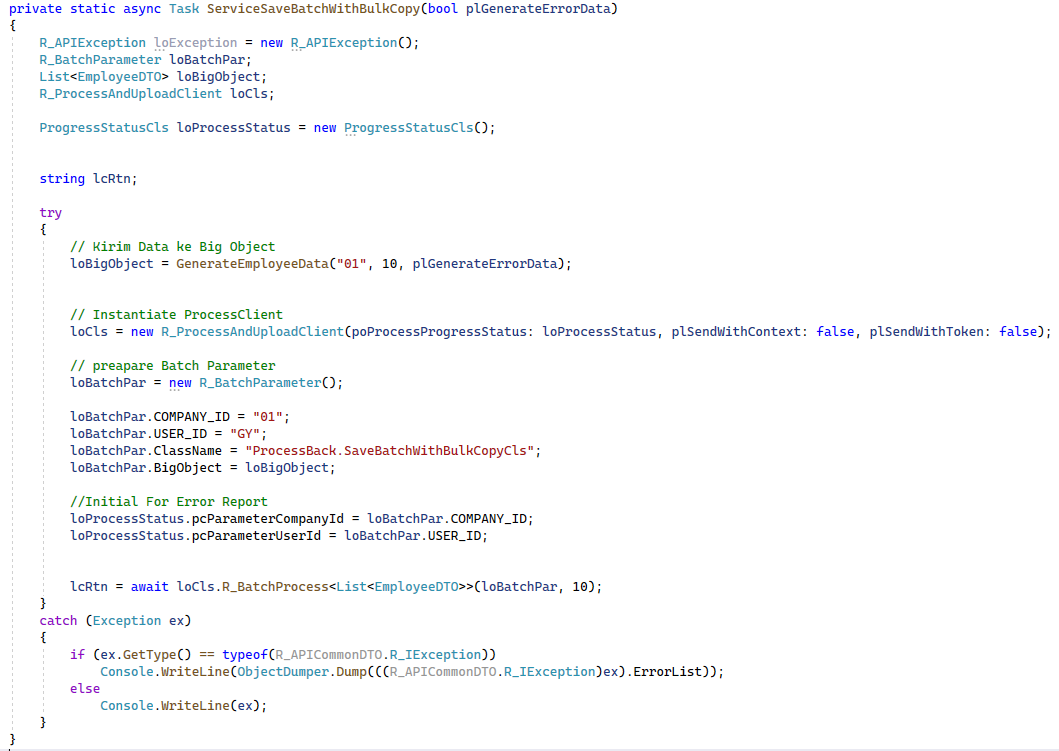
1. R\_AttachFile<BO>( poUploadParameter)
   1. Merupakan function yg dipakai untuk melakukan penyimpanan attachment file yg besar supaya dapat dikirim ke Back dan disimpan ke Database
   2. Biasanya bisa berupa dokumen atau report pdf yg besar.
   3. BO
      1. Type bebas asal inherits dari class
      2. BO merupakan type dari Big Object (salah satu property parameter) yg dapat dipakai sebagai data tambahan atau pengganti File kalau File tsb sdh berupa data Byte Array
   4. Parameter poUploadParameter
      1. Type R\_UploadPar
      2. Dapat dilihat di Common Library Part
   5. Return
      1. Task

**Function ServiceBatchProcess dan ServiceSaveBatchWithBulkCopy**

Merupakan function untuk meng-implementasikan Batch Process Data dan Save Bulk Data , function ini akan mengembalikan Task karena kita akan memanggil API secara asynchronous.

Kedua function ini akan memakai function R\_BatchProcess





Untuk melakukan Batch Process Data maka diperlukan

##### Function R\_BatchProcess

1. R\_BatchProcess<BO>(poBatchParameter, pnFinalStep)
   1. Merupakan function yg dipakai untuk melakukan Batch Process
   2. Sebenarnya function ini juga dapat dipakai untuk melakukan Bulk Saving ke database tergantung implementasi dari back class nya
   3. BO
      1. Type bebas asal inherits dari class
      2. BO merupakan type dari Big Object property yg akan dipakai sebagai data yg akan diproses di back
   4. Parameter poUploadParameter
      1. Type R\_UploadPar
      2. Dapat dilihat di Common Library Part
   5. Return
      1. Task<string>
      2. Hasil nya berupa Process Id yg berisi GUID
      3. Diperlukan Task untuk pemanggilan secara asynchronous

##### Function R\_GetErrorProcess

Fungsi ini untuk mendapatkan semua bisnis proses Exception yang terjadi dalam proses berdasarkan proses Id

1. R\_GetErrorProcess(poErrorStatusParam)
   1. Parameter
      1. poErrorStatusParam
         1. Merupakan parameter key yg dipakai untuk mendapatkan semua pencatatan validasi terhadap proses dengan Key proses tertentu
         2. Type R\_UploadAndProcessKey
            1. Lihat Common Library Part
   2. Return
      1. Type List<R\_ErrorStatusReturn>
         1. Lihat Common Library Part

#### Class Progress Status

Class tsb akan dipakai untuk melakukan pelaporan status dan error pada semua proses yang menggunakan class tsb.

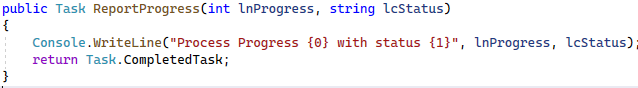
Class ini akan dipakai sebagai salah satu parameter pada waktu melakukan initialisasi dari class R\_ProcessAndUploadClient.

##### Implementation

Class ini akan meng-implement interface R\_IProcessProgressStatus yang akan mempunyai 3 method

###### Function ReportProgress(pnProgress, pcStatus)

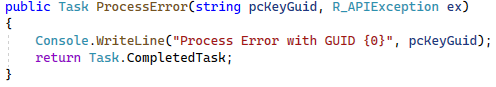
Merupakan fungsi untuk melaporkan progress dari setiap proses



1. Parameter
   1. pnProgress
      1. Type int
      2. Merupakan parameter yg berisi informasi untuk menampilkan progress step dari suatu proses
   2. pcStatus
      1. Type string
      2. Merupakan parameter yg berisi informasi status
2. Return
   1. Task
   2. Kalau didalam function ini kita tidak memanggil suatu method secara asynchronous, maka kita harus mengembalikan Task.CompletedTask

###### Function ProcessError(pcKeyGuid, ex)

Merupakan fungsi untuk melaporkan error yg terjadi kalau dalam proses terjadi error



1. Parameter
   1. pcKeyGUID
      1. Type string
      2. Merupakan proses ID dari proses tsb
   2. Ex
      1. Type R\_APIException
      2. Merupakan Exception yg kita dapat kesalahan proses yg terjadi
2. Return
   1. Task
   2. Kalau didalam function ini kita tidak memanggil suatu method secara asynchronous, maka kita harus mengembalikan Task.CompletedTask

###### Function ProcessComplete(pcKeyGuid, poProcessResultMode)

Merupakan fungsi yg dipakai untuk melaporkan kalau proses sudah selesai tanpa error

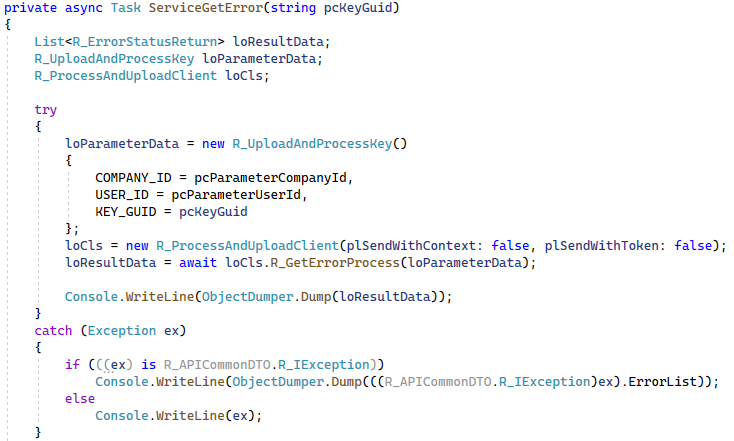


1. Parameter
   1. pcKeyGUID
      1. Type string
      2. Merupakan proses ID dari proses tsb
   2. poProcessResultMode
      1. Type enumeration eProcessResultMode
         1. eProcessResultMode.Success
            1. Menandakan proses selesai dengan sukses
         2. eProcessResultMode.Fail
            1. Menandakan proses selesai tapia da kegagalan bisnis proses dari proses tsb, mungkin dikarenakan adanya validasi bisnis proses

##### Function Helper

###### Function ServiceGetError

Merupakan function private yg dipakai sebagai helper untuk mendapatkan semua Exception dari validation bisnis proses



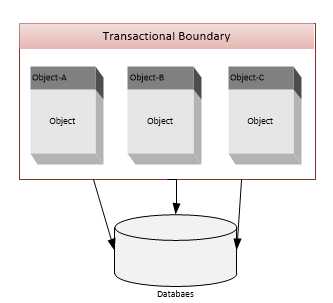
##### Property

Ada 2 property public dengan scope class yg merupakan property untuk melengkapi Key dari parameter Key R\_UploadAndProcessKey

1. pcParameterCompanyId
   1. Type string
   2. Berisi informasi Company Id dari suatu proses
2. pcParameterUserId
   1. Type string
   2. Berisi informasi User Id dari suatu proses

# Transaction Scope

Transaction Scope digunakan untuk memberikan scope atau boundary terhadap beberapa fungsi atau object yg melakukan manipulasi terhadap data di database.



Biasanya manipulasi data di database akan melakukan manipulasi data dari beberapa record dari sebuah table atau lebih.

Maka kegunaan dari transaction scope ini adalah

untuk melindungi kesatuan dari data, misalkan kita mau memindahkan data dari 1 table ke table lain

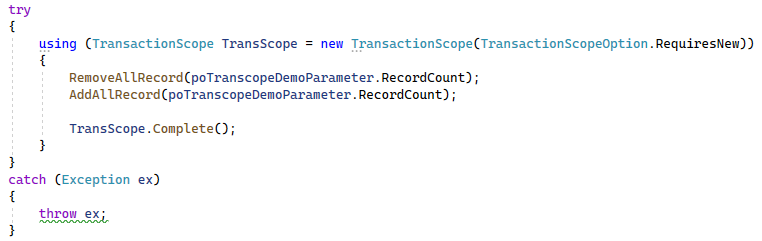
Kita harus memastikan data berpindah utuh dari table yg 1 ke table yg lain, sehingga misalkan terjadi error saat memindahkan data, jangan sampai data sudah dihapus dari table yg 1 tapi belum sempat dipindahkan ke table yg lain, sehingga data menjadi corrupt.

## Class TransactionScope

Untuk melakukan Transaction Scope, maka .NET Framework menyediakan service TransactionScope dari namespace System.Transactions.

Untuk menggunakan transaction Scope, maka kita dapat menggunakan pembatasan using block dengan menggunakan option saat melakukan instantiate

1. TransactionScopeOption.RequiresNew
   1. Memulai dengan Transaction Scope yg baru
2. TransactionScopeOption.Required
   1. Mengikuti TransactionScope diatas nya, kalau tidak ada, maka akan membentuk yg baru
3. TransactionScopeOption.Suppress
   1. Tidak mengikuti TransactionScope
   2. Biasa nya block ini ada didalam block Required/RequiresNew, jadi ada bagian yg tidak masuk dalam transaction scope.
   3. Biasa dipakai untuk menulis log, dimana log tetap terjadi walaupun rollback



### Cara kerja TransactionScope

1. Transaction Scope Block akan ada didalam Try Block
2. Method2 yg masuk dalam TransactionScope akan ada di dalam block using
3. Kalau tidak ada error, maka kita harus menjalankan Complete supaya transaction block tsb di-commit
4. Kalau ada Exception, maka akan keluar ke Catch block dan otomatis akan me-rollback Transaction Scope
5. Semua Connection untuk masing2 method harus sudah ditutup didalam Block TransactionScope

## Skenario

Kita akan membuat API untuk melakukan pemindahan record dari table yg 1 ke table yg lain dengan mencatat nya di table log.

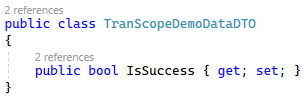
Untuk melakukan demo transaction scope yg meliputi 3 behavior

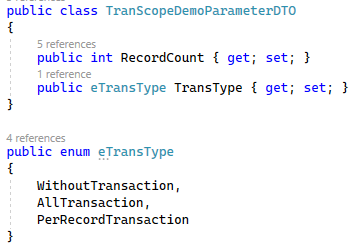
1. Tanpa menggunakan transaksi
2. Menggunakan transaksi dalam suatu collection data secara bersamaan
3. Menggunakan transaksi untuk per record

## TranScopeCommon

1. Penamaan dan Type Project
   1. Penamaan TranScopeCommon
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Standard 2.1
2. Reference Assembly R\_APICommonDTO
   1. Digunakan untuk Result DTO

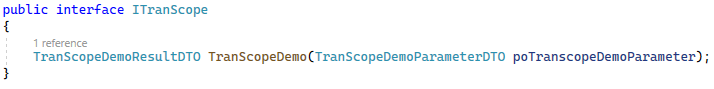
### Entity

1. TranScopeDemoDataDTO
   1. Hanya berisi data untuk memperlihatkan apakah method yg dijalankan sukses atau tidak
   2. 
2. TranScopeDemoResultDTO
   1. Merupakan Result dari method, result ini mengandung data yg bertype TranScopeDemoDataDTO dan meng-inherits dari R\_APIResultBaseDTO
   2. Graphical user interface, text

      Description automatically generated
3. TranScopeDemoParameterDTO
   1. Merupakan parameter dari method
   2. Property
      1. RecordCount
         1. Meng-identifikasikan berapa jumlah record yang akan dipindahkan
      2. TransType
         1. Type dari cara pemindahan
   3. Enumeration eTransType
      1. WithoutTransaction
         1. Method dijalankan tanpa transaksi
         2. Kalau ada error maka record yg di-delete akan hilang tapi record yg harus ditambahkan tidak terjadi
      2. AllTransaction
         1. Method dijalankan menggunakan transaksi untuk suatu collection sekaligus
         2. Kalau ada error maka seluruh record pada table yg akan di-delete akan dikembalikan seluruh nya
      3. PerRecordTransaction
         1. Method dijalankan menggunakan transaksi untuk per record
         2. Kalau ada error, maka hanya record yg mengalami error yg tidak jadi di-delete
   4. 

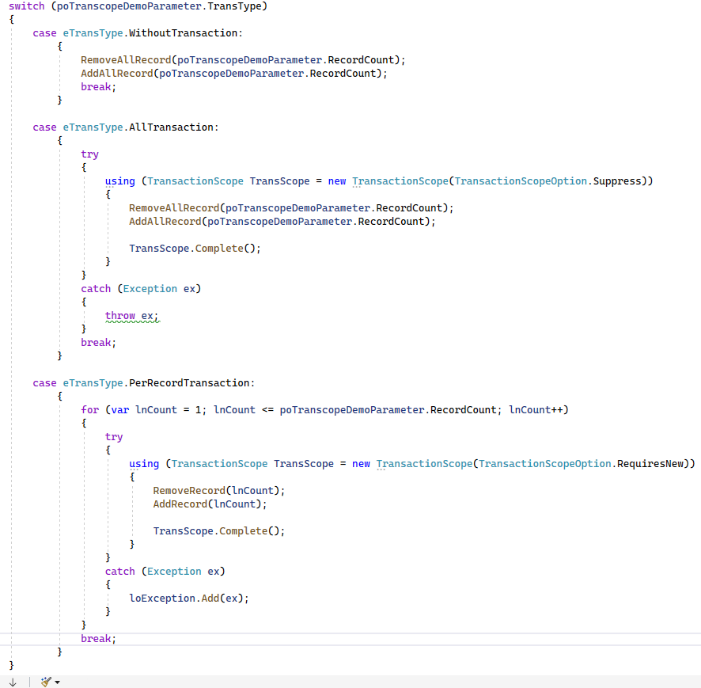
### Interface

Untuk interface kita hanya mempunyai sebuah fungsi yaitu TranScopeDemo



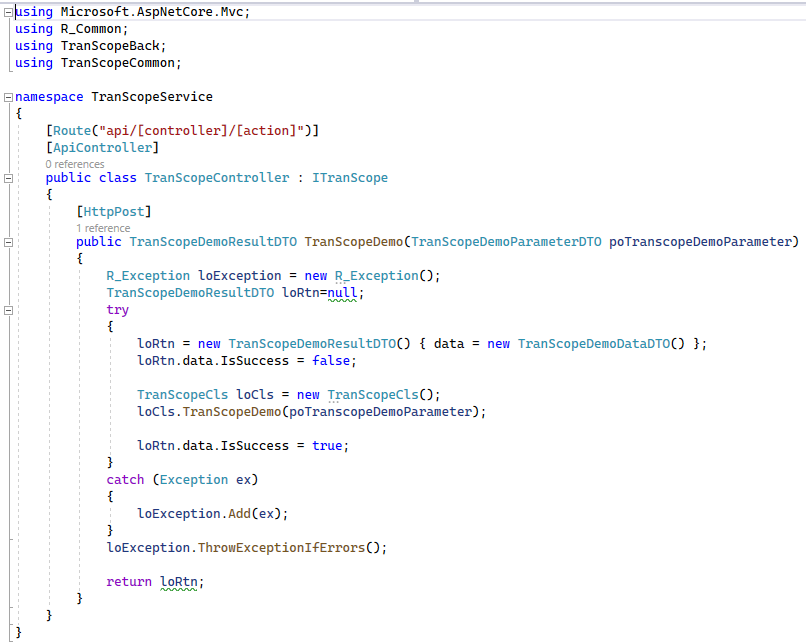
## TranScopeBack

Merupakan Project yg berupa kumpulan bisnis process untuk menghasilkan data yg diminta oleh Controller API, dimana sudah dituangkan fungsi2 nya pada Interface yg ada di common.

1. Penamaan dan Type Project
   1. Penamaan TranScopeBack
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Reference
   1. Assembly R\_APICommon
      1. Dipakai untuk utility umum
      2. Dipakai untuk Exception framework
   2. Assembly R\_APIBackEnd
      1. Dipakai untuk berhubungan dengan database
   3. Project TranScopeCommon
3. Penamaan Class
   1. Sesuai dengan project, kata Back diubah menjadi Cls
   2. TranScopeCls
   3. Mempunyai fungsi untuk menghasilkan data sesuai dengan interface yg akan diimplementasikan oleh controller di project Service
4. 
5. Method
   1. TranScopeDemo
      1. Merupakan method utama bersifat public yang nantinya akan menjalankan method2 private tergantung dari Enumeration pada parameter
      2. 
   2. RemoveAllRecord(int pnCount)
      1. Merupakan method private yg dipakai untuk melakukan delete record sebanyak pnCount parameter
      2. Pada method ini melakukan 2 hal
         1. Mendelete sejumlah record pada TestCustomer table sesuai dengan jumlah parameter pnCount
         2. Menambahkan record2 yg di delete ke dalam Table TestCustomerLog untuk dicatat.
      3. Untuk penambahan pada table TestCustomerLog di dalam scope option Suppress yg artinya penambahan ke table tsb tidak ikut pada scope transaction
      4. 
   3. Method AddAllRecord(int pnCount)
      1. Merupakan method private yg dipakai untuk melakukan penambahan record sebanyak pnCount parameter
      2. Pada method ini menambahkan sejumlah record pada TestCopyCustomer table sesuai dengan jumlah parameter pnCount
      3. Error kita generate secara sengaja pada count kelipatan 3
      4. 
   4. Method RemoveRecord(int pnCount)
      1. Merupakan method private yg dipakai untuk melakukan delete single record pada record yg id nya sesuai dengan parameter pnCount
      2. Pada method ini melakukan 2 hal
         1. Mendelete single record pada TestCustomer table yg id nya sesuai dengan parameter pnCount
         2. Menambahkan record yg di delete ke dalam Table TestCustomerLog untuk dicatat.
      3. Untuk penambahan pada table TestCustomerLog di dalam scope option Suppress yg artinya penambahan ke table tsb tidak ikut pada scope transaction
      4. 
   5. Method AddRecord(int pnCount)
      1. Merupakan method private yg dipakai untuk melakukan penambahan record single yg id nya sesuai dengan pnCount parameter
      2. Pada method ini menambahkan single record pada TestCopyCustomer table yg id nya sesuai dengan parameter pnCount
      3. Error kita generate secara sengaja pada count kelipatan 3
      4. 

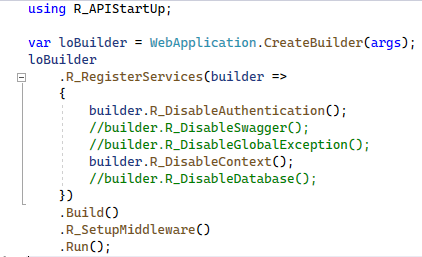
## TranScopeService

Merupakan project service yg berisi sekumpulan controller yg akan mendukung API sesuai dengan interface nya

1. Penamaan dan Type Project
   1. Penamaan TranScopeService
   2. Language C#
   3. Platform open (Linux)
   4. Type project Library
   5. Framework Net 6.0
2. Edit project file dan tambahkan
   1. <ItemGroup>
   2. <FrameworkReference Include="Microsoft.AspNetCore.App" />
   3. </ItemGroup>
3. Reference Assembly
   1. R\_APICommon
      1. Untuk ExceptionFramework
4. Reference Project
   1. TranScopeCommon
   2. TranScopeBack
5. Buat Class Controller
   1. Class Controller tergantung dari banyak nya class Back
   2. Buat dengan penamaan Business Process ditambahkan dengan Controller
      1. TranScopeController
   3. Using
      1. R\_Common
      2. Microsoft.AspNetCore.Mvc
   4. Tambahkan annotation di class
      1. [Route("api/[controller]/[action]")]
      2. [ApiController]
   5. Inherit dan interface
      1. Inherit ControllerBase
      2. Interface ITranScope
   6. 

## API\_TranScope

Merupakan project API TranScope

1. Penamaan dan Type Project
   1. Penamaan API\_TranScope
   2. Language C#
   3. Platform Open (Linux)
   4. Type project ASP.NET Core Web API
   5. Framework NET 6.0
   6. HTTP
2. Reference Project
   1. Reference Service Project
      1. TranScopeService
3. Reference semua collection assembly baik Back maupun Common
   1. Semua assembly yg ada di back directory sudah termasuk common directory
   2. Reference semua assembly di directory “RuntimeLibrary/Back” semua assembly yg mempunyai awalan R\_ saja
4. Menggunakan Private realta Nuget
   1. Pakai Realta.BackCollectionNuget
5. Perubahan pada Class Program.cs
   1. using R\_APIStartUp
   2. Untuk penggunaan StartUp ada 5 hal yg dapat di-disable
      1. Authentication
      2. Swagger
      3. GlobalException
      4. Context
      5. Database
   3. Karena kita akan menggunakan Exception, Swagger dan Database, maka kita akan melakukan disable untuk point yg lain
   4. 
6. Penambahan pada AppSetting
   1. Karena kita akan berhubungan dengan database, maka kita akan tambahkan konfigurasi database di bagian AppSettings.json
   2. Kita dapat melakukan setting untuk beberapa database di bagian R\_DBConfigs, semua nya akan dijelaskan pada dokumentasi R\_Db
   3. Graphical user interface, text, application

      Description automatically generated