مطالب

[مقدمه 2](#_Toc144238622)

[Terminology 2](#_Toc144238623)

[DTO 2](#_Toc144238624)

[View Model 2](#_Toc144238625)

[Command/Query 2](#_Toc144238626)

[Blazor Component/Page 2](#_Toc144238627)

[کلاس partial 2](#_Toc144238628)

[بخشهای Framework 3](#_Toc144238629)

[Code Generator 3](#_Toc144238630)

[Security Manager 3](#_Toc144238631)

[CQRS Engine 3](#_Toc144238632)

[License Manager 3](#_Toc144238633)

[Base Class Library 3](#_Toc144238634)

[ستاپ Framework 4](#_Toc144238635)

[Connection String 4](#_Toc144238636)

[Show Toast Notifications 4](#_Toc144238637)

[Source Code Folder Paths 4](#_Toc144238638)

[Code Generator 5](#_Toc144238639)

[Back-End 5](#_Toc144238640)

[پنجره Model Details 5](#_Toc144238641)

[پنجره CQRS – Query Details 7](#_Toc144238642)

[پنجره CQRS – Command Details 9](#_Toc144238643)

[پنجره Custom Full CRUD 9](#_Toc144238644)

[Front-End 9](#_Toc144238645)

[پنجره Blazor Component Generator 9](#_Toc144238646)

[پنجره Blazor Page Code Generator 13](#_Toc144238647)

[پنجره Functionality 14](#_Toc144238648)

# مقدمه

در این framework کدهای back بر پایه CQRS نوشته شده‌اند و نسخه .NET آن، 6.0 میباشد. همانطوری که در مستندات قبلی اشاره شد، در معماری نرم‌افزار چندین روش برای سازماندهی ماژولهای یک نرم‌افزار وجود دارد. مشهورترین آنها، Service Base و CQRS است. به دلیل اینکه قبلا مزایا و معایب هریک بیان شده، از تکرار مکررات پرهیز میکنم.

کدهای front ها بر پایه Blazor نوشته شده‌اند. Blazor یک framework مبتنی بر وب است که توسط مایکروسافت ایجاد شده است. با استفاده از Blazor، می‌توانید برنامه‌های وب را با استفاده از زبان‌های برنامه‌نویسی مانند #C به جای javascript ایجاد کنید. دلیل نامگذاری این فریم‌ورک "Blazor" نیز از ترکیب واژه‌های "Browser" و "Razor" به دست آمده است. این framework به توسعه‌دهندگان دات‌نت اجازه می‌دهد برنامه‌های وب مدرن و پیشرفته‌ای را با بسازند که می‌توانند به صورت بومی در مرورگر اجرا شوند. Blazor WebAssembly، یک نسخه از Blazor است که به صورت کاملاً مستقل از سرور، در مرورگر اجرا می‌شود. و تا حد زیادی برنامه‌نویس را از javascript بی‌نیاز میکند. مایکروسافت امیدوار است بطور کامل javascript را کنار بزند.

# Terminology

در نرم‌افزار، یک سری کلمات فنی وجود دارند که به دلیل استفاده از آنها در MES-Infa، آنها را مختصرا توضیح میدهم

## DTO[[1]](#footnote-1)

کلاسهایی هستند که تا سطح امکان، فقط از fieldها و propertyها تشکیل شده و عموما برای جابجایی داده‌ها بین لایه‌ها مختلف نرم‌افزار استفاده می‌شوند. این کلاسهای به دلیل سبک بودن، برای serialization ایده‌آل هستند. از این کلاسها در لایه سرویس و لایه‌های CQRS استفاده میشود.

## View Model

کلاسهایی که بیشتر حاوی اطلاعاتی هستند که در UI نمایش داده میشوند و شباهتهایی به DTO دارند با این تفاوت که پیچیده‌تر هستند و میتوانند شامل سورس‌های عملیاتی و instance کلاسهای دیگر نیز باشند.

## Command/Query

احتمالا نیاز به توضیح ندارند زیرا پایه اصلی معماری CQRS هستند. به هر یک از این بخشها، اصطلاحا یک segregation گفته میشود.

## Blazor Component/Page

این کلاسها نیز نیاز به توضیح ندارند. تنها نکته اینکه page، در این framework، شامل مجموعه‌ای از componentهاست.

## کلاس partial

در حالت کلی، کلاس partial کلاسی است که هم نام کلاس دیگری باشد و هر دو دارای پیشوند partial باشند. ولی در Code Generator،تمام کلاسها partial هستند. منتهی در terminology این framework، کلاس partial، کلاسیست که توسط generator، ایجاد شده و نباید توسط برنامه‌نویس تغییر کند. بنابراین، لطفا توجه داشته باشید که کلاسی که نام فایل، patial دارد، نباید دستکاری شود، زیرا در generateهای بعدی overwrite خواهد شد. ولی کلاسهایی که نام فایل partial ندارند، فقط در صورتی توسط generator، ایجاد میشوند که در فولدر، از قبل وجود نداشته باشند.

# بخشهای Framework

این Framework از بخشهای زیر تشکیل شده:

## Code Generator

در این بخش، با کمک برنامه داخلی Infra Code Generator، شما میتوانید نزدیک به کل کدهای Back-end و Front را ایجاد نمایید. بخشهای باقیمانده از کد که توسط ICG[[2]](#footnote-2) ایجاد نمیشوند، business برنامه هستند که در لحظه تولید کد، فقط برنامه‌نویس میتواند آنها را بنویسد و عملا امکان تولید آن توسط ICG وجود ندارد. کدهایی که در این قسمت ایجاد میشود، بر پایه CQRS بوده و شامل کدهای مربوط به CRUD بر پایه اطلاعات خام و یا اطلاعات schemaی موجود در database است.

## Security Manager

در این بخش، مدیریت دسترسی به اجرای سورس‌ها در داخل برنامه پیاده‌سازی شده است. به این نحو که کد ایجاد شده، شامل یک شناسه یکتا است که توسط کاربر ICG، به کد داده میشود. این کد در زمان فراخوانی command یا query، توسط یک middleware به نام Security Manager، چک میشود. در صورتی که کاربر یا برنامه‌ای که در حال حاضر به برنامه لاگین کرده، به این بخش از کد دسترسی نداشته باشد، پیش از اجرای command یا query مربوطه، اصطلاحا intercept شده و access denied میشود.

## CQRS Engine

با R&D که در زمان تحلیل برنامه کردم، متوجه شدم که تمام CQRS Engineهای موجود برای .NET، مزایا و معایب خاص خود را دارند و به این دلیل که میخواستم هیچ مشکلی در engine وجود نداشته باشد، همچنین در صورت نیاز به extend، امکان آن وجود داشته باشد، از یکی از پروژه‌های open-source مایکروسافت ایده گرفته و با برطرف کردن کاستی‌های اختصاصی آن برای برنامه MES، آنرا بطور کاملا، مجددا نوشتم. در حال حاضر، کل سورسهای تولید شده، بر پایه همین engine، تولید میشوند. البته امکان تغییر این رویه به سادگی ممکن است. یعنی میتواند از هر engine دیگری نیز استفاده نمود.

## License Manager

در واقع یک middleware ساده است که از لحاظ عملیاتی کاری انجام نمیدهد. منتهی با قرار گرفتن در مسیر ورودی و خروجی‌ها، تمام اتفاقات برنامه را رصد میکند. متعاقبا میتوان کدهای مربوط به licensing را در آن نوشت. مثلا با استاندارد JWT، آن را کامل کرد و به عنوان مثل، تعداد کاربران برنامه را محدود نمود.

## Base Class Library

جهت راحتی کار developer، تعداد بسیار زیادی کلاسهای کاربردی، اعم از serviceها، helperها، practical design patternها، collectionها با کاربری اختصاصی، exceptionها، loggerها، handlerها و مقدار زیادی دیگری از کلاسها و ماژولها در این بخش گنجانده شده‌اند در صورت استفاده، تمام کارهای عمومی ساده و پیچیده، همچنین و تکراری برنامه‌نویس را به سهولت انجام میدهند. بخشهای زیادی از BCL دارای unit test هستند که هم از لحاظ سلامتی، صحت‌سنجی شوند و هم روش استفاده از آنها، در قالب یک sample، و البته با جزییات استفاده، قابل مشاهده باشند.

در ادامه راهنمای استفاده از کل framework را مینویسم.

# ستاپ Framework

بسیار ساده است و منوی Tools > Settings، قابل انجام است.

## Connection String



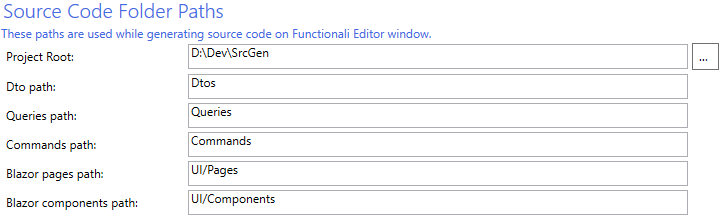
بخش اول، مربوط به Connection String است و جهت اتصال framework به دیتابیس استفاده میشود.

## Show Toast Notifications



خود من، در حین کار با برنامه، از تعداد زیاد پیامهایی که میداد، خسته شدم. بخشی از آنها را حذف کردم. ولی بخشی نیز اجبارا باید نمایش داده بشوند. پیامهایی که صرفا جهت اطلاع هستند و ارزش دیگری ندارند و نباید کاربر را دچار وقفه کنند را توسط Tost Notification نمایش میدهم که کاربر مجبور به زدن یک button برای بستن پنجره‌ای خاص نشود. ولی حتی پس از مدتی، این پیامها نیز خسته‌کننده شدند. بنابراین تصمیم گرفتم نمایش آنها را اختیاری کنم. جهت مشاهده یک نمونه Toast Notification، روی  کلیک کنید.

## Source Code Folder Paths



در این بخش، شما باید محلی را جهت ذخیره سورهای تولید شده توسط Functionality را ست نمایید. بخش Root Project یک فولد فیزیکی از هارددیسک است که باید قابل دسترسی باشد. (اگر وجود نداشته باشد، توسط framework ایجاد خواهد شد.) مابقی بخشها، نام فولدرهایی هستند که با ترکیب با Project Root معنادار میشوند و مورد استفاده قرار میگیرند. پس نباید یک آدرس کامل یک فولدر باشند. این فولدرها نیز در صورت عدم وجود، ایجاد توسط framework، میشوند. توجه داشته باشید که این بخش فقط توسط Functionality مورد استفاده قرار میگیرد و در بخشهای دیگر کاربرد ندارد.

نکته مهم اینکه حتما در پایان کار، کلید  را بزنید.

# Code Generator

بزرگترین بخش MES-Infra است که سورهای مورد نیاز برنامه‌نویس را ایجاد و متادیتاهای آن را نگهداری میکند.

بطور کلی، این برنامه، DTOها، View Modelها، Query Handlerها و Command Handlerها، را تولید میکند. در ادامه به شرح جزییات آن میپردازم.

## Back-End

### پنجره Model Details

جهت تولید یک segregation[[3]](#footnote-3)، در ابتدا باید DTOهای مربوطه ایجاد شوند. DTOهای مورد نیاز، شامل Parameterها یک اجرا segregation و Result آن است. برای این منظور باید به پنجره Model Details برویم. (Back-End > Manage Models…). این پنجره، در ابتدا، کل دیتابیس را scan میکند تا لیست جداول را جهت ایجاد DTO، داشته باشد.

#### DTOی bound و unbound

در این پنجره، شما میتواند DTOهای bound و unbound ایجاد کنید (). توسط این دو نوع DTO در اتصال آنها به یک table است. به عبارت دیگر، اگر قصد دارید بر روی یک یا چند table، عملیات CRUD انجام دهید، باید یک DTOی bound ایجاد کنید. ولی اگر قصد دارید یک business را پیاده‌سازی کنید که مستقیما با جداول دیتابیس انطباق ندارد یا پیچیدگی آن از جداول دیتابیس بیشتر است، میتوانید از DTOی unbound استفاده کنید. با آیکن  میتوانید DTOی bound ایجاد کنید. با آیکن  میتوانید DTOی unbound ایجاد کنید.

#### Explorer

در بخش explorer دو تب وجود دارد.

##### تب Tables

لیست جداول دیتابیس را میتواند مشاهده کنید که در ایجاد DTOی unbound، کاربرد دارد. این جداول مستقیما از دیتابیس خوانده میشوند. با انتخاب یک جدول و کلیک بر روی آیکن ، یک DTOی bound، ایجاد میشود.

##### تب Models

لیست مدل‌ها (در اینجا، همان DTOها) را مشاهده میکنید. جهت ویرایش یک DTO، پس از انتخاب آن، روی  کلیک کنید.

#### جزییات DTO

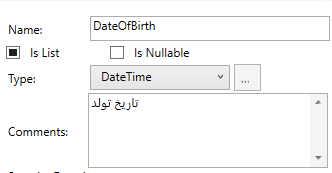
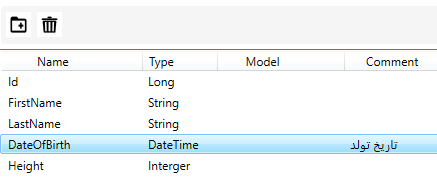


نام جدول با پسوند Dto، نام DTO را ایجاد میکند. توجه داشته باشید که تمام DTOها، باید شامل نام، module و namespace باشند. به دلیل اینکه نام module یکبار برای همیشه ایجاد میشود و CRUDی روی آن نیست، من فرض را بر این گرفتم که نام moduleها، در دیتابیس فیکس شده‌اند. دقت در انتخاب نام module بسیار مهم است. زیرا فقط DTOهای هر ماژول میتوانند در segregationهای آن ماژول استفاده شوند. در صورتی که DTOای را در یک module ایجاد کنید، در هنگام ایجاد segregation در یک module دیگر، نام DTO را نمیتوانید ببینید. بخش Comment، صرفا جهت یادداشت است و در مراجعات بعدی مورد استفاده قرار میگیرد.

یک DTO یکی از حالات زیر باشد:

1. Params: ورودی segregation است. (توجه داشته باشید که در framework، کلید کلیدی Params به عنوان ورودی segregation است و حاوی اطلاعاتی است که مستقیما در segregation مورد استفاده قرار میگیرد ولی ParamDto، یک DTO کلی است که میتواند ورودی هر چیزی باشد، اعم از API یا به عنوان ورودی Params. یعنی API، ورودی خود را از body میگیرد و در به عنوان پارامتر سازنده کلاس Params، به segregation ارسال میکند.)
2. Result: خروجی segregation است.
3. View Model، به عنوان DataContext در pageهای Blazor مورد استفاده قرار میگیرد. (متعاقباً در این مورد بیشتر توضیح خواهم داد.)
4. Source، داخلی framework است و در framework، مورد استفاده قرار میگیرد. در قسمت Functionality این مورد را شرح خواهم داد. فقط این نکته را عرض کنم که Source، به عنوان منبع اطلاعات اولیه مورد نیاز Functionality است و به معنای Source Code نیست.

#### Properties

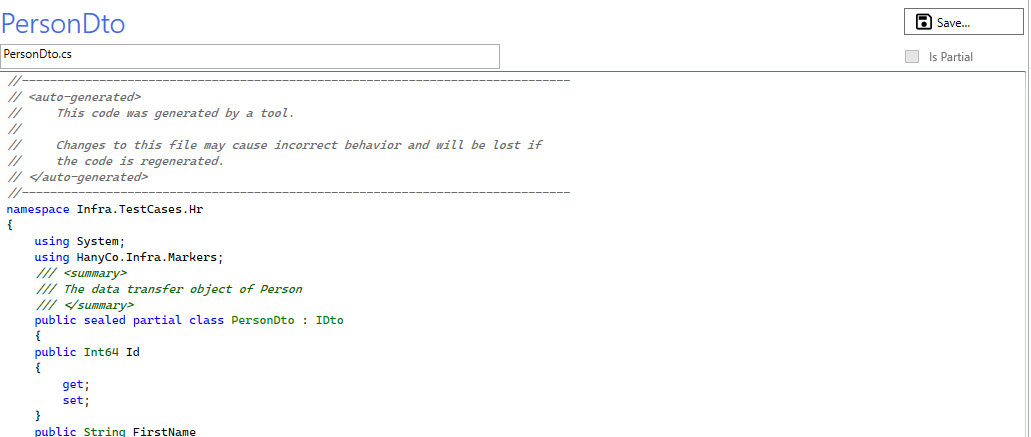
در این بخش، propertyهای DTO تعریف میشوند. در این قسمت، هیچ چیزی فیکس نیست و قابل تغییر است. شما میتوانید بسته به منطق برنامه، این propertyها را کم یا زیاد کنید، آنها را rename کنید و یا نوع آنها را تغییر دهید. بصورت پیش‌فرض، این اطلاعات، انطباق با اطلاعات جدول دیتابیسی دارد. Framework، نوع property را تبدیل به نوع .NETی کند. ولی اگر نتیجه مد نظر شما نبود، میتوانید آنرا تغییر دهید. همچنین میتوانید بنا به تمایل، property جدید به لیست اضافه کنید و نوع آنرا مشخص کنید. یکی از انواع propertyها، DTO است و زمانی کاربرد دارد که شما میخواهید یک جدول را با جدول دیگر relation زده و آنرا در قالب DTO، در یک DTO دیگر داشته باشید.

یک property میتواند list و یا nullable هم باشد.

#### Security Descriptor

فعالیت این بخش، در این مرحله بسیار ساده است. شما تعدادی کد امنیتی به DTO اضافه میکنید که بعدا توسط middleware مربوطه بررسی میشود که آیا کاربر جاری برنامه، دسترسی به این کدهای امنیتی دارد یا خیر. علت امکان تعدد کدها، امکان ورود از قسمتهای برنامه است. یعنی ممکن است این DTO در بخشی متعددی استفاده شود که بسته به محل، ممکن است کاربر دسترسی داشته یا نداشته باشد.

#### ایجاد و ذخیره سورس‌کد



در این بخش شما میتواند با آیکنهای اقدام به ایجاد و ذخیره کدها کنید. پس از ایجاد، شما میتوانید در تب Results، کدهای تولید شده را ببینید.

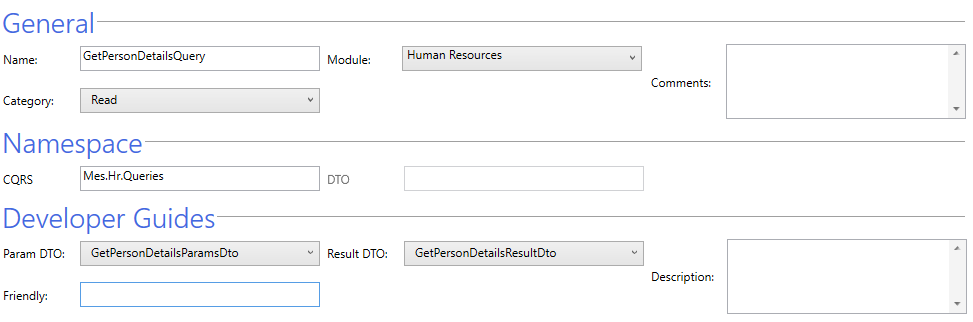
بصورت پیش‌فرض تمام DTOها، بطور کامل در اختیار framework هستند و برنامه‌نویس نباید اقدام به تغییر در آنها کند. زیرا در تولیدهای بعدی، ممکن است این تغییرات، overwrite شوند.

در هنگام ذخیره کد، framework، محل ذخیره را سوال میکند. سورس‌کد DTO، یک خروجی بیشتر نیست پس فقط یک فایل ذخیره خواهد شد.

### پنجره CQRS – Query Details

پس از تعریف DTOهای ورودی و خروجی یک segregation، بر حسب نوع آن segregation، میتوان اقدام به ایجاد آن کرد. در این پنجره، میتوانید segregationهای از نوع query تعریف کنید.

#### اطلاعات اولیه query



تعریف query بسیار ساده است. نام آن را انتخاب میکنید. عموما از دو حالت خارج نیست:

یا یک لیست رکوردهای موجود در یک جدول را میخواهیم که به دلیل احتمال زیاد بودن رکوردها، بهتر است فقط فیلدهای اصلی در DTO باشد، که اطلاعات زیادی از سمت Database Server به App Server و بعد از App Server به UI نرود. (این query را Get All می‌گویند)

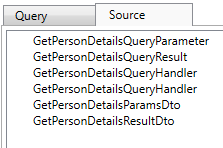
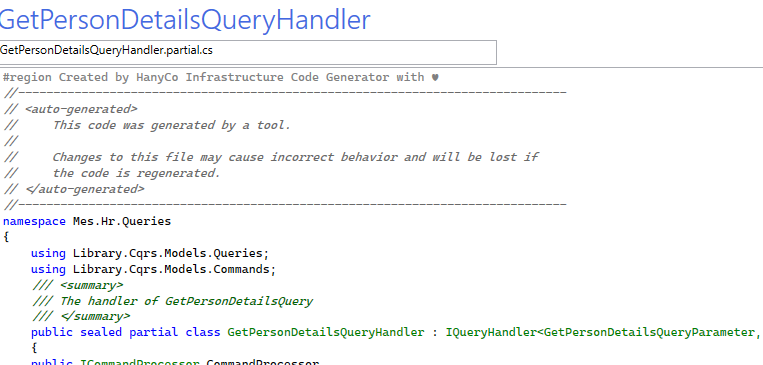
یا جزییات یک رکورد خاص را میخواهیم. در این صورت تمام اطلاعات آن رکورد باید دریافت و ارسال شود. (این query را Get By ID می‌نامند)

البته در برنامه، محدود به این DTO ها نیستیم و میتوانیم هر نوع DTO دیگری را نیز تعریف کنیم.

نیاز به یادآوری نیست که بر اساس module این query، بخش Param DTO و Result DTO، تکمیل میشوند.

جهت دسته‌بندی segregationها، category این query را انتخاب میکنیم. با این حال که یک query، فقط میتواند Read باشد، من دست برنامه‌نویس را باز گذاشته‌ام که در صورت تمایل، این query بتواند در دسته‌بندی دیگر هم ظاهر شود. در صورتیکه بخواهیم microserviceی کار کنیم، ممکن است بخواهیم برخی از segregationها را در دسته‌بندی‌هایی غیر از دست‌بندی اصلی، در container آپلود کنیم.

#### ایجاد و ذخیره سورس‌کد



طبق روال سابق کار میکند با این تفاوت که چندین فایل ایجاد میکند. نکته بسیار مهم این است که DTO ایجاد شده در مرحله قبل، مستقیما به query پاس داده نمیشود. بلکه query خودش پارامتر خودش را دارد و DTO ساخته شده در مرحله قبل، در قالب property به query داده میشود. علت آن این است که ممکن است بخواهیم در کنار DTO، اطلاعات دیگری را نیز به query پاس بدهیم.

نکته مهم اینکه بصورت پیش‌فرض، تمام queryها در transaction اجرا میشوند و در پایان اجرای query، آن transaction، عمدا rollback میشود که ناخواسته اطلاعاتی در دیتابیس ذخیره نشود. البته مواردی مثل logging و exception handling از این قانون تبعیت نمی‌کنند و میتوانند در دیتابیس خاص خودشان، ذخیره شوند.

در این بخش، تمام فایلها جز یک مورد partial هستند. تنها مورد، handler هست که انتظار میرود برنامه‌نویس بر اساس منطق برنامه، کد آن را بنویسد. البته برای کمک به برنامه‌نویس، تمام ابزارهای مورد نیاز، در اختیار گذاشته شده. جهت استفاده از segregationهای دیگر، processorهای مربوطه به query پاس داده میشود که در صورت نیاز، بتواند از commandها یا queryهای دیگر را نیز فراخوانی کند.

### پنجره CQRS – Command Details

این پنجره، از لحاظ ظاهر بسیار مشابه [پنجره CQRS – Query Details](#_پنجره_CQRS_–) است. بنابراین من موارد فوق را مجددا تکرار نمیکنم. فقط نکته قابل عرض اینکه، گرچه این دو پنجره ظاهرا شبیه به هم هستند، ولی تفاوتهای ساختاری مرا مجبور کرد که این دو پنجره را از هم جدا کنم.

### پنجره Custom Full CRUD

عملا منسوخ شد زیرا مفهوم Functionality تمام امکانات این پنجره را همراه با بسیاری امکانات دیگر، در اختیار قرار داد. پس من به بیان جزییات این پنجره نمی‌پردازم. و به زودی این پنجره را حذف خواهم کرد.

## Front-End

بخش فرانت، سورس‌های مربوط به Blazor را تولید میکند. در کل، این بخش به دو قسمت تقسیم میشود. یک بخش مربوط به componentهاییست که قرار است روی pageها قرار بگیرند، قسمت دیگر هم pageهاییست که قرار است بر اساس routeای که در همین بخش ایجاد میشود، در منو بنشینند.

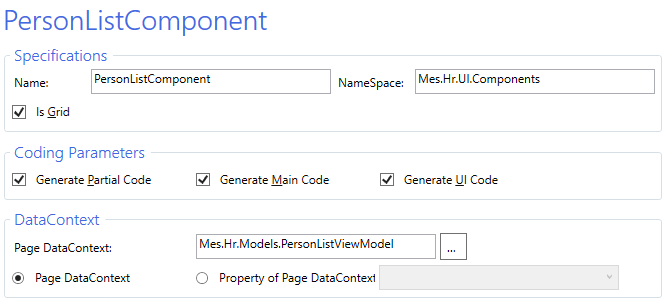
### پنجره Blazor Component Generator

در این پنجره بر اساس View Modelهای ایجاد شده در بخش ایجاد DTO، componentها ایجاد میشوند. البته ناگفته نماند که میتوان componentهای unbound هم ایجاد نمود که به هیچ ViewModelی وصل نیستند. در این صورت، بر اساس متادیتای دریافتی از کاربر، سورس، تولید شده و این متادیتا در دیتابیس، جهت تولیدهای بعدی، ذخیره میشوند. گرچه این کار موضوعیت ندارد، صرفا جهت باز بودن دست برنامه‌نویس، این امکان فراهم شده.

#### General

نحوه کارکرد این پنجره در حالت bound به این ترتیب است:

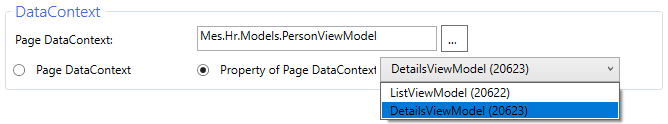
ابتدا با آیکن  دایالوگ انتخاب ViewModel باز میشود. پس از انتخاب ViewModel به طور خودکار بر اساس اطلاعات ViewModel تمام اطلاعات پر میشوند.



در واقع همین حالا میتوان اقدام به تولید سورس کرد. منتهی در صورت تمایل میتوانید تغییراتی در آن بدهید. شاید مهمترین تغییر که گاهی لازم است داده شود، در pageهای (تولید شده) لیستی است. اگر تیک  زده شود، سورس تولید شده، یک page لیستی تولید خواهد کرد.

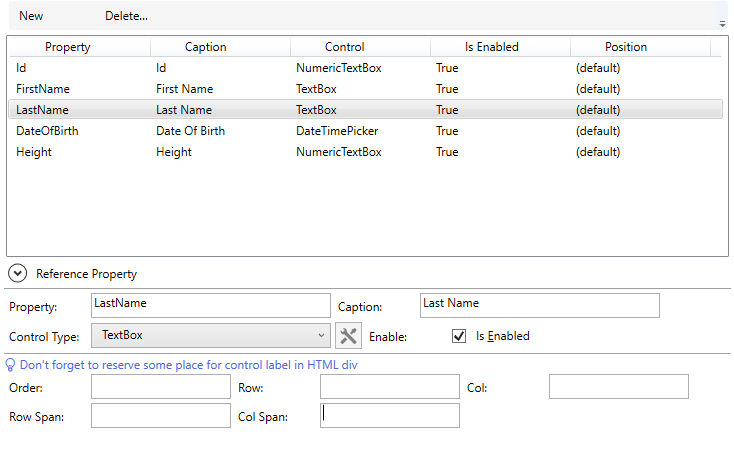
موارد زیر در تنظیم DataContext مهم هستند:

درست است که در زمان ایجاد، ViewModel، سوال میشود ولی در صورت تمایل، میتوان آنرا تغییر داد که در این صورت کل اطلاعات نیز با ViewModel جدید، تغییر خواهند کرد. پس لزومی ندارد در صورت تغییر نظر، یک component جدید ایجاد شود.



در صورتی که یکی از propertyهای یک ViewModel، یک ViewModel دیگر باشد، میتوان آن property خاص را به عنوان DataContext انتخاب کرد. این امکان، از آن جهت قرار داده شده که بتوان pageهای master/detail هم ایجاد کرد. در این گونه pageها، دو component روی page قرار میگیرد. یکی لیست entityها را نمایش میدهد، دیگری با تغییر رکورد انتخاب شده در component لیست، اطلاعات رکورد انتخابی را نمایش میدهد که در صورت نیاز، اطلاعات، تغییر نموده و در دیتابیس، ذخیره شود.

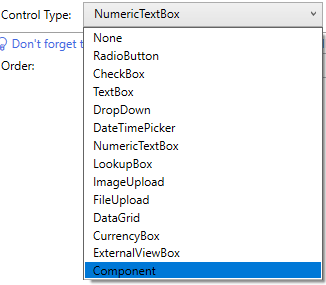
#### Properties



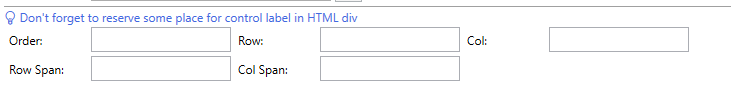
واضح است که در این component باید کنترلهای bindشده به propertyهای data context نیز قرار گیرند. در تب Properties میتوانید این کار را انجام دهید. مضاف بر اینکه میتوانید انتخاب کنید عنوان کنترل چه باشد، کدام کنترل، در کجا قرار بگیرد، فعال/غیرفعال باشد. چه نوع کنترلی برای هر property استفاده شود.

با scroll در لیست propertyها، میتوانید نحوه قرار گیری property انتخاب شده را تغییر دهید. اولین چیزی که باید تغییر کند، Caption کنترل است که منجر به تغییر عنوان کنترل (به صورت پیش‌فرض با کمک یک label) میشود.

همچنین میتوانید نوع کنترلی که قرار است آن property را نمایش دهد نیز انتخاب کنید. در صورتی که این property خودش یک DTO باشد، میتوانید یک component از پیش طراحی شده دیگر را برای نمایش آن، انتخاب کنید.



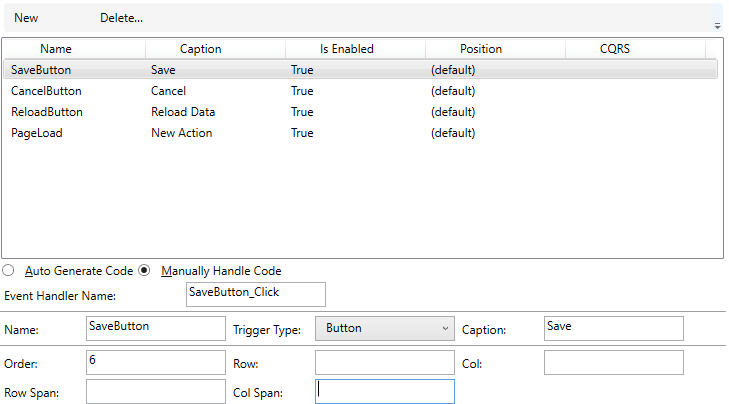
چیدمان Pageی که قرار است ساخته شود، با کمک bootstrap، انجام خواهد شد.



بر همین اساس، شما میتوانید چیدمان کنترلهای روی component را هم تنظیم کنید. فقط به یاد داشته باشید که در قسمت Col، حتما جای خالی برای label هم بگذارید که بصورت خودکار به سورس component اضافه خواهد شد.

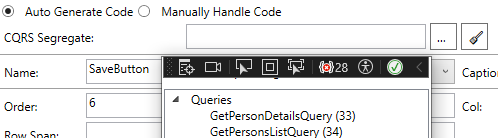
#### Actions

هر componentی میتواند حاوی button نیز باشد. و یا بخواهد در زمان load، اطلاعات خود را initialize کند. برای این منظور، میتوانید از actionها استفاده کنید.



پس از ایجاد action با کمک New، میتوانید انتخاب کنید. هر action میتواند خصوصیات زیر را داشته باشد:

Auto Generate Code:



در صورتی که فقط قرار است یک segregate فراخوانی شود، از این گزینه استفاده کنید. نکته بسیار مهم اینکه DataContext، به عنوان ورودی، به segregate پاس داده خواهد شد. پس بصورت پیش‌فرض، IConverter به عنوان ورودی باید به component داده شود تا با دادن ViewModel، یک Param از converter، تحویل گرفته شود. ولی اینکه این Param، با segregate، همخوانی دارد یا خیر، تست نمیشود. پس انتخاب segregate صحیح، بر عهده برنامه‌نویس است.

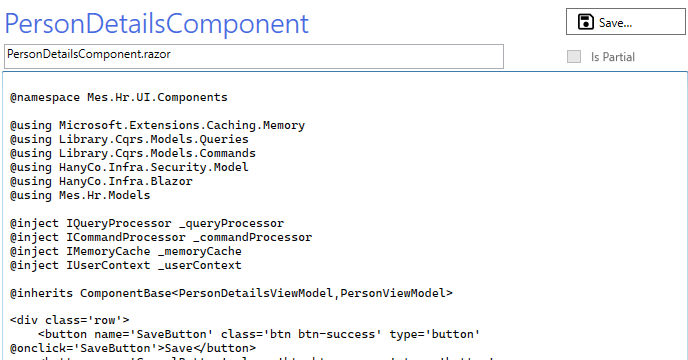
Manually Handle Code:



در این صورت، یک Handler به component اضافه خواهد شد تا برنامه‌نویس، شخصا اقدام به handle کلیک کند.

#### ایجاد و ذخیره سورس‌کد

با زدن کلید سورس‌کدها تولید می‌شوند.

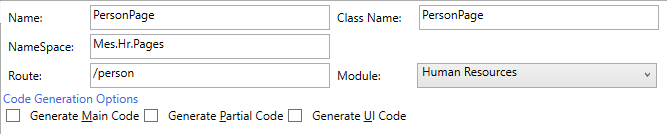


روال، همان روال پنجره‌های دیگر است.

### پنجره Blazor Page Code Generator

#### General

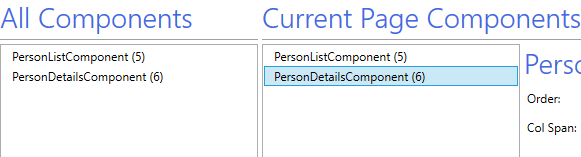
در این پنجره میتوانید با چیدمان componentها در کنار هم، یک page داشته باشید.



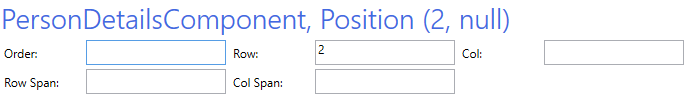
نام page، صرفا جهت اطلاعاتی دارد و کاربرد تولید کد ندارد. Class Name نام کلاس Code Behind این page خواهد بود. Route مسیری است که با کلیک بر روی Menu Item مربوطه، این page در این route در دسترسی خواهد بود.

#### Components

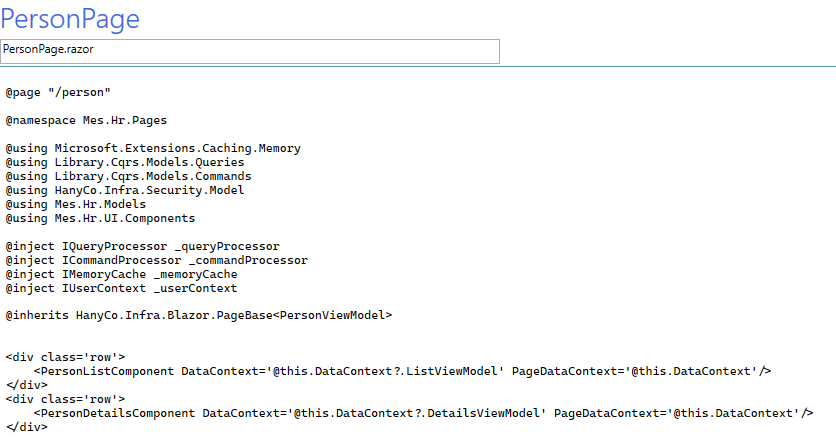
لیست تمامی componentهای موجود در module انتخابی در دسترس است. با انتخاب هریک، میتوان آنرا در page جای داد.



چیدمان page هم مانند component بر با کمک bootstrap انجام میشود.



#### ایجاد و ذخیره سورس‌کد



همه چیز مانند روالهای پیشین است و تفاوتی وجود ندارد.

### پنجره Functionality

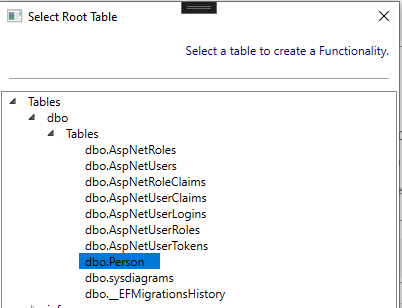
این پنجره صرفا جهت تسریع، سهولت و استانداردسازی کار برنامه‌نویس ایجاد شده. تمامی امکاناتی که در Code Generator وجود دارد و تا کنون بحث شده، بطور بصورت اختصاری در این پنجره، قرار داده شده. با یک Functionality شما میتوانید تمام عملیات back و front یک CRUD رو یکجا ایجاد کنید. در این بخش، به هدف استانداردسازی، دست برنامه‌نویس بسته شده و امکان بسیار محدودی در زمینه کنترل کد ایجاد شده، به برنامه‌نویس داده شده.

#### ایجاد یک Functionality

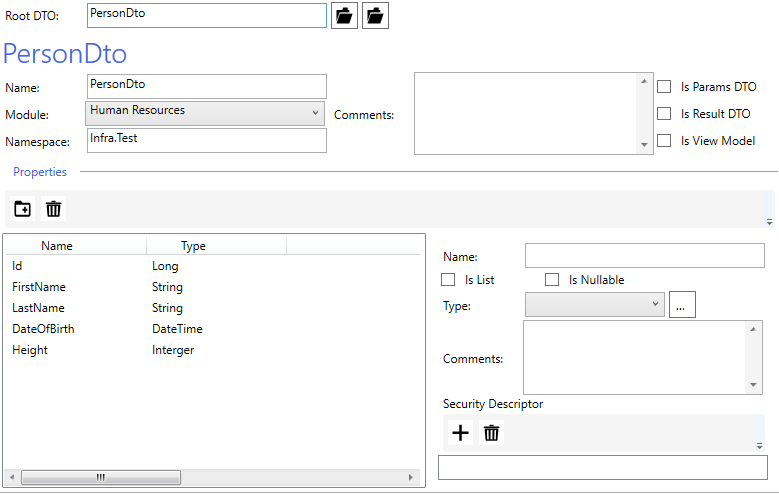
مانند اکثر پنجره‌ها، خروجی میتواند bound یا unbound باشد. تنها مزیت این پنجره این است که در حالت bound، شما میتوانید از DTOهایی که قبلا توسعه پنجره ایجاد DTO، ایجاد کرده‌اید انتخاب کنید یا یک table دیتابیسی را مستقیما انتخاب کنید و DTO مربوطه، در لحظه، ایجاد شود سپس به تغییرات propertyهای DTO بپردازید.

##### ایجاد از طریق جدول دیتابیسی

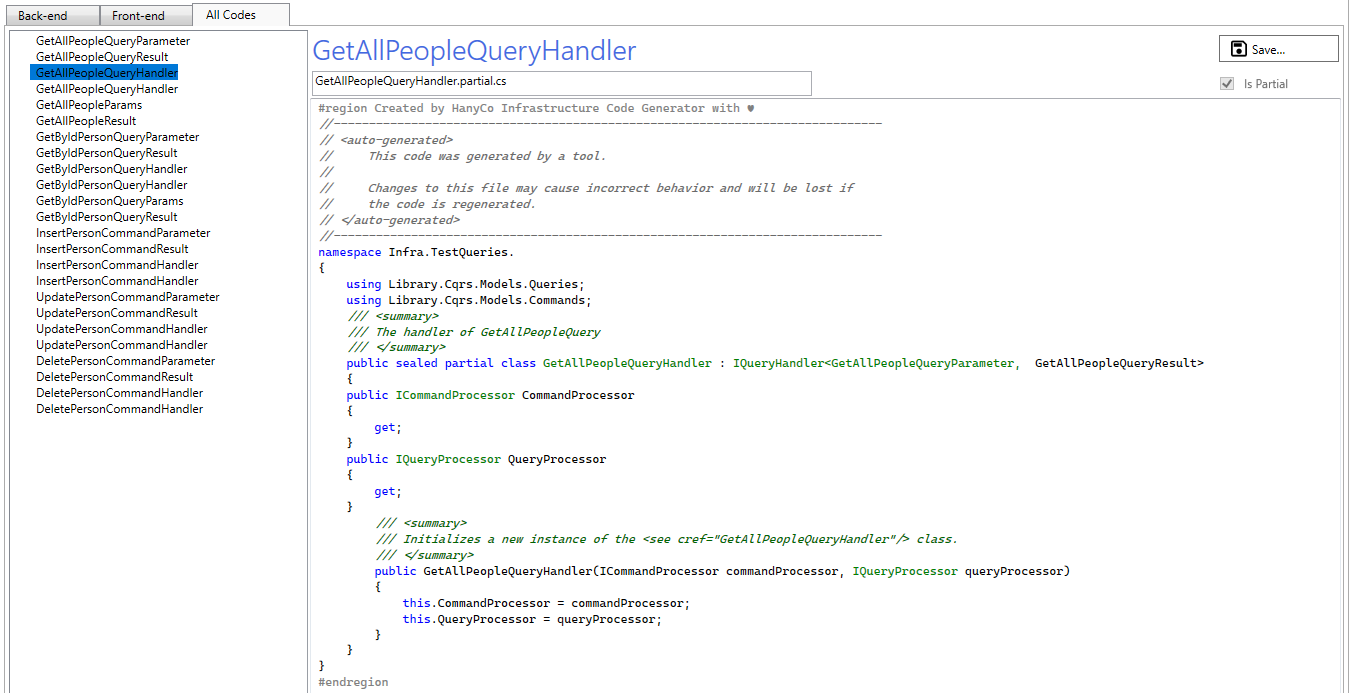
با کلیک آیکن  پنجره زیر باز خواهد شد و شما میتوانید یک جدول را انتخاب کنید:



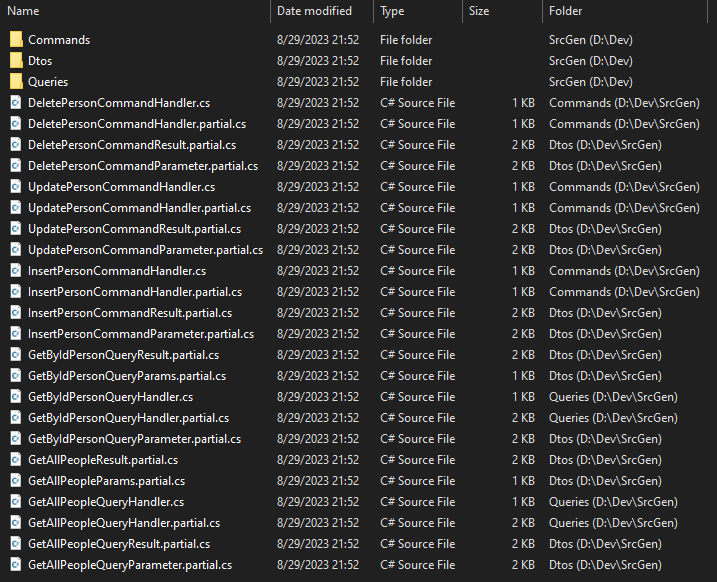
پس از انتخاب جدول، فیلدهای Module و Namespace باید پر شوند. میدانید که Namespace باید تابع قوانین نام‌گذاری در C# باشد. در صورت تمایل میتوانید اقدام به ویرایش propertyها نمایید. نحوه ویرایش، دقیقا منطبق با نحوه ویرایش propertyها در پنجره تعریف DTO میباشد.



سپس با کلیک بر روی  در Toolbar، تمام DTOها، ViewModelها، Segregationها ساخته میشوند. با کلیک بر روی  سورس‌کدها نیز تولید میشوند. میتوانید سورس‌کدها را در تب Results ببینید:



با زدن روی ، سورسهای ایجاد شده، ذخیره می‌شوند. محل ذخیره کدها، قبلا در پنجره Setting، تنظیم شده است.



لازم به توضیح هست تنها بخش باقیمانده: به خاطر یک مشکل کوچک (بسته نشدن tagهای HTML در razor pageها)، من فعلا این بخش را غیر فعال کردم. هم‌اکنون مشغول حل این مشکل هستم و امیدوارم ظرف یکی دو روز آینده، با مرتفع شدن این مشکل، با یک دموی کامل، رضایت شما را جلب کنم.

1. Data Transfer Object [↑](#footnote-ref-1)
2. Infra Code Generator [↑](#footnote-ref-2)
3. یک Command یا Query با تمام اجزا­ [↑](#footnote-ref-3)